LUIS FREDDY MUÑOZ SANABRIA

UN MARCO DE PROCESO DE SOFTWARE DISCIPLINADO BASADO EN LA ARQUITECTURA y EN XP PARA VSE (Very small Enterprise) "XP/ARCHITECTURE"

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES
MAESTRIA EN CIENCIAS EN COMPUTACION

Popayán
2013

LUIS FREDDY MUÑOZ SANABRIA

UN MARCO DE PROCESO DE SOFTWARE DISCIPLINADO BASADO EN LA ARQUITECTURA y EN XP PARA VSE (Very small Enterprise) "XP/ARCHITECTURE"

Tesis presentada a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca
Para optar el Título de

Magister en Ciencias de la Computación

Director:
Doctor: JULIO ARIEL HURTADO

Popayán,

2013

A Dios

Y a mi Familia

AGRADECIMIENTOS

A Dios por su ayuda permanente y permitirme caminar hacia el propósito que tiene trazado para mi.

A mi Señora esposa y mis hijos por soportar tantos momentos de ausencia familiar.

A mi madre y mis hermanos a quienes merezco mucho respeto y admiración.

A mi Director Por su incondicional acompañamiento, valiosa orientación y enseñanzas para mi vida profesional.

A los miembros del grupo IDIS, porque con su apoyo, ejemplo y disciplina me motivaron a terminar este proyecto.

A la Fundación Universitaria de Popayán en cabeza del señor Rector Padre Mario Alfredo Polo Castellanos quien ha confiado en mí y me han apoyado permanentemente en todos mis proyectos.

RESUMEN

Con el auge de las metodologías agiles se da respuesta a los desarrollos rápidos en ambientes de bajos recursos y de gran incertidumbre, incluyendo practicas básicas de calidad. Los mayores problemas de calidad provienen de los requisitos, así que las metodologías agiles abordan de manera directa este problema con ciclos de desarrollo cortos, orientados al valor y con la participación del cliente. Uno de los problemas identificados en la comunidad de los métodos ágiles, es la dificultad para escalarlos cuando el proyecto es mediano o grande. En particular se refiere a la complejidad del producto y al tamaño del equipo. La mayoría de los proyectos de software que reporta la literatura son proyectos de pocas personas y en cuyos desarrollos los atributos de calidad no son reportados como relevantes. La arquitectura es concepto clave en un proyecto de software mediano que intente usar metodologías ágiles. La ausencia de una orientación hacia la arquitectura dentro de las metodologías ágiles, no permite que se destaquen decisiones tempranas de diseño que tendrán un profundo impacto en todo el trabajo de ingeniería del software.

Dado que la escala esta relacionada a la complejidad y al tamaño del equipo, deben incluirse elementos técnicos y de gestión que permitan descomponer de manera metódica el proyecto en unidades mas simples, gestionables y construibles con las practicas del proceso ágil. Este proyecto presenta Extreme Programming with Architecture - XA como un modelo de proceso de software orientado a la arquitectura que soportado bajo las reglas y principios de la metodología Extreme Programming - XP, busca escalar XP a proyectos de mediana complejidad y equipos mas grandes que 10±2 personas.

Palabras Claves: Ingeniería del software, proceso software métodos ágiles

ABSTRACT

With the rise of agile methodologies is in response to the rapid developments in low-income environments, high uncertainties, including basic quality practices. The biggest problems come from quality requirements, so agile methodologies directly address this problem with short development cycles, value-oriented and customer engagement. One of the problems identified in the community of agile methods is the difficulty to scale them when the project is medium or large. In particular it relates to product complexity and size of equipment. Most software projects reported in the literature are few projects and developments whose quality attributes are reported as not relevant. The architecture is a key concept in a medium software project seeking to use agile methodologies. The absence of an orientation towards architecture in agile methodologies, highlighting not allow early design decisions that will have a profound impact on all software engineering work.

Since the scale is related to the complexity and size of the team should be included technical and management elements that allow methodically break down the project into simpler units, manageable and constructible process with agile practices. This project presents Extreme Programming with Architecture - XA as a process model oriented software architecture that supported under the rules and principles of the methodology Extreme Programming - XP, XP scalar looking medium complexity projects and teams larger than 10 ± 2 people.

Keywords: software engineering, software process agile methods

Tabla de contenido

Listado de figuras

Listado de tablas

Listado de ilustraciones

CAPITULO I: Introducción

En este capítulo, se define el problema que inspiró el proyecto de maestría, contiene la pregunta de investigación, la hipótesis que se desea demostrar y los objetivos y aportes planteados; también relaciona de manera resumida la metodología seguida para el logro de los propósitos del proyecto, al mismo tiempo define la estructura de documentos generados tanto en la monografía como en los anexos, finalmente se presentan aclaraciones sobre los logros alcanzados en relación con los objetivos propuestos.

1.1 Definición del problema

En Colombia, la industria de software está creciendo vertiginosamente. A la fecha actual, la industria de software colombiana aporta al producto interno bruto el 3.5%; la venta de software ha crecido un 275%, las microempresas un 68% (de 1 a 10 empleados) y las pequeñas empresas un 37% (de 11 a 50 empleados). (Fuente: FEDESOFT “Fortalecimiento y Desarrollo Integral de la ISTIR en Colombia”). En las últimas décadas, la investigación en ingeniería de software ha brindado buenas prácticas a la tradicional industria de software mundial. Sin embargo, de acuerdo a estos datos la industria de software colombiana (y en general en el mundo) se compone de VSE (Very Small Enterprises), para las cuales la mayoría de estas prácticas no son aplicables directamente.

Dadas estas condiciones, es pertinente investigar sobre mecanismos que hagan ver interna y externamente a nuestra industria de software nacional como una industria

competitiva. La competitividad es un balance delicado entre calidad y productividad. La calidad requiere de definir estrategias para el control, el aseguramiento y la gestión de la calidad, lo cual significa esfuerzo y costo. La productividad requiere de equipos preparados y procesos adecuados.

1.1.1 Contexto General del Problema

Con el auge de las metodologías ágiles se ha venido dando una respuesta a los desarrollos rápidos en ambientes de bajos recursos y de gran incertidumbre [3], incluyendo prácticas básicas de calidad. Los mayores problemas de calidad provienen de los requisitos [1][11], así que las metodologías ágiles abordan de manera directa este problema con ciclos de desarrollo cortos, orientados al valor y con la participación del cliente [5]. Sin embargo desaparecen algunos aspectos considerados importantes cómo la documentación, la cual es necesaria para evolucionar y mantener los productos en el mercado. Incluso en el desarrollo, la documentación se hace necesaria como un mecanismo de comunicación y organización del producto [2], en particular cuando su complejidad no es tan baja.

Las metodologías ágiles han presentado problemas de escalabilidad en el tamaño del equipo y la complejidad de las soluciones[4]. Dado que la documentación arquitectónica facilita de manera temprana organizar la solución para que satisfaga los requisitos más relevantes y para organizar equipos de desarrollo, la arquitectura se abre como concepto complementario nivel técnico y de gestión en un proyecto de software mediano que intente usar metodologías ágiles [19]. La ausencia de una orientación hacia la arquitectura dentro de las metodologías ágiles, no permite que se destaquen decisiones tempranas de diseño que tendrán un profundo impacto en todo el trabajo de ingeniería del software. Tampoco se construye un modelo que aunque relativamente pequeño e intelectualmente comprensible permita verificar cómo está

estructurado el sistema y de cómo trabajan juntos sus componentes. La metáfora del sistema [11] es una aproximación inicial a la arquitectura que sirve para soluciones simples y ésta puede ser usada como punto de partida a una descripción arquitectónica.

La priorización de los requisitos, es una práctica clave en los métodos ágiles, sin embargo está orientada a generar valor más rápidamente, pero es un valor orientado al cliente, que no necesariamente beneficia las empresas durante el desarrollo y mantención de software. La idea de una industria de software competitiva es que produzca soluciones, al tiempo que fortalece su producción, es decir, que pueda reutilizar componentes, estructuras y decisiones de diseño, así como obtener productos que después puedan ser mantenidos adecuadamente [3]. Así, la ausencia de diseño bajo esquemas ágiles, puede generar productos poco competitivos, algo rígidos para un mercado que demanda muchos cambios en las aplicaciones en la medida en que sus negocios lo requieren.

1.1.2 Formulación del Problema

Uno de los problemas identificados en la comunidad de los métodos ágiles, es la dificultad para escalarlos cuando el proyecto es mediano o grande. En particular la comunidad se refiere a la complejidad del producto y al tamaño del equipo. La mayoría de los proyectos de software que reportan la comunidad ágil son proyectos de pocas personas y en cuyos desarrollos los atributos de calidad no son reportados como relevantes. En este caso sucede que la gran mayoría de decisiones arquitectónicas son de baja complejidad puesto que normalmente se construyen sobre una plataforma que ya tiene la mayoría de decisiones arquitectónicas tomadas.

Son proyectos de alta complejidad aquellos donde se requiere definir una solución

con atributos de calidad muy exigentes. Son ejemplos, un sistema de control de tráfico aéreo (alta confiabilidad), un lenguaje específico de dominio para la simulación de sistemas de evento discreto (alta modificabilidad), un sistema transaccional para un banco internacional (alta disponibilidad y seguridad). Un sistema de mediana complejidad se encuentra en un punto medio entre estos extremos, es decir, tiene cierto grado de exigencia en los atributos de calidad, así como en las tecnologías disponibles para su desarrollo. Por ejemplo un sistema que permita monitorear el tráfico de camiones de una empresa de distribución de encomiendas (confiabilidad media) o un framework de simulación, que no es altamente modificable, pero exhibe buena capacidad para la simulación de varios modelos basados en el enfoque de simulación definido en el framework.

Por otro lado, dado el tamaño del equipo, las metodologías sugieren prácticas más colaborativas. Sin embargo cuando el tamaño del equipo crece se hace más difícil gestionar y coordinar el equipo [29]. Se entiende en este proyecto un equipo mediano como aquel que se mueve alrededor de 16±4 personas.

Así hemos llegado a la formulación de la siguiente pregunta de investigación:

¿Cómo escalar XP en proyectos de mediana complejidad y equipos de desarrollo medianos?

Escalar un método significa lograr que el proceso brinde los beneficios para un contexto más grande para el que fue propuesto. Dado que la escala está relacionada a los atributos de calidad y al tamaño del equipo, deben incluirse elementos técnicos y de gestión que permitan descomponer de manera metódica el proyecto mediano en unidades más simples gestionables y construibles con las prácticas del proceso ágil.

1.2 Los aportes del proyecto

Esta investigación aporta los criterios necesarios para que a partir de los principios y valores del manifiesto ágil (XP como referencia) se facilite la escalabilidad a proyectos medianamente complejos con equipos medianamente numerosos. Esto se expresa de manera concreta en un modelo de proceso, una extensión de Extreme Programming con un enfoque holístico y con métodos de arquitectura tales como QAW y ADD, cuyos resultados no solamente generan valor al cliente (característica del manifiesto ágil), sino que también, le generan valor a la organización de desarrollo. Además el propósito es que se pueda convertir en un modelo que ayude en un futuro a fortalecer la industria del software regional. Específicamente, los aportes de esta propuesta son:

- A la comunidad académica le brinda un nuevo proceso, ***XP/Architecture o XA,*** para el desarrollo de proyectos aplicables al aula de clase teniendo en cuenta enfoques de ingeniería de software relevante y prácticas básicas que han funcionado en la industria y que fortalecerán la formación en el pregrado.

- A la comunidad investigativa se entrega un caso de integración de prácticas disciplinadas con prácticas ágiles, así como experiencias del desarrollo centrado a la arquitectura en un contexto ágil.

A la comunidad industrial se le aportará herramientas de trabajo orientado a la generación de valor con un significado más amplio, entregar valor al cliente y a la empresa desarrolladora a través de XA.

1.3 Objetivos

1.3.1 General

- Diseñar un marco de procesos de software basado en XP y en los métodos de arquitectura QAW [15] y ADD [16] para el desarrollo de software que involucre proyectos medianos en complejidad y en equipos de desarrollo de tamaño mediano.

1.3.2 Específicos

- Proponer un proceso de software basado en la arquitectura y en requisitos de calidad denominado ***XP/Architecture o XA***, que apoyado en métodos de arquitectura facilite la comprensión común de las soluciones (manejo de la complejidad) y la comunicación entre los miembros del equipo (manejo del tamaño del equipo).

- Especificar formalmente el proceso de software ***XP/Architecture*** con SPEM2.0[1] usando EPF[2], donde se muestre una extensión de Extreme Programming (XP) en proyectos de mayor complejidad.

- Validar ***XP/Architecture*** a través de 2 casos de estudio, 2 proyectos de desarrollo de software con requisitos de calidad medianamente complejos y

[1] Software Process Engineering and Systems Metamodel. Es un proceso standard definido por la OMG para el modelado de procesos de Software. Disponible en http://www.omg.org/spec/SPEM/2.0/
[2] Eclipse Process Framework. Es una herramienta para la especifiación de procesos usando SPEM2.0 como base conceptual. Disponible en: http://www.eclipse.org/epf/

equipos de más de 8 personas en el contexto de cursos de ingeniería de software o relacionados con el aprendizaje de una metodología de desarrollo.

1.4 Método de investigación

El proyecto diseña un marco de procesos de software, basado en XP y en los métodos de arquitectura QAW [15], ADD [16] para el desarrollo de software que involucre proyectos medianos en complejidad y en equipos de desarrollo medianamente complejos. Para lograrlo, se usa como marco el *método general de investigación científica*, específicamente el método científico De Mario Bunge [2727]. Dada la particularidad de la investigación en el área de ingeniería de software, se pusieron en práctica las bases metodológicas de investigación científica en ingeniería de software recopiladas por Mary Shaw [28] desde datos empíricos a partir de conferencias y revistas relevantes en el área. De acuerdo a la clasificación de Shaw, la pregunta de investigación de este proyecto es del tipo "¿cómo crear un artefacto X?, siendo nuestro artefacto el software.

El resultado obtenido de acuerdo a la misma clasificación de Shaw es "Un procedimiento o técnica para hacerlo mejor", el cual corresponde a **XP/Architecture o XA**. La validación fue de tipo "Experiencia" dado que se validó en dos estudios de caso en el ámbito académico. Este tipo de validación requiere diseñar y ejecutar y estudiar casos prácticos.

El método científico seguido se basa en el modelo MCIS – Método Científico en Ingeniería de Software propuesto por Hurtado [30], el cual se ha aplicado de manera empírica en algunos proyectos de investigación dentro del grupo IDIS basado en los hallazgos de Shaw. El método, de acuerdo a la Figura 1 define tres fases principales(Exploración, Formulación y Validación), las cuales se recorren a través de

iteraciones de investigación que incluyen un conjunto de actividades básicas.

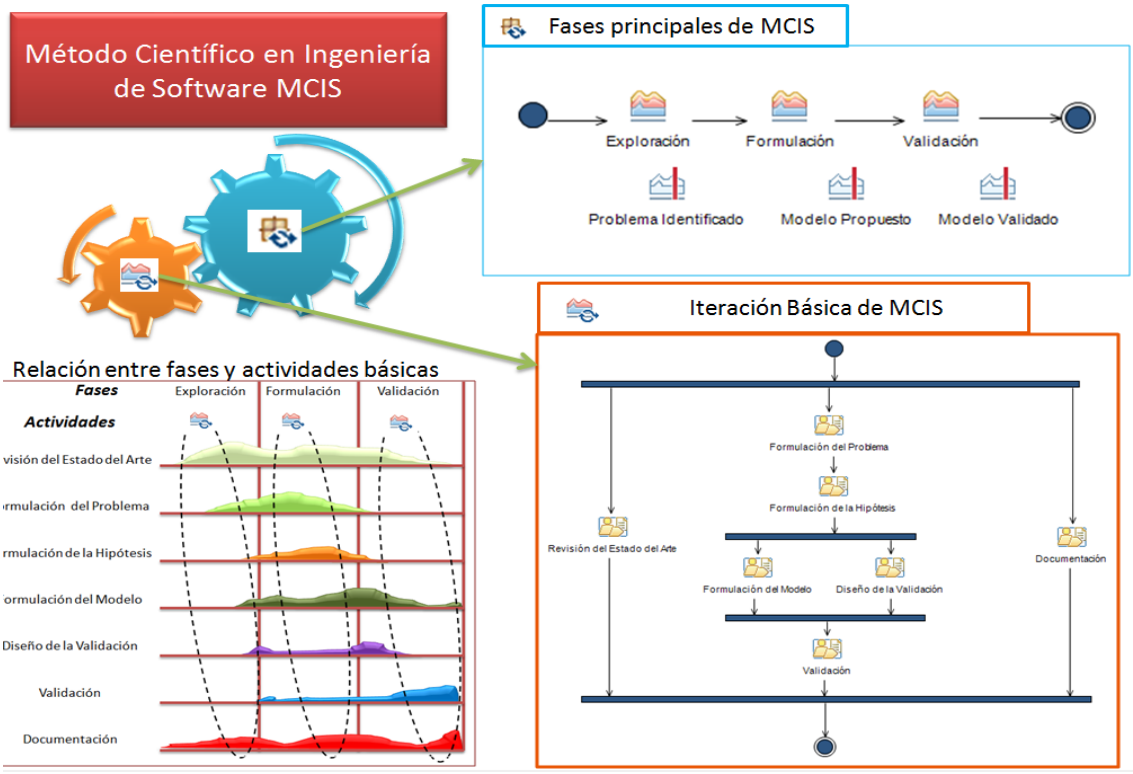

Figura 1.1 Métodos Científico en Ingeniería de Software - MCIS

Dependiendo de la fase en que se vaya, se hace énfasis en las actividades respectivas. Así, el ciclo de investigación no es lineal y permite incrementalmente ir avanzando en la investigación, lo cual sucede en la práctica. Los detalles de las fases, iteración y relación entre éstas en el MCIS se pueden visualizar en la Figura 1. A continuación se detallan las actividades básicas planificadas haciendo una adaptación particular de MCIS a éste proyecto.

1.4.1. Actividades

A continuación se describen las principales actividades de este modelo con las tareas respectivas adaptadas al contexto de este proyecto:

Actividad 1: Planteamiento del problema

- **Reconocimiento de los hechos:** se revisaron experiencias y problemas encontrados sobre métodos ágiles en grupos grandes y sistemas complejos. Este reconocimiento se hizo a través de la recopilación de estudios científicos e industriales del área.
- **Descubrimiento del problema:** consistió en la revisión del problema de la escalabilidad de XP en la literatura científica. Particularmente las principales conferencias han identificado la escalabilidad en general de los métodos ágiles como un problema por resolver.
- **Redacción del estado del arte:** a partir de la investigación exploratoria sobre otros trabajos relacionados con el tema. Esto permitió construir un marco teórico que de soporte a las demás fases del proceso investigativo.
- **Definición formal del problema de investigación.** Esto se hizo que a través de una pregunta se concretara la investigación.

Actividad 2: Hipótesis y Construcción del Modelo Teórico

- Planteamiento de la hipótesis. Se definió la hipótesis central de la investigación. La hipótesis preliminar planteada fue XA permite escalar XP a través de prácticas de arquitectura en un contexto académico.

- Selección y formalización de los insumos de proceso (procesos y fragmentos de proceso). Es decir el proceso XP[3] y los activos de proceso relacionados con los métodos de arquitectura.
- Completar el estado del arte a partir de la investigación exploratoria complementaria sobre otros trabajos relacionados con el tema. Esto permitió construir un marco teórico más completo que dio soporte a las demás fases del proceso investigativo.
- Construcción del marco de proceso de software basado en XP y orientado a métodos de calidad. Dicha construcción consistió en un modelo de proceso especificado en SPEM 2.0 y definido en Eclipse Process Framework.

Actividad 3: Evaluación del Modelo

La evaluación del modelo fue realizada a través de estudios de caso siguiendo la metodología de Runeson y Höst:

- **Diseño de los estudios de caso.** Aquí se establecieron los indicadores y métricas que se aplicaron para alcanzar las conclusiones relevantes del proyecto.
- **Aplicación de los casos de estudio.** Se hizo la aplicación del marco de proceso de software en el entorno académico con equipos de 10±2 participantes en proyectos de mediana complejidad (se describen al interior de este documento). Se recolectaron las mediciones basadas en las métricas definidas (descritas en los anexos).
- **Obtención de los datos** a través de mediciones las métricas fueron aplicados y así obtenidos los indicadores que permitieron responder al cumplimiento de los objetivos de la investigación.

[3] Disponible para EPF en http://www.eclipse.org/epf/downloads/xp/xp_downloads.php

- Generación de un reporte de validación.

Actividad 4. Análisis del Modelo, Conclusiones Empíricas y Validación de la Hipótesis.

- Se describe el análisis de resultados.
- Se realizó una búsqueda de soportes racionales y empíricos que se contrasten o complementen los resultados.
- Obtención de conclusiones empíricas y racionales.
- Confrontación de las conclusiones con las hipótesis.
- Reajustes pertinentes al modelo.
- Trazado para trabajos futuros.

Actividad 5. Documentación

- Elaboración del presente documento de tesis.
- Documentación del proceso ***XP/Architecture - XA*** en Eclipse Process Framework.
- Documentación de los estudios de caso.
- Elaboración de los artículos científicos.

1.5 Estructura de los documentos generados

El cuerpo principal de este trabajo está constituido por la monografía del proyecto de maestría organizado de la siguiente manera:

CAPITULO I: Introducción
CAPITULO II: Marco Teórico y Estado del Arte

CAPITULO III: XP/Architecture(XA)

CAPITULO IV: Validación del modelo

CAPITULO V: Conclusiones y trabajos futuros

Los anexos generados como soporte al proyecto de maestría se organizaron de la siguiente forma:

ANEXO A: Extremme Programming – XP version 1.0

ANEXO B: XP/Architecture version 1.0

ANEXO C: Diseño del caso de estudio versión 1.0

ANEXO D: Resultados de la validación

ANEXO E: XA: Una Extensión de XP para Soportar Prácticas de Arquitectura (Ponencia 7ccc)

ANEXO F: Publicación revista indexada.

CAPITULO II: Marco teórico y estado del arte

En este capítulo se presenta el marco teórico sobre el cual se soporta el presente proyecto de maestría. En él, se describen los aspectos generales de las metodologías ágiles y entre ellas Extreme Programming (XP) como fundamento de estudio del proyecto.

Las metodologías ágiles se han convertido hoy, en un referente para el desarrollo de software para los equipos pequeños [4]. Por ello, para esta investigación, es necesario tener en cuenta cuales son las características primordiales de tales metodologías y cuales sus posibilidades de uso dentro de desarrollos disciplinados y orientados al valor.. Interesa identificar las limitantes para que escalen a sistemas más complejos y equipos más grandes. Particularmente, fue necesario estudiar los trabajos previos que han intentado fortalecer este tipo de problemas con un enfoque de arquitectura, qué es sobre el cuál orientamos esta propuesta.

2.1. El Manifiesto Ágil

Kent Beck, quien propuso el método *Extreme Programming* [5], se reunió en Salt Lake City con un grupo de críticos a las exigencias de las metodologías clásicas para tratar sobre técnicas y procesos para desarrollar software. Como resultado de esta reunión surgió el manifiesto ágil[4]. En la reunión se definió el término “Métodos Ágiles” para definir a los métodos que estaban surgiendo como alternativa a las

[4] http://agilemanifesto.org/iso/es/

metodologías formales o clásicas, a las que las consideraban excesivamente "pesadas" y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo. El ciclo de vida general se puede ver en la Figura 2.

Figura 2.2 Aproximación al ciclo de vida ágil y sus principales componentes.

En dicha reunión se manifestó: "Estamos poniendo al descubierto mejores métodos para desarrollar software, y ayudando a otros a que lo hagan". Con este trabajo se llegó a valorar:

- A los individuos y su interacción, por encima de los procesos y las herramientas.
- El software que funciona, por encima de la documentación exhaustiva.
- La colaboración con el cliente, por encima de la negociación contractual.
- La respuesta al cambio, por encima del seguimiento de un plan.

Las metodologías ágiles, son enfoques de desarrollo de software que promueven prácticas adaptativas en lugar de predictivas, centradas en las personas y en los

equipos, iterativas, orientadas hacia la entrega de valor, de comunicación intensiva, y que requieren que el cliente se involucre en forma directa. Constituyen también una solución a la medida, con una elevada simplificación que, a pesar de ello, no renuncia a las prácticas esenciales para asegurar la calidad del producto [6]

Por tanto, para las metodologías ágiles, es importante comprender los requisitos del cliente con el objeto de ejecutar un desarrollo de software exitoso, pero que en la mayoría de los casos, se presentan los siguientes problemas:

– Los usuarios raramente tienen la habilidad para articular lo que realmente necesitan.
– Las necesidades de los usuarios cambian cuando ellos ven el potencial del sistema y comprenden lo que el sistema puede hacer por ellos.
– Tanto la tecnología como el dominio cambian, y mientras más tiempo tome el desarrollo, más cambiarán.

De ahí que, intentar conocer todos los requisitos del producto al comienzo del proyecto no es factible a un costo razonable. En general, las metodologías se centran en la especificación de requisitos, intentando extraer al máximo los deseos del usuario para entregar un producto lo más cercano a la realidad. Sin embargo, las metodologías ágiles imponen un ciclo de vida más iterativo con el objeto de generar valor rápidamente y validar el avance con software funcionando, más que documentación.

Por las exigencias del usuario y la forma como interviene en los proyectos desde las perspectivas ágiles, los requisitos son siempre crecientes respecto al desarrollo de software; y debido a la ambigüedad con que se tratan las necesidades de usuario, en la mayoría de los casos estos dos factores (requisitos v/s desarrollo) presentan un comportamiento exponencial (ver figura 3)

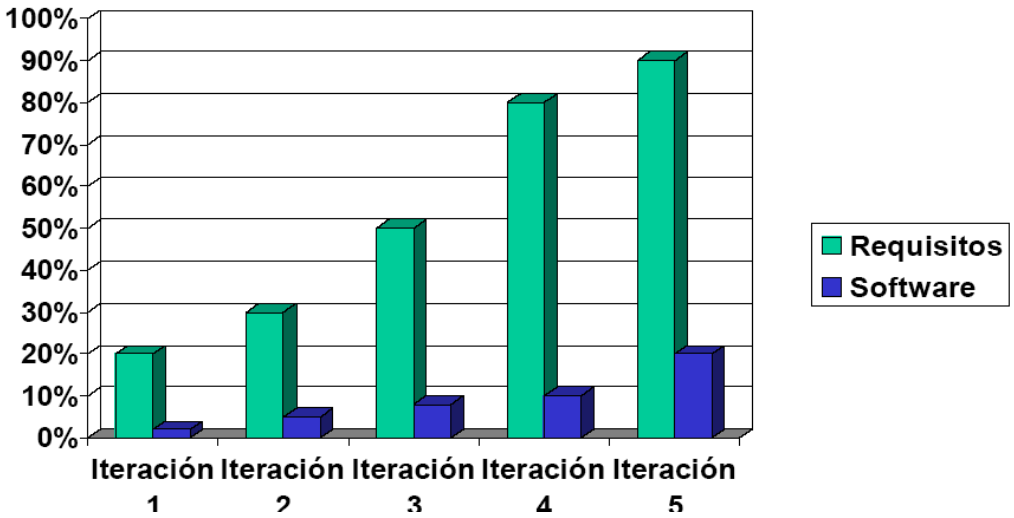

Figura 2.3 Requisitos y software (tomado de Agile Shift System Engineering).

Para el establecimiento de unos principios de diseño en ingeniería [7] que permitan la escalabilidad, esos principios deberían basarse en los siguientes criterios:

- Facilitar la comunicación entre todas las partes interesadas en el desarrollo de un sistema.
- Destacar decisiones tempranas de diseño que tendrán un profundo impacto en todo el trabajo de ingeniería del software.
- Constituir un modelo relativamente pequeño e intelectualmente comprensible pero que a medida que crece el sistema, esos modelos relativamente pequeños puedan acoplarse y verificar de cómo está estructurado el sistema y de cómo trabajan juntos sus componentes.
- Mejorar en la calidad del producto: el diseño de software disciplinado y orientado al valor, debe dar como resultado productos de mayor calidad, más competitivos en un mercado que demanda productos de fácil uso.

2.2. Programación Extrema XP (Extreme Programming).

Dentro de la industria del software, las metodologías ágiles más usadas son Extreme Programming (XP), Scrum, Crystal Method, Feature-Driven Development (FDD), Lean development (LD) y Agile Unified Process [8]. De todas ellas, Extreme Programming (XP) se destaca como la más conocida y aplicada respecto a su flexibilidad, simplicidad, adaptabilidad, prácticas de colaboración y su excelente técnica [9]. Además es la que mejor satisface los principios y valores del manifiesto ágil [10]. Este modelo es propuesto y analizado por Kent Beck, en [11] donde expone las ventajas de un contrato con alcances opcionales.

Figura 2.4 Prácticas Agiles en Extreme Programming.

La figura 4 presenta las prácticas que se definen en XP enmarcadas en los ciclos de una metodología ágil. Estas prácticas se mueven entre entender los deseos del cliente, estimar el esfuerzo, auto-organizarse y auto-gestionarse, crear la solución y entregar el producto final al cliente. Dentro de ese ciclo de vida bastante dinámico, se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto. Se trata entonces de realizar ciclos de desarrollo cortos (llamados iteraciones), para proyectos pequeños con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de desarrollo, pero utilizando el conjunto de principios y prácticas que caracterizan a XP [12].

2.3. Métodos de arquitectura propuestos por el SEI

La arquitectura es un concepto que ha comenzado a tener mayor relevancia en la industria de software y es considerada un área madura dentro de la ingeniería de software [28]. Un ejemplo práctico es el Proceso Unificado de Desarrollo, el cual en su segunda fase, llamada Elaboración, tiene como objetivo el diseño de una arquitectura [13]. Sin embargo, el proceso unificado no provee una metodología concreta para su captura y diseño, sino que se limita sólo a considerar su importancia a través de artefactos que acompañan el análisis y el diseño y algunas pautas generales de priorización y desarrollo de la arquitectura.

En las últimas décadas, el SEI - Software Engineering Institute ha dedicado una de sus líneas de investigación a los aspectos relacionados con la arquitectura de software [14]. De estas investigaciones han surgido una serie de métodos enfocados al análisis, diseño y evaluación de la arquitectura de software. Un proceso general que presenta y relaciona éstos métodos se muestra en la figura 5.

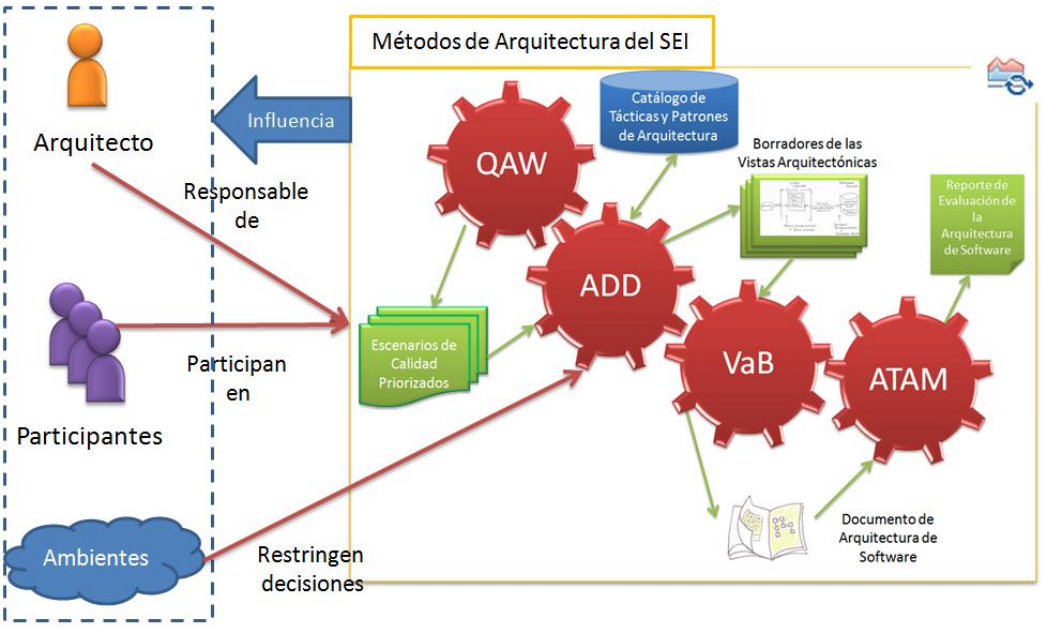

Figura 2.5 Métodos de Arquitectura propuestos por el SEI.

A continuación se describen brevemente cada uno de los métodos de arquitectura:

● **QAW (Quality Attribute Workshop)** [15]**:** Es un método enfocado a la captura de requerimientos que determinan la arquitectura. Estos requerimientos son los atributos de calidad que son descritos a través de un "escenarios" que incluyen información para analizar el comportamiento esperado frente a un estímulo relevante para el atributo de calidad analizado. El escenario completo incluye la fuente del estímulo, estímulo, el artefacto afectado, entorno en que se encuentra el artefacto, la respuesta al estímulo y una medida de la respuesta.

● **ADD (Attribute Driven Design)** [16]**:** Una vez que se han capturado escenarios y se han priorizado, se procede a realizar el diseño de la arquitectura siguiendo un enfoque iterativo de descomposición del sistema en componentes cada vez más

pequeños. Durante ADD, se van tomando escenarios y se van tomando decisiones de diseño (usando soluciones conceptuales llamadas “patrones” y “tácticas”) que permitan satisfacer los requerimientos descritos por los escenarios.

● **VaB (Views and Beyond)** [17]**:** Como resultado de ADD, se obtienen distintas estructuras del sistema, formadas de componentes y sus relaciones. Estas estructuras se documentan a través de “vistas” que muestra una perspectiva particular del sistema. Dado que la comunicación de la arquitectura presenta un papel fundamental en el desarrollo, los aspectos relacionados con su documentación son primordiales.

● **ATAM (Architecture Tradeoff Analysis Method)** [18]**:** El método ATAM permite revelar qué tan bien logra satisfacer los atributos de calidad a la arquitectura y revela además que riesgos, puntos sensibles y compromisos están involucrados en la arquitectura.

La presente investigación, pretende integrar los métodos QAW y ADD en el proceso XP. Se aprovechará el enfoque iterativo y de descomposición del sistema para la arquitectura fundamentado en ADD.

2.4 Trabajos Relacionados

Existen en la literatura aproximaciones de autores que han intentado establecer criterios basados en la arquitectura para plantear soluciones en etapas tempranas dentro de las metodologías ágiles. A continuación se describen brevemente estos trabajos, así como su diferencia con respecto a esta propuesta:

2.4.1 Modelo CA o C3A

Ethan Hadar & Gabriel M Silberman [19] proponen una metodología basada en la arquitectura llamada también CA o C3A, basada fundamentalmente en tres niveles. Esta metodología está diseñada bajo artefactos de UML [31] y cada nivel presenta su funcionalidad en el sistema:

El Nivel 0 considerado como el nivel mas importante, en donde además de especificar los requerimientos, se plantea una etapa de seguimiento y otra de reportes (en diagrama de paquetes de UML) que se hace a través de una API para GUI. Es a través de este nivel que se escriben los “contratos” o “documentación de una página”, como soporte y seguimiento al proyecto. (Ver Figura 6)

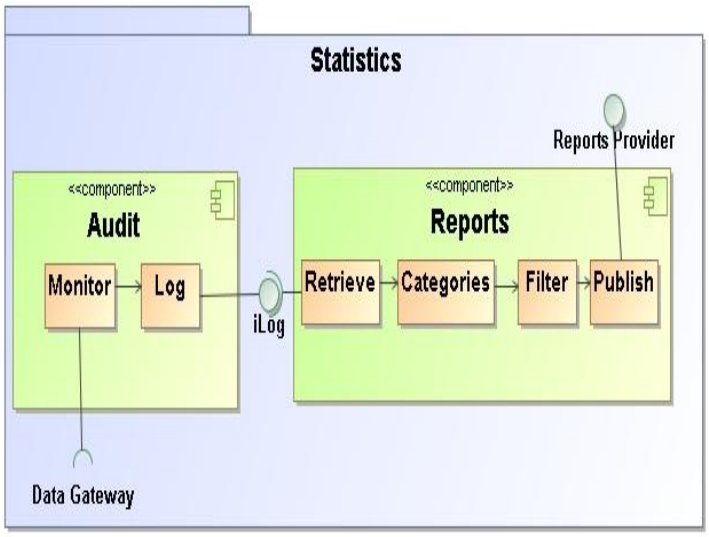

Figura 2.6 (Tomado de Agile Architecture Methodology: Long Term Strategy Interleaved)

El nivel 1 proporciona la misma estructura de información que el nivel 0, pero es mucho más detallado. Este nivel se conecta a través del “contrato” con el nivel 0, para llevar con más detalle los requisitos del sistema. Aquí son muy importantes los parámetros, protocolos, mensajes, canales de comunicación y los puertos que usará la aplicación. Para este nivel es muy importante "lo que hay que hacer", seguido por

"Lo que se requiere para lograrlo."

El nivel 2, es el nivel más alto de abstracción, incluso esta propuesto como opcional. Se ocupa de las interfaces de un componente, basado en los patrones en los cuales se esté soportando la aplicación. Cumple con la función iterativa, hasta completar el código del sistema. En muchas ocasiones este nivel toma importancia, cuando es necesario especificar los componentes del primer nivel. Esto es posible, cuando la complejidad del sistema requiera del diseño a un nivel tres y así sucesivamente

El ciclo de vida propuesto por la CA, basado en siete pasos y considerados de "Bajo Riesgo", está organizado en dos fases: fase de evaluación (escuchar, observar y ver), y fase de evolución (mejorar, analizar, reflexionar y arrancar (Ver Figura 7)

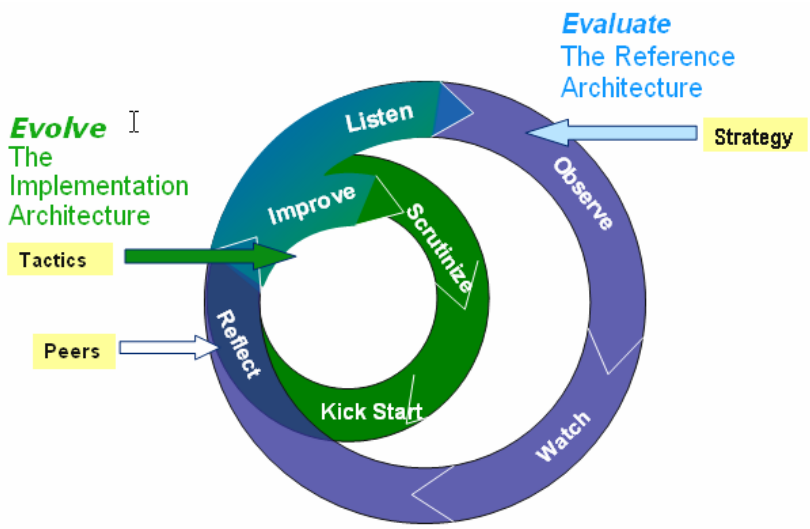

Figura 2.7 (Tomado de Agile Architecture Methodology: Long Term Strategy Interleaved)

Cada una estas etapas cumple lógicamente con sus propias funciones en el sistema. Cabe anotar que este ciclo pretende alcanzar todo el desarrollo desde la especificación de requerimientos incluso hasta una etapa de mejora o de reflexión como es denominada en la etapa de arranque.

Aunque esta propuesta pretende aplicar los conceptos de las metodologías ágiles que ofrecen una solución casi a la medida para una gran cantidad de proyectos pequeños y con requisitos muy cambiantes [7] puesto que una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo [20].

Sin embargo, éste método no muestra las prácticas bien definidas respecto al desarrollo de los requisitos y el diseño de la arquitectura. Tampoco la estrategia está asociada a algún método en particular y además, carece de experimentación que la respalde. A Diferencia de este enfoque, ***XP/Architecture*** o XA especifica explícitamente un proceso definido, bajo los principios y prácticas de XP y los métodos ADD y QAW con el fin de escalar a proyectos complejos sin perder tal filosofía.

2.4.1 Enfoques Centrados en el Usuario

El Ciclo de vida en estrella (arquitectura centrada en el usuario) propuesto por Hix y Hartson [32] (ver Figura 7) un procesos de desarrollo centrado en el usuario, basado en criterios de Interacción Humano computador (IPO). Este ciclo (Representado en la Figura 6), se fundamenta en la evaluación de cada actividad. Igual que las metodologías en cascada [33], no se puede cambiar de actividad hasta tanto no se haya evaluado completamente la anterior. Ello no prescribe cual es el orden o secuencia de cada actividad. Esto destaca el nivel altamente interactivo de la propuesta.

Las restricciones son reducidas al mínimo y con el objeto de cumplir con los tiempos estimados por el manifiesto ágil, es posible empezar con un diseño rápido así no se

haya terminado de especificar los requisitos. Esta propuesta, da mucha importancia al desarrollo de las IU (Interfaces de usuario) proponen incluso unos esquemas que debe tenerse en cuenta para el diseño de estas interfaces.

Aunque esta propuesta trata muy bien las técnicas de especificación arquitectónica, no trata con profundidad las tareas de análisis de sistemas, las cuales son comunes al desarrollo de la interacción con el usuario importante dentro de las metodologías ágiles y al desarrollo de la parte interna del sistema. Tampoco define claramente cuáles serán las actividades prácticas e indispensables para el diseño de una buena arquitectura, ni mucho menos (aunque su filosofía está basada en el paradigma de cascada), fortalece el paso de proyectos pequeños a proyectos complejos tanto en desarrollo como en número de desarrolladores.

Figura 2.8 (Tomado de Hix y Hartson 2003)

El diseño centrado en el usuario de Constantine y Lockwood [34] proponen una metodología, basada en una colección de actividades coordinadas que contribuyen a la arquitectura, agrupadas en el método de Diseño Centrado en el Uso. El usuario es importante en estas tareas y para ello, el modelo de actividades de diseño centradas en el uso incluye algunas actividades que corresponden al proceso de desarrollo más amplio: diseño de Objetos, Construcción Concéntrica e Iteración Arquitectural, junto con actividades de usabilidad puras, como modelado de tareas o modelado del contenido de la Interfaz. Estos modelos propuestos son bastante atractivos; y aunque pretenden ser muy cercanos al requerimiento de los usuarios, se basan en criterios Orientados a Objetos (RUP), lo que hace que la propuesta se aleje de toda

posibilidad de tiempos, factor importante para la entrega del producto al cliente, fundamento del manifiesto ágil. En concreto, la técnica de casos de uso esenciales, que son un pilar del enfoque centrado en el uso, es una reinterpretación de la técnica de casos de uso, la cual es muy popular en el desarrollo orientado a objetos. Los casos de uso esenciales pueden, por tanto, servir de puente para la integración de la arquitectura en el proceso de desarrollo.

2.4.2 Enfoques Centrados en la Arquitectura

Si bien es cierto las metodologías ágiles y entre ellas Extreme Programming, ha ganado bastante popularidad desde hace algunos años, debido a que son una muy buena solución para proyectos a corto plazo, en especial, aquellos proyectos en donde los requerimientos están cambiando constantemente, es muy cierto también que en proyecto a largo plazo, el aplicar estas metodologías no dan tan buenos resultados. De ahí que el diseño de la arquitectura de software se haya convertido en una práctica muy importante para el desarrollo de software [23]. Tener una buena arquitectura implica que nuestro sistema tiene atributos de calidad que nos van a dar un valor muy importante en el software. Si se definen actividades que fomenten el uso de métodos para el desarrollo de la arquitectura [15][16], en un proceso de desarrollo de software, se puede obtener muchos beneficios con respecto al producto que se desarrolla.

Constantine y Lockwood [34] ofrecen una serie de consejos, admitiendo que no hay una única forma de enfocar una integración sólida de la metodología con la arquitectura. Por tanto, dejan el tema de la integración para ser resuelto de forma particular en cada caso. Indican que "buenas estrategias de integración de la arquitectura en el ciclo de vida acomodan de forma conjunta las nuevas prácticas con las antiguas, modificando las prácticas actuales para incorporar la arquitectura en los procesos de diseño y análisis, al tiempo que se particulariza el diseño centrado en el

uso a la organización y sus prácticas" [13].

Si bien es cierto, esta propuesta hace un acercamiento mas a la arquitectura, y aunque se fundamenta en artefactos de metodologías orientadas a objeto, que han demostrado trabajar bien en proyectos complejos, no es muy clara en temas de arquitectura como tal, es decir el hibrido formado entre metodologías ágiles y UP[34], hecho por Constantine y Lockwood no favorece ni el postulado del manifiesto ágil en criterios de agilidad, ni incorpora prácticas de arquitectura más allá de los artefactos básicos basado en el modelo 4+1 Vistas[24].

Andreas Kornstädt and Joachim Sauer [23], basados en sus experiencias reales, demuestran que una vista arquitectónica común permitiría solucionar problemas de comunicación en un proyecto de desarrollo ágil. Aunque no se propone un proceso que combine métodos ágiles con métodos de arquitectura, ésta es una evidencia concreta que respalda éste proyecto, puesto que defiende la idea de que la arquitectura es un artefacto relevante para escalar el proyecto a grupos más grandes.

Rolf Njor Jensen, Thomas Möller, Peter Sönder, and Gitte Tjörnehöj[24] proponen las "Historias de desarrolladores", las cuales son análogas a las historias de usuario, para representar los requerimientos y decisiones de arquitectura. Este nuevo componente es agregado a XP con el objeto de plantear una arquitectura basada en las “historias de desarrolladores”. Las historias de desarrolladores describen los cambios, experiencias, unidades de propiedades visibles para desarrolladores del software. En contraste, las historias de usuario describen unidades de funcionalidad visible para el usuario del software. La representación física de una historia de desarrollador es una tarjeta que puede tener un color diferente a las historias de usuario. El propósito de las historias de desarrolladores presenta dos características: primero, que son una herramienta para la planificación y expresan las demandas

concretas de refactorización. Segundo, obliga a los desarrolladores a reflexionar sobre el diseño del sistema, y efectivamente construir un entendimiento compartido de los elementos importantes de la arquitectura. Sin embargo, las “historias de desarrolladores” no se preocupan por solidificar una arquitectura como artefacto, el cual permita representar la descomposición del sistema de acuerdo a los requisitos de calidad. Tampoco se integra explícitamente algún método de arquitectura.

J. Reifer et Al. [11] recogen el criterio de varios expertos quienes dicen que no es posible escalar los métodos ágiles sino se usan metáforas mixtas (agiles y rigurosas). Estos autores proponen un equipo de arquitectura y subdividir el resto del equipo en equipos mas pequeños de desarrolladores, aclarando que dependiendo del tamaño del software, la comunicación y los otros principios del postulado ágil se pueden hacer tan complejos que en la relación cliente y equipos XP pueden perderse los objetivos del sistema.

2.4.3 Estudios de productividad de las metodologías ágiles (en particular de XP)

Como se ha mencionado en páginas anteriores, en los últimos años las metodologías ágiles han irrumpido con fuerza como un intento de despojar al desarrollo software a las estrictas y rígidas estructuras y arquitecturas planteadas por las metodologías tradicionales, y no son pocas las organizaciones que en estos momentos, debido a la productividad que apuntan con creciente interés a las metodlogías ágiles [4]. La novedad de estas metodologías hace que, aunque existen evidencias de los beneficios que están proporcionando a proyectos de pequeña envergadura, de todas maneras (y es el propósito de esta investigación) resulta difícil escalar a grandes proyectos. Algunos estudios recientes indican que la productividad y calidad del software aumenta aplicando los principios y valores que las rigen. No obstante, la

mayoría de estos estudios se limitan a narrar observaciones cualitativas. Entre los que utilizan datos empíricos para apoyar sus conclusiones, los resultados son tan dispares como una mejora del 337% en la productividad en [56] y un decremento del 44% en [57]. Por este motivo, desde las organizaciones que promueven el desarrollo ágil de aplicaciones se solicita la realización de estudios sobre metodologías ágiles que permitan constatar o reprobar sus beneficios sobre todo cuando deben aplicarse a proyectos de mediana complejidad con equipos de las mismas características.

De ahí que la productividad de los equipos XP han sido poco reportadas en la literatura. Según lo reportado por [19], la productividad fue de 0,003 historias de usuario persona hora (32 historias de usuario, en 3.5 meses por un equipo de 14.7 personas). Por otro lado de acuerdo a [20], la productividad de un equipo XP es de 17 NLOC-P-H1 fue notablemente mejor que un proyecto desarrollado en forma tradicional (10.3 NLOC-persona-hora). Extender ésta parte y hablar no sólo de los aspectos cualitativos sino de los casos con un mayor detalle.

CAPITULO III: “XP/Architecture”

Si quieres construir un barco, no llames a la gente a recoger leña, a dividirles el trabajo y darles órdenes. En lugar de ello, enséñales a añorar el inmenso e infinito mar.

(Antoine De Saint-Exupery)

En este capítulo se hace una descripción del modelo propuesto en ésta tesis: XP con Arquitectura (XA). Una solución al problema de escalar XP a proyectos más complejos, es decir equipos de desarrollo grandes y requerimientos de negocio\tecnologías complejos.

En el resto de éste capítulo el modelo de proceso XA es caracterizado y descrito como un modelo de proceso siguiendo un enfoque de descomposición Top-Down usando elementos del lenguaje SPEM[58] para representar procesos.

3.1 Que es XP/Architecture (XA)

XA es una metodología ágil sostenida por los valores y principios propios de la programación extrema (XP), que agrega algunas prácticas de arquitectura, cuyo objeto primordial es posibilitar el uso del enfoque ágil a proyectos de mediana complejidad y con equipos de más de 8 personas donde la programación extrema ha reportado tener dificultades para escalar [35].

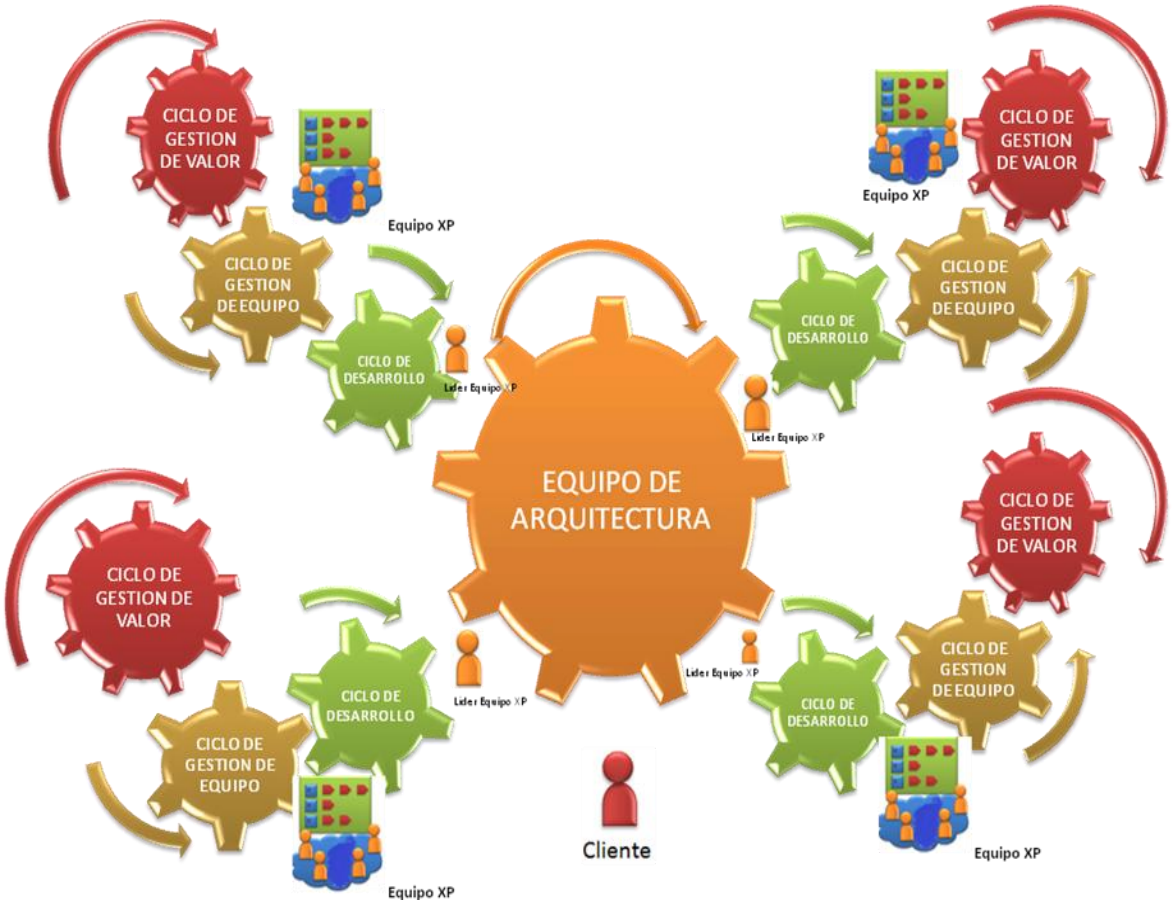

Figura 3.9 El ciclo de vida de Xp/Architecture

Al igual que XP, para XA son muy importantes los 4 valores propuestos por el manifiesto ágil [36], por ello XA ha extendido su definición y propone un nuevo valor que formará parte de los pilares que asegurarán una arquitectura y un desarrollo ágil en proyectos de mediana complejidad:

Figura 3.10 Valores de XP/ARCHITECTURE

- **La organización de equipos XP**: Debido a la magnitud del proyecto, este debe ser organizado en sub equipos XP, tantos cuantos lo amerite el proyecto, cada equipo seguirá los valores y principios de XP [37] y al menos un equipo tendrá más énfasis en la arquitectura de software, considerándolo de valor para el negocio de desarrollo y quien seguirá las prácticas extendidas definidas en XA. A los equipos que sigan los principios de XP se denominarán ***equipos XP*** y al equipo que siga además las prácticas de arquitectura se denominarán ***equipo XA***.

- **La comunicación**: Basada en una alta interacción entre los equipos de trabajo XP, los miembros del equipo XA y de estos con el cliente, la comunicación requiere establecer los canales para que el conocimiento fluya de forma emergente, pero estructurada [21]. El cliente debe ser miembro activo del equipo XA como fuente permanente de los requisitos del sistema. Este equipo está conformado por un miembro de cada equipo XP, quienes a su vez hacen de clientes en su respectivo equipo XP. Al cliente del proyecto se le denominará **Cliente XA** y al cliente de cada equipo XP, miembros a su

vez del equipo XA, se le denominará **Cliente XP**. Son los clientes XP los agentes que establecen la relación inter-grupo y por tanto son los elementos claves de la comunicación en XA.

- **La simplicidad**: Aplicada a todos los aspectos de XA, desde la fase inicial, con arquitecturas muy sencillas y basadas en prácticas básicas, incluyendo la refactorización de código. Para XA, lo importante es hacer “Lo que el cliente necesita”, debe “hacerse de la forma más sencilla”, pero sin perder la posibilidad de la integración continua y de valor.

- **La retroalimentación**: Sucede en muchos momentos, pero los dos momentos más significativos; el primero por parte del equipo de trabajo XA hacia el cliente XA, con el fin de brindarle avance sobre el desarrollo del sistema y conocer su apreciación, y desde el cliente XP hacia el equipo XP en los aportes al desarrollo del proyecto. El segundo momento relevante ocurre entre cada miembro de los equipos XP, y los miembros del equipo de arquitectura XA, con el fin de brindarle avance sobre el desarrollo del sub-sistema y conocer su apreciación, y desde el cliente XP hacia el equipo XP en los aportes al desarrollo del sub-proyecto.

- **La valentía**: Para enfrentarse a los continuos cambios que se presentan en el transcurso de la actividad, que implica a todos los miembros tanto del equipo XA como los equipos XP.

3.2 Reglas propias de XA

A partir de los valores se plantean una serie de prácticas que sirven de guía para los desarrolladores en esta metodología. Una de los aspectos más importantes para XA

son principios.

Figura 3.11 Principios de XP/ARCHITECTURE

Además de los doce (12) principios que caracterizan a XP[25], existen otros que también se caracterizan por su grado de simplicidad y por su enfoque en la práctica y que competen solo a los equipos XA con el objeto de alcanzar la escalabilidad de los métodos ágiles:

- **Gestión ágil XA**: Cada equipo XA y XP tienen un líder de proyecto quienes gestionan de forma ágil el proyecto. Para la integración de las prácticas de gestión el líder del proyecto XA se reúne periódicamente (reunión diaria de gestión) con los líderes de cada proyecto XP para abordar los problemas y enfoques de gestión del proyecto**.**

- **Unificación de código en el equipos XA:** Al igual que con los equipos XP, existe para el equipo de arquitectura una propiedad colectiva de código [11], pero con el objeto de hacer menos complejo el desarrollo y cumplir con los criterios ágiles en cada equipo existe la propiedad colectiva del código del sub-sistema que tiene a cargo (la propiedad colectiva de código se mantiene como práctica). XA se apoya de prácticas de arquitectura para la distribución de subsistemas, distribución de los requisitos y posterior validación e integración. XA puede verse como un equipo ágil que subcontrata el desarrollo de componentes con equipos XP.

- **Mecanismo para determinar requisitos:** Una vez establecido un incremento en la arquitectura, por el equipo XA, se le comparte a los equipos XP los requerimientos del subsistema aprovechando la interacción de un miembro XA como cliente del equipo XP.

3.3 Alcance de XA

XA es conveniente para proyectos de mediana complejidad y con equipos de más de 8 personas, además, XA posibilita la agilidad que ofrecen las metodologías ágiles [38], por tanto aunque el proyecto tenga cierto grado de complejidad, es posible que satisfaga la flexibilidad en los requerimientos. Es decir, se adhiere al ambiente cambiante que se presenta con las necesidades del cliente. De ahí que XA está encaminada hacia los desarrollos que requieren de cambios continuos en los requerimientos en el transcurso de un proyecto.

Los proyectos realizados bajo esta metodología cumplen con lo estrictamente necesario en su funcionalidad a cada momento necesario: “hacer lo que se necesita cuando se necesita”, precipitarse o adelantarse a las tareas que se han establecido

previamente con el cliente, conllevan a complejizar el sistema, alejándolo del concepto de simplicidad [39].

3.3 El Modelo Holístico de XA

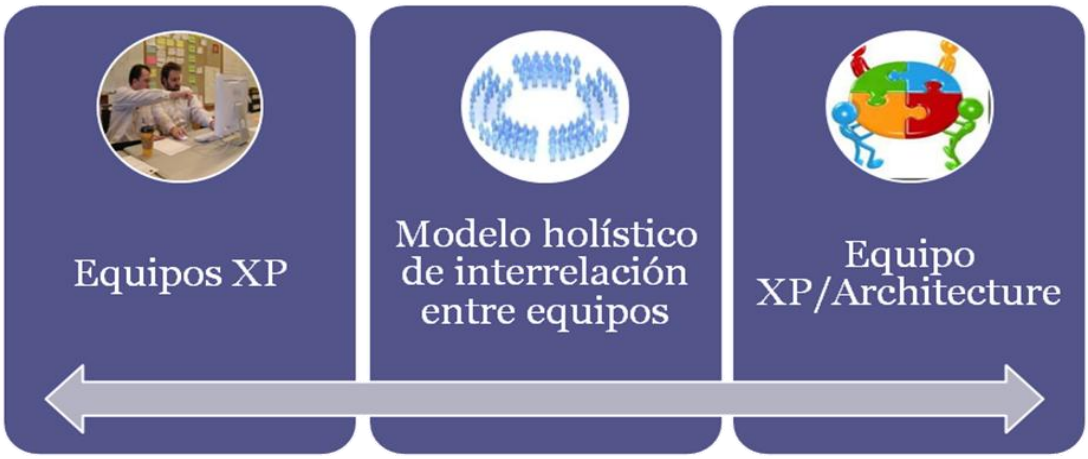

Figura 3.12 El modelo holístico de XA

Teniendo en cuenta que todo modelo Holístico se basa en sus inter-relaciones [59], para XA, es importante la sinergia generada por el modelo y esto se debe a que cuando los equipos XP aplican sus prácticas, los resultados se reflejan en XA donde son fortalecidas con las practicas del XA, y su vez vuelven a los equipos XP a través del cliente (el representante de XP en XA) esto hace que se genere cada vez mas valor tanto para el cliente del sistema como para los equipos XP y XA.

La sinergia entre los equipos hace que se re-creen en cada una de sus actividades y que sus resultados generen conocimiento para los equipos; de ahí que sería imposible explicar el modelo analizando los resultados de uno de los equipos XP o solamente del equipo XA. DE ahí que para estudiar el valor generado por el modelo es necesario analizar a los equipo en su conjunto aplicando XA.

3.4 El ciclo de vida de XA

XA al igual que cualquier proceso de desarrollo de software, sigue un conjunto de fases e iteraciones como se muestra en las Figuras 12 y 13. XA cuanta con cuatro fases: Exploración, Planificación, Desarrollo con Arquitectura, Entrega Final. Una iteración define el trabajo técnico del grupo XA y embebe iteraciones paralelas del proceso XP, las cuales son planificadas y gestionadas de forma independiente por cada equipo XP, pero coordinadas a través de un modelo holístico de XA que facilita la comunicación y sincronización hacia la generación continua de valor.

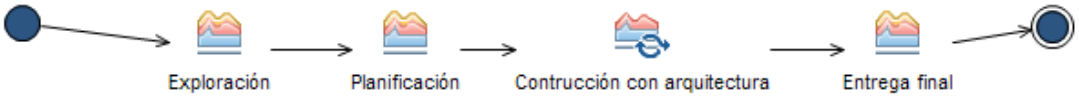

Figura 3.13 Fases de XA

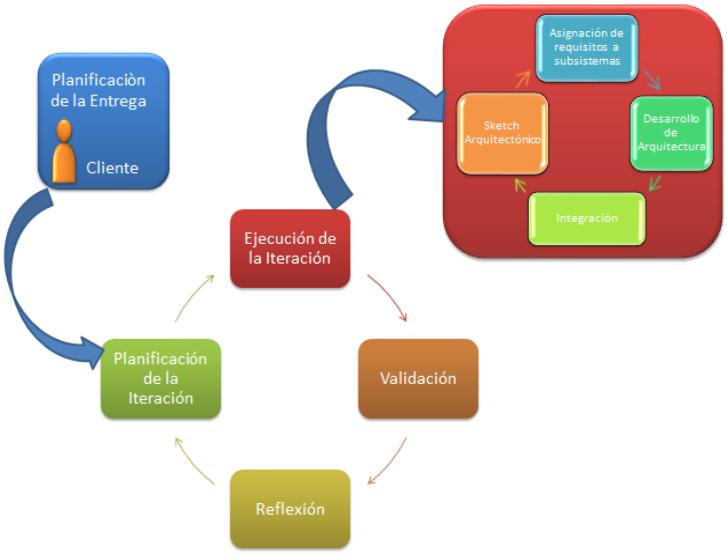

Figura 3.14 Iteración conceptual en XP/ARCHITECTURE

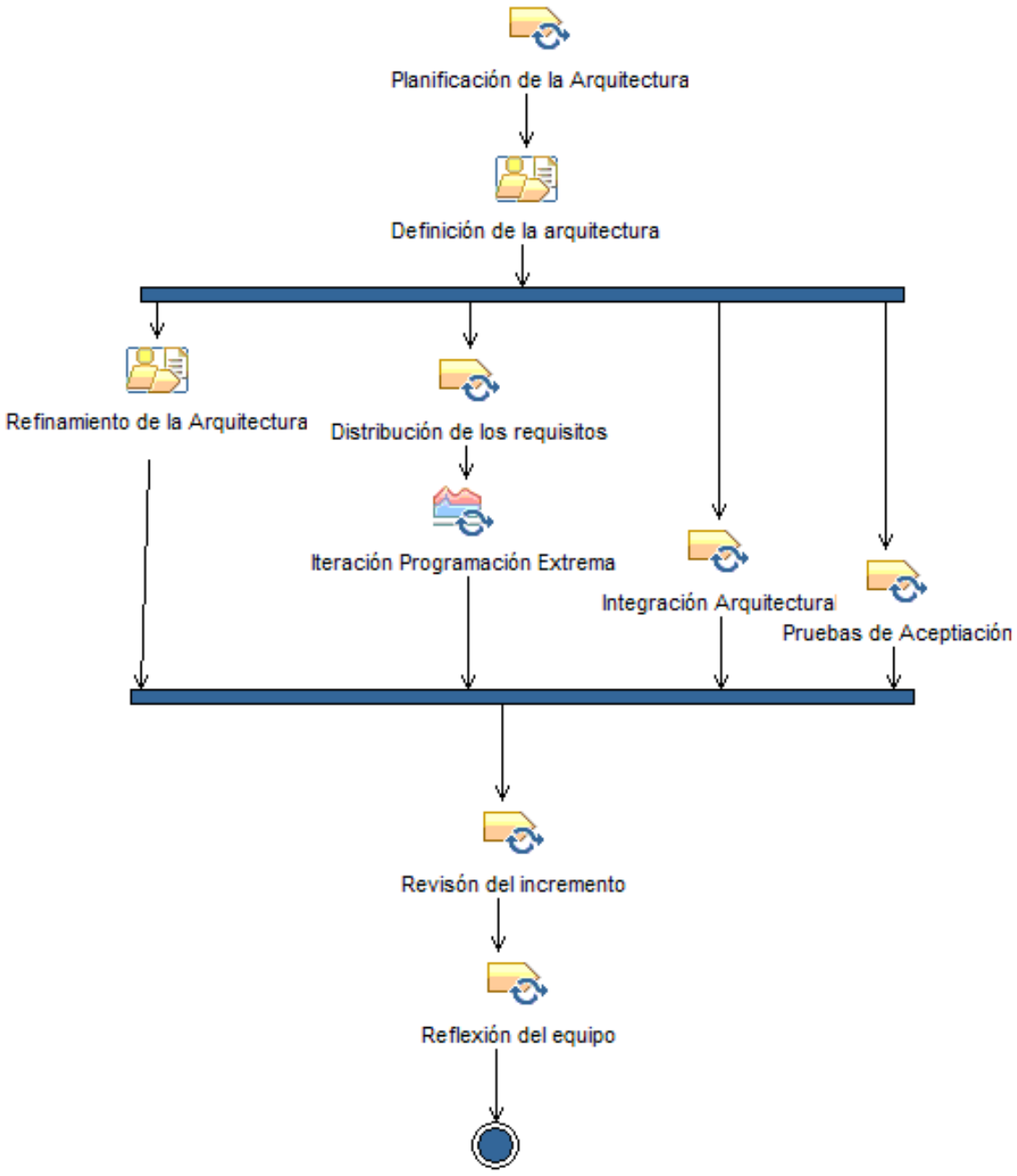

Figura 3.15 Iteración en forma detallada de XA

Cada una de las fases del ciclo de vida en XA se desarrollan de la siguiente forma:

- **Exploración:** se reúnen los miembros del equipo XA, con el cliente XA para establecer los requisitos del sistema. En esta fase se delimita el sistema.

- **Planificación de la Arquitectura**: se reúnen los miembros del equipo XA, se establecen los objetivos del proyecto, y se define una arquitectura metafórica del sistema.

- **Construcción con arquitectura:** se priorizan los requisitos del sistema y se desarrolla un borrador de la arquitectura planteada por el equipo de arquitectura para esos requisitos, se modulariza el sistema y se distribuyen los requisitos a los equipos XP que participen en el proyecto. La definición de arquitectura se desarrolla paralelo al desarrollo de los equipos XP. Continuamente se va integrando el sistema (ver figura 6), sin embargo hay una fase de integración donde se refina, refactoriza y prueba la integración. Continuamente al final de cada iteración se presenta al cliente los resultados y con el fin de obtener una valiosa retroalimentación.

- **Entrega Final**: El equipo XA presenta al cliente un producto afirmando que no hay mas historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final y no se realizan más cambios en la arquitectura. La terminación del proyecto ocurre cuando los requerimientos han sido satisfechos, los equipos XP han entregado todas sus historias. Acciones validadas por el cliente y el equipo XA. Esto quiere decir que ya hay un producto final preparado para publicarse como versión definitiva. En esta fase el producto implementa todas las funciones del diseño

y se encuentra libre de cualquier error que suponga un punto no consolidados en el desarrollo.

Cada equipo XP se mueve paralelo a este ciclo de vida, de acuerdo a la metodología XP **¡Error! No se encuentra el origen de la referencia.**, sincronizándose a través del cliente XP miembro del equipo XA al que se le encarga comprender y comunicar los requerimientos del respectivo sub-sistema.

3.5 Los roles en XA

Para XA es necesario definir roles con el objeto de organizar quienes se encargaran de cada una de las actividades que deben realizarse en el transcurso del proyecto y estos roles son por:

- **El cliente XA**: Es quien determina los requisitos del sistema y define qué es lo se va a desarrollar.
- **Líder del proyecto XA:** Quien tiene interacción directa con el cliente XA, Recibe directamente del cliente XA los requisitos del sistema y los comparten (Junto con el cliente XA) a los demás miembros del equipo XA.
- **El Equipo XA**: compuesto por un representante de cada equipo XP (interface cliente a cada equipo XP), el líder del proyecto XA y el Cliente XA.
- **Los Equipos XP**: con sus roles y actividades según lo dispone esta metodología ágil [3].

El trabajo por pares y la rotación de las personas del grupo es importante en XA para evitar entre otros, problemas relacionados con la pérdida de conocimiento, que por cierto se convierte en una fortaleza dentro del equipo XA; para ello los representantes del equipo de arquitectura en cada equipo XP, al igual que los miembros XP, se rotarán para desarrollar las tareas dentro del equipo XA.

3.6 La arquitectura en XA

Al igual que los criterios de las metodologías ágiles, XA tiene en cuenta para el diseño: la Simplicidad, la metáfora del sistema, Tarjetas de clase, responsabilidad, colaboración (CRC cards) y la Refactorización (Refactoring) [40]. En forma concreta XA ha adaptado y adoptado en forma simplificada los métodos de arquitectura Attribute-Driven DesignADD [15] y Quality attribute Workshop (QAW) [16] debido a que posibilitan criterios de calidad para soportar la arquitectura y documentar los requisitos del sistema que aparecen una vez iniciado el proyecto.

3.6.1 Conceptos de Arquitectura

Se entiende la arquitectura como la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución [60], siendo su objetivo principal el de aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. De ahí que dentro de las practicas de XA sea tan importante establecer mecanismo de comunicación dado el tamaño del equipo, quienes sobre pasan en número por los propuestos por las metodologías ágiles.

XA plantea que se desarrolle una arquitectura basada en atributos de calidad los que anteriormente se conocían como aquellos que son parte de los requerimientos no funcionales de una aplicación, la cual captura las múltiples facetas de cómo los requerimientos funcionales de una aplicación son logrados [61]. Para XA los atributos de calidad son los que hablan de las características específicas que debe tener el sistema [62]; para que tengan sentido los requerimientos de atributos de calidad deben ser específicos de cómo una aplicación debe lograr una necesidad dada,

puesto que la calidad está basada en el grado que posee el software para combinar sus atributos. De ahí que sean importantes para XA los escenarios de calidad como mecanismos para expresar los atributos de calidad y que consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco del sistema a desarrollarse.

3.6.2 Que es QAW (QUALITY ATTRIBUTE WORKSHOPS)

QAW [16] es una manera de documentar, descubrir y dar prioridad a la calidad de un sistema de atributos al principio de su ciclo de vida.

QAW ayuda a los arquitectos del sistema para obtener la máxima información posible sobre la calidad y ayuda también en la definición de la estructura del sistema; QAW es un método usado para generar priorizar y refinar escenarios de atributos antes que la arquitectura del software sea completada.

QAW propone un plan de arquitectura basado en escenarios donde se conjugan los requerimientos del sistema, los objetivos del negocio, el equipo de trabajo y los stackholders, cuyos beneficios incluyen: Una lista de drivers de arquitectura y una lista de priorización y refinación de escenarios que consolidan el sistema.

En XA, los requerimientos funcionales se definen mediante historias de usuario, los atributos de calidad se explican mediante escenarios de calidad, conocidos como Historias de arquitecto que se exponen en el kanban del equipo XA los cuales han sido concertados en el ejercicio de la realización del producto entre el equipo de arquitectura y el cliente

Para el cumplimiento de QAW el equipo XA realiza la siguientes actividad:

- ***Confirmar con claridad los requerimientos:*** se trata de tener una lista priorizada de requerimientos, expuesta por el cliente y clasificada por el equipo XA basada en los objetivos y la misión del negocio, en atributos de calidad, restricciones de diseño y requerimientos funcionales. En esta etapa, se hace una lista jerarquizada de tales requisitos, con el objeto de priorizar cuales tendrán más impacto en la arquitectura y por tanto susceptibles de ser candidatos para la siguiente fase.

- Identificar los requerimientos de mayor importancia para el cliente y de alto impacto para la arquitectura y expresarlos como Historias de arquitecto. Para ello se tendrán en cuenta los criterios expuestos en el Kanban [41] del sistema.

3.6.3 ADD (ATRIBUTE DRIVEN DESIGN) y XA

ADD es un método de arquitectura [15] basado en atributos de calidad que el software debe cumplir. Sigue un proceso recursivo de descomposición del sistema y que en este caso favorece al equipo XA. Para su aplicación, el equipo XA y el cliente ya han venido desarrollando los requisitos del sistema basados en requerimientos funcionales, restricciones de diseño, y atributos de calidad [15]. Particularmente en XA, cuando QAW ha sido aplicado y a partir de la retroalimentación de las anteriores iteraciones XA y actividades de refactorización arquitectónica.

Las restricciones son decisiones de diseño de estricta obligación a tener en cuenta debido a todos los factores externos que rodean el sistema. Para el cumplimiento de ADD el equipo XA realiza las siguientes actividades:

- Se eligen los elementos del sistema para el diseño. Para ello se tiene en cuenta ***las Historias de Arquitecto*** y la descomposición propuesta

previamente del sistema. El equipo de arquitectura tomara cada uno de estos elementos como fuentes de información para el diseño de una estrategia arquitectónica que para facilitar la comunicación con los equipos XP cuando se le distribuyan los módulos del sistema.

- Elegir ***Patrones y Estrategias Arquitectónicos*** acordes a las Historias de arquitecto propuestos para el diseño de los elementos de alto impacto seleccionados. Aquí se deben tener en cuenta aspectos como restricciones, requerimientos, atributos de calidad, elementos de comunicación y artefactos de arquitectura. Se debe establecer un modelo que ha de servir de patrón a todos los elementos de diseño del sistema.

- A partir de estos patrones y estrategias, se instancian e implementan los elementos abstractos en la plataforma de desarrollo para crear los nuevos elementos del sistema. A cada nuevo elemento, se le asignara sus responsabilidades y funciones pertinentes al sistema.

- Para integrar las partes abstraídas, se definen y programan las interfaces de los elementos instanciados para estandarizar la información y servicios bajo los cuales los módulos se comunicarán.

- Verificar y distribuir los requerimientos y realizar las restricciones para los elementos instanciados. Estas restricciones se convertirán en los requisitos para los módulos internos. Repetir estas actividades hasta que se haya alcanzado una adecuada descomposición arquitectónica del sistema.

3.7 Practicas XA: extendiendo las prácticas XP con ADD y QAW.

Estas prácticas han sido combinadas y compactadas con las prácticas base de XP [3]. El anexo ABC presenta una descripción detallada de las prácticas paso a paso como una guia de conducción de QAW y ADD.

3.7.1 Complemento de QAW a XP para soportar XA

Tabla 4.1 QAW - XP y XP/ARCHITECTURE

JUEGO DE LA PLANIFICACION	
EXTREME PROGRAMMING(XP)	**XP/ARCHITECTURE (XA)**
Es una práctica continua, donde el cliente se pone de acuerdo con el equipo XP. Desde que empieza el desarrollo del producto, se requiere que el grupo y el cliente tengan una visión general y clara de que es lo que se quiere y se va a hacer, es decir, deben entender y estar de acuerdo con lo que el "otro" plantee. En el transcurso del proyecto se realizan diferentes reuniones, con el fin de organizar las tareas y nuevas ideas que surjan por parte del cliente y del equipo [40]	Las Historias de usuario que resultan de XP, se complementan con las Historias de arquitecto proporcionados por QAW [16] donde se tienen en cuenta los escenarios y sus seis componentes en una forma liviana. La priorización de escenarios mismos dan información adicional para el cliente y los desarrolladores con el objeto de ayudarles elegir las historias de de arquitectura claves y las historias de usuario relacionadas. En esta fase se seleccionan los drivers de arquitectura, muy útiles en la elección de las historias de usuario durante la práctica del juego de Planificación.
EL CLIENTE *IN SITU*	
EXTREME PROGRAMMING(XP)	**XP/ARCHITECTURE (XA)**
El cliente debe estar dispuesto para el equipo de desarrollo con el	El cliente es acompañado por los otros participantes relevantes de los

objeto de solucionar las preguntas o dudas que se puedan presentar a medida que se realice el proyecto. El en sí mismo es los requerimientos en vivo y es quien valida si la entrega es útil o no	equipo XP y XP con el fin de evaluar los requisitos del sistema y las soluciones planteadas al mismo. Esto se hace en las reuniones de planificación de la iteración.
PRUEBAS CONTINUAS	
EXTREME PROGRAMMING(XP)	**XP/ARCHITECTURE (XA)**
Cuyo objetivo es verificar la funcionalidad y estructura de cada componente individualmente. Con ello se prueba cada historia de usuario basados en pruebas de unidad y de aceptación con la participación del cliente en este caso el representante del equipo XP.	Se trata de verificar el correcto ensamblaje entre los diferentes componentes del sistema una vez que han sido probados unitariamente con el fin de comprobar que responden a la arquitectura propuesta y los requisitos del cliente. Para ello, se mantiene las pruebas unitarias y de integración de XP, pero se agregan las pruebas de arquitectura con las Historias de arquitecto.

3.7.2 Complemento de ADD a XP para soportar XA

Tabla 4.2 QAW - XP y XP/ARCHITECTURE

JUEGO DE LA PLANIFICACION	
EXTREME PROGRAMMING(XP)	**XP/ARCHITECTURE (XA)**
Es una práctica continua, donde el cliente se pone de acuerdo con el equipo XP. Desde que empieza el desarrollo del producto, se requiere que el grupo y el cliente tengan una visión general y clara de que es lo que se quiere y se va a hacer, es decir, deben entender y estar de acuerdo con lo que el “otro” plantee. En el transcurso del proyecto se realizan diferentes reuniones, con el fin de organizar las tareas y nuevas ideas que surjan por parte del cliente y del	ADD define la arquitectura basado en atributos de calidad, y que se deben aplicar en las primeras iteraciones de XP Para ello se eligen los controladores de arquitectura de una lista de atributos, basados en los requisitos del cliente, lo que permite definir también las historias de usuario. Para efectos de la planificaciones iniciales, se escoge bien sea metáforas o se usan patrones arquitectónicos típicos para facilita.

equipo [3]	
METAFORA DEL SISTEMA	
EXTREME PROGRAMMING(XP)	**XP/ARCHITECTURE (XA)**
Las utilizamos con el objeto de establecer un lenguaje unificado entre el equipo XP y el cliente. La idea es mejorar la comunicación y hacer de los requerimientos historias muy sencillas y claras tanto para el equipo XP como para el Cliente [42].	El método ADD proporciona un enfoque definido para el diseño de la arquitectura en términos de descomposición del sistema en módulos más pequeños usando diferentes perspectivas del sistema y basándose en las Historias de arquitecto.
DISEÑO SIMPLE	
EXTREME PROGRAMMING(XP)	**XP/ARCHITECTURE (XA)**
Los diseños complejos no tienen cabida dentro de XP, porque generalmente no aportan soluciones claras al desarrollo del producto. Esto no quiere decir que no se debe hacer diseño, XP exige que se haga por el contrario un diseño pero muy sencillo y ágil que pueda ser entendido por todos los miembros del equipo [11]	La necesidad de ADD de satisfacer las Historias de arquitecto se basa en definiciones previas, por lo que requiere aplicar un método una refactorización a nivel arquitectónico, en este caso ayudado por la aplicación de ciertos patrones y estrategias de arquitectura.
REFACTORIZAR	
EXTREME PROGRAMMING(XP)	**XP/ARCHITECTURE (XA)**
Se realiza durante todo el proceso de desarrollo. El código se revisa de forma permanente para depurarlo y simplificarlo, buscando la forma de mejorarlo [22]	La necesidad de ADD de satisfacer las Historias de Arquitecto se basa en soluciones tempranamente establecidas, por lo que requiere aplicar continuamente prácticas de refactorización a diferentes niveles de granularidad. En el caso arquitectónico esto es ayudado por la aplicación de ciertos patrones de arquitectura.

3.8 Actividades de XA respecto a la arquitectura

Figura 3.16 Actividades principales relacionadas con la arquitectura en la iteración de la arquitectura

Al igual que XP [43], XA propone unas actividades que forman parte de su ciclo de vida y que ponen en evidencia las prácticas anteriormente anotadas:

3.8.1 Actividad 1: Analizar Los Requisitos Arquitectónicos

Figura 3.17 Tareas del análisis de requisitos arquitecturales (Aplica QAW)

Esta actividad basada en criterios de calidad de QAW, propone que se realicen las siguientes tareas:

- **Identificar historias del arquitecto**: El cliente y el equipo de arquitectura (XA) identifica las historias del arquitecto,

- **Analizar historias del arquitecto:** El equipo XA, analiza las Historias de arquitecto, las prioriza y expresa siguiendo un esquema liviano que permita expresar escenarios de calidad. que sean relevantes a la arquitectura identificando los valores propios del manifiesto ágil [3] [44]: costo, tiempo, calidad pero aplicada a la unidad de desarrollo. En esta tarea, el equipo XA, manifestará los requerimientos funcionales basados en las historias de usuario relevantes para construir las piezas arquitectónicas y los requisitos de calidad, expresados por historias de arquitecto, donde se describirán por medio de escenarios de calidad propios de QAW [16] pero en un formato libre y metafórico escogido por el equipo de arquitectura.

3.8.2 Actividad 2: Diseñar arquitectura

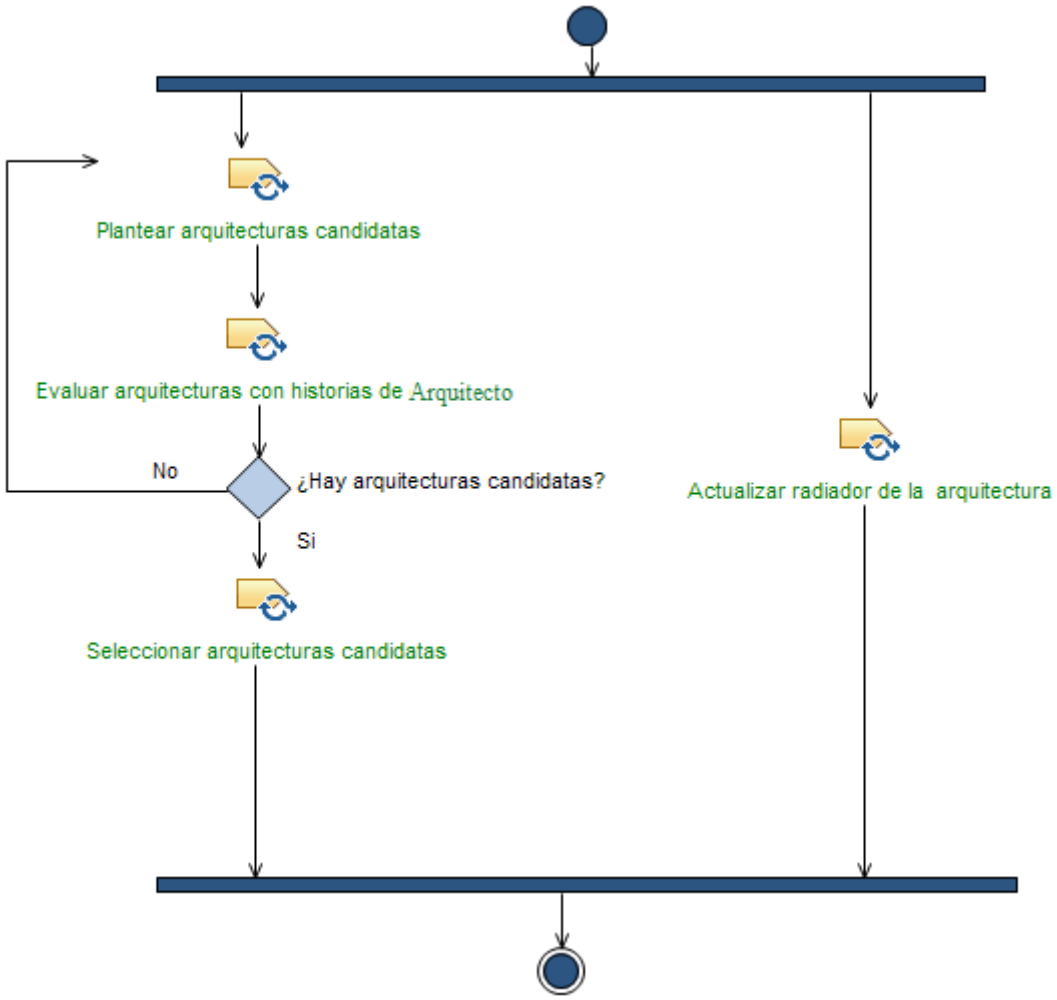

Figura 3.18 Diseñar Arquitectura (Aplica ADD)

- **Plantear Arquitecturas candidatas:** El equipo XA basado en los objetivos y la misión del negocio, en las Historias de arquitecto, y en las historias de usuario relevantes [48], plantea arquitecturas priorizando cuales son las más adecuadas al producto y bajo las restricciones del proyecto. Una arquitectura

candidata también puede surgir de implementaciones iniciales que resulten de iteraciones previas arrojadas por el trabajo preliminar de los equipos XP.

- **Evaluar Arquitecturas con las historias del arquitecto:** en la actividad anterior, se definieron las historias arquitecto los cuales sirven para evaluar las arquitecturas planteadas por el equipo XA, con el objeto de seleccionar las candidatas del sistema, correspondiente a la siguiente tarea. Aquí el equipo XA tendrá en cuenta la descomposición del sistema bajo las arquitecturas propuestas.

- **Seleccionar e instanciar arquitectura candidatas:** El equipo de arquitectura toma como arquitecturas candidatas aquellas de más alta relevancia para el sistema y de mejor impacto para el proyecto, basado en las Historias de arquitecto y las historias de usuario relevantes Para ello se propondrán tácticas, estrategias y patrones de arquitectura que satisfarán los requisitos.. Además se deben generar las vistas básicas para expresar la instanciación de la arquitectura bajo las decisiones tomadas por el equipo XA.

3.8.3 Actividad 3: Implementar Arquitectura

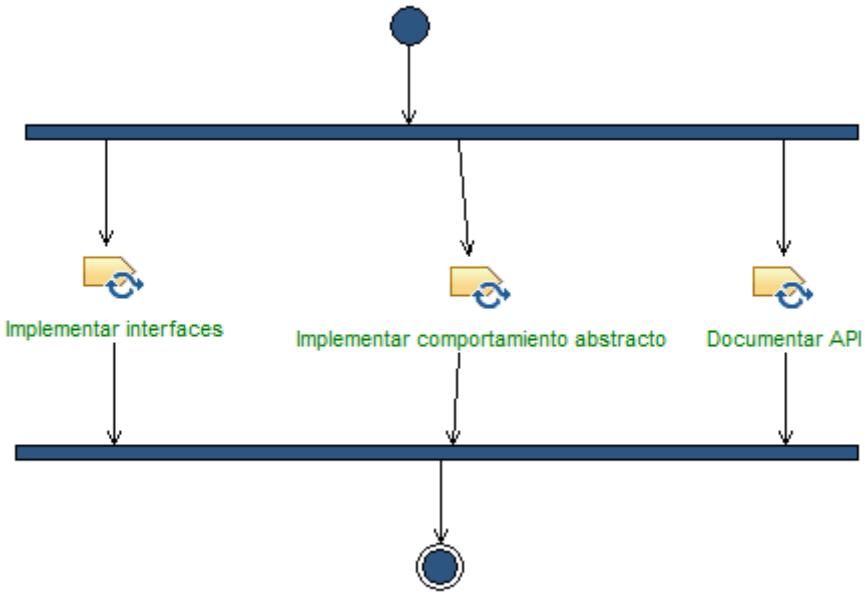

Figura 3.19 Implementar Arquitectura

- **Implementar interfaces**: Una vez desarrollado el concepto de arquitectura, cada equipo XP, llevará a líneas de código sus diseños, de tal forma que muy temprano se pueda mostrar al cliente módulos funcionales.
- **Implementar comportamiento abstracto**: El equipo XA, sincronizará las tareas realizadas por los equipos XP, para ello es necesario tener un repositorio de aplicaciones y de manejo de versiones, con algunas restricciones sobre las partes arquitectónicas.
- **Documentar API**: El equipo XA presentará a los equipos XP, las estrategias y especificaciones necesarias para la documentación de las API. La documentación se fundamenta en los criterios ágiles de XP, y esta actividad se hará al tiempo que el equipo está entregando resultados del sistema.

.

3.8 Pruebas

Para XA, Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Estas actividades podrán ser implementadas en cualquier momento de dicho proceso.

Los equipos XP seguirán las prácticas de pruebas continuas basadas en TDD [45] con base en los criterios del proyecto es decir

- **Elegir un componente a implementar:** se elige de una lista de tareas los componentes a desarrollar que se cree que nos dará mayor conocimiento del problema y que a la vez sea fácilmente implementable [46].
- **Escribir una prueba:** Se comienza escribiendo una prueba para el requerimiento [47].
- **Verificar que la prueba falla:** Si la prueba no falla es porque el requerimiento ya estaba implementado o porque la prueba es errónea.
- **Escribir la implementación:** Escribir el código más sencillo que haga que la prueba funcione. Se usa la metáfora "Déjelo simple" ("Keep It Simple, Stupid" (KISS)) [48].
- **Ejecutar las pruebas automatizadas:** Verificar si todo el conjunto de pruebas funciona correctamente.
- **Refactorización:** Este paso es muy importante para XA puesto que se utilizará principalmente para eliminar código duplicado. Se hacen de a una vez un pequeño cambio y luego se corren las pruebas hasta que funcionen [39] [49].
- **Actualización de la lista de requerimientos:** Se actualiza la lista de requerimientos tachando el requerimiento implementado. Asimismo se agregan requerimientos que se hayan visto como necesarios durante este ciclo y se

agregan requerimientos de diseño (P. ej que una funcionalidad esté desacoplada de otra).

XA mantendrá la coherencia del desarrollo desde la holística del mismo, para ello se fundamentara en los siguientes ejercicios de pruebas:

3.8.1 Prueba de Unidad

La prueba de unidad se centra en probar de forma individual cada uno de los módulos de un sistema. Para ello XA mantendrá un control sobre las interfaces del módulo, la lógica de la aplicación, los atributos de calidad, los escenarios de calidad que llevarían al límite el funcionamiento del mismo, y las historias de arquitecto que producen las iteraciones y las condiciones para asegurarnos de que todos funcionan en su momento, el manejo de errores para asegurarnos de que los mensajes son entendibles y el módulo es resistente a fallos. Es importante durante esta prueba hacer una revisión de los cálculos u operaciones que módulos realiza, con el objetivo de detectar errores de tipo lógico o de tipo matemático, que aunque no harían fallar el sistema, si producirían resultados incorrectos en el corto y mediano plazo.

3.8.2 Prueba de integración

Existen muchas razones por las cuales un grupo de módulos que funcionan correctamente de forma independiente ocasionen problemas al trabajar juntos, entre las que podemos encontrar:

- El acceso a estructuras de datos globales
- La duplicidad en nombres de variables
- La actualización de datos
- La no uniformidad en el diseño de la interfaz

Para ello XA propone aplicar una prueba de integración, desde un enfoque incremental, es decir, ir integrando los módulos poco a poco, en repositorios comunes a los equipos XP y XA con lo cual podemos aislar la detección de errores. Un enfoque no incremental como lo es el enfoque “big bang”, consiste en integrar todos los módulos y una vez juntos probar el sistema esperando la explosión de errores que se puede generar.

3.8.3 Prueba de validación

XA entiende por validación, la verificación del funcionamiento del software de acuerdo a los requerimientos del cliente. El resultado de esta prueba es crucial dado que si el software no cumple con las expectativas del cliente, todo el esfuerzo y trabajo no tendrán ningún significado y aunque pudiera verse como una prueba arbitraria no es así, ya que solamente si no hemos implementado los mecanismos de calidad necesarios en el proceso de desarrollo está prueba no tendría que modificar aspectos estructurales del software sino solamente aspectos superficiales y de interfaces.

3.8 XP/Architecture (XA) en Eclipse Precess Framewok (EPF)

El correcto desarrollo de un producto de software debe seguir ciertas pautas homogeneas que permitan cumplir con los objetivos planteados en las etapas iniciales del desarrollo. Desde las perspectivas ágiles, el proceso debe ajustarse a los tiempos y costos estimados, para obtener los beneficios que se pretende alcanzar. XP/Architecure (XA) propone un proceso de software que busca equilibrar estos parámetros propuestos por Extremme porgramming (XP). Con el fin de lograr una formalziación concreta de la propuesta metodológica, ésta investigación se

enfocó a poder brindar un Plugin de Método del XA en la herramienta Eclipse Process Framework Composer, donde se modela un Proceso que representa las actividades y productos involucrados en XA..

Para desarrollar el Plugin se estudió el metamodelo SPEM y para su implementación se utilizó la herramienta EPF Composer. Dicho desarrollo se complementó con el análisis de los nuevos conceptos de la Ingeniería de Procesos de Software.

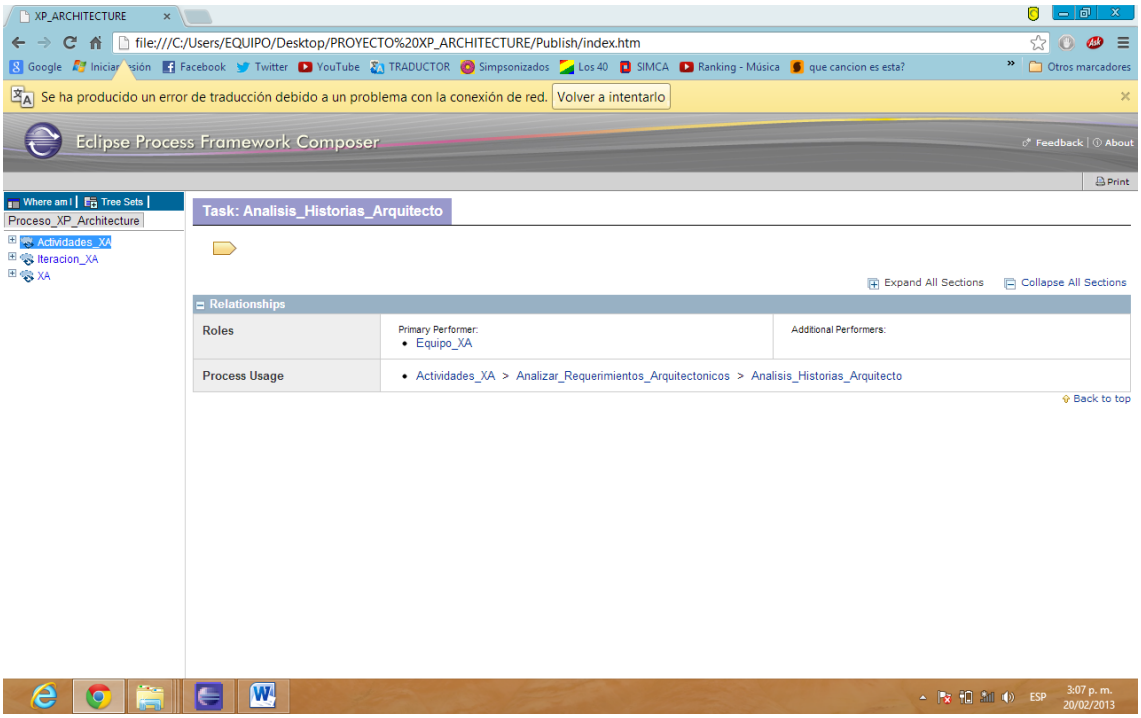

Figura 3.20 XP/ Architecture en Spem 2.0

XP/ARCHITECTURE propone una serie de patrones de procesos con sus respectivos metodos de contenido, que permiten al grupo de arquitectura orientar el desarrollo de la aplicación a partir de las directrices del modelo, ademas se observa de forma grafica cada actividad, iteracion y fases de XA.

XA tiene muy bien definidos sus roles, tareas y productos de trabajo; y crea su

configuración para asignarla a los patrones de proceso que luego en una categoría personalizada, agrupa dichos patrones de proceso. Esto permite obtener gráficamente cada una de las actividades y los artefactos planteados para XP/Architecture

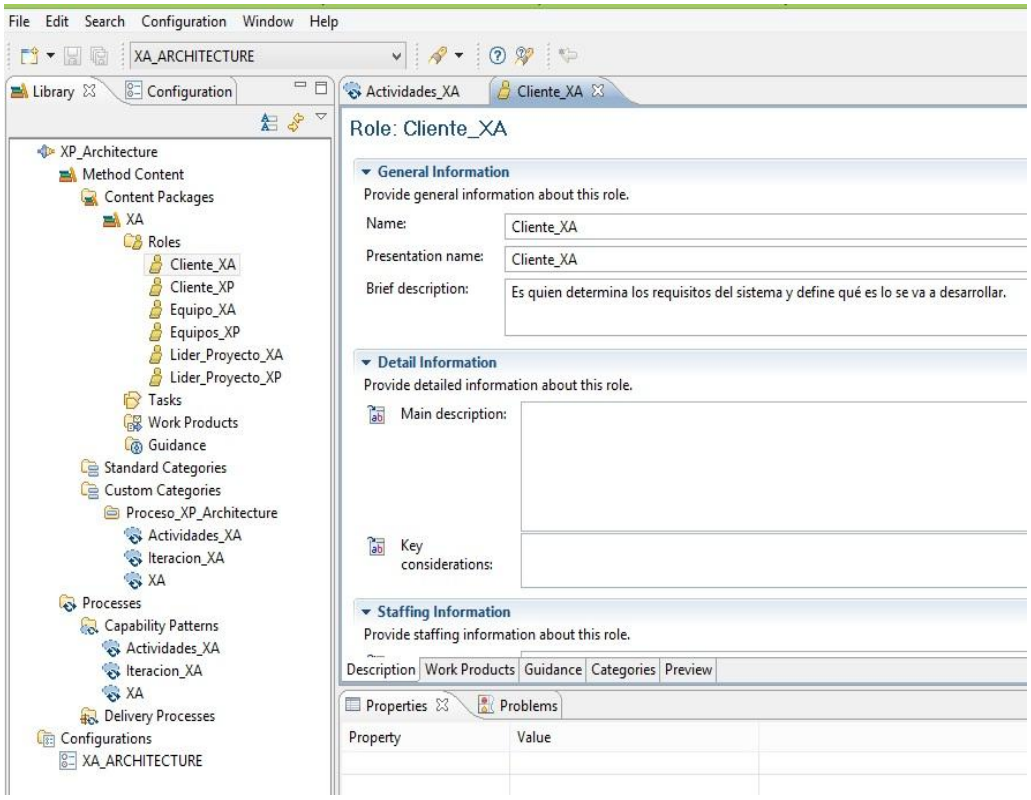

Figura 3.211 Roles Tareas y productos en XP/ Architecture

Se puede notar a través de Eclipse Process Framework Composer(EPFC) que XA publica un proceso de desarrollo que permite concebir y desarrollar un producto software en un ambiente ágil y con un modelo holístico XP concurrente, soportado por prácticas de arquitectura.

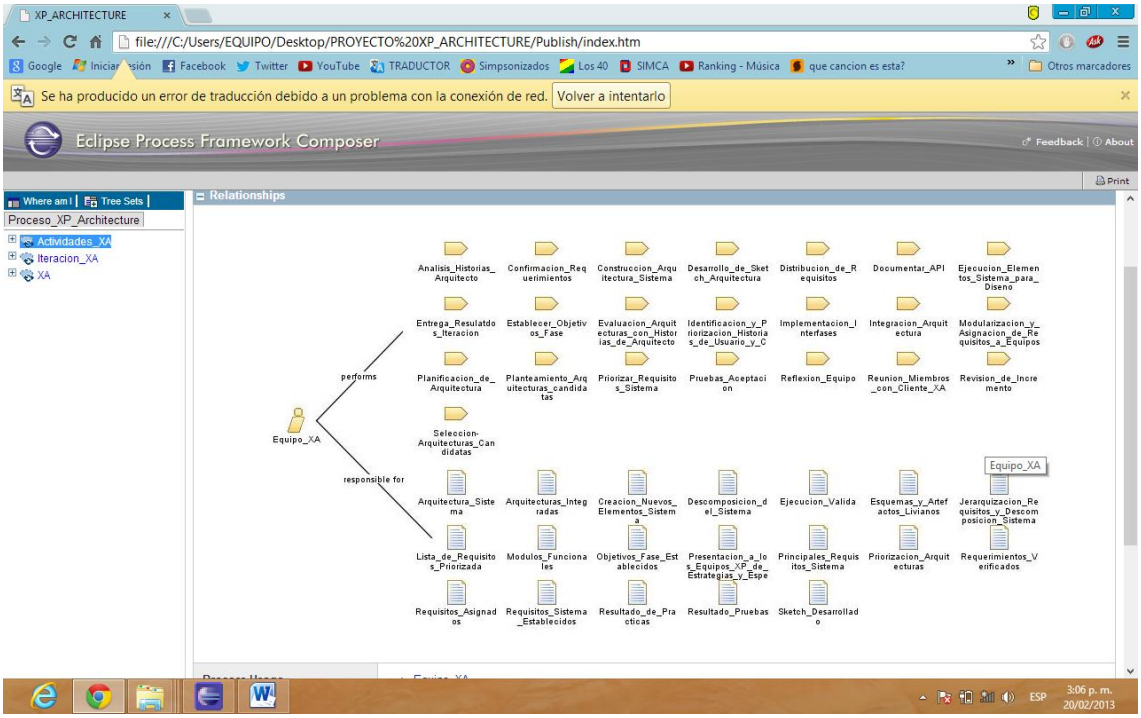

Figura 3.22 Tareas en XA

Las actividades del modelo propuesto (XA) representan una unidad de trabajo general dentro del proceso y mantienen la estructura interna formada por agregación de elementos conformados por las tareas, los productos de trabajo entre otros, tal como se muestran en las figuras del capítulo tres (3) explicando a XA. De ahí que estas actividades representen las entradas y salidas del proceso.

Gracias al uso de SPEM 2.0, se puedo disponer de una manera gráfica y coherente del modelo (XA) y sus parámetros para la orientación del equipo, lo que permitió facilitar la comprensión y comunicación del mismo, facilitar la reutilización, y facilitar la gestión del proyecto a través del modelo planteado.

CAPITULO IV: Evaluación del método XA

En este capítulo se hace una descripción de dos estudios de caso propuestos en los objetivos específicos de este proyecto para la aplicación del modelo (XP/Architecture).

El presente trabajo, presenta una descripción de los estudios de caso referentes a la automatización y seguimiento de los proyectos de investigación gestionados por el Centro de estudios e Investigaciones (CEIN) de la Fundación Universitaria de Popayán, además se explica cual fue el diseño que se trazó para el desarrollo de la aplicación, lo equipos de trabajo y una descripción de los resultados en los dos casos realizados con los diferentes miembros de los equipos de trabajo.

El desarrollo de este caso se llevo a cabo bajo las directrices de Runeson et al [53] y la guía “diseño del caso de estudio V1” desarrollada por el autor de este documento, que se puede encontrar en los anexos de esta investigación.

4.1 Descripción de los estudios de caso:

En esta sección se ilustra cada uno de los componentes que se tuvieron en cuenta para los estudios de caso. Se hace una descripción del contexto donde se desarrollaron los estudios de caso: El centro de estudios e investigaciones (CEIN) de la Fundación Universitaria de Popayán), se hace también una descripción del proyecto que se desarrolló: Un sistema para el Control y Seguimiento de Proyectos

de Investigación (CSPI). Por último la descripción de los equipos que colocaron en práctica el modelo propuesto por esta investigación. Los estudios de caso corresponden al mismo proyecto de desarrollo ejecutado por dos equipos diferentes. Sin embargo, el primer caso de estudio fue usado para validar el diseño del estudio de caso y mejorar sus debilidades y el segundo fue para evaluar el método propuesto. Por eso ambos estudios de caso presentan un mismo diseño.

4.1.1 Descripción del Contexto

La Fundación Universitaria de Popayán es una Institución de Educación Superior de carácter privado, con condiciones de calidad aprobada por el Ministerio de Educación Nacional.

Dentro de su estructura administrativa se encuentra el CENTRO DE ESTUDIOS E INVESTIGACIONES (CEIN), área encargada de gestionar y direccionar los proyectos de investigación desarrollados por las 14 facultades que tiene la Universidad.

El CEIN se encarga de gestionar los proyectos desde varias perspectivas: por un lado, da el aval para desarrollarlos cuando los proyectos realizan en convenio con otras instituciones o son fruto de alguna convocatoria, una vez cumplan con los requisitos de la facultad y que son verificados por el CEIN; por otro, realiza convocatorias internas con el objeto de motivar y apoyar el desarrollo investigativo de la institución tanto para los grupos consolidados y reconocidos por COLCIENCIAS, como para los semilleros de investigación que son grupos jóvenes que desarrollan proyectos generalmente para dar solución a problemas internos o regionales. En estos casos el CEIN aporta los recursos directamente al proyecto y establece condiciones de seguimiento y cumplimientos de entrega a los grupos para verificar la inversión económica dada al proyecto [52]. Para ello ha planteado varias estrategias

entre ellas y la que es de interés a ésta investigación es la sistematización y seguimiento de los proyectos de investigación del CEIN.

4.1.3 El equipo Trabajo

Para el desarrollo del caso, se realizaron dos proyectos para lo cual se escogieron los estudiantes matriculados al curso electivo profesional en ingeniería del software, de quinto año del programa de Ingeniería de Sistemas de la Fundación Universitaria de Popayán. Un total de 19 estudiantes. Con quienes se organizaron los equipos XP y XA.

La mayoría de los estudiantes tenían algún conocimiento de las herramientas con la que se desarrollaría la aplicación propuesta para el proyecto. No obstante, vale la pena comentar que Algunos de ellos demostraron que tenían más experiencia en las herramientas (30%) usadas en el ejercicio y en el desarrollo con metodologías ágiles (20%). Esto hizo que la propagación de conocimiento fortaleciera al equipo. Hay que anotar también que la familiaridad de los estudiantes era bastante alta debido a que se conocían ya desde hace más de cuatro años, esto permitió durante el proyecto establecer una muy buena comunicación, además porque debido a su transitar por la carrera, aprendieron los mismos estilos de programación y uso de código, lo que permitió la unificación del mismo en el momento del desarrollo.

Debido a la familiaridad de los equipos de trabajo, al conocimiento de sus capacidades de programación y de la metodología propuesta para el trabajo. La distribución de los equipos y del proyecto entre los equipos no fue difícil, esto fluyo entre los miembros de cada equipo es decir, voluntariamente cada quien según sus afinidades se reunieron y escogieron la parte del proyecto que mas les interesó.

Los anteriores fueron factores que favorecieron el desarrollo del proyecto y que lo fortalecieron al momento de aplicar el modelo propuesto por XP/Architecture

4.2. Diseño del caso de estudio

En esta sección, se describe el diseño del caso de estudio, aclarando que el método propuesto se ha aplicado en un proyecto académico, manteniendo las directrices de la investigación empírica donde se estudia un fenómeno dentro de su contexto real [53]. Para ello se definió una hipótesis con sus respectivos indicadores de validación, y con el fin de conocer sus valores, se definió un marco de medición y un conjunto de instrumentos de recolección de información.

4.2.1 Diseño de los Estudios de Caso

Para desarrollar esta investigación, se siguió la propuesta de Runenson y Host [53], para ello se partió de la pregunta de investigación principal que se intenta resolver: ¿Como escalar un proceso ágil en proyectos de mediana complejidad y equipos de desarrollo medianos? Junto con la revisión de la literatura científica relacionada, se definió como hipótesis: XA mantendrá en el caso de estudio el orden de los índices de productividad y satisfacción reportados para XP, pero en proyectos de mediana complejidad y con equipos de más de 10±2 personas. Para soportar esta investigación se definieron los indicadores, métricas e instrumentos de recolección de datos como se presentan en la Tabla 3.

Tabla 4.3 INDICADORES, METRICAS, FUENTES DE INFORMACION E INSTRUMENTOS

INDICADOR	DESCRIPCION DEL INDICADOR	METRICAS INDIRECTAS	FUENTE DE INFORMACIÓN	INSTRUMENTO
Satisfacción(S)	Este indicador permite medir el grado de aceptación de XP/XA para los distintos participantes del proyecto.	Nivel Aceptación del Método, Nivel de Participación del Cliente	La informaciòn se obtendra de una encuesta realizada al cliente, los equipos XP y el equipo XA	ENCUESTA
Calidad (C)	Este indicador mide la capacidad del software para proporcionar resultados correctos basados en los requisitos de usuarios.	Nivel de conformidad del Cliente con el Producto (CCP)	Cliente	ENCUESTA
		Satisfacción de los Requisitos (NSR)	Producto Software	REGISTRO DE DEFECTOS DEL PRODUCTO ASOCIADOS A LOS REQUERIMIENTOS
Productividad del equipo (PE)	Este Indicador mide la capacidad de los equipos XP para conducir el desarrollo de software en una forma eficiente.	Productividad (P)	Gráfico de avance del proyecto	Burndown Chart

En este caso de estudio se usaron y diseñaron los siguientes instrumentos:

Encuesta al Cliente: Con el fin de indagar el grado de participación del cliente y su compromiso respecto al proyecto y el equipo XA.

Encuesta a equipo XP: Para conocer el grado de aceptación del modelo propuesto entre los miembros del equipo.

Encuesta a Equipo XA: Con el objeto de saber el grado de receptividad del modelo y su dinamismo frente al proyecto y los otros actores.

Encuesta de conformidad: Al cliente para conocer bajo criterios de calidad el resultado del proyecto frente a los requisitos

Lista de chequeo: Al producto.

Planilla de Defectos: Donde se hará un registro general de defectos del producto a medida que este se va desarrollando

Evaluación de la Arquitectura: Con el fin de conocer la flexibilidad de la arquitectura propuesta.

Los diseños de éstos artefactos se encuentran en los Anexos ”Resultados de la validación” descritos al final de éste documento.

4.3 Caso Preliminar:

XA fue aplicado a un grupo de 18 estudiantes de quinto año del programa de Ingeniería de sistemas de la Fundación Universitaria de Popayán, en el marco de un laboratorio de ingeniería de software, donde se propone el desarrollo de una aplicación de gran escala, ver Figura 4.1. El proyecto a desarrollar consistió en una aplicación distribuida para la inscripción y seguimiento de proyectos de investigación.

El desarrollo se realizo con tecnología JEE, Web Services y Junit. El cliente fue un profesor de la institución designado por el comité curricular.

Ilustración 4.1 Estudiantes primer estudio de caso

4.3.1 Desarrollo del caso:

Para la aplicación de XA, los estudiantes recibieron una capacitación inicial de 12 horas dividida en tres partes. La primera parte abordó la temática de las metodologías agiles (XP), con el objeto unificar términos de referencia. En la segunda parte se abordó el tema de arquitectura, particularmente en lo que respecta sus conceptos, vistas, estilos y a los métodos ADD y QAW.

En la tercera parte, se les presentó XA, dando a conocer al grupo los parámetros e instrumentos que se llevarían a cabo para el proyecto del curso. Para respaldar la capacitación y el desarrollo del proyecto se les entregó dos reportes técnicos desarrollados específicamente para introducir XP y XA. El proyecto se desarrolló en clases con sesiones de 4 horas cubriendo un total de 48 horas de desarrollo. La Figura 4.2

Ilustración 4.2 Recibiendo capacitación

Los estudiantes se auto-organizaron en 3 equipos XP de 6 integrantes, cada equipo escogió su representante cliente/arquitecto ante XA. El equipo XA fue organizado por los tres representantes de los equipos XP. En la dinámica del proyecto, mientras XA se reunió con el cliente del proyecto, los equipos XP organizaron sus radiadores de información (carteleras) en un sitio visible para todos. En todas las sesiones (12) de 4 horas cada una, se desarrolló al comienzo una reunión de 15 minutos entre el cliente y el equipo XA para recibir retroalimentación del cliente debido a que no se contaría con su participación presencial por el resto de la sesión. Sin embargo el cliente tuvo un compromiso muy alto, estuvo disponible dentro de la institución y participó activamente en el desarrollo de los requisitos.

Ilustración 4.3 Equipos organizados en parejas

Además los equipos XA y XP siguieron las prácticas de planificación de iteración y del trabajo diario de acuerdo a lo establecido por XP y XA. Durante el desarrollo del proyecto se recolectó la información de acuerdo a lo planificado. La dinámica corresponde al modelo planteado, puesto que se notó durante el desarrollo del caso un compromiso por cada uno de los miembros del equipo al seguir prácticas propuestos por el modelo XA, lográndose una adherencia del 82%, el cual puede considerarse aceptable para poder evaluar los resultados del caso de estudio.

4.3.2 Resultados del caso de estudio:

La Tabla II muestra los principales resultados obtenidos del caso de estudio; Para ello, se tuvo en cuenta varios aspectos en la medición. Con el objeto de medir la aceptación del modelo de la investigación por parte del cliente y del equipo de trabajo, se realizaron unas encuestas realizadas después de terminada cada actividad y que se anexan al presente trabajo.

Para medir los niveles de conformidad tanto en los equipos XP como de los requisitos del cliente, se realizaron encuestas cuyos resultados se exponen en la tabla III, además, se anexa una lista de defectos en cada iteración que permitieron también medir la productibilidad de los equipos XP y XA respecto de la aplicación [54]. Debido a que XA tenía que entregar Historias de arquitecto a los equipos XP, fue necesario realizar una tabla de medición para conocer el grado de satisfacción respecto a la arquitectura presentada. (Estos resultados encuestas y mediciones se presentan en el anexo "Resultados de la validación" al finalizar la presente investigación)

De ahí que se concluye que XA fue entendido en un 90% de los participantes y tuvo una aceptación del 100%. Los equipos XA comentaron muy bien las prácticas dispuestas por el modelo XA para la obtención del producto.

Tabla 4.4 RESULTADOS PRIMER CASO DE ESTUDIO

Indicador	Valor
Nivel de Aceptación de XA	100%
Nivel de participación del Cliente	95%
Nivel de conformidad del Producto	70%
Satisfacción de los requisitos	95%
Historias de Usuario	9
Historias de arquitecto	3
Productividad de los equipos XP	0.01 HU-P-H
Productividad del equipo XA	0.007 HA-P-H
Adherencia al método XA	82%

Aunque al principio se notaron ciertas dudas para el lanzamiento del proyecto debido a la incertidumbre de la dinámica del modelo y a lo novedoso para el grupo. Una de las fortalezas encontradas en todos los equipos fue el flujo de la comunicación.

Para calcular la productividad de los equipos XP se siguieron los parámetros propuestos por [55] donde

$$P = \frac{HU}{E}$$

Donde P = Productividad, HU= Historias de Usuario y E= esfuerzo

Por otro lado la Productividad de XA se calculo

$$P = \frac{HA}{E}$$

Donde P = Productividad, HA= Historias de arquitecto y E= esfuerzo

El esfuerzo se calculo multiplicando el número de horas de trabajo por el número de participantes en el proyecto

La productividad de los equipos XP y XA fueron de 0,01 y 0,007 H-P-H respectivamente, para este primer caso de estudio, lo cual es notablemente mejor a lo reportado por [54] donde la productividad fue de 0,003 historias de usuario persona hora (32 historias de usuario, en 3.5 meses por un equipo de 14.7 personas). Por otro lado de acuerdo a [55], la productividad de un equipo XP es de 17 NLOC-P-H1 fue notablemente mejor que un proyecto desarrollado en forma tradicional (10.3 NLOC persona-hora). En este estudio de caso preliminar, se desarrollaron 12600 NLOC alcanzando una productividad de 14.58 NLOC, menor a la reportada por [55] aplicando XP pero mejor que la reportada usando el enfoque tradicional.

Adicionalmente, el obtener resultados comparables con otros reportes permitió confirmar la calidad del diseño del caso y mejorar la calidad de los instrumentos utilizados.

Estos resultados permiten evaluar preliminarmente la hipótesis de que la productividad de los equipos XA mantendría el orden de productividad de los equipos XP reportados por la literatura.

4.4. Estudio de Caso Final

Igual que para el primer estudio de caso, se siguieron las mismas directrices para esta segunda experiencia en cuanto a la organización del equipo y la descripción del proyecto. Cabe anotar, que debido a los requerimientos del cliente, se hizo mas complejo el desarrollo de la aplicación, pues en esta caso, era tarea del equipo además de desarrollar los requerimientos anteriormente descritos, realizar también el seguimiento financiero de los dineros otorgados por el Centro de estudios e Investigaciones de la Universidad (CEIN) o por alguna entidad externa para financiar las investigaciones de los grupos o semilleros de investigación. Lo que llevo al equipo XA a plantear otras Historias de arquitecto y a los equipos XP a desarrollar mas Historias de usuario.

4.4.1 Contexto del Estudio de Caso 2:

Para este segundo caso, se aplicó XA a un grupo de 18 estudiantes quienes se caracterizaron porque el 60% de ellos habían culminado materias y realizaban con la Institución un diplomado en desarrollo de software y el 40% restante egresados de programas de Ingeniería de Sistemas de diferentes Universidades de la ciudad

interesados en el tema y en la propuesta de trabajo para el desarrollo del proyecto. A este grupo, se les propone también el desarrollo de una aplicación de gran escala. El proyecto a desarrollar igual que con el primer grupo consistió en una aplicación distribuida para la inscripción y seguimiento de proyectos de investigación. El desarrollo lo realizaron con tecnologías PHP y la librería JQUERY-UI, para la base de datos MySQL y como servidor web Apache. Nuevamente para este caso, el cliente fue un profesor designado por el comité curricular, representante del programa de Ingeniería de Sistemas ante el Centro de estudios e Investigaciones y quien conocía la dinámica del proyecto y los requisitos para el desarrollo de la aplicación.

4.4.2 Desarrollo del caso:

Como la mayoría de los participantes a diferencia del primer grupo, eran bastante heterogéneos debido a que no conocían la metodología para el proyecto y en el manejo de las herramientas propuestas para el desarrollo del mismo presentaba el grupo diferentes habilidades, entonces inicialmente, los estudiantes recibieron una capacitación 20 horas divididas en cuatro partes. La temática para la primera parte consistió en una charla sobre metodologías agiles centrada en Extreme Programming (XP), con el objeto unificar términos de referencia.

Para la segunda parte se trabajaron temas de arquitectura puesto que se necesitaba que el grupo conociera particularmente los conceptos, vistas, estilos y métodos propuesto por el SEI (Software Engineering Institute) pero entre ellos se enfatizó en ADD (Attribute-Driven Design) y QAW (Quality Attribute Workshop).

En la tercera parte, se presenta XA (XP/Architecture), explicando al grupo, las condiciones, los parámetros e instrumentos que se llevarían a cabo para la aplicación del método propuesto para el desarrollo de la aplicación y la dinámica del mismo.

Luego, para respaldar la capacitación y el desarrollo del proyecto se les entregó dos reportes técnicos desarrollados específicamente para introducir XP y XA.

El proyecto se desarrolló durante las sesiones de clase con una intensidad de 4 horas. El proyecto tuvo un total de 54 horas de desarrollo.

Ilustración 4.4 Un momento en la capacitación

Debido a la dinámica del modelo propuesto y del proyecto, el grupo se auto-organizaron en 3 equipos XP de 6 integrantes, la mayoría de los integrantes desconocían entre si las capacidades técnicas de cada uno, por lo que ellos mismos a través de preguntas sueltas y en voz alta indagaron esas habilidades y afinidad con respecto al desarrollo de proyectos informáticos. Una vez organizados los equipos, cada equipo escogió su representante cliente/arquitecto ante XP/Architecture (XA). El

equipo XA quedó entonces conformado por los tres representantes de los equipos XP.

4.4.3 Las Prácticas de XA

Ilustración 4.5 Reunión XA

El equipo demostró adherencia al método propuesto, debido a que tuvo en cuenta aplicar cada una de sus prácticas. Los miembros del equipo iniciaron con una reunión involucrando al cliente del proyecto. En esta reunión, se establecieron los requisitos del sistema y se analizaron los variables del proyecto: Costo, tiempo, calidad y alcance. Mientras tanto, los miembros de los equipos XP empezaron a organizar los elementos necesarios para aplicar XP al módulo que deberían desarrollar. Lo primero que hicieron fue colocar en un lugar visible para todos sus radiadores de información (carteleras). Cada que se inició una sesión (programada para cuatro horas) se realizó la reunión diaria de XP, con el de objetivo de mantener la comunicación entre el equipo, y a su vez con el equipo XA, para compartir problemas y soluciones y recibir sugerencias del cliente, ya que él no estaría en el resto de la sesión. Se sugirió realizar estas reuniones en círculo y de pie. Es

importante comentar que el cliente demostró un compromiso bastante alto con el proyecto, estuvo disponible dentro de la institución y participó activamente en el desarrollo de los requisitos del proyecto.

Ilustración 4.6 Grupos auto-organizándose

Las prácticas de XA fueron bastante notables en proyecto es así por ejemplo que para:

La gestión ágil, cada equipo XA y XP seleccionó un líder de proyecto y lo hicieron con bases en sus habilidades y conocimiento de las herramientas para el desarrollo del proyecto. Ellos gestionaron de forma ágil el proyecto. Para la integración de las prácticas de gestión el líder del proyecto XA se reunió periódicamente (reunión diaria de gestión) con los líderes de cada proyecto XP para abordar los problemas y enfoques de gestión del proyecto.

Al igual que con los equipos XP, el equipo XA organizó su propiedad colectiva de código, para ello aprovechó las herramientas del sistema organizado un repositorio al que podían acceder los líderes de los equipos XP así mismo de guardar las versiones y/o iteraciones de la aplicación. Pero con el objeto de hacer menos complejo el desarrollo y cumplir con los criterios ágiles en cada equipo XP se consideró también esta práctica organizando una propiedad colectiva de código del sub-sistema. XA se apoyó de prácticas de arquitectura para la asignación de subsistemas, distribución de los requisitos y posterior validación e integración. XA pude verse como un equipo ágil que subcontrata el desarrollo de componentes con equipos XP.

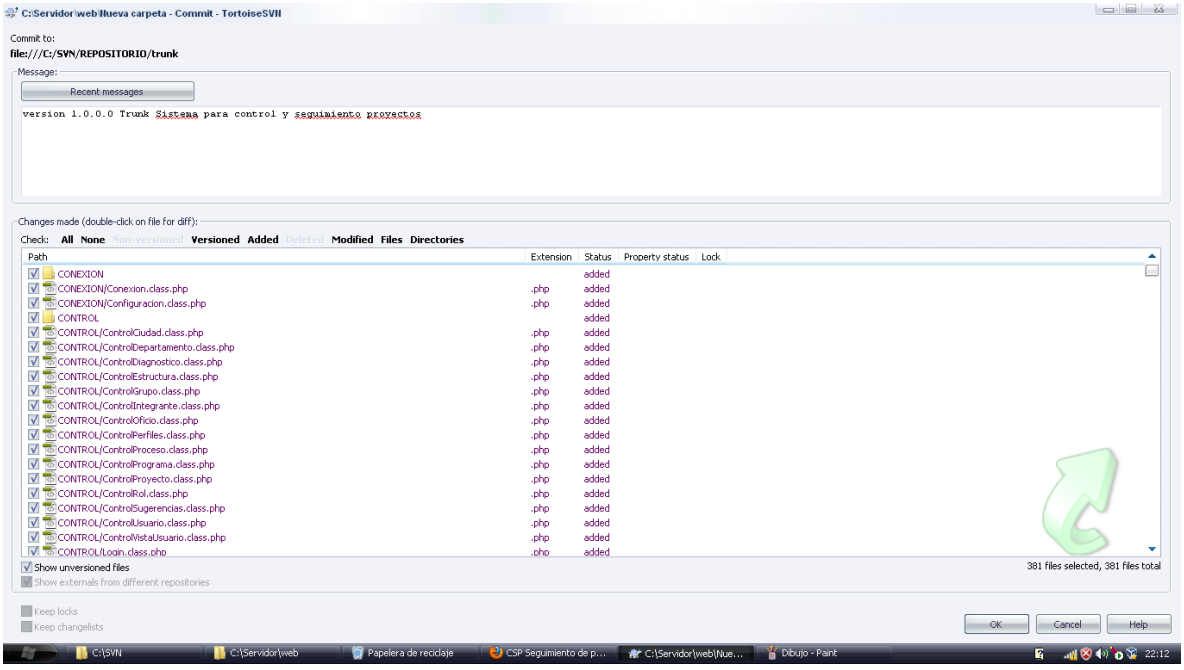

Ilustración 4.7 Repositorio de versiones

Se establecieron también los mecanismos para determinar requisitos. Una vez establecido un incremento en la arquitectura, por el equipo XA, se les compartió a los equipos XP los requerimientos del subsistema elegidos por su Cliente XA, aprovechando la iteración de un miembro XA como cliente del equipo XP.

Los equipos XP durante la planificación mantuvieron un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente XP que en este caso fue el miembro del equipo representante ante XA. El proyecto comenzó recopilando las "Historias de Usuario", del módulo que libremente había escogido cada equipo XP, con base en su experiencia y capacidades de desarrollo. Este permitió la propagación del conocimiento y con ello la unificación de criterios en cuanto a la codificación.

Proceso php ajax jquery para procesar la administración de usuarios

```php
<?php
require_once('../CONTROL/ControlVistaUsuario.class.php');
$ctrusuario = new ControlVistaUsuario();
?>

<?php
    if(isset($_POST['pos']))
    {
        $res = $ctrusuario->CallDatosUsuario($_POST['pos']);

        if ($ctrusuario->CantidadRegistros($res) != 0)
        {
            while($reg = $ctrusuario->Row($res))
            {
               $arraypersona = array('cedula'=>$reg[0], 'nombres'=> $reg[1], 'apellidos'=> $reg[2], 'rol'=>utf8_encode($reg[3]), 'estado'=>$reg[4],'email'=>$reg[5]);
               $listaPersona[] = $arraypersona;
            }
        }else
        {
             $arraypersona = array('cedula'=>'Registro no existe', 'nombres'=>'Registro no existe', 'apellidos'=>'Registro no existe', 'rol'=>'Registro no existe', 'estado'=>
'Registro no existe','email'=>'Registro no existe');
             $listaPersona[] = $arraypersona;
        }
        echo json_encode($listaPersona);
    }?>

<?php
    if(isset($_POST['poscedula']))
    {
        $res = $ctrusuario->CallUsuario($_POST['poscedula']);

        if ($ctrusuario->CantidadRegistros($res) != 0)
        {
            while($reg = $ctrusuario->Row($res))
            {
               $arraypersona = array('cedula'=>$reg[0], 'nombreuno'=> $reg[1], 'nombredos'=> $reg[2], 'apellidouno'=> $reg[3], 'apellidodos'=> $reg[4], 'telefono'=> $reg[5],
```

Ilustración 4.8 Estilo de código unificado de los equipos

Una vez obtenidas las "historias de usuario", los equipos XP evaluaron rápidamente el tiempo de desarrollo de cada una, además se tuvo en cuenta de especificar el

detalle mínimo en cada historia como para que los programadores realizaran una estimación poco riesgosa del tiempo que llevará su desarrollo. Si alguna de ellas tiene "riesgos" que no permiten establecer con certeza la complejidad del desarrollo, se realizaron pequeños programas de prueba (*"spikes"*), para reducir estos riesgos. Una vez realizadas estas estimaciones, se organizó una reunión de planificación, con los diversos actores del proyecto, para establecer un plan o cronograma de entregas (*"Release Plan"*) en los que todos estuvieron de acuerdo.

Una vez acordado este cronograma, mientras los equipos XP iniciaban el desarrollo (más exploratorio), el equipo XA empezó la fase de construcción con arquitectura, en cada una de éstas se desarrolló, probó e instaló en un repositorio dispuesto y administrado por el equipo XA al finalizar cada jornada. Los equipos XA realizaron cada una de las etapas propuestas por el modelo:

En cuanto a la primera etapa la de Exploración: se reunieron los miembros del equipo XA, con el cliente XA para establecer una aproximación inicial a los requisitos del sistema. Para la planificación de la arquitectura, se reunieron los miembros del equipo XA, y se establecieron los objetivos de la fase, y se definieron las "Historias de Arquitecto". En cuanto a la construcción con arquitectura, se priorizaron y refinaron las historias de arquitecto y de usuario, y se desarrolló un sketch de la arquitectura planteada por el equipo de arquitectura para esos requisitos, se modularizaron y se distribuyeron los requisitos a los equipos XP. La definición de arquitectura se desarrolló paralelo al desarrollo de los equipos XP. Continuamente se fue integrando el sistema.

En esta fase el equipo XA aplicó los métodos de arquitectura ADD y QAW para ello, los requerimientos funcionales se definieron mediante Historias de Usuario y las Historias de Arquitectura se explicaron mediante escenarios de calidad que se exponen en el radiador del equipo XA los cuales se concertaron en el ejercicio de la

realización del producto entre el equipo de arquitectura y el cliente. Para el cumplimiento de ADD el equipo XA realizó las siguientes actividades:

- Los requerimientos (Historias de Arquitectura e Historias de Usuario) fueron refinados basándose en los objetivos y la misión del negocio, en atributos de calidad, restricciones de diseño y requerimientos funcionales. La priorización dio una mayor importancia para el cliente y de alto impacto para la arquitectura.
- Se propuso una solución arquitectónica candidata y se publicó para los equipos XP, quienes tomaron cada quine lo que les correspondía basados en la misión de su proyecto.
- Se eligieron los patrones y las estrategias más adecuadas que satisficieron los elementos de arquitectura propuestos para el diseño de los elementos de alto impacto seleccionados.
- A partir de estos patrones, se instanciaron y trazaron estrategias para crear los nuevos elementos del sistema. Se liberó ésta nueva versión de la arquitectura a los equipos XP.
- Se esbozaron las interfaces de los elementos instanciados para mostrar el flujo de información. Las interfaces fueron implementadas y liberadas a los equipos XP.
- Por último se verificaron y refinaron los requerimientos y se realizaron las restricciones para los elementos instanciados. Estas restricciones se convirtieron en el objeto de versiones futuras.

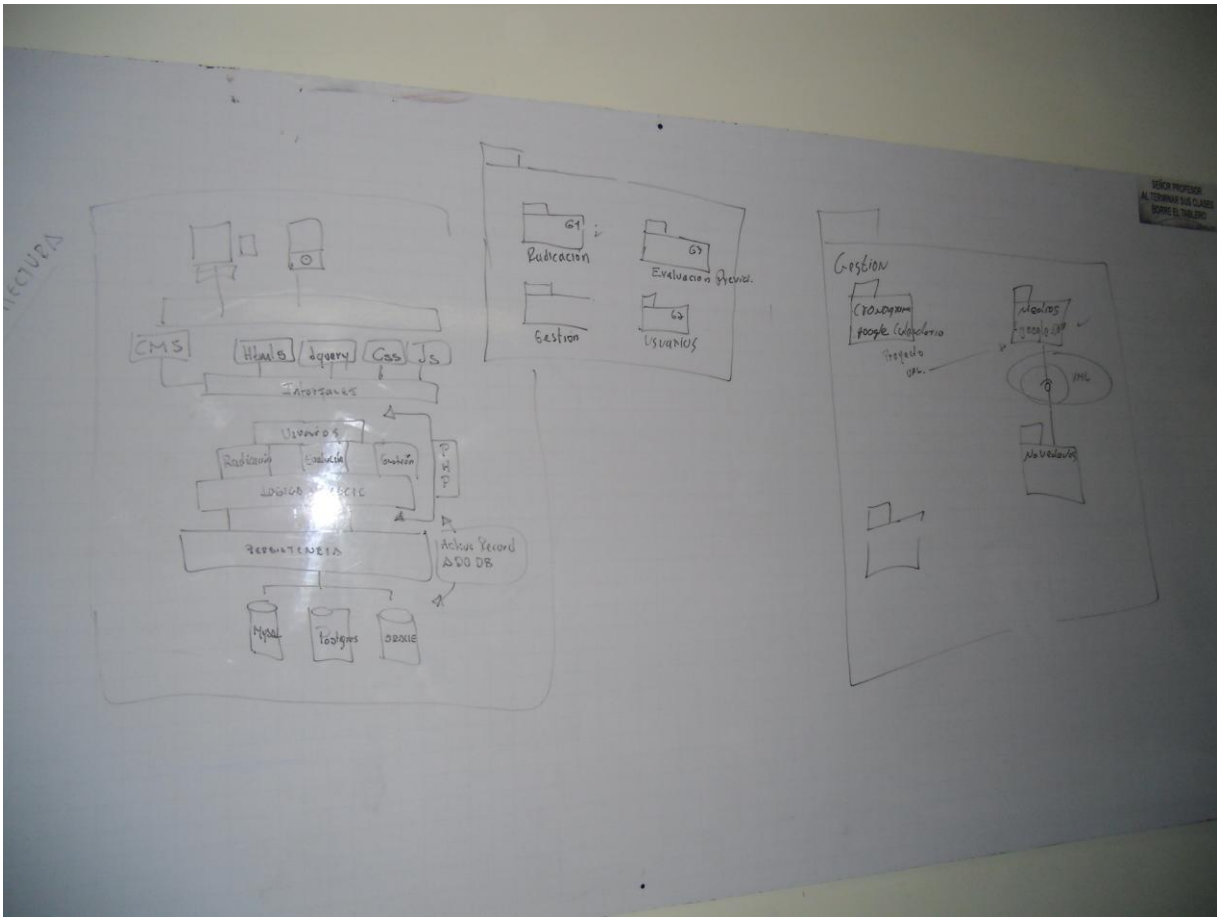

Ilustración 4.9 Aplicando ADD

En cuanto a QAW, ayudó a los arquitectos del sistema (el equipo XA) a obtener la máxima información posible sobre la calidad y a definir la estructura del sistema; QAW se usó para genera, priorizar y refinar escenarios de atributos (Historias de arquitecto) antes que la arquitectura del software fuera completada. Para ello a través de QAW se propuso un plan de arquitectura basado en escenarios donde se conjugaron los requerimientos del sistema, los objetivos del negocio, el equipo de trabajo y el cliente.

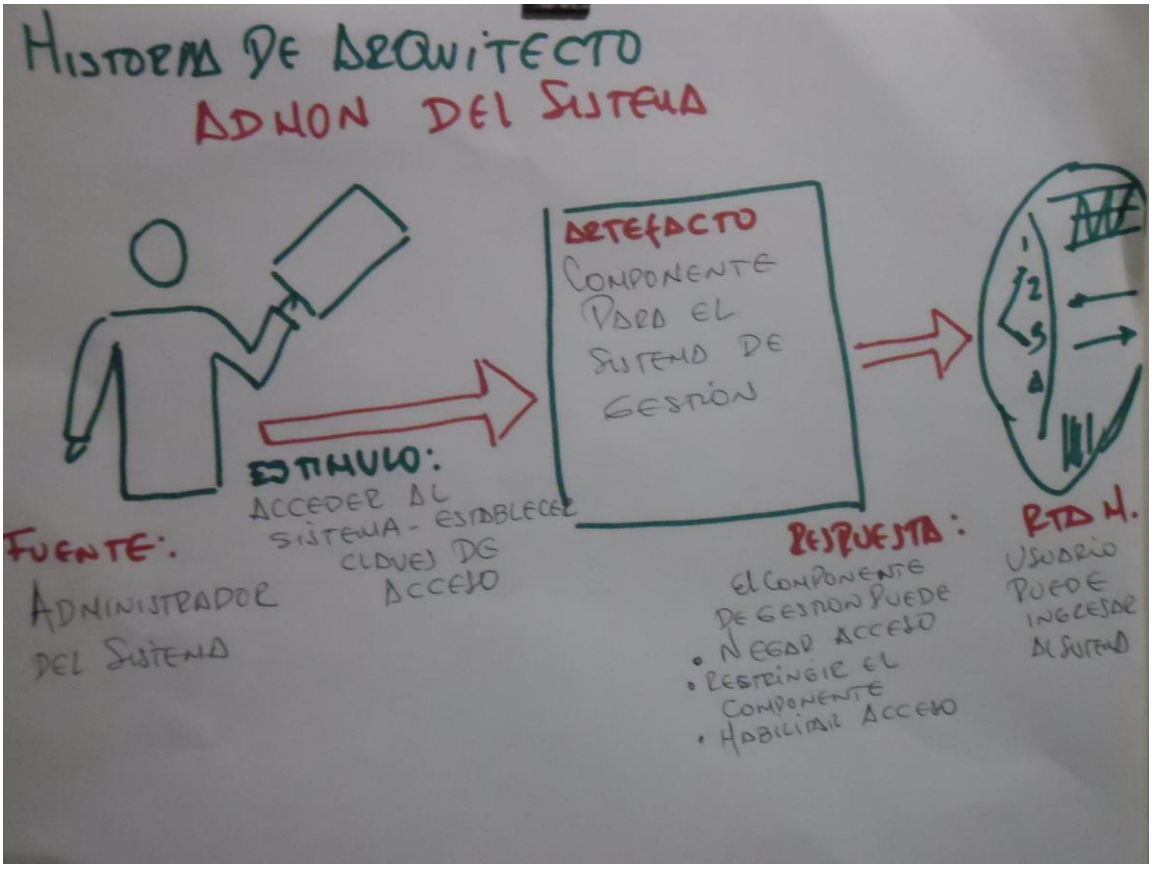

Ilustración 4.10 Historias de arquitecto (QAW)

La última fase de XA, **Entrega Final** del proyecto, el equipo XA presentó al cliente los resultados de cada iteración, quién validó lo ejecutado. Tanto el cliente como el equipo XA firmaron un acuerdo al validar los resultados de la iteración y sirvió de requisito para el paso siguiente del ciclo, hasta que maduró el proyecto.

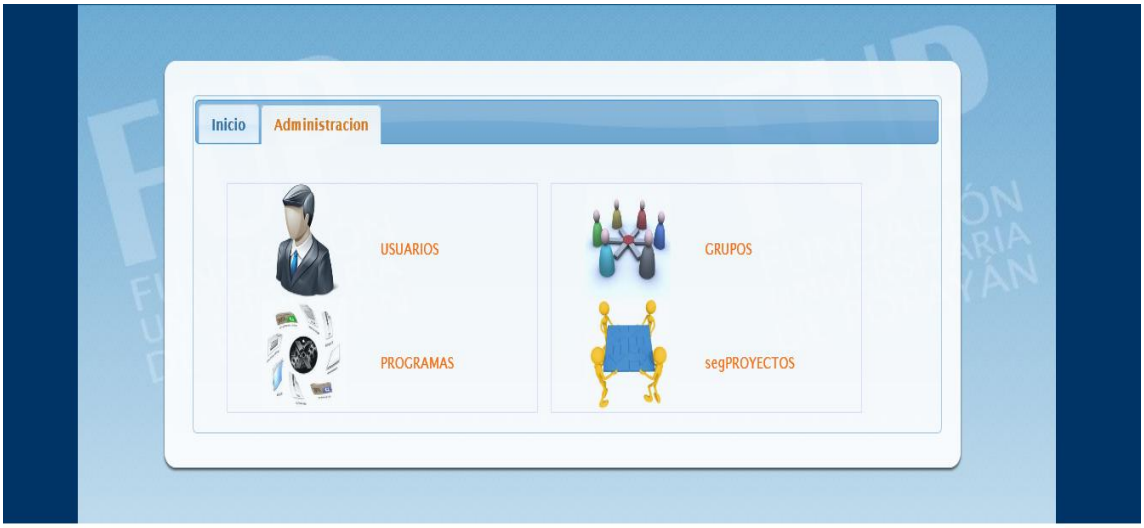

Ilustración 4.11 Interfaz final administrador

4.4.4 Resultados del Caso de Estudio Final:

La Tabla 5 muestra los principales resultados obtenidos en este segundo caso de estudio; Se pudo comprobar que XA fue entendido en un 95% de los participantes y tuvo una aceptación del 100%. Los equipos XA comentaron muy bien las prácticas dispuestas por el modelo XA para la obtención del producto.

Tabla 4.5 RESULTADOS SEGUNDO CASO DE ESTUDIO

Indicador	Valor
Nivel de Aceptación de XA	100%
Nivel de participación del Cliente	97%
Nivel de conformidad del Producto	87%
Satisfacción de los requisitos	97%
Historias de Usuario	14
Historias de Arquitecto	3
Productividad de los equipos XP	0.017 HU-P-H
Productividad del equipo XA	0.018 HA-P-H
Adherencia al método XA	93%

4.4.5 Análisis de Resultados del Caso de Estudio Final:

En este segundo caso, se notó una mayor experiencia de los equipos de trabajo en cuanto a desarrollo de software, aunque al principio debido a la dinámica del modelo hubo algunas dudas, pero que se resolvieron con los seminarios de capacitación que se les brindó a todos los miembros que participaron en el proyecto.

A diferencia del primer caso de estudio, en este segundo ejercicio los participantes no se conocían muy bien, aspecto que favoreció en cierta media el trabajo del proyecto para la aplicación del modelo debido a que la comunicación debió incrementarse desde la presentación de cada uno de los miembros del proyecto y hasta conocer sus habilidades para la solución del mismo, lo que llevó la auto conformación de los equipos y a crear una dinámica de trabajo motivada por ellos mismos.

Al igual que en el primer caso, para medir la productividad de los equipos XP y XA, siguieron los indicadores propuestos en la Tabla I y se evaluó con diferentes encuestas realizadas al cliente, a los equipos y al mismo software. Con base en estas mediciones se obtuvieron los siguientes resultados:

Basados en las fórmulas anotadas para el primer estudio de caso, la productividad de los equipos XP fue de 0,017 H-P-H y la del equipo XA fue de 0,018 H-P-H, nuevamente se puede analizar que los resultados son notablemente mejor a lo reportado por [54] donde la productividad que fue de 0,03 historias de usuario persona hora (32 historias de usuario, en 3.5 meses por un equipo de 14.7 personas). Además, por otro lado de acuerdo a [55], la productividad de un equipo XP es de 19 NLOC-P-H fue notablemente mejor que un proyecto desarrollado en

forma tradicional (10.3 NLOC P-H). En este caso de estudio, se desarrollaron 19600 NLOC alcanzando una productividad de 20.16 NLOC, menor a la reportada por [55] aplicando XP pero mejor que la reportada usando el enfoque tradicional.

A partir de los dos estudios de caso es posible afirmar que en el contexto de un curso de ingeniería de software, que la productividad de los equipos XA mantiene el orden de productividad de los equipos XP reportados por la literatura, lo cual permite afirmar que XA permite escalar XP en ese mismo contexto.

4.4.6 Resultados Cualitativos del proyecto

La alta familiaridad de los estudiantes permitió durante el proyecto establecer una muy buena comunicación, además, lo que ayudó a la unificación del mismo en el momento del desarrollo, la propagación de conocimiento y el fortalecimiento del equipo. La distribución de los equipos y del proyecto entre los equipos no fue difícil, esto fluyó entre los miembros de cada equipo es decir, voluntariamente cada quien, según sus afinidades se reunieron y escogieron la parte del proyecto que más les interesó. Los anteriores fueron factores que favorecieron el desarrollo del proyecto y que lo fortalecieron al momento de aplicar el modelo propuesto por XP/Architecture. al evaluar los equipos XP como el equipo XA, ellos afirman una entera satisfacción al modelo propuesto y están de acuerdo con los resultados obtenidos aduciendo la familiaridad del equipo. El equipo XA aduce que en estos casos es necesario fortalecer la comunicación, es decir debe conformarse códigos claros de comunicación y procurar la unicidad de los miembros de los equipos en cuanto a prácticas de programación y conocimiento de las herramientas esto hará, afirman ellos, que el proyecto empiece a madurarse muy temprano por muy complejo que sea.

CAPITULO V: Conclusiones, Limitaciones y Trabajos Futuros

En este capítulo se describen las conclusiones de la investigación alrededor de XA, en particular, la posibilidad de escalar el método Extreme Programming - XP en los contextos de estudio, se revisa el alcance de los objetivos propuestos y se explica la evaluación de las hipótesis propuestas. Además, se evalúan algunas limitaciones de la investigación, y los trabajos futuros que servirán para complementar y continuar la investigación.

5.1. Conclusiones

XP ha mostrado trabajar para proyectos de baja complejidad y equipos pequeños (5±2 personas) [36]. Esta tesis ha abordado una propuesta que permite escalar el método XP a través de un modelo holístico con centro en la arquitectura de software. La propuesta ha sido denominada como XP con Arquitectura o XA, la cual fue aplicada en dos casos de estudio en el contexto de cursos de último año del programa de ingeniería de sistemas de la Fundación Universitaria de Popayán en los que participaron 18 personas en promedio para el desarrollo de cada proyecto. En estos estudios, XA mostró que mantiene los índices de productividad mientras mantiene la filosofía de los métodos ágiles, logrando escalar en el marco de dos proyectos de mediana complejidad.

De la definición de XA y su aplicación en los estudios de caso se pueden observar los siguientes aspectos:

- La comunicación entre el equipo XA y los equipos XP, debe ser muy clara y coherente. Los miembros del equipo XA deben conocer claramente los objetivos del proyecto y los requisitos de negocio del cliente. La participación de los miembros del equipo XA como clientes de cada equipo XP fue decisoria para que éste aspecto se lograra en la práctica.

- Los representantes del equipo XA deben no solamente ser el elemento de conexión entre el equipo de arquitectura y los equipos XP, sino que deben ser los dinamizadores, motivadores del equipo. Desde una perspectiva técnica, tener un equipo responsable y orientador de la infraestructura del sistema causa confiabilidad en todos los equipos XP y al Cliente XA para alcanzar las metas del proyecto.

- Al final de cada iteración, a través de la reflexión, los miembros del equipo XA comprendieron su papel para lograr la integridad del modelo holístico, su responsabilidad como arquitectos y en la comunicación con el cliente.

5.2 Limitaciones

Desde la perspectiva del investigador, es necesaria una mayor aplicación a equipos heterogéneos en prácticas industriales y con clientes menos dispuestos medir la eficacia del modelo propuesto de una forma más generalizada. Aunque el XA logró aplicarse adecuadamente en los proyectos, los resultados podrían haberse afectados por las condiciones del equipo. Particularmente por las capacidades y experiencias de los equipos en métodos ágiles y arquitecturas de software. Otro aspecto que no fue considerado fue la evaluación de la arquitectura, tanto como parte del modelo XA, como del modelo de evaluación del estudio de caso. Si bien se ha definido el

proceso XA, su aplicación revela la necesidad de contar con herramientas que soporten mejor la gestión de los proyectos, en particular debido al trabajo concurrente de los equipos de trabajo.

5.3 Trabajos Futuros:

Uno de los principales trabajos futuros es extender la evaluación empírica a más casos de estudio en contextos diferentes. Particularmente, desarrollar nuevos casos en un contexto industrial donde las variables que se presentan en este tipo de ejercicios son difíciles de controlar. La aplicación del modelo a casos de estudios académicos ha permitido establecer que para corroborar los resultados obtenidos en esta investigación será necesario considerar otras variables de contexto como las establecidas por [11] para hacer una comparación más objetiva, por ejemplo, la experticia de los participantes en los diferentes aspectos: procesos, métodos, tecnologías y arquitecturas de software. Todas estas consideraciones serán adoptadas para mejorar el diseño de los nuevos estudios de caso para ser aplicados a los casos industriales.

Adicionalmente XA, debe ser extendido con prácticas de evaluación de arquitecturas con el fin de agregar otro aspecto de validación al producto en el contexto ágil y que su vez sea un instrumento valioso para evaluar la calidad de XA a través de la evaluación de la calidad de la arquitectura en los nuevos estudios de caso. Desde una perspectiva de soporte tecnológico, el modelo XA exhibe la necesidad de contar con herramientas de gestión acordes a las dinámicas de desarrollo concurrente del modelo holístico**.**

Referencias bibliográficas

1. Costello, R. J. and Liu, D. 1995. Metrics for requirements engineering. In Selected Papers of the Sixth Annual Oregon Workshop on Software Metrics (Silver Falls, Oregon, United States). W. Harrison and R. L. Glass, Eds. Elsevier Science, New York, NY, 39-63.

2. Booch G, Jacobson I, and Rumbaugh J. The Unified Modelling Language for Object-Oriented? Development (version 0.91) Rational Software Corporation. 2007

3. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. “Agile software development methods Review and analysis”. VTT Publications. 2007.

4. Qumer, A., Henderson-Sellers, B. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. 2007

5. Beck, K.. .Extreme Programming Explained. Embrace Change., Pearson Education, 2005.

6. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. .Agile software development methods Review and analysis.. VTT Publications. 2007.

7. Matt Light, The first Key to Project Success Is Collaborative Requirements Definition and Manegement. 2008

8. Begel, Andrew. Perceptions of agile software development at Microsoft. 2008.

9. Highsmith J. Agile software development ecosystems. 2007

10. Karla Méndescalo, Elsa Estevez, Pablo Fillotrani. Un framework para evaluación de metodologías ágiles. 2008

11. Beck, Kent. Extreme Programming explained. Reading, Mass: Addison-Wesley Longman, Inc. 2006.

12. Dibá, T., Dingsoyr, T. Empirical studies of agile software development: A systematic review. 2008

13. Rick Kazman, Mark Klein, Robert L. Nord. Tailorable Architecture Methods. CMU Institute.2006

14. Instituto de Ingeniería de Software. Disponible en: www.sei.cmu.edu. Ultima visita, julio de 2012

15. Rob Wojcik, Felix Bachmann, Len Bass, Paul Clements, Paulo Merson, Robert Nord, Bill Wood. Attribute-Driven Design (ADD), Version 2.0. 2007.

16. Mario R. Barbacci, Robert Ellison, Anthony J. Lattanze, Judith A. Stafford Charles B. Weinstock, William G. Wood. Quality Attribute Workshops (QAWs),Third Edition. 2007.

17. Paulo Merson, Robert Nord, Bill Wood. Software Architecture for Software-Intensive Systems. 2007.

18. Rick Kazman, Mark Klein,Paul Clements ATAM: Method for Architecture Evaluation. 2007

19. Ethan Hadar, Gabriel M. Silberman: Agile architecture methodology: long term strategy interleaved with short term tactics. OOPSLA Companion. 2008.

20. Highsmith J., Orr K. “Adaptive Software Development: A Collaborative Approach to Managing Complex Systems”. Dorset House. 2006

21. Andreas Kornstädt and Joachim Sauer. Tackling Offshore Communication Challenges with Agile Architecture-Centric. 2007.

22. Beck, K., Beedle, M., Bennekum, A., Cunningham, W., Fowler, M. y otros. Manifesto for Agile Software Development. (2001)

23. Tackling Offshore Communication Challenges with Agile Architecture-Centric Development. Andreas Kornstädt and Joachim Sauer. Software Engineering Group, Department of Informatics, University of Hamburg. 2007

24. Rolf Njor Jensen, Thomas Møller, Peter S¨onder, and Gitte Tjørnehøj. Arquitectura y Diseño en eXtreme Programming; Presentación de "Historias de desarrolladores". 2007.

25. Extrem Programming. Disponible en: www.extremeprogramming.org. Última visita, Julio de 2012.

26. Mario R. Barbacci, Robert Ellison, Anthony J. Lattanze, Judith A. Stafford.

27. Bunge, Mario. La ciencia y su método y su filosofía. siglo XXI Editores. México. 2000.

28. Shaw, Mary and Clements, Paul. The Golden Age of Software Architecture. IEEE Software. Volume 23, Number 2. March 2006. Pages 31-39.

29. Pendharkar, P. C. and Rodger, J. A. 2009. The relationship Between Software Development Team Size and Software Development Cost. COMMUNICATIONS OF THE ACM Volume 52, Number 1. January 2009. Pages 141-144.

30. Hurtado Alegría, Julio Ariel. Método Científico en Ingeniería de Software. Reporte Técnico. Disponible en: http://www.dcc.uchile.cl/~jhurtado/mcis.pdf.

31. LarKman, C.: UML y Patrones, Tercera Edición: Prentice-Hall, 2008

32. D. Hix, H.R. Hartson. Developing User Interfaces. Ensuring usability trough. Product and Process. New York (USA). 2003

33. Patricio Letelier y Mª Carmen Penadés. “Metodologías ágiles para el desarrollo de software: Extreme Programming (XP)”. 2007

34. Constantine, L. L., and Lockwood, L. A. D. Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. 2004

35. Software Architecture- Centric Methods and Agile Development Robert L. Nord

and James E. Tomayko, Software Engineering Institute 2006.

36. Scaling Agile Methods Donald J. Reifer, Frank Maurer, and Hakan Erdogmus.
37. Beck, K.: Extreme Programming Explained: Embrace Change, Second Edition. Addison Wesley Professional (2004).

38. www.extremeprogramming.org. Revisado en Julio de 2012.

39. www.extremeprogramming.org/rules/unittests.html. Revisado en Julio de 2011.

40. www.extremeprogramming.org/rules/pair.html. Revisado en Enero de 2012.

41. Shingō, Shigeo (1989). A Study of the Toyota Production System from an Industrial Engineering Viewpoint. Productivity Press.

42. Qumer, A., Henderson-Sellers, B. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. 2007.

43. Dibá, T., Dingsoyr, T. Empirical studies of agile software development: A systematic review. 2008.

44. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A.,Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. Manifesto for agile software development. Accessed at http://agilemanifesto.org 2009.

45. Bill Venners. Test-Driven Development A Conversation with Martin Fowler,

2002

46. http://c2.com/cgi/wiki?TestDrivenProgramming Revisado octubre de 2011

47. http://c2.com/cgi/wiki?ContinuousIntegration Revisado Octubre de 2011.

48. http://www.testdriven.com/ Revisado octubre de 2011.

49. Micah Alles, and others. Presenter First: Organizing Complex GUI Applications for Test-Driven Development. 2006.

50. Roy W. Miller, Christopher T. Collins. Acceptance Testing. 2006.

51. Fundación Universitaria de Popayán. Proyecto Educativo Institucional. 2003. Disponible en: http://www.fup.edu.co.

52. Fundación Universitaria de Popayán. Acuerdo 012 de 2003. Disponible en: http://www.fup.edu.co.

53. P. Runeson and M. H¨ost, “Guidelines for conducting and reporting case study research in software engineering,” Empirical Softw. Eng., 2009.

54. L. Layman, L. Williams, and L. Cunningham, “Exploring extreme programming in context: An industrial case study,” in Proceedings of the Agile Development Conference, ser. ADC ’04. Washington, DC, USA: IEEE Computer Society, 2004.

55. F. Maurer and S. Martel, “On the productivity of agile software practices: An industrial case study,” International Workshop on Economics-Driven Software Engineering Researh(EDSER, Tech. Rep., 2002.

56. Dalcher, D., Benediktsson, O., y Thorbergsson, H., “Development Life Cycle Management: A Multiproject Experiment”, Proceedings of the 12th International Conference and Workshops on the Engineering of Computer Based Systems (ECBS’05), 2005

57. Wellington, A., Briggs, T., y Girard, C.D., “Comparison of Student Experiences with Plan-Driven and Agile Methodolog 2007

58. Francisco Ruiz, Javier Verdugo. Guía de Uso de SPEM 2 con EPF Composer Versión 3.0. 2008.

59. Aristóteles, La Métaphysique, trad. Annick Jaulin, PUF, 1999

60. IEEE Std 1471-2000

61. Barry W. Boehm. Software Engineering economics 1982

62. IEEE Std. 1061