Entorno Metodológico Formal de Ingenieria de Requerimientos para Mejorar la Calidad de los Sistemas de Informacion

Germán Huanca Ticona
Postgrado en Informática
Universidad Mayor de San Andrés - UMSA
La Paz, Bolivia
germanh_4@hotmail.com

Resumen. En este articulo se abordo de forma general la problemática relacionada con la ingeniería de requisitos, y más específicamente con la ingeniería de requisitos de sistemas de información. Como resultado de la investigación realizada se obtiene un proceso formal con etapas que permite analizar y posteriormente sintetizar estos sistemas en fusión a los requerimientos que estos deben cumplir.

Keywords— analisis; formal; proceso; requerimiento; síntesis.

V. INTRODUCCION

Se inicio el tratamiento del proyecto, con la revisión de herramientas que fueron proporcionados en el área de los métodos formales para el desarrollo y verificación de sistemas de software, notándose que, aunque se han realizado importantes avances, aun no se ha producido una transferencia efectiva de dichos resultados a la industria del software. Por otro lado muchos métodos formales han sido desarrollados como una alternativa a las prácticas de Ingeniería de Software actuales.

A partir de la revisión que se hizo se pudo evidenciar varias herramientas formales que apoyan al desarrollo de sistemas de software, tales como UniForM Workbench, AutoFocus y Concurrency Factory. A partir de estas experiencias se empezó a plantear aspectos, que nos permitan fundamentar las especificaciones formales necesarias para la aplicación de estos métodos y sean escritas en un lenguaje accesible a diseñadores y programadores. Es por ello que consideramos que la utilización de un lenguaje gráfico facilitará la integración de los métodos formales a las prácticas industriales del software.

En una segunda etapa en el desarrollo del proyecto, se centro en determinar un punto de análisis y verificación de los sistemas de software, notando que ese punto viene ha ser la arquitectura y los componentes de un sistema de software. En vista que los sistemas de software han tenido arquitectura desde que el primer programa fue dividido en módulos, y los programadores han sido responsables de establecer las interacciones entre dichos módulos y lograr así determinadas propiedades globales para estos sistemas.

VI. ANTECEDENTES

La “crisis del software” muestra la lenta evolución que ha tenido la industria del software que data más de 40 años. En la OTAM en los años de 1967 y 1968 se hicieron dos reuniones con el fin de resolver este problema en donde difícilmente resulta ponerse de acuerdo y optar por un estándar completamente definido.

Los avances en el desarrollo de software han revolucionado muchas actividades de la vida diaria. Desafortunadamente, la mayoría de los sistemas existentes en nuestros días presentan fallas en alguna etapa de su uso y no entregan soluciones adecuadas a los problemas para los cuales fueron creados. Esto sucede, principalmente, porque los requerimientos del usuario son mal entendidos o cambian durante el desarrollo del software. Además, tanto el mantenimiento como reutilizar y modificar algunas partes del software son tareas complicadas y tediosas, aún más complicadas son cuando no se utiliza adecuadamente alguna metodología de desarrollo.

Por otro lado, las fases de captura y análisis de requerimientos de usuario han recibido en general poca atención por parte de las metodologías de desarrollo de software, que aparentemente creemos ocasiona problemas como:

- La incapacidad de entregar el software que el cliente requiere y en el tiempo establecido. Este problema se presenta por lo general cuando los requerimientos de un sistema de información en el proceso de su desarrollo cambian debido a que estos requisitos no fueron comprendidos por las personas involucradas
- Al presentarse el problema anterior este acarrea en el costo parcial o total del desarrollo del software por lo que existe

incapacidad de entrega del mismo dentro los presupuestos establecidos y sin sobregiros.

- Dedicación de muchos recursos al mantenimiento del producto software o en su defecto acortan el tiempo de vida del mismo.
- Fracaso en el desarrollo de proyectos software, ocasionando con ello mala inversión económica y de tiempo.

De todo lo expuesto, se plantea la pregunta:

¿Cómo coadyuvar en las primeras etapas, del desarrollo de sistemas de información, para garantizar el cumplimiento de los requerimientos mínimos de los usuarios?

Objetivo

Desarrollar un modelo formal de procesos iterativos como apoyo en la ingeniería de requerimientos, donde se identifican tres actividades principales: elicitación, observación y certificación, permitiendo con ello mejorar la calidad de los productos software.

Hipótesis

La incorporación de un modelo formal de procesos iterativos en la ingeniería de requerimientos permitirá clarificar30 lo que espera un usuario/cliente de un producto software y con ello se mejorará la calidad de los mismos.

VII. MÉTODOS FORMALES

Una de las características deseables de todo sistema es la confiabilidad, definida como la habilidad de un programa para realizar una función requerida bajo condiciones establecidas por un periodo de tiempo establecido. Con el fin de mejorar la confiabilidad de los sistemas producidos, se han desarrollado métodos formales que se aplican en cada paso del proceso de desarrollo a fin de asegurar la correcta construcción del software. Estas técnicas de verificación formal revisten especial interés, pues permiten demostrar con certeza que el sistema desarrollado realiza las funciones requeridas, es decir, cumple con ciertas especificaciones.

Por otro lado, la verificación formal de sistemas de software es una disciplina relativamente antigua en las ciencias de la computación; su antecedente mas destacado se remonta a los trabajos de Floyd, Hoare y Dijkstra sobre verificación de programas a fines de los años 60. Si bien originalmente los métodos formales fueron aplicados solamente a la verificación a posteriori del desarrollo del sistema, debido a que los errores del software son más baratos de reparar mientras más temprana sea su detección en el ciclo de vida del software y a que algunos estudios muestran que los errores de seguridad se presentan más frecuentemente asociados a análisis inadecuados de los requerimientos, es que actualmente estos métodos se han generalizado a todas las etapas del desarrollo de sistemas de software.

El enfoque general de los métodos formales es abstraer los sistemas en modelos matemáticos, establecer la especificación del sistema deseado en términos de propiedades matemáticas que se deben cumplir y luego demostrar formalmente que los modelos producidos verifican dichas propiedades.

VIII. LENGUAJES BASADOS EN ESPECIFICACIONES ALGEBRAICAS.

El álgebra relacional es un formalismo con un mayor poder expresivo que las lógicas modales, pero que puede ser utilizado para darle una semántica a estas y otras lógicas no clásicas. De esta manera, se puede crear un marco relacional uniforme en el cual aplicar, y aún mejorar, las técnicas desarrolladas originalmente en dichas lógicas particularmente útiles son las álgebras Fork, desarrolladas por Baum, Frias, Haeberer y Veloso, pues poseen la característica de ser finitamente axiomatizables y representables.

El objetivo de diseñar un sistema de software es dominar la complejidad de dicho sistema, por lo que, independientemente del paradigma utilizado, el proceso de diseño involucra desarrollar una visión global del sistema a ser desarrollado, establecer su estructura, descomponiéndolo en sub componentes manejables y estableciendo relaciones e interconexiones entre esos componentes, desarrollar representaciones concretas de los datos y especificar aspectos algorítmicos de los procesos.

La estructura es una característica fundamental del software, que permite la descomposición de un sistema en subsistemas más simples, dividiendo la complejidad total del sistema en unidades mas pequeñas y manejables. Las interrelaciones definidas entre dichos subsistemas dan forma a la estructura del sistema. Una estructura modular de este tipo clarifica el diseño y permite realizar abstracción sobre ciertos detalles del sistema.

De esta manera a través de la forma de abstraer a los sistemas de información, considerado en este trabajo permite ver como una herramienta intelectual y por medio de ella tratar con conceptos en vez de instancias particulares de esos conceptos. Durante las etapas de definición de requerimientos y de diseño, la abstracción permite separar los aspectos conceptuales del sistema de los detalles de implementación. En particular, durante la etapa de diseño arquitectónico la abstracción permite tratar consideraciones estructurales antes que detalles algorítmicos, reduciendo la complejidad del proceso de diseño.

IX. CONCLUSIONES

POR LO TANTO PARA EL CUMPLIMIENTO DEL OBJETIVO, EN PRIMER LUGAR, SE CONSIDERARA A LA ARQUITECTURA DE LOS SISTEMAS DE SOFTWARE PARA ELEVAR EL NIVEL DE ABSTRACCIÓN, FACILITANDO LA COMPRENSIÓN DE LOS SISTEMAS DE SOFTWARE COMPLEJOS. EN SEGUNDO LUGAR, A TRAVÉS DE ESA ABSTRACCIÓN SE AUMENTAN LAS POSIBILIDADES DE REUTILIZAR TANTO LA ARQUITECTURA COMO LOS COMPONENTES QUE APARECEN EN ELLA. POR ÚLTIMO, LA NOTACIÓN UTILIZADA TIENE UNA BASE FORMAL ADECUADA, ES POSIBLE ANALIZAR LA ARQUITECTURA DEL SISTEMA, DETERMINANDO CUÁLES SON SUS PROPIEDADES AÚN ANTES DE CONSTRUIRLO.

EN ESTE SENTIDO, EL DOMINIO IDENTIFICADO A TRAVÉS DE LA ARQUITECTURA DE UN SISTEMA SOFTWARE, ASOCIAMOS AL MODELO KRIPKE (MODELO QUE TIENE COMO BASE Y FUNDAMENTO EN LA LÓGICA HIBRIDA, DENOMINATIVO QUE SE DA A LA FUSIÓN ENTRE LÓGICA MODAL Y PROPOSICIONAL) PERMITIÉNDONOS DESCRIBIR POR MEDIO DE UN CONJUNTO DE FÓRMULAS BIEN FORMADAS LAS PROPIEDADES QUE SE DESEA QUE EL SISTEMA CUMPLA.

FINALMENTE, A TRAVÉS DEL ALGEBRA RELACIONAL AL APLICAR EN EL DENOMINADO PROGRAMA DE LA LÓGICA ALGEBRAICA, PERMITE TRADUCIR LOS PROBLEMAS LÓGICOS EN CUESTIONES ALGEBRAICAS, PARA LUEGO USAR LA 'MAQUINARIA' DEL ALGEBRA UNIVERSAL PARA RESOLVER EL PROBLEMA ALGEBRAICAMENTE Y TRASLADAR LA SOLUCIÓN OTRA VEZ A LA LÓGICA ORIGINAL, DE ESTA FORMA LA VERIFICACIÓN DEL SISTEMA SOFTWARE SE HACE MÁS EFECTIVA. COMPLEMENTARIAMENTE AL TRABAJO, OTRO ELEMENTO QUE SURGE CON EL MODELO PLANTEADO PARA LA ESPECIFICACIÓN Y VERIFICACIÓN DE SISTEMAS DE INFORMACIÓN, ES QUE ESTA SE PLASMA EN UN DOCUMENTO DEL SISTEMA SOFTWARE QUE SE TRATA, EL CUAL DE ACUERDO A LAS NORMAS ESTÁNDARES ISO/IEC, LA DOCUMENTACIÓN QUE SURGE EN CUALQUIER PROCESO DE DESARROLLO DE SOFTWARE, ES UN FACTOR PREPONDERANTE PARA QUE ESTE CUMPLA CON LOS REQUERIMIENTOS O ESPECIFICACIONES DEL MISMO.

REFERENCIAS

ARNEDO, J., CABOT, J., GUITART, I. (2010). "ESCANEANDO INFORMÁTICA", EDITORIAL UOC. BARCELONA.

Boehm, B. Bose, P., Horowitz, E. y Lee, M. J. (1994). "Software Requirements as Negotiated Win Conditions". En Proceedings ofthe First International Conference on Requirements Engineering. Disponible en http://sunset.usc.edu/TechRpts/Papers/NGPM-Requirements93.ps.

Calero, C. Moraga, A. y Piattini, M. (2010). "Calidad del producto y proceso software", RAMA-EDITORIAL, Madrid.

Chang, L. y Lin, T. (2000). "Political Games of Users and MIS Professionals in the Information Systems Development Process". Americas Conference on Information Systems, Long Beach, California

Durán, A., Amaya, A. y Toro, M. (1997). "Design of an Automatic Generator of Object–Relational Persistency Mechanisms". En Actas de las II Jornadas de Ingeniería del Software, San Sebastián.

García, F. J. (2000). "Modelo de Reutilización Soportado por Estructuras Complejas de Reutilización Denominadas Mecanos". Tesis doctoral, Universidad de Salamanca.

INCOSE. (2000). Tools Survey: Requirements Management (RM) Tools. Informe técnico, International Council on Systems Engineering, 2000. Disponible en http://www.incose.org/ tools/tooltax.html.

ISO/IEC. (1995). " Information Technology–Software Life Cycle Processes". International Standard 12207 : 1995, International Organization for Standarazition.

Pressman, R. S. (2010). "Ingeniería de Software", 7ma. Ed., McGraw-Hill Interamericana de España S.L.

Somerville, I. (2005). "Ingeniería de Software", 7ma Ed. PEARSON EDUCACION S.A. Madrid.

IEEE (2000). IEEE Guide for Software Reviews and Audits. IEEE/ANSI Standard 1028–2000, Institute of Electrical and Electronics Engineers, 2000.