Generación Automática de Código de Interfaces CRUD, en Entornos Web a partir de una Base de Datos, para Ambientes de Software Libre

Juan Francisco Mendoza Moreno,
Daniel Arenas Seleey.
Maestría en Software Libre,
Universidad Autónoma de Bucaramanga UNAB – Universidad Oberta de Cataluña UOC
Bucaramanga, Colombia
jfmendozam@gmail.com

Resumen **– Proyecto de investigación de maestría para la creación de un software CASE (Ingeniería de Software Asistida por Computador) que permita generar el código fuente de aplicaciones, tipo sistemas de información, basadas en interfaces CRUD (*Ingresar, Consultar, Actualizar y Borrar*), generadas directamente a partir de una base de datos (motor mySQL) y cuyo licenciamiento está basado de acuerdo con la comunidad de software libre. El proyecto fue contextualizado de acuerdo con los intereses y expectativas de una muestra de la comunidad de Software Libre de Boyacá (Colombia). El software se desarrolló sobre metodología de creación o adopción de software CASE (concretamente Tools-CASE) tratada por Mario Piattini, et al. Para el desarrollo de software se utilizó un marco de trabajo, basado en el proceso iterativo incremental, como es el caso de SCRUM.**

Palabras clave **- CRUD, Software Libre, SCRUM, Tools-CASE, CASE.**

Abstract - This is a master's research project for the development of a CASE (Computer Aided Software Engineering) software, your goal is produce applications source code (information systems) with CRUD (Create, Read, Update and Delete) interfaces, this interfaces are generated directly over a database (mySQL engine), your license is open source. The project was created according to expectations of a sample of the Boyacá's (Colombia) Free Software Community. The software was developed with a methodology for creation or adoption of CASE software (Tools-CASE) treated by Mario Piattini, et al. For the software development used a framework based on incremental iterative process, as is the case SCRUM.

Keywords— CRUD, Free Software, Open Source, SCRUM, Tools-CASE, CASE.

I. INTRODUCCIÓN

En la Ingeniería de Software como resultado de la llamada "Crisis del Software", se generaron profundas reflexiones sobre la actividad de esta ingeniería, que a su vez propusieron fundamentos teóricos, técnicas y buenas prácticas para poder crear software. Cabe mencionar a los lenguajes de cuarta generación: "software para crear más software" de forma automática (o semiautomática). Se empieza a ofrecer al desarrollador de software herramientas que minimizan su esfuerzo de programación, que permiten disminuir la posibilidad de cometer errores de programación y que le ofrezcan un panorama de interés sobre el propósito del software a crear, en lugar de preocuparse por las especificidades de la herramienta de programación.

De otra parte, el conocimiento es un derecho, un deber y un bien de la humanidad [17], gracias a él, el ser humano ha podido concretar la creación de su entorno, el mundo actual es muy diferente debido a la intervención humana y en el futuro será aún más diferente. Es muy interesante la creación y el uso colectivo de la inteligencia. Cualquier ser humano puede trascender por el conocimiento heredado que deja a sus congéneres y a las futuras generaciones. En la actualidad, el software se consolidó como herramienta propicia para generar y usar el conocimiento. Corrientes socio-económicas parecen que ven de forma distorsionada este principio humano, prima el interés económico y propenden para que el conocimiento sea restringido solo a una porción de la humanidad, aquella que tenga la solvencia económica para usufructuarlo. La protección de los derechos de autor se ha visto confundida con una irracional apropiación del conocimiento malversando el principio de patentar la autoría sobre el conocimiento. El movimiento del software libre tiene como objetivo dar a conocer las libertades de uso, creación y adaptación del software para que el conocimiento sea un derecho, un deber y un bien de la humanidad.

Sin embargo, la labor de desarrollar software es una tarea difícil que requiere esfuerzos por parte del desarrollador, en cuanto a tiempo e innovación intelectual. El desarrollo de aplicaciones basadas en la recuperación de información de una base de datos, mediante un administrador de base de datos (DBMS), requiere de una serie de interfaces al usuario, para que éste a su vez pueda realizar las cuatro operaciones más comunes: Crear, Recuperar, Actualizar y Borrar un registro de información, su acepción en el idioma inglés es más conocida como CRUD (*Create, Retrieve, Update, Delete*). Este tipo de aplicaciones, conocidas por algunos como sistemas de información, en realidad están conformadas en su mayoría por estas interfaces, para permitir al usuario manipular la información de la base

de datos, de acuerdo a perfiles y permisos de acceso a los datos. El desarrollo de estas interfaces es muy repetitivo y puede ser extenuante, además es probable cometer errores de programación.

Precisamente, para generar de forma automática o semiautomática, este tipo de sistemas de información con interfaces CRUD, existe un gran repertorio de herramientas. Algunas de ellas generan el software a partir de diagramas UML, otras de una forma más fácil, permiten generar el software a partir de una base de datos sin necesidad de utilizar este tipo de diagramas. Este tipo de herramientas son poco comunes para ambientes de software libre.

El caso de investigación de este proyecto, precisamente radica en la anterior situación: Proponer a la comunidad de software libre una herramienta que permita generar de forma automática aplicaciones con interfaces CRUD, a partir de una base de datos, con el motor mySQL.

Con este software, la comunidad de software libre tendrá la oportunidad de generar aplicaciones de sistemas de información y a su vez modificar manualmente el código fuente resultante, para empoderar y personalizar la aplicación, de acuerdo con los resultados de las etapas iniciales del ciclo de vida de la ingeniería de software.

La aplicación propuesta en este proyecto, consta de cuatro grandes módulos: parametrización de la aplicación generada, conexión a la base de datos, interpretación del diccionario de datos y generación del código fuente en el lenguaje de programación escogido.

Además el proyecto hace un análisis de las herramientas de similar función, ya sean de software libre o de licencia privativa, para tener en cuenta aspectos técnicos que deben ser considerados en el diseño de este tipo de herramientas.

El estudio de investigación se aplicó a una muestra de población de la comunidad de software libre de Boyacá, para tener en cuenta sus intereses y expectativas con este tipo de proyectos de software y para su vez satisfacer primordialmente las necesidades de esta comunidad.

Para el desarrollo de software se tuvo en cuenta metodologías para el desarrollo de herramientas CASE y metodologías de desarrollo ágil de software, concretamente SCRUM.

II. METODOLOGÍA

El interés de la investigación es probar que si el uso de una herramienta de generación automática de software de tareas básicas reducirá el tiempo de desarrollo de aplicaciones libres en entornos Web que requieren de una base de datos.

El bosquejo metodológico del proyecto se puede apreciar en la Fig. 1.

Fig. 1. Bosquejo metodológico del proyecto (Fuente el autor)

A. Metodología Apropiación herramientas CASE

La generación automática de código sugiere el uso de una metodología CASE, que para este caso, el plan detallado de implantación de herramientas CASE [1] proporciona métodos adecuados para el desarrollo y uso de este tipo de herramientas, tal y como se aprecia en la Fig. 2.

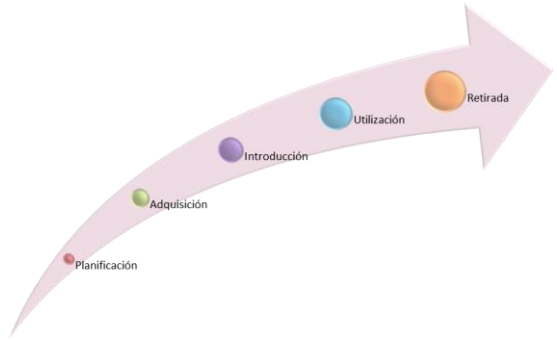

Fig. 2. Metodología apropiación CASE (Fuente [1])

Son cinco las fases que se tienen en cuenta a la hora de desarrollar una herramienta CASE. La primera de ellas es la ***Planificación***, donde se establecieron los estándares y procedimientos necesarios para la consecución del proyecto. También en esta fase se determinó el calendario del proyecto, el personal requerido, que de acuerdo con SCRUM, se pudieron identificar los siguientes roles:

- **Product Owner**: Conocido como el cliente del proyecto, en este caso fueron identificados miembros de la comunidad de software libre de Boyacá, quienes son los interesados en usar el software resultado del proyecto, por lo tanto fueron tenidas en cuenta sus sugerencias al respecto.
- **Scrum Master**: Es el facilitador preocupado para

que todos los miembros del equipo SCRUM sigan las reglas y den continuidad al proceso SCRUM. Es la persona que asegura que la lista de requisitos se cumpla de forma priorizada, además él organiza las reuniones de iteración, de sincronización, de demostración y de retrospectiva. Por último, está atento a eliminar los impedimentos para que el trabajo se pueda realizar y protege su equipo de interrupciones externas.

• **Team**: Es el equipo de trabajo, cuyo tamaño se sugiere entre 5 y 9 personas, pero para el caso del proyecto fue realizado por tan solo una persona que cumplía varios perfiles. Es necesaria la auto-organización para seleccionar y cumplir con los requisitos por cada iteración, identificando las tareas necesarias y estimando el esfuerzo. Por último, demuestra al cliente el cumplimiento de los requisitos por cada iteración.

Además, en esta primera fase se identificó el costo del proyecto, el sistema que se desarrollará (en este caso, la herramienta generadora de software para interfaces CRUD) y el análisis del riesgo.

En la segunda fase, la de ***adquisición***, se busca la herramienta CASE que satisfaga las necesidades estipuladas, en caso contrario, como en este proyecto, se crea la herramienta. Es necesario determinar la infraestructura, el proceso de desarrollo, que en este caso es basado en el ciclo de vida del software, las técnicas y metodologías de desarrollo de software.

Una vez desarrollada la herramienta, viene la tercera fase de ***Introducción***, donde es importante la organización del personal, la instalación y adaptación de la herramienta, la formación y las pruebas y evaluación. Para este caso, se utilizó la estrategia de introducción propia de la comunidad de software libre, la publicación de la herramienta en un repositorio de software libre que brinde los servicios necesarios para dar a conocer la aplicación y poder asistir al usuario de la misma.

Aunque no está desarrollada en este proyecto, la cuarta fase la de ***Utilización***, es compromiso tácito dentro de la comunidad de software libre la asistencia al usuario y las futuras migraciones o actualizaciones de las aplicaciones, no solo por parte del autor del software, sino por parte de la comunidad entera.

Otra fase no desarrollada en este proyecto, la quinta fase la de ***Retirada***, es una fase inevitable en cualquier ciclo de vida del desarrollo del software, cuando este quede obsoleto ya sea por su atraso tecnológico o porque quedó relegado del uso e interés por parte del usuario.

B. Marco SCRUM

Pero a su vez, es necesario el desarrollo de software, por la naturaleza de software libre y por la necesidad del desarrollo rápido, la metodología Agile SCRUM, proporciona los elementos necesarios para el desarrollo de esta aplicación. Se hizo el seguimiento del proyecto, con la herramienta Trello [2], de acuerdo con las características del marco SCRUM, como se puede apreciar en la Fig. 3.

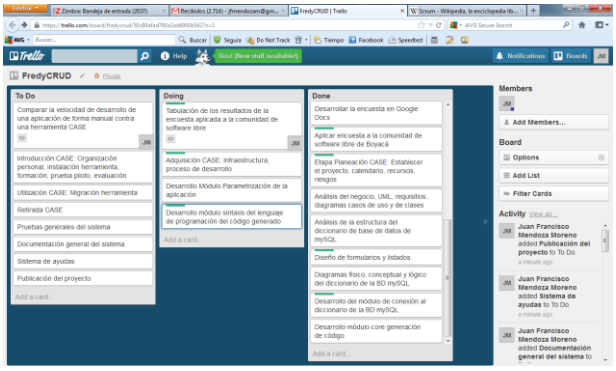

Fig. 3. Seguimiento del proyecto mediante Trello, una herramienta SCRUM (Fuente [2])

El uso de estas dos metodologías es requerido porque la investigación exige probar una herramienta con la muestra de la población, que en la actualidad no existe cumpliendo con la totalidad de esas características (que genere código de forma automática, que cree interfaces CRUD, que su punto de partida sea una base de datos ya construida, y que sea en ambientes de software libre).

C. Metodología del proyecto de Investigación

Se empezará desarrollando las etapas de la investigación cuantitativa que empieza con la ***formulación del problema de investigación***, basándose en preguntas referentes a la trascendencia de esta investigación y su pertinencia. De lo anterior se dedujo el problema de una necesidad de tipo disciplinar, que permitió la definición del título y sus objetivos.

En la fase de ***exploración*** se extrae la revisión de la literatura y se construye el marco teórico de la investigación, se busca un esquema de software libre para apoyar a la misma comunidad, como algunas soluciones propietarias lo hacen y el por qué es una de las estrategias de implantación a seguir, teniendo en cuenta fundamentación teórico - práctica.

Dentro del ***diseño*** de la investigación se obtiene el tipo de estudio que se realizará, en este caso descriptivo, analiza cómo es y cómo se manifiesta un fenómeno y sus componentes. Permite detallar el fenómeno estudiado básicamente a través de la medición de uno o más de sus atributos. Se escogió el diseño apropiado de la investigación, en el diseño experimental donde se construye una realidad.

A continuación se procede con la ***extracción*** de la muestras, teniendo en cuenta que sea representativa o que se refleje el conjunto o universo que se va a estudiar, reproduciendo de la manera más exacta posible las características de éste. Su tamaño sea estadísticamente proporcional al tamaño de la población. El error muestral

se mantenga dentro de límites aceptables. Se utilizará muestras probabilísticas donde los resultados son generalizables a la población.

Para la ***recopilación de datos*** se debe efectuar los tres procedimientos o técnicas más usadas en este proceso, la observación, la entrevista y encuestas. En la observación se utilizará en cada empresa en la observación de los procesos habituales que interviene en la investigación. Se harán entrevistas a equipos de desarrollo de software libre, técnicos y usuarios finales para tener la información de primera mano sobre los problemas principales. Las entrevistas se harán a los usuarios finales.

Se efectuará el ***análisis de la información de la investigación***, donde primero se escriben los datos estadísticos recopilados (representación escrita), tabulación de la información ordenada con los datos numéricos en filas y columnas, con las especificaciones correspondientes acerca de su naturaleza, es necesaria la representación gráfica de los datos. Se diseñará el diseño del informe final, donde se mostrarán las concusiones de la investigación, obtenidas en un análisis detallado de los datos.

De otra parte, para el desarrollo de la herramienta informática se propone marcos de desarrollo rápido, como es el caso de Agile SCRUM, que permiten el desarrollo "ágil" del proyecto de software. Se basan en los principios de la simplicidad, la comunicación, la retroalimentación y el coraje para implicar a todo el equipo (y a los usuarios o clientes) en la gestión del proyecto [3].

III. Desarrollo Metodológico del Proyecto

El proyecto se propuso en tres objetivos:

- Identificar los requerimientos que debe reunir una herramienta de generación automática de código considerados por los desarrolladores de software libre de la región (Boyacá - Colombia)
- Realizar un diagnóstico comparativo del uso de herramientas actuales, de software libre o propietario, para la generación automática de código a partir de una base de datos
- Crear una herramienta de automatización para la generación de código de interfaces CRUD, a partir de una base de datos, en ambientes de desarrollo de software libre.

A. Requerimientos de una Herramienta de Generación Automática de Código Considerados por los Desarrolladores de Software Libre de la Región

El grupo de "Software Libre de Boyacá", se define a sí mismo como: "*El grupo de Software Libre Boyacá es un punto de encuentro de saberes, ideas, iniciativas y actividades en torno al Software Libre ubicado en el contexto de Boyacá pero sin desconocer la necesidad de interactuar con el exterior*" [4]. Es un grupo de personas voluntarias que pretende difundir y adaptar el software libre en el Departamento de Boyacá y sus regiones circunvecinas, conformado por entusiastas y expertos en el software libre. La misión de la comunidad se puede apreciar en seis actividades relacionadas con el software libre:

- Difusión y promoción del modelo
- Formación de usuarios, empresas y administraciones
- Generación de confianza
- Asesoramiento a empresas
- Promoción del tejido empresarial TIC de la región
- Coordinación del proyecto distribución del sistema operativo

Se escoge dentro del grupo de Software Libre de Boyacá, la muestra de la población objetivo, necesaria para indagar por las características contextualizadas en la región, en especial en el desarrollo de software, grupo que apreciaría, en un principio, por disponer de herramientas de generación automática de software libre que beneficien a la propia comunidad. Aunque inicialmente el instrumento estadístico se aplicaría con los miembros en general del grupo, su enfoque de interés son los miembros desarrolladores o involucrados en el desarrollo de software del mismo grupo. La fuente primaria de la investigación está constituida por la aplicación del instrumento estadístico a la muestra de la población objeto de estudio. Con los resultados de esta encuesta se perfila la identificación de los requerimientos que debe reunir la herramienta de generación automática de código contextualizada con las necesidades de la comunidad de software libre en la región. Sin embargo, otras fuentes de información, especialmente las relacionadas con el estándar de herramientas CASE, propuesta en la norma ISO / IEC 14102, como el trabajo monográfico de Elison Roberto Imenes [5], el Método GQM [6], el paper de "I Gusti Made Aditya" [7], el artículo de indicadores organizacionales para la selección de Herramientas CASE [8], Criterios para la Selección de Herramientas de Ingeniería de Software en PYMEs [9] y el estado de arte de este proyecto, ayudaron a identificar estos requisitos, como se puede apreciar en la Fig. 4.

Fig. 4. Encuesta de percepción para desarrollar la herramienta (Fuente: El autor)

Para determinar los requerimientos que debe reunir una herramienta de generación automática de código, considerados por los desarrolladores de software libre de la región, se parte de los resultados de este instrumento estadístico, de los cuales se puede concluir que:

- Todos alguna vez han desarrollado software, un 60% ha utilizado plataformas Java para desarrollo y un 30% a ASP.NET
- El 60% de ellos ha desarrollado en entornos de software libre, principalmente con Java y PHP, como plataformas de desarrollo, con un 20% cada una
- El 80% ha utilizado herramientas CASE, alguna vez, para ayudar su desarrollo de software, con herramientas como *plugins* de IDE como NetBeans o Visual Studio, Power Designer, Enterprise Architect, cada uno de ellos con un 20%; y Symfony y argoUML.
- Para el 60% les gusta usar las herramientas CASE, por cuanto les parecen muy útiles. El 10% la utiliza de forma indiferente, mientras que el 30%, no les gusta por cuanto prefieren hacer su labor de desarrollo de forma tradicional.
- Aprecia como ventajas principales del uso de herramientas CASE porque reducen el esfuerzo de programación y evitan tareas repetitivas (cada una con un 80%), además pueden reducir el tiempo de programación (70%) y permite la estandarización del código (60%)
- Como desventajas ven complicado entender el estilo de programación (80%), posibilidad de generar código ineficiente (60%) y el posible mantenimiento complicado (50%).
- Prefieren que el código generado sea en las primeras etapas del ciclo de vida de software y ojalá automáticamente a partir de una base de datos (20%)
- Principalmente usarían herramientas CASE para la generación de interfaces del usuario (50%) y navegabilidad del mismo en el sistema (60%), dotadas con un sistema de seguridad (70%)
- Es indiferente la plataforma para la cual se genere código: Web (100%), Móviles (90%) y escritorio (30%).
- Exigen que el código generado debe estar bien documentado (80%), estandarizado (70%) y bien estructurado (50%).

La herramienta propuesta en este proyecto de investigación se clasificaría como *Tools-CASE*, porque parte del diccionario de una base de datos previamente creada, concebida como una herramienta de programación para software de sistemas, es decir solamente es una herramienta generadora de código, no modela, ni tampoco controla el proyecto de ingeniería.

El proceso de validación o selección de una herramienta CASE, debe pasar por los siguientes procesos:

Fig. 5. Proceso de evaluación y validación herramienta CASE (Fuente [5])

Los requerimientos detectados para este tipo de herramientas fueron agrupados en los siguientes ambientes:

- Componentes de la aplicación
- Requerimientos de la comunidad de software libre
- Implementación
- Pruebas
- Usabilidad
- Eficiencia
- Mantenibilidad
- Portabilidad
- Criterios generales

B. Diagnóstico Comparativo del Uso de Algunas Herramientas Actuales, para la Generación Automática de Código, a partir de una Base de Datos

A partir del estado de arte realizado en esta investigación y con el apoyo de material bibliográfico, principalmente de [5], [9] y [10], se pudieron analizar

algunas herramientas CASE que generaban el código a partir de una base de datos, en la mayoría de los casos usando versiones de prueba, porque son herramientas con licencia privativa, y en otros casos, basándose en el análisis previo que algunos autores hayan hecho.

Las herramientas analizadas fueron:

- e.World PHPMaker
- Sybase PowerDesigner
- Sparx Enterprise Architect
- Visual Paradigm for UML
- Oracle SQL Developer Data Modeler
- VisualWade
- Jboss Company Hibernate

Cada herramienta fue analizada con los siguientes criterios:

- Usa un SGBD (Sistema Gestor de Base de Datos)
- Tipo de licencia
- Usa filosofía de arquitectura abierta
- Genera utilidades de software, procedimientos de lectura, bibliotecas de fuentes y crea especificaciones
- Lenguaje de programación del código generado
- Ofrece interfaz con otras herramientas
- Dispone de utilidades gráficas, capaces de generar diagramas de proyecto y especificaciones de diccionario
- Provee capacidades de prototipado
- Genera automáticamente un ámbito de aplicación de las especificaciones de diseño físico de las especificaciones del proyecto
- Genera documentación para el usuario final
- Ofrece diferentes opciones de estilo o temas para el usuario final
- Maneja roles de usuarios y esquemas de seguridad para ellos
- Posibilidad de exportar los reportes a otros formatos, o de enviar vía correo electrónico
- Permite formas de navegación en la aplicación resultante
- Ofrece ingeniería inversa

Aunque las herramientas analizadas anteriormente pueden generar el código de la nueva aplicación a partir de una base de datos existente, la mayoría de ellas requieren invocar manualmente un proceso de ingeniería inversa para la lectura del diccionario de datos del motor. Herramientas que no requieren invocar al proceso de ingeniería inversa y que se destacan por la funcionalidad de conexión directa, están PHPMaker e Hibernate. PHPMaker tiene una licencia privativa, mientras que Hibernate aunque maneja una licencia de código abierto, trabaja mediante *plugin* en ambientes integrados de desarrollo (IDE) como es el caso de NetBeans y Eclipse, restringiendo el desarrollo del software a ese tipo de IDE, lo cual hace perder la característica de generación automática de software, ya que requiere de actividades manuales.

Por facilidad de uso y potencia al generar código, se destacan herramientas con licencia privativa, como es el caso de PHPMaker y PowerDesigner. PHPMaker se preocupa por brindar más opciones de diseño de interfaz al usuario, con un amplio sistema de navegación por menús, multiplicidad de opciones de conversión de formatos y envío, así como la seguridad de la información, mediante el perfilamiento de usuarios.

Con respecto a la documentación, son muy destacables las posibilidades que brinda PHPMaker con respecto a las demás herramientas analizadas. Brinda información dentro del código, como documentación externa de acuerdo con estándares de la ANSI.

PHPMaker se muestra como la herramienta más completa y sólida para generar código de forma automática a partir de una base de datos, lo puede hacer en diferentes lenguajes de programación, pero la herramienta es diferente, aunque es de la misma casa, por ejemplo, a parte de PHPMaker, existen ASPMaker, JSPMaker y CFMMaker, por citar algunos ejemplos. Pero esta herramienta no tiene licencia apropiada para la comunidad de software libre.

C. Creación de la Herramienta de Automatización para la Generación de Código de Interfaces CRUD, a partir de una Base de Datos, en Ambientes de Desarrollo de Software Libre

1) SCRUM

Para permitir el desarrollo ágil de la aplicación se hizo uso del marco SCRUM, cuyas actividades fueron seguidas mediante la herramienta Trello [2], el tablero principal (board - dashboard) se denominó con el nombre del proyecto “FredyCRUD”, este tablero se divide verticalmente en tres columnas, la primera “*To Do*”, que contiene las actividades “por hacer”, la columna de en medio “*Doing*” indica las actividades que se están haciendo a la fecha, y la tercer columna “*Done*”, contiene las actividades finalizadas (hechas). Y horizontalmente se encuentran los “*Sprints*” o hitos del proyecto (determinados por una pestaña de color verde). El primer *sprint* se definió de acuerdo al primer objetivo del proyecto, es decir identificar los requerimientos que debe reunir una herramienta CASE de este tipo. El segundo *sprint* tiene que ver con el segundo objetivo del proyecto, es decir la generación de un diagnóstico comparativo de herramientas CASE actuales que sean similares a la herramienta desarrollada en este proyecto. El tercer objetivo, por su amplitud y exigencia, requiere de varios *sprints* que están determinados por cada una de

las etapas de desarrollo de una herramienta CASE y por cada una de las fases del ciclo de vida de Ingeniería de Software para el desarrollo de la aplicación.

El autor del proyecto, asumió de forma paralela los roles de *ProductOwner* (interlocutor del cliente), *ScrumMaster* (facilitador), Desarrollador y en forma conjunta el rol de *Manager*. Los *stakeholders* (clientes) fueron asumidos por algunos miembros de la comunidad de software libre de Boyacá, compartiendo el rol de *Manager*. Aunque suene paradójico, se realizaban reuniones por cada *sprint* (*Daily Scrum*), que como los todos roles eran asumidos por el autor del proyecto, estas reuniones se convertían en un punto de interrupción (*breakpoint*), para evaluar de forma muy rápida el avance en el cumplimiento de las actividades. Entre cada *sprint* era necesario planear el siguiente *sprint* (Spring *Planning Meeting*) donde se replanteaban estrategias para poder llevar a cabo ese *sprint*. En el cumplimiento de un *sprint* se aplicaba el *Sprint Review Meeting*, para revisar el trabajo realizado y para aprovechar mostrar el avance del proyecto a los *skateholders*.

En la segunda fase del desarrollo de una herramienta CASE, la Adquisición, existe una gran actividad que consiste en el desarrollo de la aplicación, donde se aplican todas las etapas del ciclo de vida del desarrollo de software.

2) *Ciclo de vida del software*

Comprendido dentro de la segunda fase de desarrollo de herramientas CASE, concebido por la ISO [11] como "*El Marco de Referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso*".

Dentro de los procesos del ciclo de vida del software [18], se pueden mencionar tres grandes grupos:

- Procesos principales: Comprenden la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento
- Procesos de soporte: Comprenden la documentación, la gestión de la configuración, el aseguramiento de la calidad, la verificación, la validación, la revisión en equipo, la auditoría y la resolución de problemas
- Procesos de la organización: Comprenden la gestión, la mejora, la infraestructura y la formación

El modelo a seguir este ciclo de vida fue el iterativo incremental, donde al final de cada ciclo se entrega una versión del software mejorada. Estos ciclos se van repitiendo hasta obtener un producto satisfactorio. La evaluación del producto por parte del usuario se da en cada iteración. El desarrollo de software en este proyecto es un proceso empírico, no definido, por lo tanto, se necesita de mayor control en la aplicación. Cada iteración se conoce como *Sprint* y dentro de cada *Sprint* existen reuniones diarias muy rápidas (*Scrum*). Esta metodología fue escogida por cuanto los requisitos son cambiantes y emergentes, el equipo de trabajo se auto-organiza y el proceso debe ser ágil y escalable.

En las fases iniciales del proyecto para facilitar la comprensión, visibilidad y documentación, se hizo necesario el uso de software de apoyo para las fases de análisis y diseño principalmente, como herramientas UML y herramientas como PowerDesigner, que es una solución gráfica de modelamiento empresarial que soporta metodologías y notaciones estándar.

3) *Análisis de Requisitos y Diseño de Software*

El trabajo inició con el modelamiento del Proceso del Negocio (BPMN), luego el modelo de requisitos, el modelo orientado a objetos, el modelo XML, el modelo de movimiento de datos y los modelos físicos, conceptuales y lógicos, soportado con la herramienta PowerDesigner.

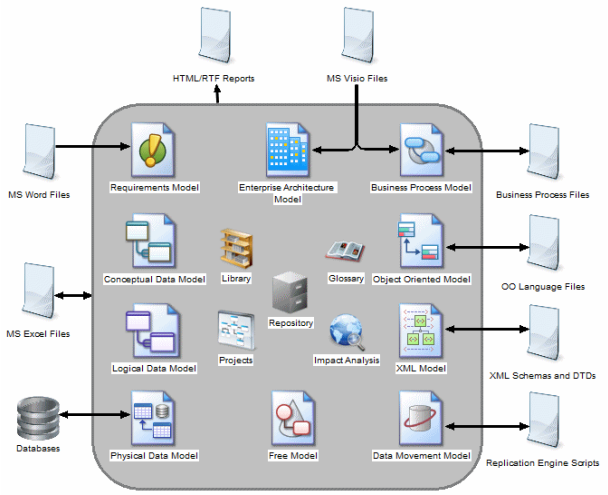

Fig. 6. Modelamiento del proyecto en PowerDesigner (Fuente *OnLine Help PowerDesigner*)

Según este modelamiento, las principales etapas del proyecto son:

- Crear nueva aplicación. Donde se invoca el software para la generación del nuevo sistema de información
- Parametrizar aplicación. Donde se configura y personaliza la nueva aplicación
- Conexión DB. Se establece la comunicación con el motor de la Base de Datos (mySQL)
- Generación de la Aplicación. Se obtiene el código fuente de la nueva aplicación

- Modificación de la aplicación. Posibilidad para que el desarrollador modifique y empodere el código fuente de la nueva aplicación generada
- Uso de la aplicación. Utilización de la nueva aplicación generada.

Con respecto al modelamiento para conocer el Negocio, se desarrollaron varios modelos:

- Organigrama Con doce perfiles de integrantes identificados
- El Diagrama de Comunicación del Negocio, donde se analizan las interacciones de los perfiles identificados con las etapas del negocio.
- Diagrama de Planeación de la ciudad. Se establecen tres grandes áreas: Fuente de datos, Aplicación y Relación con el cliente
- Diagrama orientado al servicio, con cuatro capas: Negocio, Servicio, Aplicación y Tecnología
- Diagrama arquitectura de la aplicación. Tiene en cuenta la arquitectura de tres componentes: Sistema de Información, el Servidor de Aplicaciones y el Servidor de la Base de Datos.
- Diagrama de Infraestructura Tecnológica, analiza los dos extremos (*sides*): cliente y servidor
- Mapa de procesos. Agrupados en dos macro-procesos: los de administración y los de operación. Son nueve procesos en total.

Un modelo de requerimientos (RQM) permite analizar cualquier clase de requerimientos por escrito y asociarlos con los usuarios y grupos quienes los implementarán y con los objetos de diseño de otros modelos. Se puede usar los RQM para representar cualquier documento estructurado (por ejemplo, las especificaciones funcionales, el plan de prueba, los objetivos del negocio, entre otros) y se pueden importar o exportar las jerarquías de los requerimientos.

El Modelamiento Orientado a Objetos permite analizar un sistema de información a través de casos de uso, análisis estructurales y de comportamiento y en términos de desarrollo, se usa el Lenguaje Unificado de Modelamiento (UML). Se puede modelar, aplicar ingeniería inversa y generar código en distintos lenguajes de programación. Se desarrollaron los siguientes diagramas UML:

- Diagramas de Casos de uso
- Diagramas Estructurales:
  - Diagrama de Clases
  - Diagramas de Estructura Compuesta.
  - Diagrama de objetos
  - Diagrama de paquetes
- Diagramas Dinámicos
  - Diagrama de Comunicación
  - Diagrama de Secuencia
  - Diagrama de Actividades
  - Diagrama de Estados
  - Diagrama de Interacción
- Diagramas de Implementación:
  - Diagrama de Componentes
  - Diagrama de Despliegue
  -

Con respecto al Modelamiento de Datos, se diseñaron tres modelos:

- Modelo Físico de Datos o PDM
- Modelo de Datos Conceptual y Lógico

4) Codificación

El Lenguaje de programación escogido para la codificación fue Java, por cuanto se ha constituido como la base de cualquier aplicación en red y es un estándar mundial de desarrollo, desde aplicaciones, pasando por juegos, hasta aplicaciones empresariales. Tiene más de 9 millones de desarrolladores en el mundo. Su ecosistema es maduro y su rendimiento muy sólido y ofrece portabilidad en diversos entornos [12].

Por lo tanto, el paradigma de programación es la orientada a objetos, donde impera que los programas deben ser diseñados, precisamente la disciplina de la Ingeniería de Software se preocupa por la construcción de programas correctos, que trabajen y que estén bien escritos. El ingeniero de software intenta utilizar métodos aceptados y suministrados para analizar el problema a ser resuelto y para diseñar el programa que resolverá el problema. El concepto central radica en el objeto, que es una clase de módulo que contiene datos y subrutinas [13]. En la programación orientada a objetos, ese objeto es una clase de entidad autosuficiente que tiene un estado interno (datos que contiene) y que puede responder a mensajes (llamadas a sus subrutinas). La programación orientada a objetos aprovecha la ingeniería de software al inicio con la identificación de objetos involucrados en un problema y los mensajes que esos objetos deberían responder. El programa resultante es una colección de objetos cada uno con sus propios datos y su propio conjunto de responsabilidades. Los objetos que tienen los mismos tipos de datos y responden a los mismos mensajes de la misma forma pertenecen a la misma clase, aunque pueden existir objetos similares de clases diferentes. En resumen la programación orientada a objetos puede ser una herramienta superior de desarrollo de programas y una solución parcial para el problema del reuso de software.

Como entorno de desarrollo (IDE) fue escogido a NetBeans, producto de Oracle y con más de 100 socios, es un entorno de desarrollo integrado libre, hecho principalmente para lenguaje Java, aunque soporta otros lenguajes de programación. Es un entorno de desarrollo - una herramienta para que los programadores puedan

escribir, compilar, depurar y ejecutar programas. Está escrito en Java. Existe además un número importante de módulos para extender el NetBeans IDE [14].

La documentación del código es apoyada por Javadoc . Javadoc es una herramienta para generar documentación API en formato HTML desde comentarios de documentación en el código fuente. Javadoc se descarga como parte del Java 2 SDK. El Doclet genera el documento HTML y se construye en la herramienta JavaDoc [15].

Las pruebas de software se hicieron mediante JUnit [16]. JUnit es un sencillo framework para escribir pruebas repetibles. Es una instancia de la arquitectura xUnit para frameworks de prueba unitarias.

La gestión del versionamiento se hace mediante Subversión en un servidor Google.

CONCLUSIONES

Dentro de los requerimientos de una herramienta de generación automática de código considerados por los desarrolladores de software libre de la región están agrupadas en varios ambientes de acuerdo con los componentes, la comunidad de software libre, implementación de la herramienta, pruebas sobre la misma, usabilidad, eficiencia, mantenibilidad, portabilidad, entre otros requerimientos.

Al aplicar el instrumento de recolección de información a la muestra de la comunidad de Software Libre de Boyacá, manifiesta que un tipo de herramientas de generación automática de código son muy necesarias para su labor porque les permite contribuir más con la comunidad con soluciones efectivas y adecuadas con la necesidad de la región. A los entusiastas les permite adentrarse y colaborar más efectivamente.

Con respecto al diagnóstico comparativo del uso de algunas herramientas actuales, para la generación automática de código, a partir de una base de datos, fueron muchas las herramientas encontradas, pero la mayoría de ellas tienen licencias privativas. De las pocas herramientas con licencia adecuadas a la comunidad de software libre, pocas de ellas generan código de forma automática a partir de una base de datos, porque principalmente trabajan a partir de diagramas UML. Y las únicas herramientas con licencia libre, que generan código a partir de una base de datos, son aquellas que funcionan como componentes de un IDE (Ambiente Integrado de Desarrollo) lo que implica mayor trabajo manual de codificación y pocas posibilidades de personalización.

Este diagnóstico comparativo realizado denota la necesidad que tiene la comunidad de software libre de disponer este tipo de herramientas, para aportar aún más en satisfacer las necesidades de la región. A su vez se presenta como un sustento teórico que justifica la realización de estos tipos de proyectos de investigación

Para la creación de la herramienta de automatización para la generación de código de interfaces CRUD, a partir de una base de datos, en ambientes de desarrollo de software libre, se encontraron muchos problemas, de los cuales el que más impacto generó fue la falta de personal para conformar el equipo de desarrolladores, porque el software solamente fue desarrollado por una sola persona (el autor). Como estrategia, se hizo uso de métodos de desarrollo ágil de software como SCRUM, que aunque exige un equipo entre 5 y 9 personas, el único autor del proyecto tomó como estrategia manejar varios perfiles de usuario para seguir los procesos SCRUM. Aunque el proyecto de desarrollo tomó más tiempo de lo estimado, fue factible su realización.

El software fue analizado y diseñado de una forma muy sencilla, tan solo en tres grandes módulos: Parametrización de la nueva aplicación, conexión al diccionario de datos y generación del código de la nueva aplicación. Sin embargo, en la etapa de codificación la complejidad de cada uno de estos módulos fue mayor, verificable por la cantidad de código creado.

El modelamiento del software en las fases de análisis y diseño requirió de apoyo de software para facilitar la labor y disminuir la complejidad del proyecto. Sin embargo, hay que reconocer que herramientas con licencias privativas fueron muy prácticas para realizar el trabajo. Queda como recomendación a la comunidad de software libre tener en cuenta esta necesidad para futuros proyectos de investigación.

Efectivamente la hipótesis que se había planteado en el proyecto con respecto a que el uso de una herramienta de generación automática de software de tareas básicas reduciría el tiempo de desarrollo de aplicaciones libres en entornos Web que requieren de una base de datos se cumplió. Era de suponerse por cuanto codificar interfaces CRUD de forma manual es una tarea que requiere mucho tiempo, es rutinaria y propensa a cometer errores de codificación.

RECONOCIMIENTOS

Especiales reconocimientos al asesor de este proyecto de investigación de maestría, Ingeniero Daniel Arenas, por su permanente asistencia y colaboración.

REFERENCIAS

[1] PIATTINI, M. G., CALVO-MANZANO, J. A., CERVERA, J., & FERNÁNDEZ, L. (2004). Análisis y Diseño de Aplicaciones Informáticas de Gestión. México: Alfaomega Rama.

[2] TRELLO. Trello. Recuperado de http://trello.com Junio 2012.

[3] HERNÁNDEZ, J. M., & al., e. (2007). Ingeniería del Software en Entornos del Software Libre. Catalunya: Fundació per a la Universitat Oberta de Catalunya.

[4] Software Libre de Boyacá. Definición. Recuperado de http://softwarelibreboyaca.org/index.php/es/quienes-somos, en Junio de 2012

[5] ROBERTO IMENES, Elison. Seleção de Ferramentas CASE. Jaguariúna. 2006

[6] SOLIGEN, Rini van y BERGHOUT, Egon. The Goal / Question / Metric Method: A Practical Guide for Quality Improvement of Software Development. The McGraw-Hill Companies, London, England, 1999
[7] I Gusti Made Aditya. CASE Tools Comparison. 2009
[8] MENDOZA, Luis Eduardo, et al. Organizational Indicators for CASE Tools Selection: A Case Study. Revista Colombiana de Computación – RCC. Editorial UNAB
[9] RIVAS, Lornel, et al. Criterios para la Selección de Herramientas de Ingeniería de Software en PYMEs. Universidad Simón Bolívar. Revista de la Facultad de Ingeniería UCV. Vol. 25, No 1, pp. 89-104. 2010
[10] INEI. (1999). *Herramientas CASE*. Perú: Editorial INEI
[11] ISO. ISO 12207-1. Ciclo de Vida del Software
[12] Oracle. Java. Recuperado de http://www.oracle.com/lad/technologies/java/overview/index.html el 12 de Diciembre de 2012
[13] ECK, David J. et al. Introducción a la Programación utilizando Java. Nueva York, 2012
[14] Oracle. NetBeans. Recuperado de http://netbeans.org/index_es.html el 12 de Diciembre de 2012
[15] Oracle. Javadoc. Recuperado de http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html el 12 de Diciembre de 2012
[16] JUnit.org. Junit. Recuperado de http://junit.sourceforge.net/ el 12 de Diciembre de 2012
[17] MENDOZA MORENO, Juan Francisco, "Reflexión humanista de la ingeniería de sistemas" En: Colombia. 2011. Evento: II Encuentro Nacional de Programas de Ingeniería de Sistemas Ponencia: Reflexión humanista de la ingeniería de sistemas Libro II Encuentro Nacional De Programas De Ingeniería De Sistemas, Editorial De La Escuela Colombiana De Ingeniería, p.73 - 74 , v.1, fasc.1
[18] SANTAMARIA GRANADOS, Luz y MENDOZA MORENO, Juan Francisco. "Herramientas en 3D para el modelado de escenarios virtuales basados en Logo - estado del arte". En: Colombia Ciencia E Ingeniería Neogranadina ISSN: 0124-8170 ed: Prueba v.19 fasc.2 p.77 - 94 ,2009