INTRODUCCION

A pesar de ser una disciplina del conocimiento relativamente nueva, el desarrollo de software ha debido enfrentarse en múltiples ocasiones a retos en los cuales fue necesario mirar atrás y reevaluar el trabajo realizado previamente. Acontecimientos como el renombrado Y2K son ejemplos de estas situaciones y atestiguan las cuantiosas inversiones en esfuerzo y dinero que los procesos de ingeniería inversa requieren. Sin embargo, esto no es una eventualidad aislada que ocurre cada siglo o más, por el contrarío, muchas empresas se enfrentan a diario al desafío de actualizar sus sistemas de información, bien porque su negocio evoluciona, surgen cambios legales o de procedimientos, por la necesidad de incluir nuevas funcionalidades o simplemente por decisiones de actualización tecnológica.

Cuando las organizaciones emprenden procesos de renovación de sus sistemas de información suelen enfrentarse a una triste realidad: aún cuando inicialmente su software haya sido documentado con disciplina, gran parte del conocimiento reflejado en el aplicativo no es reproducible fácilmente. Son muchos los casos en los cuales las correcciones y mejoras que se han realizado a los sistemas, se han ejecutado bajo presión de tiempo, enfocándose en atacar un problema puntual y terminan implementándose según el estilo del desarrollador de turno. El problema se hace aún mayor cuando no se tienen procesos de control de cambio ni disciplina de actualización de la documentación.

Si el proceso de actualización de software corresponde a la necesidad de implementar mejoras en las funciones claves del sistema, integrar a éste con

otros aplicativos o reemplazarlo por nuevos desarrollos, el conocimiento que se posee de la tecnología utilizada por el sistema no es suficiente. De hecho, puede ser más valioso el conocimiento de las reglas de negocio modeladas por el software dado que éstas serán las que garanticen que funcionalmente la integración sea exitosa o, en el caso de migraciones, que el aplicativo nuevo y el anterior sean equivalentes desde la perspectiva del negocio.

Las reglas de negocio son fundamentales para el software puesto que definen o restringen aspectos de la organización. Son directivas que introducen una necesidad o una obligación que cobija conductas, acciones, prácticas y procedimientos de la empresa, estos sencillos enunciados terminan siendo el núcleo de los aplicativos. Si bien el software no sólo modela reglas de negocio, éstas son el elemento central que sustenta la operación del mismo y sólo cambian cuando el propio negocio evoluciona, los demás requisitos modelados por los aplicativos, funcionales o no, pueden cambiar a la luz de necesidades y tecnologías puntuales, pero si una regla de negocio deja de ser soportada por la aplicación, la posibilidad de enfrentar problemas serios en la organización es prácticamente un hecho.

Sin embargo, la identificación de reglas de negocio modeladas en el software no es una tarea fácil, entrelazadas con éstas se encuentran reglas no esenciales, símbolos del lenguaje que no corresponden a términos del negocio, aspectos ligados a la tecnología informática utilizada y sobreespecificaciones, todo esto hace del proceso de extracción y documentación de reglas de negocio una actividad difícil de llevar a cabo.

Este trabajo pretende apoyar la labor de identificación y documentación de reglas de negocio a partir de código fuente, ofrece herramientas que facilitan la labor de ingeniería inversa y explora la utilidad de modelos ontológicos para la

documentación del conocimiento adquirido. Así mismo se explora la conveniencia de utilizar metodologías de trabajo similares a *Synchronized Refinement.*

OBJETIVOS

OBJETIVO GENERAL

Ofrecer herramientas que faciliten el análisis de código fuente, permitiendo al analista identificar entidades y relaciones presentes en un sistema, expresarlas como reglas de negocio y documentarlas con la ayuda de una ontología para facilitar su futuro uso y referencia.

OBJETIVOS ESPECIFICOS

- Generar un analizador léxico que permita Identificar en código fuente, los elementos que puedan representar entidades y relaciones de un dominio.

- Establecer un esquema de representación de los elementos identificados ofreciendo cierto grado de independencia respecto a tecnologías y lenguajes de programación utilizados.

- Ofrecer herramientas que faciliten la lectura y navegación del código fuente por medio de vistas y anotaciones en el código.

- Diseñar mecanismos que permitan asociar elementos de código con su representación en una ontología, al tiempo que facilitan la evolución de la misma y permiten la navegación entre ambos modelos.