Una Propuesta para Desarrollo Dirigido por Modelos en Entornos Ágiles

Alberto Cortez [1,2,3] , Carlos Martinez [1] , Claudia Naveda [1,2,3] ,
Alejandro Vazquez [1] , Matías Luna [3] , Raúl Varela [1]

.

UTN Facultad Regional Mendoza, Ingeniería en Sistemas de Información,

1 Laboratorio de Auditoria y Seguridad de TIC,
2 Universidad de Mendoza, Instituto de Informática, 3 Universidad del Aconcagua.
{cortezalberto, , claudialaboral, mtsluna }@gmail.com ,{ avazquez,
Raul.varela carlos.martinez]@ frm.utn.edu.ar

Resumen

El Desarrollo de Software Dirigido por Modelos y el Desarrollo de Software Ágil son dos paradigmas de ingeniería de software que contribuyen al veloz desarrollo de aplicaciones.

La integración de ambas perspectivas generan un nuevo enfoque denominado Desarrollo de Software Dirigido por Modelos Ágil (en inglés, Agile Model Driven Development) [1], que permite acelerar el proceso a partir de una perspectiva ágil.

Estudios anteriores han propuesto la integración de MDD **(**en inglés, Model Driven Development)[2**]** *y Agile, sin embargo, estos enfoques no cubren el desarrollo completo del ciclo, por ejemplo, para incluir la integración contínua o son específicos de un dominio de la aplicación. Al abordar este problema proponemos describir un Marco Referencial de procesos integrados, complementado con herramientas que siguen el enfoque de Desarrollo de Software Dirigido por Modelos para que sea aplicado conjuntamente a un proyecto Ágil, aportando un alto valor de rendimiento y calidad a las empresas que lo adopten*

Palabras Clave

Desarrollo de Software Dirigido por Modelos, Métodos Ágiles, Integración Contínua, Calidad de Software. Generación de Código.

1. Introducción

En la actualidad, los métodos ágiles son ampliamente utilizados en la industria del software. La integración contínua forma parte del proceso de desarrollo, la cual se encarga de mantener la línea base del código siempre en un estado desplegable, soportado por la infraestructura.

Uno de los problemas actuales que se presenta, es que los líderes de proyectos invierten un valioso tiempo en crear los modelos iniciales que utilizan al comienzo de un proyecto, pero durante las sucesivas iteraciones no se actualizan reflejando los cambios producidos en el código fuente creado por los desarrolladores. La falta de sincronía entre el modelo y producto de software elaborado, establece inconsistencias que impactan negativamente en el proceso de desarrollo. Generando de este modo problemas futuros para el mantenimiento, calidad y eficiencia.

Los beneficios que proporcionan Agile y MDD pueden llevar a un mejor desarrollo de software afirman Vijayasarathy et al. [3]. En segundo lugar, el MDD se puede considerar como un proceso de desarrollo lento, lo que puede ser una barrera para su adopción en la industria, y agregarle agilidad facilitará su adopción. Los métodos de Desarrollo Dirigido por Modelos surgieron como un nuevo paradigma de desarrollo de software, donde los modelos juegan un papel fundamental. Tienen como objetivo elevar el nivel de abstracción, y aumentar la automatización en la generación de código, mejorando la productividad, la portabilidad, la interoperabilidad y el mantenimiento de los sistemas [4, 5]. De esta manera, MDD permite optimizar las tareas más críticas del proceso de desarrollo, basándose en los modelos como instrumento primario para generar código y otros artefactos [6].

Uno de los problemas recurrentes que presenta MDD es la falta de sincronización entre los modelos y el código fuente. Es decir, cuando el desarrollador genera código fuente en base a un modelo y luego lo modifica, agregando: métodos, atributos, clases, etc., el modelo inicial permanece inmutable sin reflejar los cambios.

Este problema se potencia en los métodos ágiles, cuya naturaleza es adaptativa, y donde los cambios son eventos esperados que generan valor para el cliente [7]. Ellos imponen un enfoque iterativo para desarrollar sistemas de manera incremental, siguiendo los principios del Manifiesto Ágil [8].

Scrum [9] es uno de los métodos ágiles más populares en la actualidad. Caracterizado por plantear iteraciones cortas (Sprints) para realizar entregas de software funcional.

De la integración de las prácticas Ágiles y el desarrollo de software dirigido por modelos, surgió una nueva iniciativa denominada Desarrollo de Software Dirigido por Modelos Ágil que permite acelerar el proceso a partir de una perspectiva ágil. Sin embargo, existen diferencias

fundamentales y conflictos entre los métodos ágiles y MDD que hacen que su integración sea desafiante, especialmente cuando se aplica el enfoque de Entrega Continua (DC por sus siglas en inglés).

La práctica DC promueve la automatización de los cambios en el código antes de alcanzar la fase de producción, y suele estar acompañada de una Integración Continua (CI por sus siglas en inglés). Ésta última permite implementar todos los cambios en el código en un entorno de pruebas o de producción después de la fase de compilación. Cuando la integración continua se implementa de manera adecuada, los desarrolladores disponen siempre de un artefacto para su implementación que ha sido sometido a un proceso de pruebas estandarizado. De esta forma, la integración, verificación y validación continua, dentro de las prácticas ágiles ayudan a la organización a comprobar que se construye el producto con una calidad aceptable pero sin ser sometido a una rigurosa formalización.

El presente artículo propone la definición de un marco referencial de procesos integrados soportado por herramientas automáticas, en el contexto AMDD, para su aplicación en proyectos ágiles Scrum. La propuesta permite solucionar la inconsistencia entre los modelos y el código fuente asociado al enfoque MDD, aplicando ingeniería inversa y estableciendo una evolución dinámica y paralela de los mismos.

El código fuente se genera a partir de modelos de clases UML que aplican patrones de diseño y arquitectónicos, acordes a las buenas prácticas sugeridas dentro de la Ingeniería de Software. De esta manera, el presente trabajo no solo contribuye a aumentar la eficiencia y productividad en las organizaciones, sino también a mejorar la calidad de los procesos y productos de software resultantes.

El documento está organizado de la siguiente forma: después de la introducción, en la Sección 2 se presentan los trabajos relacionados. En la Sección 3, se explica la Metodología de investigación abordada en la presente investigación. En la sección 4 se describe la propuesta y por último en 5 se exponen las conclusiones y trabajos futuros.

2. Trabajos relacionados

La idea básica detrás de la combinación de ambos métodos, MDD y prácticas ágiles, es crear sistemas que puedan responder rápidamente a cambios frecuentes, se proponen diferentes enfoques para resolver los requisitos mencionados; la agilidad se enfoca en aspectos metodológicos que conciernen a un producto, mientras que MDD centra su estudio en un aspecto arquitectónico definido por su variante específica MDA (en inglés, Model Driven Architecture) [10] que separa las características del sistema de su implementación en una plataforma técnica. Advirtiendo la importancia de metodologías ágiles y MDD en el desarrollo de software de sistemas, muchos trabajos se centraron en combinar estas áreas por lo que en esta sección se presentan algunos trabajos previos.

Kulkarni et al. discuten en su trabajo [11], por qué la metodología ágil no se puede utilizar con MDE (en inglés, Model Driven Engineering) [12], entonces proponen una modificación a hacer sobre metodologías ágiles de manera de combinarlas con MDE. A diferencia de este planteo, el presente trabajo describe un nuevo proceso de desarrollo de software que combina Scrum y MDE, los autores propusieron el uso de Meta-Sprints que se ejecuten en paralelo a Sprints para validar los modelos, y sugieren dos o tres meses como escalas de tiempo para meta-sprints, en los cuales los clientes deben proporcionar retroalimentación sobre modelos y prototipos, algo que es opuesto a los principios ágiles; los cuales sugieren una retroalimentación por parte de los clientes en períodos cortos.

Nakicenovic en [13] presenta un proceso AMDD que fue desarrollado considerando prácticas ágiles. Se aplica tanto ingeniería directa como inversa para responder a dos objetivos: la aceleración del proceso de reingeniería de la solución MDD, cómo beneficiarse de la agilidad al producir una solución MDD en un período de tiempo corto. El artículo describe un enfoque que combina MDD y metodologías ágiles basado en su versatilidad. La implementación del enfoque se realizó en el proyecto Market Server Capabilities (MSC) propuesto por la empresa SunGard.

Basso et al. presentan una propuesta que combina los enfoques de MDA y metodologías ágiles en el contexto del Prototipado de Aplicación Rápida (RAP por sus siglas en inglés) [14]. RAP permite la validación de los requerimientos de software, antes de las pruebas de aceptación, con el fin de obtener una respuesta rápida de los clientes. Este enfoque tiene en cuenta los principios de metodologías ágiles en el contexto de MDA y basándose en la metodología RAP para generar modelos y Frontend, fundados en el patrón MVC (en inglés, Model View Controller) , la implementación del enfoque se apoyó en metodología Scrum y MDA para la generación de sistemas de información web. Los autores pretenden asegurar varios beneficios dentro de este enfoque; mejor organización del código fuente, simplicidad en los cambios del código fuente, diseño simple y rápido de modelos, y beneficios de la reutilización de diseño de modelos. Pero este enfoque no puede ser generalizado a todo tipo de sistemas de software, de hecho, se dedicó para desarrollar Sistemas de Información Web. No se detalla en este enfoque cómo integrar MDA y Scrum, por ejemplo, no se propone dónde usar cada nivel de MDA en la metodología Scrum.

Alfraihi en [15] analiza el desafío de combinar métodos ágiles y MDD. Propone un framework para implementar la agilidad como MDD, y estipula recomendaciones, pautas, y procedimientos para poder usar Agile MDD en la práctica. Se observa que incluso aunque en dicho trabajo sugiere algunas prácticas para implementar Agile MDD, no tiene en cuenta la arquitectura de MDD, MDA; ni explica cómo beneficiarse de los diferentes niveles de abstracción para producir sistemas de software sostenibles.

En su propuesta Essebaa et. al [16] presentan el proceso V Life Cycle y lo usan para definir los pasos del desarrollo de sistemas. Proponen una combinación de MDA y Scrum + V Life Cycle. Modelo V significa modelo de Verificación y Validación. Al igual que el modelo de cascada, el ciclo de vida V es un camino secuencial de ejecución de procesos.

Cada fase debe ser completada antes de comenzar la siguiente; basado en el documento de requisitos que contiene especificaciones del sistema, el equipo de desarrolladores comienza el trabajo de diseño y luego la real implementación del código y el equipo de testing comienza con la planificación, escritura de casos de pruebas y pruebas de scripting. Ambas actividades comienzan en paralelo.

La propuesta presentada en este trabajo, se diferencia de 13, 14, 15 y 16, en que el Marco de Referencia aquí planteado describe un flujo de trabajo integrado entre MDD, Scrum, brindando los beneficios de la integración contínua, tomando como soporte herramientas propietarias de ingeniería inversa y automatización de código. Que permitan colaborar con la integración de los flujos de trabajo y metodologías que actualmente las empresas adoptan en su proceso de desarrollo..

3. Metodología de la Investigación

El método de investigación se diseña en base a los elementos planteados en los siguientes trabajos: Wohlin et al. [17], Marcos [18] y Delgado et al. [19]. Para la validación se utiliza una adaptación de un método de investigación llamado investigación en acción técnica [20, 21], la cual parte de un artefacto tecnológico y busca la forma de validarlo.

Se seguirá un proceso contínuo, iterativo e incremental que pasa por varios ciclos de refinamiento y validación. Este proceso se aplica en la sección 4. donde se explica la propuesta. El resultado es el desarrollo del entorno propuesto y la validación de los componentes del mismo. Siguiendo las pautas de la investigación en acción técnica, se definen ciclos compuestos por varias actividades: investigación del problema, diseño del tratamiento, validación del diseño, implementación del tratamiento y evaluación.

La actividad de investigación del problema, se lleva a cabo mediante el análisis del estado del arte planteado por Alfraihi et. al[19]. Esta actividad permite determinar el objetivo de la investigación.

La actividad de diseño de tratamiento es en este caso el Marco de Referencia de Procesos integrando los principios de MDD y las prácticas de Ágil de Scrum. El marco referencial debe ser soportado por herramientas que permitan generar código fuente, utilizar ingeniería inversa para establecer sincronización con los modelos de software, y chequear la consistencia entre los modelos y el código fuente.

Esta propuesta va afectar a los desarrolladores de software que van a definir criterios o métricas. Esto va permitir evaluar y contrastar los objetivos de los desarrolladores contra el producto de las herramientas propuestas. En esta fase se deben obtener los criterios y métricas.

En la actividad de validación del diseño, se evalúa el Marco de Referencia propuesto en contraste con los criterios definidos por los desarrolladores en la fase de investigación del problema. También se debe evaluar si sería efectivo o útil dicho marco si cambia el problema.

La validación se va aplicar en distintas iteraciones. Primero en experimentos de laboratorio controlados donde se utilizan distintos procesos y modelos genéricos. Luego se probará con casos de estudio basados y aplicados en procesos reales de una empresa de desarrollo.

El método de prueba planteado permite obtener una medida del funcionamiento del Marco de Referencia y sirve para determinar si la propuesta está funcionando de acuerdo a los objetivos y criterios planteados en la investigación del problema.

Por último, la actividad: Implementación del Tratamiento y Evaluación, consiste en la transferencia del entorno a los desarrolladores y a su evaluación en cuanto a: efectos, valor y sensibilidad en el contexto del problema. Se compara el producto del marco de referencia propuesto (modelos y código fuente), con los modelos y código propuesto por desarrolladores expertos.

El resultado exitoso de lo anterior permite verificar el funcionamiento del marco de referencia y cumplir con el objetivo planteado.

4. Propuesta

El presente trabajo propone en el contexto de AMDD describir un Marco Referencial de procesos integrados y para que sea aplicado a un proyecto ágil Scrum. Para lograr este objetivo es necesario la creación de herramientas propietarias que se integren a los procesos y aporten una mejora de eficiencia y calidad en los proyectos que se implementen.

El proyecto consta de 2 etapas: La primera etapa, es construir las herramientas necesarias que sirvan de base fundacional para la aplicación del Marco Referencial propuesto.

ColmenaMetamodelo: Establece la sintaxis abstracta de un lenguaje específico de dominio propietario.

ColmenaVisual: Permite modelar diagramas de clases de diversos dominios.

ColmenaGenerador: Traduce modelos a código fuente automáticamente.

ColmenaEnsayo: Genera los test automáticos, unitarios, integración y de aceptación del modelo realizado.

ColmenaReversa: Describe y analiza estructuralmente la composición del código fuente a través de ingeniería inversa.

ColmenaAuditoría: Persiste y analiza evolución de Unidades de Código en adelante UC.

ColmenaWebHooks: Orquesta los cambios que se realizan en el repositorio remoto, para su análisis y clasificación mediante ColmenaReversa y posteriormente persiste mediante ColmenaAuditoría.

En la segunda etapa se definen los procesos y roles que intervienen el el Marco Referencial a la luz del objetivo propuesto:

Determinación de Roles: Determinar las responsabilidades de cada miembro del equipo, de forma tal que cada actividad tenga un actor con el expertise correspondiente.

Descripción de los Procesos: Establecer y detallar un esquema de flujo de trabajo que dan origen al Marco Referencial.

4.1 Primera Etapa

Siguiendo los lineamientos de MDE, se describen las actividades para construir las herramientas, que constituyen los artefactos de inicio necesarios para este proyecto. Las mismas servirán de soporte al Marco Referencial.

4.2 Diseñar y construir un lenguaje específico de dominio - ColmenaMetamodelo

Se diseña un el metamodelo correspondiente que expresa la sintaxis abstracta del DSL(en inglés, domain specific language)[30], para representar un diagrama de clases, relaciones, generalizaciones, implementación interfaces, atributos, métodos, etc. La justificación de crear un metamodelo propio es ampliar significativamente la semántica de UML.
Para crear la sintaxis abstracta del metamodelo se utiliza el framework EMF/Ecore(en inglés, Eclipse Modeling Framework)[31], el cual permite diseñar y representarla en forma gráfica.
El metamodelo de la Figura 1, representa un fragmento de la sintaxis abstracta del DSL. Se muestran las diferentes EClases con los atributos de cada una de ellas. También se describen las diferentes asociaciones o interrelaciones entre las EClases (entidades) con su respectiva multiplicidad o cardinalidad. Este diagrama emplea la notación definida por Ecore. Los modelos que se construyan a partir de la herramienta Colmena Metamodelo, crearán instancias del mismo.

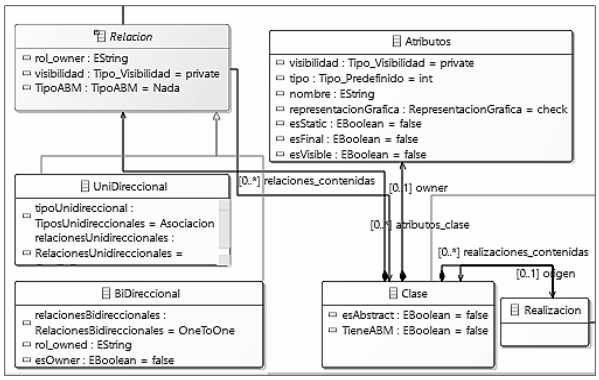

Figura 1. Colmena Metamodel

4.3 Diseñar y construir un Editor Gráfico - ColmenaVisual

Se crea un editor gráfico que exprese la sintaxis concreta del metamodelo anterior; con el fin de escribir las instrucciones y la gramática en el lenguaje de propósito específico definido DSL, se utiliza un plugin de Sirius [24],que se integra con Eclipse. Esta nueva funcionalidad permite la creación de puntos de vistas del metamodelo, contribuyendo con un conjunto de elementos de representación gráfica que facilitan la experiencia de usuario en el modelado. En la figura 2 se corresponde con la vista funcional del editor gráfico.

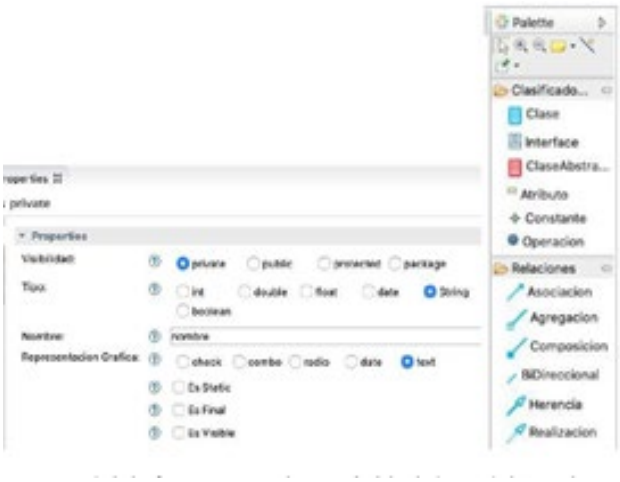

Figura 2. DSL del ColmenaVisual.

4.4 Diseñar y construir las plantillas de transformaciones de modelo a texto - ColmenaGenerador

Para completar el ciclo de vida de un proceso de transformación que obtenga como producto una aplicación ejecutable, es necesario realizar una transformación de modelo a texto. Es decir, a partir de los modelos generados se puede obtener el código fuente final que se ejecutará sobre la plataforma de destino. Los generadores de código tienen que obtener la información de los modelos y generar artefactos (p.e., código fuente) a partir de ellos. En los casos más sencillos se mapea cada símbolo del lenguaje a un determinado fragmento de código; el código generado también podría ser variable dependiendo de los valores de las entradas. La idea es que a partir del modelo se generan artefactos sin necesidad de intervención manual. Se producen artefactos de Frontend (Angular) y Backend (Spring Boot) para este proyecto.
Acceleo [23] es una herramienta de código abierto de la fundación Eclipse que permite generar código fuente a partir de un modelo determinado y de esta manera crear aplicaciones.
A continuación se presenta un fragmento de plantilla que transforma elementos del modelo a código JPA de java.

```
[template public generateMain(raiz : DiagramaClases)]
[comment @main/]
[generateMainClass(raiz)/]
[generateBase(raiz)/]
[generateEntity(raiz)/]
```

```
[generateController(raiz)/]
[generateService(raiz)/]
[generateRepository(raiz) /]
[generateConfiguration(raiz) /]
[generatePom(raiz) /]
[/template]
```

4.5. Diseñar y construir una herramienta que permita generar pruebas - ColmenaEnsayo

Siguiendo la misma metodología empleada para construir ColmenaGenerador, se realizan plantillas correspondientes de transformación de modelos a textos con Acceleo. Este artefacto permite generar clases, para la ejecución de pruebas unitarias, pruebas de integración y pruebas de aceptación.

En la Fig.3se muestra un fragmento de código de dichas transformaciones

4.6. Diseño y creación de una herramienta que analice código de Ingeniería Inversa - Colmena Reversa

La Ingeniería Inversa es el proceso de analizar artefactos de software existentes con el objetivo de extraer información y proveer vistas de mayor nivel de abstracción que faciliten la comprensión, el análisis e independencia del lenguaje de implementación, del sistema subyacente. ANTLR, es un generador automático de analizadores sintácticos LL(*) de código libre, desarrollado por Terrance Paar de la universidad San Francisco. A partir de la gramática formal expresada en una notación similar a BNF, ANTLR genera un Analizador para el lenguaje capaz de construir automáticamente Árboles de sintaxis concretos, así como el código para recorrerlos.

ColmenaReversa es una herramienta basada en ANTLR, con capacidad de análisis sintáctico. Su función principal es la clasificación de cada unidad de código (Clase e Interfaz) en adelante "UC". La clasificación permite describir el estado de cada UC en términos de Nombre, paquete, atributos, métodos, herencia, relaciones y cardinalidad con otras UC.

La aplicación de esta herramienta sobre un Dominio, permite obtener una instantánea del estado de cada una de ellas, en un momento dado.

4.7 Diseñar y construir herramienta de orquestación - ColmenaWebHook

Este subsistema es encargado de interconectar y ordenar mensajería entre la plataforma Github mediante su api, con ColmenaReversa y ColmenaAuditoría.

Cada vez que un miembro del equipo de desarrollo, ejecuta el comando push de Git, ColmenaWebhook, será notificada. Esta notificación implica la realización de un escaneo de aquellas **UC** involucradas en la operación, enviando las mismas hacia colmenaReversa para obtener el estado de las mismas. Posteriormente el conjunto de estados resultado serán asociados a la información del evento que originó el análisis (comando push) ,para su posterior persistencia en ColmenaAuditoría. La asociación entre los estados de un conjunto de UC y un evento en adelante "EE," es la unidad de información transferida hacia Colmena Auditoría.

En la figura 3 se ejemplifica el funcionamiento, representando un equipo de trabajo. En el flujo (1) cada integrante ejecuta git push de sus ramas individuales, posteriormente durante el flujo (2) envía una notificación con información hacia ColmenaWebHook, a continuación en el flujo (3) se clasifica la estructura de cada UC mediante ColmenaReversa, el resultado obtenido conjuntamente con la información de GitHub es persistida en ColmenaAuditoría en el flujo (4).

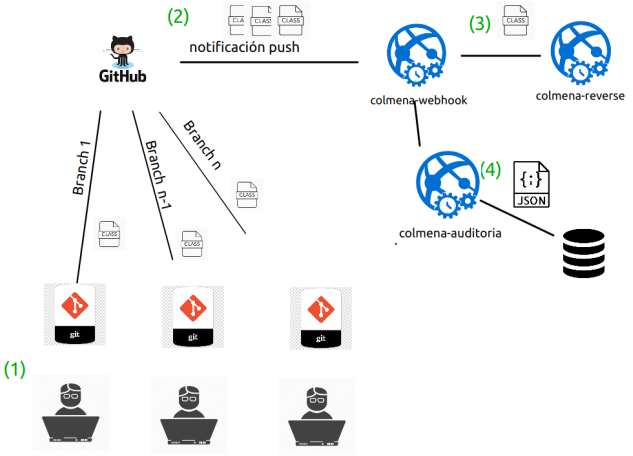

Figura 3. Iterconexión entre flujos.

4.8. Diseño y creación de una herramienta que permita auditar y comparar código ColmenaAuditoria

El sistema ColmenaAuditoría, brinda soporte completo de persistencia y análisis de cambios en el estado de una UC, asociado a un evento. Un evento es un suceso determinado en el tiempo, como por ejemplo una operación "git push" en una rama de trabajo. La asociación entre estados de UC y eventos, permite analizar la evolución de varias unidades de código UC, a través de sucesivas modificaciones realizadas mediante ramas de trabajo en los diferentes entornos de desarrollo, para cada proyecto involucrado.

Cada estado-evento (EE), representa la transformación, que introduce una mejora expresada en funcionalidad, seguridad y/o calidad. La traza completa de la evolución del proyecto es el conjunto de todos los EE.

El conjunto de los EE de una misma rama para un intervalo de tiempo, en adelante "Unidad de Evaluación" UE, representa la evolución total de la rama para el intervalo considerado.
La principal función de ColmenaAuditoría, es brindar soporte al análisis de una UE, permitiendo la valoración de la evolución de cada UC (clase o interfaz) contenida en dicha unidad de evaluación.
El Líder de Proyecto encargado de aprobar o rechazar el código que integra a cada rama, mediante la UE, puede valorar la composición estructural de cada UC.

4.9 Segunda Etapa

4.9.1 -Descripción del Marco Referencial de Procesos:

4.9.1.1- Determinación de Roles

Las actividades de nuestro proceso están asignadas a un número limitado de roles, y un desarrollador puede desempeñar múltiples roles. Se identifican los siguientes roles y responsabilidades;

Dueño del Producto: Asegura que el equipo aporte valor al negocio.

Maestro de Scrum:,Ayudar al Product Owner (a comunicarse con el equipo de trabajo, a gestionar el backlog).

- Ayudar al Equipo de Trabajo (a mantener el foco, a ser autosuficiente, a aprovechar los artefactos de **Scrum**)
- Ayudar a la Organización (a entender y adoptar **Scrum**, a planificar implementaciones dentro de la organización

Modelador:Se encarga de modelar el sistema.;

Líder de Equipo: Es el referente central tecnológico del equipo

Desarrollador: Programador en tecnologías de FrontEnd y Backend.

Asegurador de Calidad: Se encarga de realizar todas las pruebas.

4.9.1.2- Descripción de los Procesos

El marco propuesto comienza con la fase de selección de requisitos prioritarios y termina con la fase de auditoría. Las fases son las siguientes: (i) selección de requisitos prioritarios, (ii) prototipado y asignación , (iii) desarrollo, (iv) integración y auditoría. Durante el desarrollo, sigue un enfoque iterativo e incremental.

El presente marco no está ligado a un formalismo particular, sin embargo, proponemos adoptar estándares, para estar abierto a múltiples herramientas y promover la interoperabilidad de los modelos. Una descripción general del proceso y sus actividades se presentan en la Fig.4 y Fig, 5, mientras que las diferentes fases y sus etapas se explican en las siguientes subsecciones.

i)Fase Requisitos prioritarios:

Se crean todas las historias de usuarios y se priorizan para comenzar el Sprint. Se utiliza GitHub como herramienta de soporte para crear las tareas y hacer el seguimiento de las mismas.
Participantes: Dueño del Producto y Maestro de Scrum.

ii)Fase Prototipado y versionado:

a) Diseño:
Se elabora el modelado del Sprint correspondiente en diagramas de clases utilizando ColmenaVisual.

b) Transformación:
A través de ColmenaGenerador y ColmenaEnsayo se aplican las transformaciones automáticas a código para el Frontend y Backend, según las tecnologías seleccionadas.

c) Versionado:
Se almacena en un repositorio el modelo diseñado (clases) en formato de intercambio de modelos XMI (en inglés, XML Metadata Interchange) [29] como así también el código fuente generado..
Se realiza el primer commit de versionado del código fuente de Frontend y Backend a la rama de desarrollo de GitHub, como punto de partida para que cada desarrollador descargue localmente su historia de usuario, código inicial y comience la tarea asignada.
Este primer push hacia la rama de desarrollo desencadena la ejecución de ColmenaWebHook.

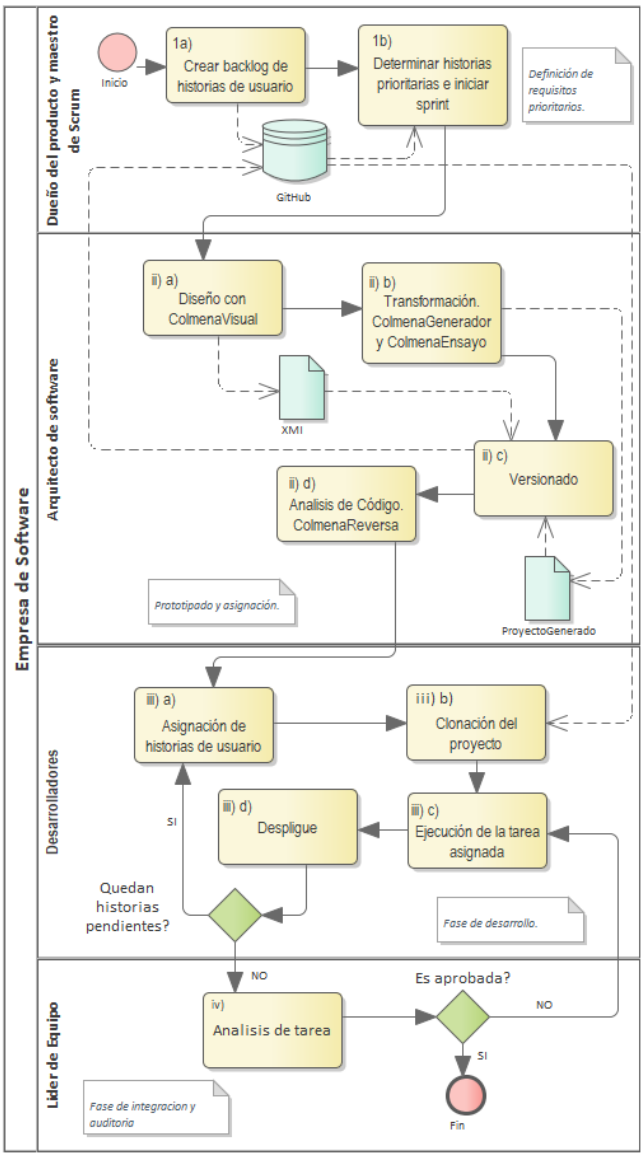

Figura 4. Marco referencial de proceso. Parte1

d) Análisis de código:

Se aplica la herramienta ColmenaReversa sobre el artefacto Backend prototipado, a fin de realizar el análisis estructural del código. Durante esta actividad se inspecciona el código fuente, analizando sintácticamente para obtener el estado de cada Unidad de Código (UC).
Posteriormente en cada git push se repetirá dicho proceso de clasificación, a fin de informar las evolución de las Unidades de Evaluación (UE), producidas en el Modelo inicial.
Participantes: Líder de Equipo. - Modelador

iii) Fase Desarrollo:
Los desarrolladores deberán iniciar el proceso inicial de codificación de la tarea asignada en su ambiente local y al terminarla realizará el despliegue hacia el servidor remoto.

a) Asignación de historias de Usuario:

Se asignan las historias de usuario correspondiente a fin de realizarse.

b) Clonación:
Cada desarrollador deberá clonar el prototipo inicial asignado, en su repositorio local.

c) Ejecución de Tarea asignada:
Cada desarrollador comienza la ejecución de la tarea asignada.

d) Despleguie a GitHub:
Cada desarrollador que ha finalizado su tarea realiza un commit y posteriormente realiza un push a su rama remota desencadenando la ejecución de ColmenaWebHook.
Participantes: Desarrollador

iv) Fase de Integración y Auditoría
En esta etapa el líder de equipo mediante colmenaAuditoria realiza la valoración de las Unidades de evaluación generadas, para una rama determinada, la aprobación del conjunto de UEs, dispara la integración del código hacia la rama correspondiente, caso contrario el desarrollador itera nuevamente la ejecución de su tarea asignada.

Participantes: Lider de Equipo.

****Despliegue y Análisis de código:**
Cada vez que un miembro del equipo de desarrollo, ejecuta el comando push, se ejecuta ColmenaEnsayo para validar el código producido con el total de pruebas creado. Por otro lado ColmenaWebhook, será notificada. ello implica la realización de un escaneo de aquellas **UC** involucradas en la operación, enviando las mismas hacia ColmenaReversa para producir el análisis y obtener el estado de las mismas.

Revisión y Auditoría
Posteriormente el conjunto de estados resultado serán asociados a la información del evento que originó el análisis (comando push) para su posterior persistencia en ColmenaAuditoría. En esta actividad el Líder de Proyecto será notificado si el código generado por el desarrollador hubiere modificado estructuralmente el prototipo inicial asignado, en caso de que suceda esta condición, procederá a notificar al desarrollador correspondiente la inconsistencia y generará otra historia de usuario para su corrección. De lo contrario significa que se puede integrar el producto generado de software a la rama de desarrollo.

Se debe iterar sobre las **fases de desarrollo e integración y auditoría hasta** que finalice el Sprint y de esta forma producir el entregable con la aceptación del cliente.
Participante: Team Leader.

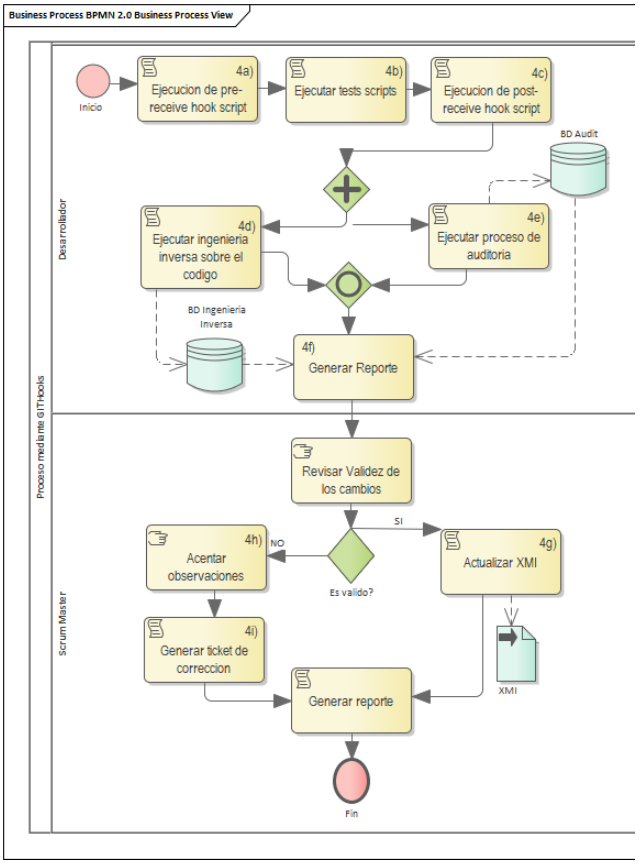

Figura 5. Marco referencial de proceso. Parte 2

5. Conclusiones y Trabajos Futuros

En este trabajo se propone un Marco Referencial de procesos integrando AMMD y Agil, Permite disminuir el desfase o falta de sincronía entre los modelos UML y el código fuente asociado. Para solucionar esto, se contribuye con el desarrollo de los artefactos: ColmenaVisual, ColmenaGenerador, ColmenaEnsayo, ColmenaReversa. ColmenaWebHooks y ColmenaAuditoría. Estos permiten lograr un desarrollo de software más productivo y de calidad integrando la metodología ágil Scrum, y MDD. Se proporcionan herramientas de ingeniería inversa que permiten inspeccionar distintos artefactos, para reconstruir el modelo subyacente al código. Se desarrolla un plugin integrado a Eclipse para el modelado gráfico y generación de código automático en distintos lenguajes. Se controla y se hace el seguimiento de los modelos en las distintas etapas del desarrollo de software: desarrollo, test y desarrollo. Se mantiene la trazabilidad de las modificaciones ocurridas en el código a lo largo de todo el proceso. Se asegura la calidad del producto final, al seguir buenas prácticas de diseño y programación. código, por la inserción de operaciones y propiedades que cada programador aporta.

En futuros trabajos nos centraremos en la evaluación del proceso de forma más completa. Con este fin, cuatro estudios de casos, desarrollando el mismo sistema, se llevará a cabo por cuatro diferentes equipos que aplican diferentes enfoques: "enfoque tradicional" (es decir un enfoque no ágil codificado a mano); enfoque "sólo MDD"; "Ágil sólo"; y el enfoque "MDD ágil". Comparación de los resultados de estos estudios de casos debería proporcionar una clara comprensión del impacto de la integración de Agile y MDD en el proceso de desarrollo.

Referencias

[1] S. Ambler, “Agile Model Driven Development (AMDD): The key to scaling agile software development,” (2020, Oct, 5)[Online]Available: http://agilemodeling.com/essays/amdd.htm

[2] OMG, “The pragmatics of model-driven development”, IEEE Softw., vol. 20, no. 5, pp. 19 25, 2003.

[3] L. Vijayasarathy and D. Turk, “Agile software development: A survey of early adopters”. Journal of Information Technology Management, 19(2):1–8, June 2008.

[4] D. Frankel. Model Driven Architecture: Applying MDA to Enterprise Computing, volume 25. Hoboken, New Jersey John Wiley & Sons, 2003.

[5] A.Kleppe, J. Warmer, and W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise. The Addison-Wesley object technology series. Primera Edición. Reading, Massachusetts Addison-Wesley, 2003.

[6] Patterns: Model-Driven Development Using IBM Rational Software Architect, Peter Swithinbank, Mandy Chessell, Tracy Gardner, Catherine Griffi n, Jessica Man, Helen Wylie, Larry Yusuf, disponible en ibm.com/redbooks.

[7] Patel, Ahmed & Seyfi, Ali & Taghavi, Mona & Wills, Christopher & Liu, Na & Latih, Rodziah & Misra, Sanjay. (2012). A Comparative Study of Agile, Component-Based, Aspect-Oriented and Mashup Software Development Methods.

[8] K. Beck and M. Fowler, Planning Extreme Programming. The XP series. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA 2000.

[9] K. Schwaber, and M. Beedle. Agile Software Development with Scrum. Agile Software Development. Prentice Hall, 2002.

[8] A. Cockburn, Agile Software Development. Agile Software Development. Addison-Wesley, 2002.

[9] K. Beck .Manifesto for agile software development, 2001. [Online]. Available: https://agilemanifesto.org/

[10] M. Siegel, MDA (2014, June,6). [Online].Available: https://www.omg.org/mda/

.[11] V. Kulkarni, S. Barat, and U. Ramteerthkar, “Early experience with agile methodology in a model-driven approach,” in Model Driven Engineering Languages and Systems, 14th International Conference, MODELS 2011,

Wellington, New Zealand, October 16-21, 2011. Proceedings, 2011, pp.578–590.

[12] Boronat A. (2019).Code-First Model-Driven Engineering: On the Agile Adoption of MDE Tooling. 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 2019, pp.

[13] M. Nakićenović, “An agile driven architecture modernization to a model-driven development solution,” International Journal on Advances in Software, vol. 5, no. 3,4, 2012.

[14] F. Basso, R. Pillat, F. Roos-Frantz, and R. Z. Frantz, “Combining mde and scrum on the rapid prototyping of web information systems,”, Int. J. Web Engineering and Technology, vol. 10, no. 3, October 2015.

[15] H. Alfraihi, “Towards improving agility in model-driven development,” in Joint Proceedings of the Doctoral Symposium and Projects Showcase Held as Part of STAF 2016 co-located with Software Technologies: Applications and Foundations (STAF 2016), Vienna, Austria, July 4-7, 2016., 2016, pp. 2–10.

[16] I. Essebaa and S. Chantit. “Model Driven Architecture and Agile Methodologies”, Proceedings of the Federated Conference on Computer Science and Information Systems pp. 939–948, ISSN 2300-5963 ACSIS, Vol. 15, 2018.

[17] Wohlin C., Runeson P., Höst M., Ohlsson M.C. , Regnell B. and Wesslén A., (2012).

"Experimentation in Software Engineering", Springer, ISBN 978-3-642-29043-5.

[18] Marcos, E., “Investigación en Ingeniería del Software vs. Desarrollo”, Grupo KYBELE. Universidad Rey Juan Carlos.(2020,Oct,5)[Online]Available: http://gidis.inf.pucp.edu.pe/recursos/InvIngSWvsDS.pdf

[19] Delgado A, Ruiz. F, Garcia I, Piattini M, (2012). Un experimento para validar transformaciones QVT para la generación de modelos de servicios en SoaML desde modelos de procesos de negocio en BPMN2. Sistides.XVII Jornadas de Ingeniería del Software y Bases de Datos.

[20] Wieringaand R, Morali A. (2012). “Technical Action Research as a Validation Method in Information Systems Design Science”. InK. Peffersand M.RothenbergerandB.Kuechler(eds.)Seventh International Conference on Design Science Research in Information Systems and Technology (DESRIST). LNCS7286.Pages220-238.Springer.

[21] M.,Cruz-Lemus J., Piattini Velthuis M. (2014). “Métodos de investigación en ingeniería del software”. Editorial Ra-Ma, 2014. ISBN 978-84-9964-507-0 .

[22] About the object constraint language specification version 2.4 [Internet]. Milford, Massachusetts:OMG (2020, Oct,5)[Online].Available:https://www.omg.org/spec/OCL/About-OCL/

[23] Generate anything from any emf model. Ottawa, Ontario:Eclipse Foundation (2020, Oct, 5)[Online]Available: https://www.eclipse.org/acceleo/

[24] The easiest way to get your own Modeling Tool [Internet]. Ottawa, Ontario:Eclipse Foundation [citado 09 Ago 2020] Disponible en: https://www.eclipse.org/sirius/

[25]Writing Queries and Interpreted Expressions. [Internet]. Ottawa, Ontario:Eclipse Foundation.[citado 09 Ago 2020] Disponible en: https://www.eclipse.org/sirius/doc/specifier/general/Writing_Queries.html

[6] Eclipse for RCP and RAP Developers.[Internet]. Ottawa, Ontario:Eclipse Foundation. (2020,Oct,5) [Online] Available:https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-rcp-and-rap-developers

[27] What is ANTLR?[Internet].Ottawa,Ontario:Eclipse Foundation. (2020,Oct,5) [Online] Available: https://www.antlr.org/

[28]Customizing Git - Git Hooks.[Internet]. Brooklyn: New York:Software Freedom Conservancy. (2020, Oct,5) [Online] Available:https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks

[29]About the xml metadata interchange specification version 2.5.1.[Internet] Milford, Massachusetts:OMG. (2020, Oct,5) [Online]Available:https://www.omg.org/spec/XMI/About-XMI/

[30] Brambilla M. , Cabot J. , Wimmer M. (2017) “Model-Driven Software Engineering in Practice”. Second Edition. Morgan & Claypool Publisher.University of Illinois at Chicago.

[31] Eclipse Modeling Framework (EMF).[Internet]. Ottawa, Ontario:Eclipse Foundation (2020, Oct, 5)[Online]Available: https://www.eclipse.org/modeling/emf/

[32]H. Alfraihi and K. Lano. The integration of agile development and model driven development: A systematic literature review. In Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development, MODELSWARD, 2017.