4 CONCLUSIONES

El proceso de extracción automática de reglas de negocio a partir de código fuente puede ser, en el mejor de los casos, sólo heurístico, dado que tras las instrucciones del software también se encapsulan reglas no esenciales, símbolos del lenguaje que no corresponden a términos del negocio, aspectos ligados a la tecnología informática y sobrespecificaciones (Baxter, 2005). Sin embargo, metodologías y herramientas para el trabajo en ingeniería reversa pueden ser de gran ayuda a la hora de identificar y documentar reglas de negocio.

La ingeniería inversa es una de las técnicas identificadas por la IEEE para la etapa de mantenimiento del ciclo de vida del software (ABRAN, 2004), su importancia radica en que propende por el mejoramiento de la documentación existente respecto a un producto finalizado (BERKMAN, 2006) buscando controlar la complejidad, generar vistas alternativas, recuperar información perdida, detectar efectos colaterales, sintetizar mayores niveles de abstracción y facilitar el reuso.

Synchronized Refinement apoya el proceso de análisis y entendimiento de un sistema aplicando iterativamente técnicas *bottom-up* y *top-down* hasta encontrar un punto medio donde se obtiene un recorrido, completo y plausible, desde el problema modelado hasta su solución. Esta técnica requiere, para tener éxito, herramientas que soporten el proceso.

La propuesta desarrollada combina tres elementos importantes para llevar a buen término el proceso de identificación de reglas de negocio en código fuente: Un esquema formal de representación de conocimiento (ontologías), una

metodología de trabajo (*Synchronized Refinement*) y herramientas que apoyan el proceso identificando elementos claves en el código fuente y facilitando la navegación entre éste y su representación ontológica (*Plug-In* de eclipse). Estos tres elementos han demostrado simplificar el proceso de ingeniería inversa al ofrecer un entorno de trabajo ágil, flexible y evolutivo, donde no es necesario realizar seguimiento al total de líneas de código para poder identificar elementos importantes en el mismo y comprender su función.

La aproximación utilizada a *Synchronized Refinement* demostró ser muy útil en el proceso de reconstrucción de reglas de negocio, especialmente, por evitar el estancamiento en una sola línea de trabajo y reducir el esfuerzo y tiempo requerido para explorar el aplicativo. Lo anterior, reforzado con una constante documentación en la ontología y su respectivo enlace con el código, permitió ir registrando reglas en el momento que fueron identificadas a pesar de no ahondar en ellas inmediatamente, luego, cuando el proceso lo ameritaba, se pudo retomar estos temas y profundizar en los mismos, reduciendo así la complejidad que implica llevar un rastro de las distintas líneas de análisis que se desprenden de un mismo fragmento de código.

El trabajo realizado permitió identificar reglas de negocio importantes para el sistema en sólo minutos y la tasa de descubrimiento de las mismas se mantuvo durante casi toda la experiencia.

No sólo el proceso de adquisición del conocimiento mejoró notablemente respecto a experiencias pasadas con ingeniería inversa, el uso de ontologías facilitó notablemente la consulta y comprensión de la información obtenida hecho que se hizo evidente al momento de listar las reglas de negocio identificadas durante la experimentación.

5 RECOMENDACIONES

- Se recomienda emprender trabajos orientados a soportar SBVR como estándar para documentación de las reglas registradas en el *Rule Viewer,* esto con el fin de ofrecer una herramienta acorde con los últimos estándares de la OMG.

- Expandir el modelo para soportar no sólo código fuente sino también objetos y procedimientos de las bases de datos, ésta puede ser una fuente adicional e importante de documentación.

- Optimizar el proveedor de contenido del *Entity Viewer* y el mecanismo de creación del archivo ".codml" puesto que el esquema actual, al ser evaluado con aproximadamente 200 archivos fuentes, demostró consumir un gran número de recursos y tiempo para cumplir su labor.

- Explorar la posibilidad de desarrollar *plug-ins* equivalentes con herramientas de libre distribución como *protégé*.