LUIS FREDDY MUÑOZ SANABRIA

AGile/ArchiTecture in Action (ÁGATA): Un Proceso Holístico y Gestionado para el Desarrollo de Software Ágil con Arquitectura Orientado a Equipos de tamaño mediano.

Tesis presentada a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para la obtención del Título de

Doctor en:
Ciencias de la Electrónica

Director:
PhD. Julio Ariel Hurtado Alegría
Universidad del Cauca, Colombia

Comisión evaluadora:
PhD. Francisco Álvarez, Universidad Autónoma Aguascalientes, México
PhD. Juan Manuel Gonzales, Universidad Autónoma de Puebla, México
PhD. Luis Merchán, Universidad San Buena ventura, Colombia
PhD. César Collazos, Universidad del Cauca, Colombia

Popayán
2017

A Dios por renovar mis fuerzas cada día
A mi amada esposa por soportar mis noches de insomio mientras daba paso a esta imaginería
A Viviana Angie Sebastián por soportar tantas fechas aplazadas
A la FUP por sus apoyo incondicional en este proyecto

Agradecimientos

Al Gran creador, por renovar cada día mis fuerzas para continuar en la marcha

A mi amada esposa y mis preciosos hijos. Ellos siempre serán el motor que me impulsa a continuar por encima de cualquier dificultad.

Gracias Julio Ariel Hurtado, que mas que un director te consideré un amigo, espero no defraudar esa confianza que depositaste en mi. Tus consejos y regaños sirvieron para formar y alcanzar este propósito.

A Francisco Álvarez, Dr. Paco siempre hubo una mano amiga lista para animar, para impulsar para aconsejar.

César gracias, pertenecer a tu grupo se convirtió en una gran fortaleza para mi vida profesional, tu insistencia y compañía me sirvió para alcanzar este proyecto.

A Libardo Pantoja, muchas veces me escuchaste y en esas muchas charlas, me sirvieron para progresar en este camino.

Al Padre Mario Alfredo Polo Castellanos, Fernando, Luis, su apoyo incondicional me animó a terminar con el proyecto.

A mis amigos del grupo IDIS, todos los días había algo nuevo que aprender de ellos.

Resumen

Las metodologías ágiles son un referente para el desarrollo de software cuando los equipos son pequeños. Cuando crecen en número, las prácticas ágiles no pueden soportar las dinámicas de los equipos de tamaño mediano. Muchos investigadores han hecho sus propuestas basadas en métodos tradicionales y/o híbridos para extender los parámetros ágiles, pero con cierto grado de dificultad en cuanto a la gestión y a la comunicación de sus miembros. Esta investigación tuvo en cuenta cuales son las características de los valores y prácticas ágiles y (Scrum y Extremme Programming), cuales sus posibilidades de uso dentro de desarrollos disciplinados y orientados al valor. Se apoyó en XP/Architecture, un modelo que basado en la arquitectura escaló XP a equipos de tamaño mediano, para identificar los límites, donde se destacan problemas en la comunicación y la gestión del equipo por su tamaño; para enfrentar los desafíos de la comunicación y la coordinación de los equipos que desean escalar los métodos ágiles. Para lograrlo se propone un marco de proceso denominado *Agile Architecture in Action (AGATA),* orientado a la gestión, la arquitectura y a la comunicación. Un método que se apoya en dos canales para dinamizar la comunicación: La arquitectura y el cara a cara, y que motivarán al equipo a mantenerse dentro de sus prácticas para entregar los resultados establecidos por los métodos ágiles.

Abstract

Agile methodologies are a reference for software development when teams are small. When they grow in numbers, agile practices can´t withstand the dynamics of large teams. Many researchers have made their proposals based on traditional and / or hybrid methods to expand the parameters, but with some degree of difficulty in the management and communication of its members. This research took into account the characteristics of values and practices and Scrum and Extremme programming, and their possibilities of use within disciplined and value oriented developments. It was supported in XP / Architecture, a model that was based on the architecture scaled XP to big equipment, to identify the limits, where they emphasize the problems in the communication and the management of the equipment by its size; To meet the challenges of communication and coordination of teams wishing to scale agile methods. Agile architecture in action (AGATA), oriented to management, architecture and communication. A method that relies on two channels to dynamize communication: architecture and face to face, and that motivated the team to stay within their practices to deliver the results established by agile methods.

Tabla de contenido

Lista de Figuras

Lista de Tablas

Lista de Ilustraciones

Capítulo 1

Introducción

> Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: **la voluntad**.
>
> Albert Einstein

En su necesidad de hacerse más competitivas las organizaciones actuales, han incrementado la demanda de la sistematización de sus procesos; esto ha hecho que haya un crecimiento vertiginoso de las necesidades de las cadenas de valor y por lo tanto, un crecimiento equivalente de la industria de software [11]. Paulatinamente, la industria de software en Colombia creció en un 29% entre los años 2012 y 2014; en los últimos doce años (2003 – 2014) el mercado de software se ha incrementado cinco veces su tamaño debido a la creciente demanda de sus productos por parte del sector de telecomunicaciones, pero también el de finanzas, gobierno, consumo masivo y manufactura. El resultado del desarrollo de la infraestructura de tecnología, ha permitido que la industria de software incremente sus ventas. Estas pasaron de los 2,6 billones a los 9,3 billones entre 2010 a 2014, respectivamente**.** En el 2014 el Producto Interno Bruto de Colombia (PIB) creció en 4,6%, y en ese mismo año la actividad de correo y telecomunicaciones aumentó 4,23%**,** con una participación en el PIB nacional del 3,2% [12].

Debido al fenómeno recurrente de crecimiento de la industria, esta ha venido evolucionando en términos de procesos, métodos y herramientas. Es así que para los años setenta (70´s), aparece el método de análisis estructurado [13] con una orientación al modelado de flujo de datos y los archivos como medios para la clasificación de la información requerida por el cliente. Para los años ochenta (80´s) las metodologías orientadas a objetos buscaron responder a la creciente complejidad de los sistemas para atender a las nuevas necesidades organizacionales [14] [17]. Apoyados de modelos como CMMI [132] o normas ISO [133], la industria del software buscaba mejorar las prácticas a través de los procesos software. Pero como una forma revolucionaria a los procesos estandarizados, enfocados hacia las personas en lugar de orientarse hacia los procesos y por la exigente competitividad de los mercados, que exigen modelos de trabajo más acordes a la velocidad de los negocios, para el año 2000 aparecen los métodos ágiles [15], que surgen como respuesta a la incertidumbre de las organizaciones de poder validar de forma temprana soluciones a sus problemas empresariales a través de ciclos de trabajo más cortos y rápidos [16].

El auge de las metodologías ágiles ha dado respuesta a los desarrollos rápidos en ambientes de bajos recursos y de gran incertidumbre [1][18], incluyendo prácticas básicas de calidad. Los mayores problemas de calidad reportados por la industria, provienen de un manejo inadecuado de los requisitos [2] [3], así que, las metodologías ágiles han abordado de manera directa este problema con ciclos de desarrollo cortos, orientados al valor y con la participación del cliente [4].

Por otro lado, estas metodologías valoran las personas sobre los procesos y la comunicación cara a cara es valorada sobre la documentación [3]; sin embargo han presentado problemas para ser aplicadas en contextos con mayor número de personas (4±3 Integrantes) y mayor complejidad de las soluciones [15].

Cuando los equipos son numerosos, (mas de 7 personas), [18] las organizaciones requieren de mayores habilidades comunicativas para lograr sus objetivos de negocio [7]. Estas habilidades son requeridas independientes al contexto, pero se hacen más necesarias en ambientes orientados a personas como es el caso de los métodos ágiles. Sin una buena comunicación, es muy poco probable que los equipos puedan obtener buenos resultados. Para la industria del software es muy necesaria esta habilidad, de ahí que William Scheffer, gerente internacional de desarrollo de San Microsystems afirme que: “Si pudiéramos decir que existe una cualidad necesaria para triunfar en esta industria sería la de comunicarse” [8], refiriéndose a la industria del software, o como afirma Jim Richman quien manifiesta que “Si pudiéramos dar un consejo, sería que uno jamás puede considerar que se ha entrenado lo suficiente en el desarrollo de la capacidad de comunicarse” [9]. Más de 200 jefes de empresas en desarrollo tecnológico en Estados Unidos concuerdan que para poder triunfar en el siglo XXI es necesario adquirir más habilidades o plantear mejores estrategias de comunicación [10].

En este capítulo, se define el problema de investigación que este proyecto de doctorado resuelve alrededor del tema de la comunicación en proyectos ágiles de tamaño mediano; contiene la pregunta de investigación, el planteamiento de la hipótesis así como la metodología seguida para el logro de los propósitos del proyecto, al mismo tiempo define la estructura del documento y relaciona los anexos que contiene la información complementaria. Finalmente se presentan aclaraciones sobre los logros alcanzados en relación con los objetivos propuestos.

1.1 Planteamiento del problema

Uno de los problemas identificados en la comunidad de los métodos ágiles, es la dificultad para escalarlos cuando el proyecto crece [19]. Como crecimiento del proyecto la comunidad se refiere al incremento en el tamaño del equipo, incremento en la duración y esfuerzo para el desarrollo del producto; debido a los requerimientos del cliente.

Para enfrentar el problema del crecimiento del tamaño del equipo, las investigaciones sugieren prácticas más colaborativas, con el fin de detener la complejidad para la gestión, la coordinación y la comunicación dentro del proyecto [20].

La mayoría de los proyectos de software que presenta la comunidad ágil son proyectos de pocas personas y en cuyos desarrollos la arquitectura no es reportada como relevante y aunque para la gestión se propone la organización de sub-equipos que siguen las prácticas ágiles, estas se quedan cortas en el establecimiento de la división, la organización entre equipos y la comunicación. En estos escenarios pequeños sucede que la gran mayoría de decisiones arquitectónicas son de baja complejidad puesto que normalmente se construyen sobre una plataforma que ya ha resuelto las decisiones arquitectónicas.

Con el crecimiento del equipo, la comunicación crece incrementalmente [21]; todas los miembros del equipo tienen sus necesidades, intereses, conocimientos, experiencias, expectativas y otras serie de motivaciones las cuales son diferentes entre ellos, aumentando la complejidad de la comunicación y por tanto la complejidad del proyecto; más cuando no existen prácticas ni canales adecuados que se hayan acordado al empezar el proyecto [22];

porque La función del canal de comunicación debe ser la de consolidar esta variedad de expectativas en una sola y de bien común, así como alinearlas de tal forma que se habilite un mecanismo de comunicación que facilite el flujo autónomo de información en el equipo.

Stahl et. al [23] reconocen cuatro necesidades que obligan a establecer patrones de comunicación en etapas muy tempranas del proyecto: 1) resolución de problemas, 2) información, seguimiento y retroalimentación, 3) creación de relaciones, y 4) toma de decisiones y coordinación. Al no establecerse esas actividades, con equipos de tamaño mediano como lo manifiesta Rudolph F. [24] lo más probable es que se pierdan el buen ambiente de trabajo que facilite la cohesión del equipo referida a mantenerlo unido en cuanto a la comunicación actualizada, la similitud de intereses y necesidades (homogeneidad), el compromiso con las tareas y con el equipo. Lo ideal es que el equipo pueda comunicar adecuadamente las prácticas y principios establecidos para el desarrollo del proyecto, mantener el consenso para la toma de decisiones producto de la reflexión e interacción y establecer un alto grado de calidad de la información que debe compartir a través de canales claros de comunicación.
El hecho de no comprender completamente las características del sistema, aumenta la incapacidad para resolver eficazmente los conflictos, aumenta el presupuesto y se retrasa el cronograma y, en última instancia, aumentan los clientes insatisfechos [25]. Comprender las características del sistema implica tener adecuados métodos para la transmisión de la información, porque en equipos de tamaño mediano, son frecuentes los problemas de comunicación; muchas veces debido a la personalidad y emotividad de cada uno de sus integrantes, o la complejidad de los mensajes. Estas dificultades individuales y grupales deben ser identificadas y analizadas oportunamente, porque constituyen barreras que afectan la comunicación y el alcance de logros por parte del equipo generando entropía al interior del grupo [24]. Estos problemas pueden llevar al deterioro de las relaciones del mismo equipo y afectar lógicamente al proyecto/producto. Lo anterior demuestra que el arte y la técnica de la comunicación suponen una profunda actitud al cambio, que debe ser motivada por canales de comunicación que generen confianza entre los integrantes del equipo.

Los equipos de desarrollo para comunicarse utilizan diferentes medios: aprovechan las conversaciones generadas en sus reuniones, mensajería electrónica o conversaciones de pasillo, suficiente para equipos pequeños, La idea es interactuar unos con otros en torno al software. Pero cuando los equipos crecen, generalmente no tienen patrones o canales que les permitan tener una identidad con el fin de difundir la información relacionada con el proyecto a un público más grande; aumentar el uso de medios electrónicos para la comunicación, afectó el comportamiento social de los desarrolladores de software [30] debido a que no tiene patrones claros que intenten unificar un grupo con diferencias individuales en torno a un proyecto. Más aún cuando la captura de la información en el proceso de diseño es un problema complejo y multidisciplinario. La aparición, aumento y aceptación de herramientas automáticas de trabajo cooperativo, han tratado de integrar el trabajo colaborativo del equipo que crece, para fortalecer de esta manera la comunicación [72] y mejorar las decisiones de diseño durante los procesos del proyecto. Pero en muchas ocasiones, estos modelos no abordan adecuadamente las prácticas sucedidas en el día a día en el que los miembros del equipo se dedican colectivamente a intercambiarse mensajes sobre los detalles del producto [32]. Si el equipo no tiene prácticas de comunicación coordinadas y adecuadas, puede entrar en el desánimo interno y por lo tanto llevar al fracaso al sistema. Entender las dificultades y preocupaciones del cliente, comprender las variables implicadas en la creación de las dificultades, explorar la causalidad entre las variables

identificadas y compartir entre los miembros de los equipos, comprenden la fase más crítica y difícil del proceso de diseño del sistema. Aunque, hay muchas ideas en la literatura para llevar a cabo esta fase, de todas maneras, falta un método cualitativo completo [33], que soporte las prácticas comunicativas en las relaciones intergrupales y que cubra completamente las dificultades de la fase.

Especialmente en grupos más grandes la comunicación es el punto crítico para recolectar y formar la información relevante, compartir el conocimiento y crear productos funcionales [34]. Contradictoriamente, el apoyo a la comunicación entre proyectos y entre organizaciones parece desempeñar un papel secundario en el desarrollo de producto en la industria del software. Las organizaciones grandes y experimentadas, no tienen enfoques sistemáticos de comunicación [35]. Estas prácticas, las han inventado con base a las necesidades a nivel de proyecto, especialmente al inicio del proceso cuando se necesita bases sólidas que lo lleven al éxito.

Por lo anterior, en este proyecto se formula la siguiente pregunta de investigación: **¿Cómo escalar los métodos ágiles en proyectos de software con equipos de desarrollo de tamaño mediano (de 2 a 9 sub-equipos)[36] y qué prácticas podrían introducirse para superar los desafíos de comunicación?**

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.

La priorización de los requisitos en los métodos ágiles está orientada a generar valor rápidamente al cliente, aunque no necesariamente beneficie las empresas de software durante el desarrollo y mantenimiento de sus activos de software. La idea es fortalecer una industria de software competitiva, que produzca valor a los clientes y al mismo tiempo empodere sus estrategias de producción y mantenimiento de software, es decir, que pueda reutilizar componentes, estructuras y decisiones de diseño, así como facilitar el mantenimiento de sus productos [1].En equipos de tamaño mediano la falta de diseño explicito, ha dificultado la comunicación de las decisiones y por tanto se terminan perdiendo en la cadena de valor de la empresa de software. La ausencia de diseño bajo esquemas ágiles, puede generar productos poco competitivos difíciles de mantener, en un mercado que demanda muchos cambios en las aplicaciones en la medida en que sus negocios lo requieren y que por tanto pueden llevar al fracaso a las empresas de software en el largo plazo y cuando los equipos crecen se carece de mecanismos claves para organizar el sistema y distribuir el trabajo en una forma razonable.

Para esta investigación, una de las estrategias para escalar las metodologías ágiles, es la de considerar la arquitectura como un mecanismo de organización, comunicación e integración, donde la descripción arquitectónica como canal de comunicación, entra a facilitar de manera temprana la organización de una solución que se satisfagan los requisitos más relevantes del software, mientras sostiene el enfoque de generación de valor para el equipo de desarrollo.

Así, la arquitectura se abre como concepto complementario a nivel técnico, un canal de comunicación que soporte la comunicación intergrupal y de gestión en un proyecto de software con equipos de tamaño mediano que intente usar metodologías ágiles [37]. Una de

las aproximaciones al modelo arquitectónico la propone Extreme Programming (XP) a través de la práctica de la metáfora del sistema [38] y que puede ser usada como punto de partida a una descripción arquitectónica.

Si bien la arquitectura es una estrategia clave para la organización de equipos en un contexto ágil, se generan algunas problemáticas respecto a la coordinación de equipos, particularmente aspectos de comunicación y de gestión del proyecto [39].

Cuando los equipos crecen es necesario establecer estrategias para que la comunicación no se diluya en la entramada complejidad del desarrollo del proyecto [44]. Así mismo se deben abordar los problemas de gestión porque si el equipo crece es necesario subdividirlo con el objeto de mantener las prácticas ágiles para lograr que estas se mantengan [40]. Esto conlleva ala sincronizada en términos de valor entre los diferentes equipos del proyecto.

1.2 Objetivos

1.2.1 Objetivo General

- Definir un proceso holístico y gestionado para el desarrollo de software ágil centrado en la arquitectura para equipos de tamaño mediano (de 2 a 9 sub-equipos)[36]

1.2.2 Objetivos Específicos

- Establecer que prácticas ágiles, de arquitectura y de gestión han sido usadas para soportar la comunicación en proyectos de software grandes
- Definir un proceso para el desarrollo de software basado en la comunicación, la agilidad, la gestión y soportado en métodos de arquitectura, denominado ***AGATA***, en el contexto de equipos de tamaño mediano.
- Evaluar el proceso de software AGATA a través de estudios de caso académicos e industriales.

1.3 Hipótesis

AGATA es un proceso software holístico y gestionado, sostenido por los valores y principios propios de la gestión de procesos (SCRUM), que agrega las prácticas de arquitectura de XP/Architecture (XA), los principios de la programación extrema (XP), basado en la comunicación, la agilidad y la gestión para el desarrollo de software ágil en el contexto de equipos de tamaño mediano (2 a 9 sub-equipos).

Con el objeto de validar este proceso, se propuso la siguiente hipótesis desde dos perspectivas: de investigación e hipótesis nula y de la siguiente manera:

Hipótesis de investigación: El proceso AGATA, incrementa el nivel de comunicación en equipos de desarrollo de tamaño mediano (2 a 9 sub-equipos) en el contexto de los casos estudiados.

Hipótesis nula: Los elementos de coordinación, gestión y arquitectura en el proceso AGATA no son suficientes para incrementar el nivel de comunicación para equipos de tamaño mediano (2 a 9 sub-equipos) en el contexto de los casos estudiados.

La hipótesis nula podría permitir proponer nuevas hipótesis sobre: (1). Extensiones futura de AGATA para la gestión de los canales de comunicación propuestos para esta investigación. (2).Extensiones futuras de AGATA para resolver los desafíos de la comunicación.

1.4 Método de investigación

El proyecto diseña un marco de procesos de software, basado en Xp/Architecure [41], para el desarrollo de software que involucre proyectos con equipos de desarrollo de tamaño mediano. Para lograrlo, se usa como marco el *método general de investigación científica*, específicamente el método científico De Mario Bunge [42]. 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 [43] 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 ***AGATA***. 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 [45], el cual se ha aplicado de manera empírica en algunos proyectos de investigación dentro de los grupos IDIS y LOGICEL, basado en los hallazgos de Shaw. El método, de acuerdo a la Figura 1.1 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.

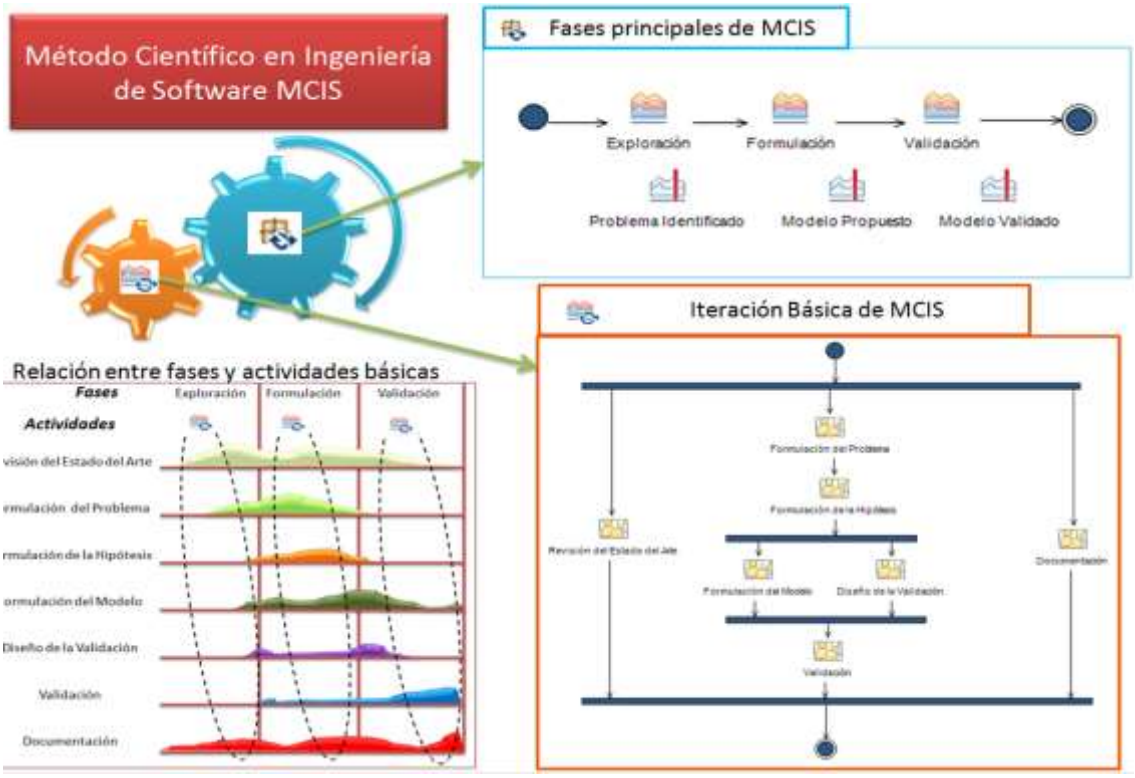


Figura 1 Métodos Científico en Ingeniería de Software - MCIS

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 1.1. A continuación se detallan las actividades básicas planificadas haciendo una adaptación particular de MCIS a éste proyecto.

1.4.1. Actividades

Las principales actividades de este modelo con las tareas respectivas adaptadas al contexto de este proyecto son:

Actividad 1: Planteamiento del problema

- **Reconocimiento de los hechos:** se revisaron experiencias y problemas encontrados sobre métodos ágiles en grupos grandes. 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 y sus dificultades en la comunicación 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 a través de una pregunta lo que permitió que 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 ***AGATA*** permite escalar XP a través de SCRUM para la gestión del equipo y de prácticas de arquitectura como canal de comunicación en un contexto académico e industrial.
- Selección y formalización de los insumos de proceso (procesos y fragmentos de proceso). Es decir los proceso XP[1], SCRUM [46] 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 SCRUM 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 [47]:

- **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 16±4 participantes (se describen al interior de este documento). Se recolectaron las mediciones basadas en las métricas definidas (descritas en los anexos). Luego se hace las mediciones en tres contextos industriales que permitieron hacer mas sólidas las conclusiones de la propuesta.
- **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.
- 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.

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

- Trazado para trabajos futuros.

Actividad 5. Documentación

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

1.5 Organización del documento

La organización del documento de trabajo de grado está dividida en 5 capítulos, los cuales se describen brevemente a continuación:

El **Capítulo 1**, capítulo actual referente a la introducción del documento, donde se especifica el planteamiento del problema, los objetivos del trabajo de grado, la hipótesis de solución y la estructura del documento.

El **Capítulo 2**, presenta los referentes teóricos necesarios para comprender la información presentada en el documento. Los referentes teóricos están basados en dos aspecto: Por un lado, se definen los conceptos básicos de los métodos ágiles desde la perspectiva de la comunicación y la arquitectura, básicamente Extreme programming y Scrum. Por otro lado, se hace un estado del arte para dar a conocer la relevancia de estas metodologías en la industria del software, y las propuestas hechas por diferentes actores para escalar los parámetros ágiles cuando los equipos crecen.

El **Capítulo 3**, presenta la propuesta de esta investigación: ***AGATA*** como marco de proceso, basado en la arquitectura, un mecanismos de comunicación y la gestión del equipo, para escalar los métodos ágiles; Extreme programming particularmente.

El **Capítulo 4**, presenta la ejecución del modelo propuesto en estudios de casos académicos e industrial, además del análisis de los resultados.

El **Capítulo 5**, presenta las conclusiones de los resultados obtenidos y su articulación con los objetivos planteados. Adicionalmente, son presentadas las actividades futuras para fortalecer el presente trabajo.

Capítulo 2

El propósito de la ingeniería del software es controlar la complejidad, no crearla.

Pamela Zave

2.1. Marco conceptual

En este capítulo se presenta el estado del arte y trabajos relacionados sobre los cuales se fundamenta esta tesis. Se describen aspectos generales de las metodologías ágiles entre ellas Extreme Programming (XP) y SCRUM, también los métodos de arquitectura propuestos por el Software Engineering Institute (SEI): Attribute-Driven Design (ADD) y Quality Attribute Workshops (QAW). Debido a que las metodologías ágiles se han convertido hoy, en un referente para el desarrollo de software para equipos pequeños [4], esta investigación, tendrá en cuenta cuales son las características primordiales de tales metodologías y cuales sus posibilidades de uso desde estrategias disciplinadas y orientados al valor. Los temas de interés de este trabajo son las limitaciones desde la perspectiva de la comunicación que dificultan la escabilidad de los métodos ágiles a equipos más grandes. Particularmente, fue necesario estudiar los trabajos previos que han intentado fortalecer este tipo de problemas con un enfoque basado en la gestión de canales de comunicación entre ellos la arquitectura, la configuración y gestión intergrupal y el cara a cara de la comunicación; temas en los cuales se orienta esta propuesta.

2.1.1. Manifiesto Ágil

Kent Beck, quien propuso el método Extreme *Programming* [5], e 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[2]. En la reunión se definió el término “Métodos Ágiles” para definir a los métodos que estaban surgiendo como alternativa a las 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 de las metodologías ágiles se puede ver en la Figura 2..

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

Figura 2 Aproximación al ciclo de vida ágil y sus principales componentes (Elaboración Propia)

.

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 valora:

- 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 prácticas predictivas, centradas en las personas y en los equipos, iterativas, orientadas hacia la entrega de valor, de comunicación frecuente, 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 [1]

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 claridad para indicar 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 con cortas iteraciones con el objeto de generar valor rápidamente y validar el avance con software funcionando, más que con su 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)

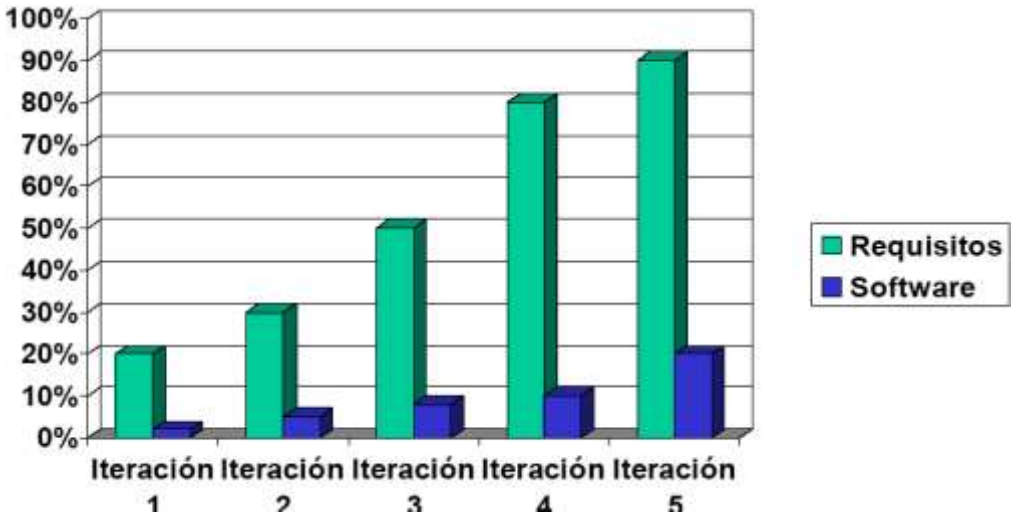


Figura 3 Requisitos y software (tomado de Agile Shift System Engineering).

Existen algunos principios de diseño en ingeniería [35] que se basan 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.

2.1.2 Los Métodos ágiles.

- **Extreme Programming (XP)**

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 [50]. De todas ellas, Extreme Programming (XP) se destaca como la más conocida y 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 [51]. Además es la que mejor satisface los principios y valores del manifiesto ágil [49]. Este modelo es propuesto y analizado por Kent Beck [52], donde expone las ventajas de un contrato con alcances opcionales.

Figura 4 Prácticas Agiles en Extreme Programming.(Elaboración Propia)

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 [53].

- **Scrum**

SCRUM tiene sus orígenes en 1986 en un artículo de la Harvard Business Review, “El nuevo juego para el desarrollo de productos” [54], donde se describe a empresas como Honda, Canon y Fuji-Xerox quienes utilizando un enfoque basados en equipos integrales para la fabricación de sus productos. El documento enfatiza sobre la importancia de dar poder a los equipos auto-organizados.

Este enfoque influenció para desarrollar muchos de los conceptos que dieron nacimiento a lo que hoy se llama Scrum. Cabe aclarar que Scrum no es un acrónimo, es un término extraído del Rugby (Deporte de contacto nacido en Inglaterra), que se refiere a la manera cómo se reinicia el juego luego de una falta o cuando el balón se ha ido fuera del campo de juego; los dos equipos forman una masa juntos alrededor del balón con sus brazos cruzados, y las cabezas bajas, luchando por la posesión del balón [134].

El enfoque de “carrera de relevos” para el desarrollo de productos entra en conflicto con el objetivo de obtener la máxima velocidad y flexibilidad. En su lugar un enfoque como el rugby – donde el equipo intenta avanzar como equipo, enviando el balón hacia atrás y luego avanzar – sirve mejor a los desarrollos competitivos que se ven hoy en día.

Los equipos que desarrollaron estos productos partían de requisitos muy generales, así como novedosos, y debían salir al mercado en menos tiempo del que se tardó en lanzar productos anteriores. Estos equipos seguían patrones de ejecución de proyecto muy similares. Jeff Sutherland [54] compara la forma de trabajo de estos equipos altamente productivos y multidisciplinares con la colaboración entre los jugadores de Rugby y su formación de Scrum. Ver figura 5 juego de rugby

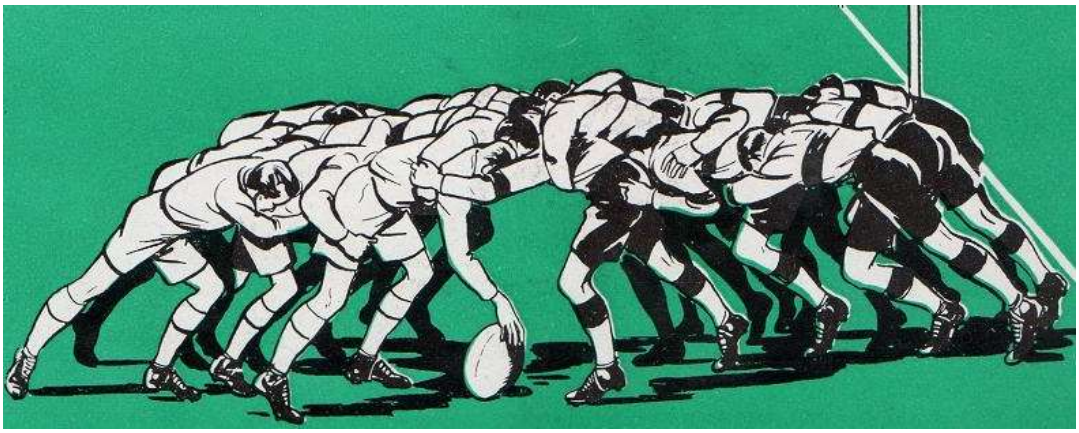


Figura 5 Rugby: orígenes de SCRUM.(Elaboración Propia)

En 1993, Jeff Sutherland [54] y su equipo en Easel Corporation crearon el marco de trabajo Scrum para ser utilizado en los procesos de desarrollo de software combinando los conceptos del artículo de 1986 con los conceptos del desarrollo orientado a objetos, un

control de procesos empírico, desarrollo iterativo e incremental, procesos de software y mejora de la productividad, así como el desarrollo de sistemas complejos y dinámicos.

En 1995, Ken Schwaber publica el primer informe sobre Scrum en OOPSLA 1995 [55]. Desde esa fecha, Schwaber y Sutherland, han escrito y publicado varias especificaciones para Scrum, incluyendo Desarrollo de Software Agile con Scrum, Gestión de Proyectos Agiles con Scrum y la Guía de Scrum.

Al igual que Extreme Programming (XP), Scrum es de las metodologías ágiles más usada para la organización de equipos en la gestión de proyectos [56]. Scrum es reconocido como un proceso ágil en el que se aplican un conjunto de buenas prácticas para la gestión de equipos que trabajan colaborativamente y obtener los mejores resultados en el desarrollo de proyecto. Scrum apoya sus prácticas entre sí para obtener equipos altamente productivos [57] En la Figura 6 se muestra la gestión de SCRUM:

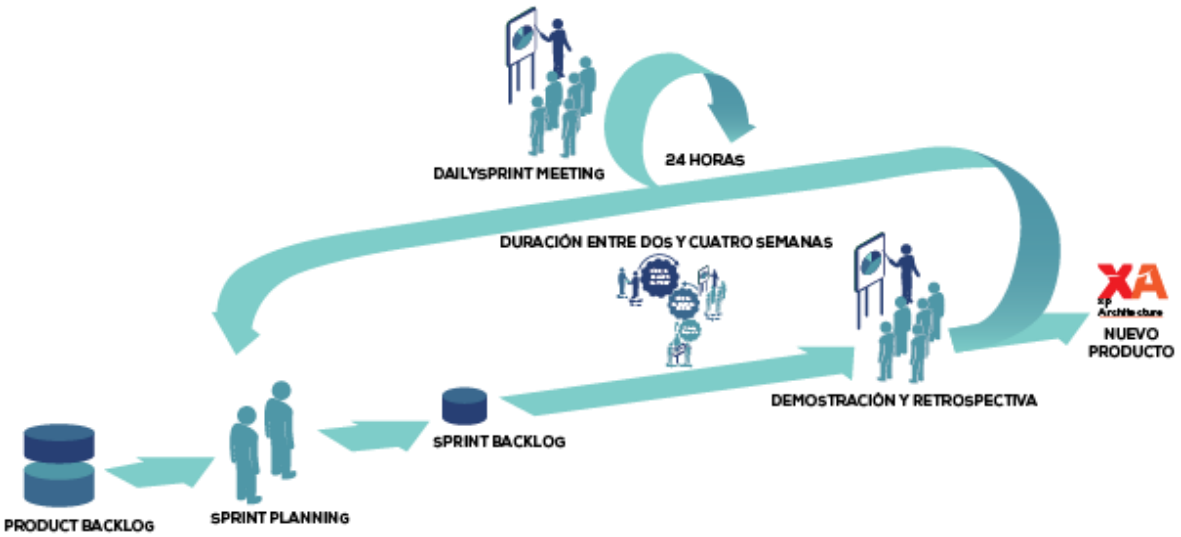


Figura 6 SCRUM.(Elaboración Propia)

Scrum se define también como un marco de trabajo de procesos [58] usado para gestionar equipos dedicados al desarrollo de productos. Cabe aclarar que Scrum no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se puede gestionar las prácticas siguiendo varias técnicas y procesos de construcción. Scrum muestra la eficacia relativa de las prácticas de gestión de producto y las prácticas de desarrollo, las cuales se pueden ir mejorando en su aplicación [59], de tal forma que los resultados cumplan con los requerimientos propuestos por el equipo de trabajo. Se reconoce como un marco de trabajo [59], debido a que Scrum define unos roles, eventos, artefactos y reglas asociadas que sirven a un propósito dentro de ese marco y potencian la efectividad en la gestión de los equipos para el desarrollo de cualquier producto.

Así, Scrum propone una metodología donde el equipo debe avanzar en conjunto, de nada sirve tener partes del producto terminado, si el producto no puede utilizarse. De esta manera la velocidad para el desarrollo de productos es la única variable que se intenta controlar. La filosofía de Scrum impulsa y resalta el trabajo en equipo, el aprendizaje constante y una estructura de desarrollo dinámico, que es flexible a los cambios durante el proceso de desarrollo.

Para esta investigación es importante el tamaño del equipo, puesto que como es bien reconocido por la literatura científica los métodos ágiles han dado resultado con equipos pequeños y proyectos de baja complejidad.

Un análisis y extensión de SCRUM y XP desde una perspectiva holística y orientado a la arquitectura se convierten en el propósito final de esta investigación para soportar los criterios y parámetros de las metodologías ágiles cuando los equipos de trabajo crecen en tamaño.

2.1.3 Arquitectura del Software

La arquitectura es un concepto que ha alcanzado su edad de oro, debido a la relevancia en la industria de software, por lo que es considerada un área madura dentro de la ingeniería de software [60]. 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 [61]. 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 metodologías ágiles, entre los participantes del proceso, no queda claro cuáles son las responsabilidades del arquitecto de software, ni cuál es la arquitectura del proyecto [27]; máxime cuando la necesidad de los clientes de reducir el tiempo de salida al mercado, lo cual obliga a las organizaciones de desarrollo de software a ser agresivas en sus calendarios de entrega; y entonces se tiende a creer que los diseños arquitectónicos no son muy relevantes en el desarrollo del proyecto, porque aumentarían los tiempos de trabajo en el proyecto y perjudicaría el cumplimiento del cronograma [37]. Definir una arquitectura conduce a reflexionar en una serie de decisiones que se toman detenidamente durante las primeras fases del desarrollo y que una vez definida, la arquitectura se convierte en el “cimiento” que soporta el software que se comienza a desarrollar.

- **Métodos de arquitectura propuestos por el SEI**

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 [21 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 estos métodos se muestra en la figura 7.

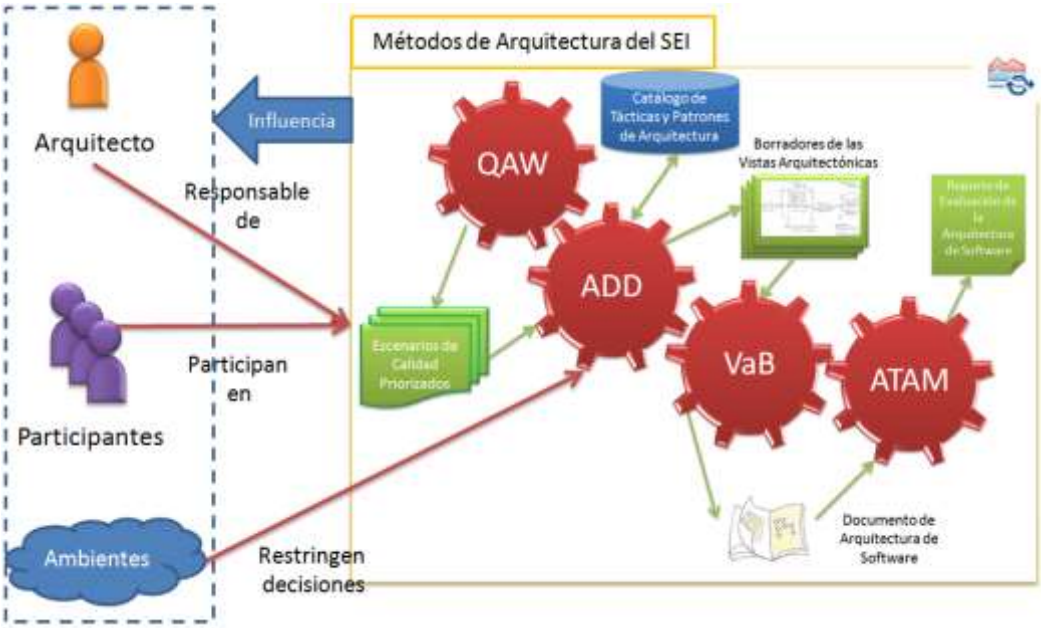


Figura 7 Métodos de Arquitectura propuestos por el SEI. (Elaboración propia)

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

● **QAW (Quality Attribute Workshop)** [62]**:** Es un método enfocado a la captura de requerimientos que determinan la arquitectura a través de un trabajo conjunto con los diferentes interesados en la solución arquitectónica. Estos requerimientos son los atributos de calidad que son descritos a través de un "escenario" 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)** [63]**:** Una vez que se han capturado escenarios y se han priorizado, se procede a realizar el diseño de la arquitectura siguiendo un enfoque recursivo 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 permiten satisfacer los requerimientos descritos por los escenarios.

● **VaB (Views and Beyond)** [48]**:** Como resultado de ADD, se obtienen distintas estructuras del sistema, formadas por componentes y sus relaciones. En este método, estas estructuras se documentan a través de "vistas" las cuales muestran 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)** [64]**:** 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 entre los atributos de calidad que están involucrados en la arquitectura.

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

2.1.1. La Comunicación en los proyectos de software

- **Comunicación en equipos de desarrollo grandes.**

Todo desarrollo de software implica una gestión de procesos que involucran personas, técnicas y herramientas. Dentro de este proceso hay una serie de sub-procesos que deben estar alineados para alcanzar con éxito el objeto del producto software [65]. La comunicación es el aspecto encargado de dicha alineación. Es decir mantener los sub-procesos en equilibrio para que personas, herramientas y técnicas trabajen coordinadamente [66].

La comunicación se puede definir como, el proceso por el cual los individuos cambian información a través de un sistema común de símbolos, signos y comportamientos [67], que como proceso, vincula todas las actividades del proyecto a través de los miembros que participan en él. La comunicación es una actividad bastante difícil y su complejidad, es directamente proporcional con el número de personas que participan en el proyecto, máxime cuando los mensajes deben ser compartidos y conocidos por todos. La Figura 8, muestra el sentido de la comunicación y su complejidad debido a su multi-direccionalidad.

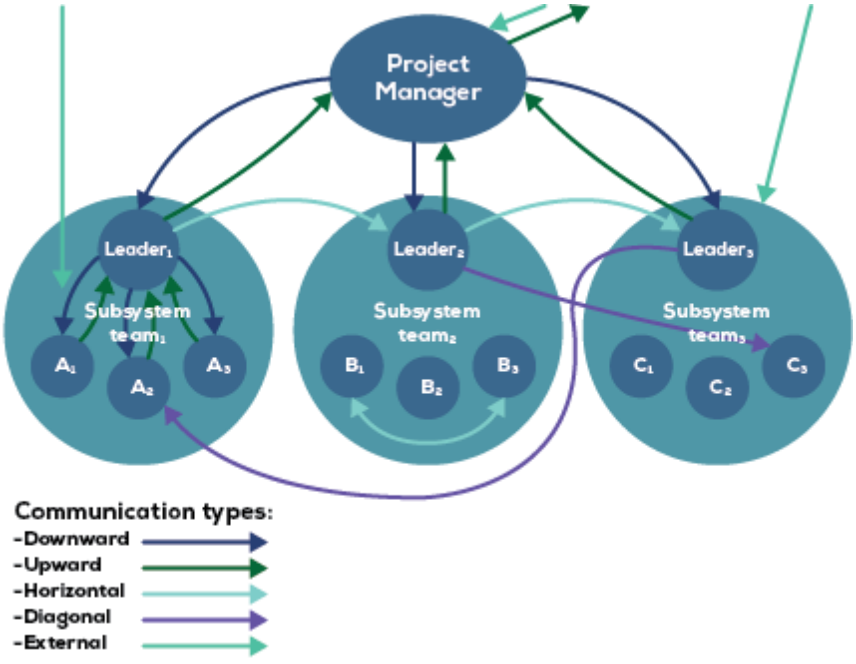


Figura 8 Un modelo de comunicación en proyectos software (Tomado de P. Flensburg)

Al ser la comunicación multidireccional, siempre tendrá un efecto sobre los elementos que participan en ella y sobre el medio [68]. De ahí que sean objetivos de la comunicación [69]:

- Transmitir información: Sea cual sea el mensaje obtiene una reacción del destinatario.

- Transferir ideas: cuyo objeto es cambiar la conducta del receptor. Una nueva reacción a partir del mensaje.
- Acordar, crear significados: basada en la acción del emisor, el receptor y los conocimientos fundamentales para crear cimientos compartidos.
- Dar y recibir retro-alimentación: posibilitan la corrección de errores, evalúa que se hizo bien o mal durante el proceso de producción.
- Co – Crear realidades: Influye en gran medida en las percepciones del emisor y el receptor

Para que se cumplan estos objetivos dentro del proceso de comunicación debe establecerse la sinergia de los elementos expuestos en la figura 9.

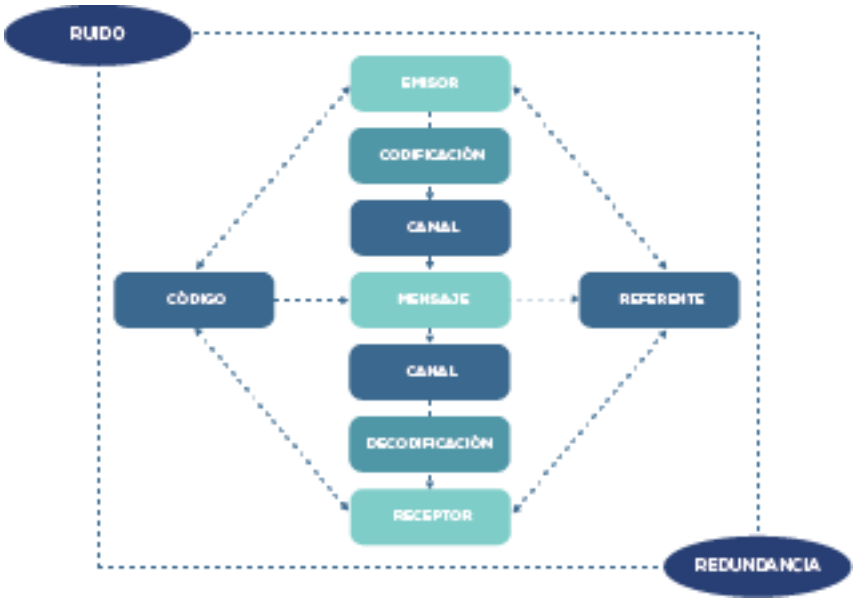


Figura 9 Elementos del proceso de comunicación. Adaptado de [70]

- **Emisor**: Quien inicia la comunicación, es el actor quién codifica y envía el mensaje.
- **Codificación:** La idea se convierte en palabras, gráficas, signos, señales u otros símbolos adecuados que dan a conocer el mensaje.
- **Canal:** Medio físico por el cual circula el mensaje.
- **Mensaje:** conjunto de información que se mueven dentro del equipo.
- **Decodificación:** Proceso en el cual los actores convierten la información en sus propios códigos para "entenderla".
- **Receptor:** El actor que decodifica y recibe el mensaje

- **Código:** Conjunto de signos, señales, con sintaxis y semántica puestos a disposición de los actores de la comunicación (emisor y receptor).

- **Referente**: Elementos extralingüísticos que acompañan el mensaje.

- **Ruido:** Perturbaciones imprevistas e imprevisibles que deterioran o dañan el mensaje.

- **Redundancia:** Elementos que aparecen en un mensaje y permiten de una u otra manera disminuir el ruido que puede ser: del código o del mismo emisor.

- **Entorno o contexto:** Ambiente en el cual se produce el mensaje para ser comprendido tanto por los actores de la comunicación, como aquellos que afectan de alguna manera ese ambiente. El contexto puede ser situacional (espacio y tiempo), socio-histórico (época), socio-cultural (valores y creencias) o lingüístico (lo que se dice y como se dice).

Cabe anotar que no todos estos elementos han sido perceptibles en el proceso comunicativo, ello ha dependido de la época y del enfoque de comunicación propuesto para las organizaciones. Los enfoques de comunicación al igual que las metodologías para el desarrollo de software han tenido su transición debido al dinamismo de las organizaciones y los clasifica Bartoli [97] en los siguientes:

- **Enfoque racionalista:** Se interesa por la información operativa; aquellos contendidos ligados al desempeño de la actividad profesional. La comunicación es exclusivamente descendente; basada en una débil transmisión de la información.

- **Enfoque de Comportamiento:** Da importancia al factor humano y a la importancia de su participación en las decisiones, la comunicación no es solamente descendente sino también horizontal y ascendente.

- **Enfoque Sociológico:** las teorías comunicativas que se derivan de este enfoque, hace una dura crítica a la burocracia no comunicativa y entre otros aspecto dan importancia a la comunicación informal.

- **Enfoque Gerencial:** Se adhiere a enfoques globales y flexibles al mismo tiempo. Propone concertación y coordinación en todos los niveles de la organización. Para ello favorece una comunicación interna fluida, participativa y que abarque todos los elementos del proceso.

A partir de estos enfoques y con base en la participación de todos elementos en el proceso de comunicación, aparecen los modelos de comunicación (Ver Figura 10), que según Lucas Marin [71] se pueden clasificar en las siguientes topologías:

Figura 10 Modos de Comunicación. Adaptado de [71].

Circulo: El flujo de la comunicación parte de un actor para volver al mismo después de pasar por otra serie de elementos incluidos en este proceso y la comunicación generalmente es Unidireccional.

Rueda: Existe un actor principal que es el que dinamiza la comunicación y mantiene el equilibrio del grupo y de los elementos de la organización.

Cadena: el flujo comunicacional es muy parecido al del círculo, pero no se llega a completar el último eslabón. Es una acción de transmisión de información descendente.

Total: Todos los sujetos se relaciona entre si ya sea de forma circular o lineal

Estrella: Se dan dos o más circuitos de comunicación entre los actores de la organización en función de los entornos e independientemente de los niveles. Es decir los miembros de un nivel no se comunican con los de otro nivel. Es típico de organizaciones estructuralmente rígidas.

Las empresas de desarrollo de software no son ajenas a estos enfoques, elementos y modelos. Si bien es cierto no son muy contundentes en el tratamiento de la información y más aún las metodologías ágiles, cuando basa su estructura comunicacional en la acción cara a cara [72], porque la comunicación, fluye a medida que se desarrollan cada una de las actividades del proyecto, también es cierto que cuando los equipos crecen no es posible mantener ese equilibrio dado que los enfoques y elementos comunicativos crecen proporcionalmente con el equipo; es el momento de involucrar nuevos elementos de comunicación (canales, codificadores, decodificadores) que permitan mantener la filosofía ágil en tanto que el equipo crece.

2.2.4. Coordinación de la comunicación en equipos de desarrollo grandes

Bohem [85] categoriza en cuatro grupos los aspectos o factores que inciden en la industria del software (ver Fig. 2.10) y los clasifica en grados de importancia según su responsabilidad en la obtención del producto: Personas, procesos, proyectos, organización.

Figura 11 Factores que inciden en la comunicación

Basados en esta clasificación, Shahane [86] reconoce 69 factores en total para todos los grupos (ver tabla 1) y que están inmersos en las filosofías ágiles: 30 factores para el grupo personas, 20 factores para procesos, 14 factores para proyecto y 4 factores para organización.

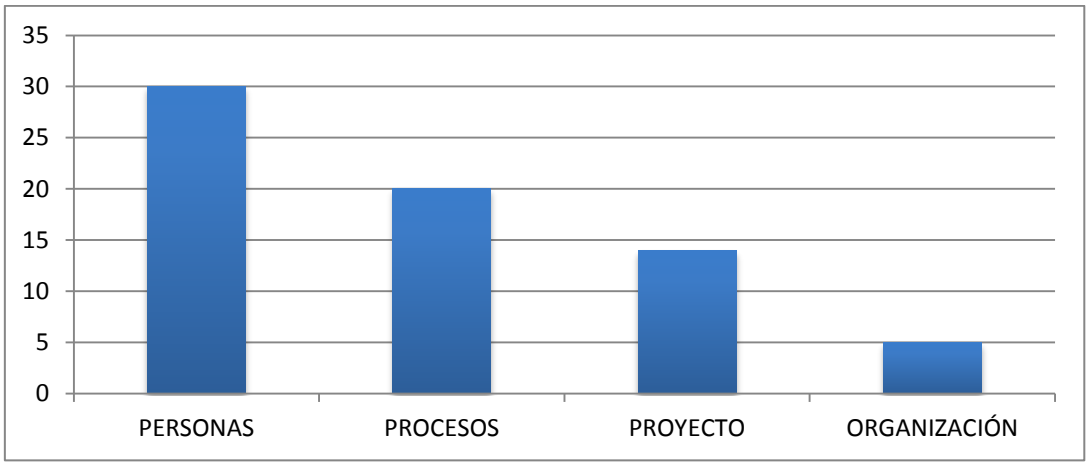


Tabla 1 Factores que inciden en el desarrollo de software

Dado que los aspectos o factores del grupo personas son los más numerosos y tal vez los más relevantes para la industria del software [85] (sin restar importancia a los otros grupos), y por otro lado, como es la comunicación el objeto de esta investigación, es interesante ver que cuando los equipos crecen, generalmente no se planifican prácticas o patrones claros de comunicación [35], simplemente extienden las actividades realizadas en cualquier desarrollo de producto software sin importar el tamaño del equipo.

Esto ha hecho que algunos factores importantes para las metodologías ágiles en la gestión de los miembros del equipo se vean afectados y por tanto afecten también el objetivo del

sistema. Cuatro de ellos son:

- **La resolución de problemas**

Figura 12 Incertidumbres en la comunicación

En equipos de tamaño mediano, los miembros se enfrentan a una gran cantidad de incertidumbres (Ver Fig. 12), que empiezan desde las primeras conversaciones cuando se trata de entender los requisitos del cliente y de ahí en adelante con la cantidad de información que se mueve subsecuentemente por el desarrollo del proyecto [87]. Incertidumbres que se convierten en situaciones problemáticas cuando no existen criterios claros de comunicación, que se olvidan fácilmente en la planificación del proyecto. Si no se ha planificado cómo y cuáles serán los canales, las estrategias y los modos de comunicación, el equipo podría fácilmente no entregar la solución del objetivo propuesto. Esto conllevaría a aumentar los costos del mismo, debido a la volatilidad de la información, mientras se resuelven los problemas de gestión entre ellos los retrasos del proyecto.

Cuando no existen prácticas de comunicación adecuadas, los miembros del equipo fácilmente pueden pasar mucho tiempo tratando de encontrar personas que puedan apalancar las diferentes fases del proyecto [35], y la barrera entre el equipo y el cliente cada vez puede ser mayor debido a la incomunicación.

- **El tratamiento de la información, seguimiento y retroalimentación,**

Los métodos ágiles vinculan al cliente como un miembro más del equipo con el objeto de obtener la información requerida del proyecto en el momento que se necesita, pero además, mantenerlo informado de su progreso [87]. Al mismo tiempo, debe existir una sinergia al interior del equipo para monitorear, y retroalimentarse de los eventos sucedidos [52]. Cuando los equipos crecen, este factor se torna difuso (Ver Fig. 13) debido a que ni el cliente ni los

miembros del equipo son conscientes de las decisiones que se vayan tomando al interior del proyecto [34]. Esto hace que la monitorización del sistema falle, debido a que el cliente se puede olvidar de informar al equipo y entre ellos, los cambios de diseño en sus decisiones iniciales, así mismo los cambios realizados mientras el proyecto avanza, o los nuevos documentos que aparecen producto del trabajo de equipo. Por tanto la comunicación debe ocurrir en todas las direcciones. Puesto que además de informar, el equipo espera la retroalimentación de su trabajo en términos de calidad, claridad y eficiencia.

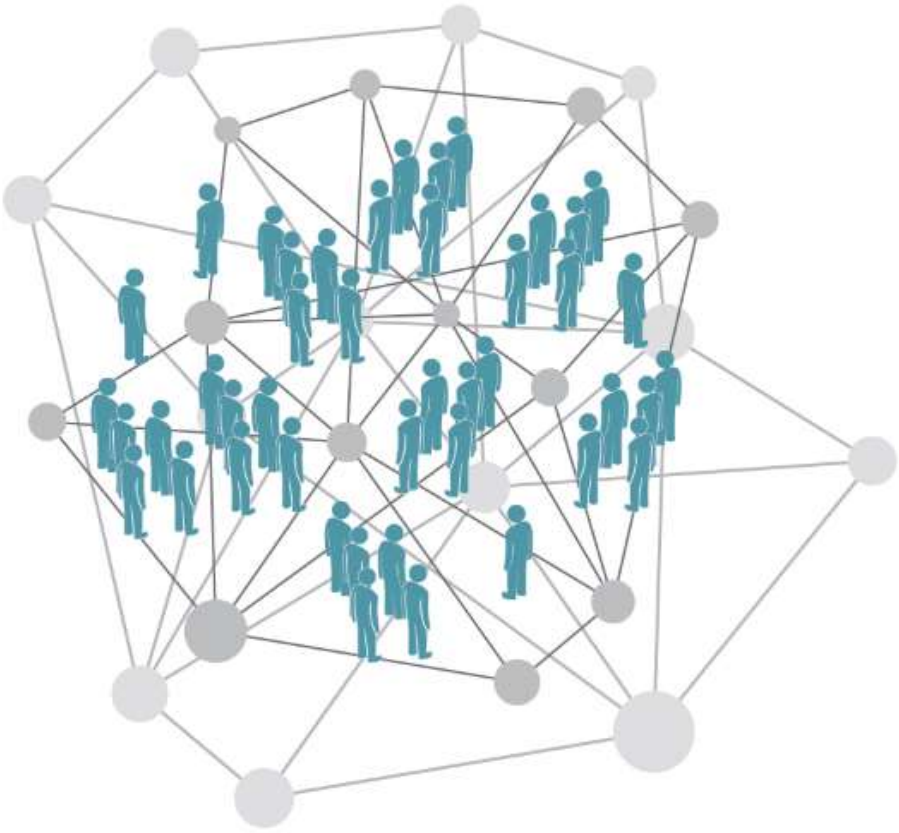


Figura 13 Comunicación cuando los equipos crecen

- **La Construcción de relaciones**

Desde la planificación del proyecto se deben establecer criterios claros de comunicación que permitan estructurar redes sólidas en las relaciones de los participantes del proyecto, que mantengan la sinergia necesaria para cumplir con los objetivos trazados. Estas relaciones permiten tener un equipo motivado, auto-gestionado que pueda lograr el propósito para el cual fue contratado. El encuentro cara a cara en las metodologías ágiles es un factor que permite establecer conexiones productivas en los equipos, y posibilita una comunicación continua entre los miembros del equipo y estos con el cliente [52]. Cuando el equipo crece, este tipo de relaciones se pierde debido a la complejidad de la comunicación por el tamaño del equipo y del proyecto [88]; lo más probable es que la calidad de la comunicación se pierda, es cuando se deben establecer canales eficientes que permitan mantener el equilibrio en los mensajes compartidos por los miembros del equipo.

- **La Coordinación en la toma de decisiones**

Un equipo que se auto-gestiona, entiende que las decisiones son del equipo [52] y que por tanto debe mantener discrecionalidad y coordinación en sus decisiones, es la mejor manera de lograr los objetivos. De ahí que sea necesario establecer estrategias de comunicación, para que cada decisión tomada, redunde en beneficio del proyecto y en el fortalecimiento del equipo.

La comunicación es uno de los valores esenciales de los métodos ágiles [52], tan esencial que Ambler S. [89] considera en sus investigaciones, que la calidad de un proyecto ágil, viene a ser el resultado de técnicas efectivas de comunicación. La Interacción cara a cara entre los miembros del equipo y estos con el cliente, conversaciones de poca formalidad, pero de una interacción social tan importante, facilitan la compresión de los requisitos de usuario al interior del equipo de desarrollo [90]. De ahí que en el desarrollo ágil, no sólo es suficiente generar y recopilar ideas, pensamientos y autoanálisis de grupo; también es relevante, la forma como se imparte la información generada en el intercambio de cada actividad. Prácticas que dependiendo de la forma como se desarrollen, son responsables del éxito o el fracaso del proyecto. La comunicación personal efectiva entre los miembros del equipo y los participantes del proyecto es una de las tareas que más se destaca en los equipos ágiles [91].

Basado en una revisión sistemática (Ver tabla .2) Marjaie y Rathod [33] hacen un listado de características extraídas del ejercicio de la comunicación y los mecanismos de retroalimentación usados por los equipos ágiles en la ejecución de un proyecto:

Interacción de equipo Planeada Interacciones no planificadas Confianza Compromiso Resultados de los proyectos (tiempo, costo, calidad) El alcance del proyecto Creatividad Ubicación La frecuencia de la interacción cara a cara Duración de la interacción cara a cara Eventos y reuniones Documentos memos y cartas Horarios Formas de presentación de gráficos (encuestas, informes de avance) Materiales de gestión de proyectos Noticias, reportajes y anuncios Políticas Las solicitudes de información Trabajo colaborativo Información general (por ejemplo, el teléfono, Listas	Diseño y análisis del diseño La planificación de grupo Confirmación en la comunicación y Respuesta rápidas Comunicación espontánea La formación de equipos Las solicitudes de información Duración de reuniones Sprint Satisfacción de la comunicación Lenguaje común en la comunicación Compromiso organizacional Satisfacción del equipo El desempeño individual Satisfacción del Proceso Frecuencia de la comunicación del equipo

Tabla 2 Factores de la comunicación ágil

A partir de las características planteadas en la tabla 2 se puede definir la comunicación para los métodos ágiles como el acto de transmitir información entre los individuos (miembros de un equipo y cliente) alrededor de un proyecto software y que se hace muy relevante debido a la necesidad permanente de la comunicación en toda la gestión del software: tanto en el desarrollo, como en las operaciones y soporte. Esto quiere decir que los desarrolladores y el cliente tienen que comunicarse permanentemente, los desarrolladores y la gente de otras áreas tienen que comunicarse, los desarrolladores y la gerencia tienen que comunicarse, en fin, todo aquel que se involucre en un proyecto software debe transmitir la información necesaria y pertinente para lograr los objetivos.

Para mantener permanentemente estos contactos, los métodos ágiles utilizan diversos modos de comunicación, Cockburn [92] describe varios modelos de comunicación. La figura 14, compara la efectividad de diferentes modos de comunicación dependiendo del canal de comunicación empleado. Es así como la línea punteada muestra opciones de comunicación cuando se está documentando, la línea continua muestra otras opciones de comunicación que se puede usar al modelar el producto. Estos valores relativos de las opciones son dependientes de la situación, es posible que una conversación o videoconferencia sea más efectiva que hacerlo cara-a-cara con una persona específica, mientras que con otra persona puede funcionar con mayor efectividad otro modo y/o medio.

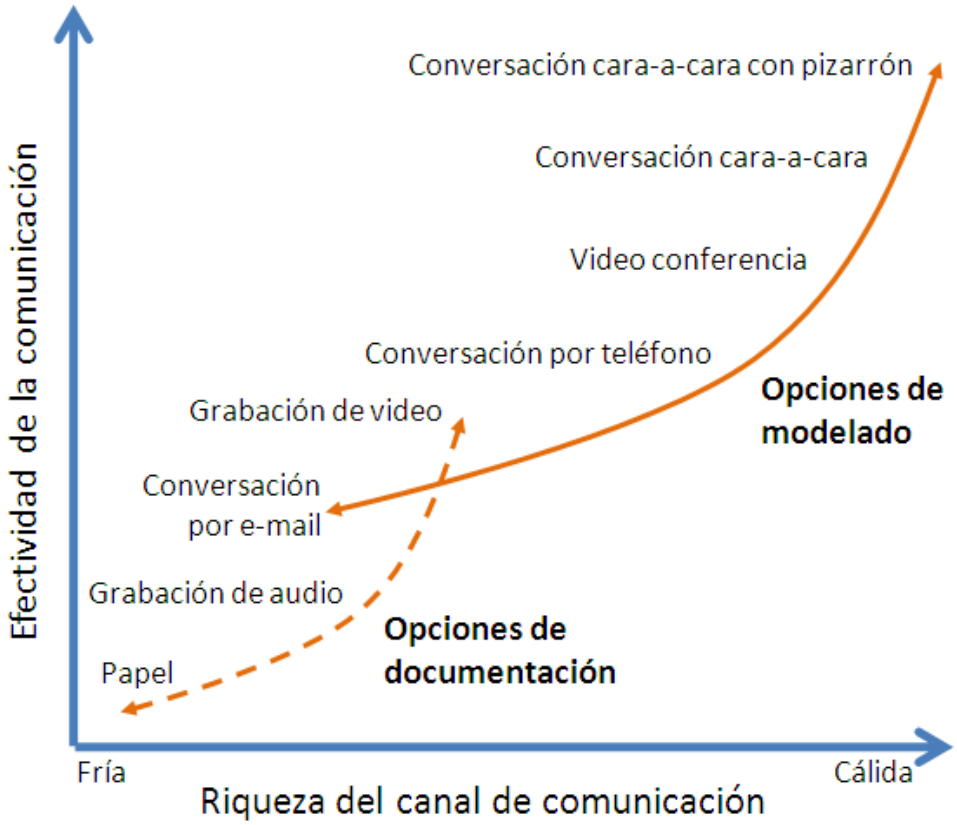


Figura 14 Adaptado de Agile Software Development Alistair Cockburn. Adaptada de Cockburn [92],

Para las metodologías ágiles la comunicación más efectiva es la que ocurre persona-a-

persona, cara-a-cara [90], en particular cuando se trabaja con un esquema de modelado compartido y concertado por los actores del proyecto (cliente – equipo). Esto quiere decir que al alejarse los actores de este esquema y de la forma como se comparte la información, se puede experimentar una caída en la efectividad de la comunicación en el proyecto; con ello, también se pierden otros beneficios como la habilidad de transmitir la información requerida para el proyecto usando otras técnicas además de las palabras, como los gestos y las expresiones faciales [91]. Se pierde la posibilidad de hacer cambios en la voz, el timbre, en el momento de la comunicación; las personas no se comunican sólo con palabras, sino en la forma cómo se dicen estas palabras. Cockburn [92] señala que el orador puede enfatizar lo que dice, cambiando su forma de comunicar: hablando más rápido, más despacio, haciendo pausas o cambiando el tono de voz. La comunicación cara a cara facilita la capacidad de responder preguntas en forma sincrónica (La figura 2.18 muestra el punto que separa la curva de opciones de modelado de la curva de documentación.), con las preguntas, se puede descubrir qué tanto han comprendido las personas de la información compartida.

De ahí que para los métodos ágiles es muy importante que las personas trabajen juntas [65], puesto que cuando las personas trabajan juntas, tanto en el espacio físico como en el temporal, existe la posibilidad que se establezca una comunicación informal, es decir, información indirecta que se transfiere por escuchar conversaciones o simplemente por estar atentos a lo que ocurre alrededor [91]; alcanzando una mayor efectividad en el desarrollo del proyecto debido a que las personas harán lo que sea para terminar el trabajo.

La importancia del principio de comunicación abierta y honesta que promueven los métodos ágiles, genera confianza en la información que se transmite (o en los miembros del equipo que continuamente la proveen). El concepto de quienes comentan que "todos podemos aprender de cualquier otra persona", es importante para el éxito del proyecto en las metodologías ágiles, porque además de romper las barreras que pueden crearse entre los participantes del proyecto, posibilita la generación de valor a partir de diferentes medios y canales de comunicación; cuando los miembros del equipo son conscientes que, cada individuo puede aprender algo de la persona con la que se está comunicando, los hace más receptivos de aquellos que piensan lo contrario. En Ágiles, este principio tiene sus raíces en el valor de la Humildad [52], un valor que una y otra vez prueba ser crítico para los desarrolladores, más aun cuando la comunicación se da en la informalidad, debido a que los miembros del equipo son personas con la experiencia suficiente para asumir con responsabilidad la tarea que se le ha asignado [52].

No sucede lo mismo cuando los equipos crecen, en el momento que se salen de las posibilidades expuestas por los métodos ágiles, La información adquiere cierto grado de complejidad que la hace directamente proporcional al crecimiento del equipo. Esto ha obligado a mejorar la forma como se transmite esa información, a organizar redes formales de comunicación, a diseñar sistemas que permitan establecer parámetros de conversación entre los equipos y a su vez los miembros de los equipos [93]. Pero la informalidad de la comunicación, basada en patrones de interacción, en amistades, en proximidad e intereses compartidos entre los miembros de los equipos, la que ha adquirido bastante madurez en los equipos ágiles, que puede ser de tipo personal o profesional [94], se torna indirectamente proporcional al tamaño del equipo, es decir que si el equipo crece, la comunicación informal se va perdiendo. De ahí que, es necesario establecer métodos y medios formales por el cual se comunicara el nuevo equipo; De no establecerse, cualquier cosa podría entorpecer un mensaje y provocar recepciones imprecisas.

2.2. Estado del Arte

2.2.1. La comunicación en métodos ágiles con equipos de tamaño mediano

Iqbal Aftab et al [30] muestran como un grupo de desarrolladores de software para mantener los parámetros ágiles, utilizan diferentes canales de comunicación para transferir grandes paquetes de información entre los miembros del equipo, actividades que generalmente se realizan a través de listas de correo, repositorios, wikis, reuniones, trabajo de pares, foros de discusión, entre otros. Describe también, como con el creciente interés en el uso de las redes sociales Twitter, Facebook, LinkedIn entre otros, los equipos desarrollaron su propia identidad con el fin de difundir información relacionada con los proyectos de software debido al número de sus miembros.

Pero cuando el equipo crece, dentro del ciclo de vida del proyecto, las etapas iniciales para la construcción del producto son bastante problemáticas; hay una serie de información que se debe compartir y que debe ser muy clara para que el proyecto alcance su objetivo [31]. De ahí que los equipos tengan que adoptar herramientas de trabajo cooperativo apoyados por computador para facilitar el tratamiento de la información, que en la mayoría de los casos no ha dado buenos resultados dado a la complejidad de la misma. Zaychik Vera y Regli William C. [31] afirman que por la cantidad de información que se mueve al interior del equipo se va tornando paulatinamente más compleja [25]. Parece ser que usar herramientas automáticas facilitaría la gestión del equipo y de la información. Los autores proponen un entorno de software al que llamaron CodeLink, una herramienta para el apoyo al diseño como actividad integradora en el desarrollo de un producto software y que busca proporcionar al equipo, un medio para asociar automáticamente los elementos de código específicos con mensajes de correo electrónico. Con esto, se buscó enriquecer la comunicación a través de canales automáticos para los equipos de trabajo sin importar su tamaño.

Existen prácticas desde la perspectiva de la gestión y la comunicación que se quedan cortas debida al tamaño del equipo en el desarrollo de un producto; McChesneya Ian R, Gallagher Séamus [32] describen y analizan las prácticas de comunicación y de coordinación de la ingeniería del software en el desarrollo de proyectos, indican que los modelos existentes en el proceso de software no abordan adecuadamente todas las situaciones que producen información relevante para el desarrollo de un producto; tampoco las actividades del día a día donde los miembros del equipo participan colectivamente cuando realizan prácticas ágiles; sin embargo, es a través de esas actividades que se logran extraer mensajes importantes para el desarrollo del proyecto incluso estrategias para la coordinación del equipo. Proponen un modelo para identificar y articular algunas de las prácticas de interacción y coordinación de los equipos de desarrollo que contribuyan al éxito de proyectos de software. El marco propuesto, también sirve como un modelo útil para la comprensión de las complejidades de los datos de campo obtenidos en la comunicación directa con el cliente, que por cierto son muchas sus dificultades y preocupaciones. La comprensión de las variables que intervienen en esas dificultades y mimetizarlas en el equipo a través de su interacción, es la fase más crítica y difícil en la dinámica del proceso de modelado de un sistema software. Desde esta perspectiva, son necesarias prácticas inclusivas y personalizadas que permita fortalecer las relaciones del equipo; Marjaie Seyed-Ali, Rathod

Urvashi [33] introducen un método basado en la comunicación del equipo con el cliente, fundamentado en prácticas, especifican las necesidades sobre los aspectos cualitativos de la dinámica de sistemas integrándolas a las metodologías existentes para facilitar el proceso iterativo de modelado. Este trabajo propone un marco sistemático con el fin de relacionar los resultados de la fase cualitativa a la fase cuantitativa del proceso de modelado software, pero que se dificulta cuando los equipos crecen en el momento que deban realizar actividades como la recolección y estructuración de la información pertinente y que tienen que compartir para configurar y poner en funcionamiento un producto. Reinhardt Wolfgang [96] menciona cómo algunos estudios destacaron el hecho de que, la comunicación informal ad hoc en los métodos ágiles es significativa en la interacción del grupo. Sin embargo no es posible lograr algún efecto positivo cuando el equipo crece. Por tanto, propone apoyarse con herramientas automáticas basada en el microbbloging entendido como una forma de pequeña escala de los blogs, generalmente compuesto por mensajes cortos y concisos, utilizado para compartir noticias, actualizaciones de estado de correos y mantener las conversaciones inherentes al proyecto.

Basados en los anteriores aspectos, se pude concluir que uno de los grandes retos de los métodos ágiles, es el de lograr su escalabilidad de la comunicación a equipos más grandes, y para ellos se han propuesto diferentes estrategias que desde la perspectiva equipo/proyecto, analizan los patrones de comunicación y la gestión de los miembros del equipo, las relaciones formales e informales en el desarrollo del proyecto, pero se escapan las decisiones de diseño importantes para la generación de producto acordes con los requisitos del cliente.

2.2.2. Arquitectura de Software como estrategia para escalar los métodos ágiles

Las metodologías ágiles han ganado bastante popularidad desde hace algunos años en la industria del software, debido a que frente a las metodologías tradicionales se convirtieron en una muy buena solución para proyectos a corto plazo, en especial, aquellos proyectos en donde los requerimientos están cambiando constantemente. En proyectos de larga duración, con equipos de tamaño mediano, aparecen los problemas de coordinación y comunicación de los proyectos que siguen las metodologías tradicionales. De modo que, para solucionar aspectos de coordinación cuando los equipos crecen, los métodos ágiles se focalizaron en el diseño de software [79], debido a que un buen diseño implica que el sistema presenta atributos de calidad que generarán valor al negocio del desarrollo de software. Si se definen actividades que fomenten el uso de métodos para el diseño, entre ellos la arquitectura, en etapas tempranas del desarrollo de software, se podría aspirar al desarrollo de productos más confiables [65][66].

Respecto al uso de la arquitectura, Constantine y Lockwood [78] ofrecen una serie de consejos, admitiendo que no hay una única forma de enfocar una integración sólida de las metodologías 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 heredadas, modificando las prácticas actuales para incorporar actividades de diseño de la arquitectura" [80]. Con su propuesta Constantine y Lockwood [78], intentan hacer un acercamiento a la escalabilidad de los métodos ágiles, proponiendo además el uso de artefactos de los métodos tradicionales que han demostrado buenos resultados para proyectos complejos; es decir, el resultado es una metodología híbrida formada entre

metodologías ágiles y UP[78], pero que tratándose de las prácticas ágiles, esta propuesta, poco ha favorecido al desarrollo de software debido que al incorporar artefactos de los métodos tradicionales, se pierden los criterios de agilidad, y las entregas tempranas del proyecto. Además no son claras las prácticas de arquitectura más allá de los artefactos básicos basados en el modelo 4+1 Vistas [81].

Basados en prácticas de software, Andreas Kornstädt and Joachim Sauer [95], demuestran que una vista arquitectónica común permitiría solucionar por un lado problemas de comunicación y por otro, atender asuntos de escalabilidad en un proyecto de desarrollo ágil con equipos de tamaño mediano. 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 apoya la idea de que la arquitectura es un artefacto relevante para escalar el proyecto a grupos más grandes.

Las "Historias de desarrolladores" de Rolf Njor Jensen, Thomas Möller, Peter Sönder y Gitte Tjörnehöj [82], análogas a las historias de usuario, son usadas para representar los requerimientos y decisiones de diseño cuando los equipos crecen. Este nuevo componente es agregado a XP con el objeto de plantear una arquitectura basada en las “historias de desarrolladores” que describen los cambios, experiencias, unidades de propiedades visibles para el equipo de desarrollo. 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 expresando 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 canal, que permita representar la descomposición del sistema de acuerdo a los requisitos de calidad para que fluya la información al interior del equipo. Tampoco se integra explícitamente algún método de arquitectura.

2.2.2. Retos de los métodos ágiles en los proyectos ágiles

Como se ha mencionado anteriormente, en los últimos años las metodologías ágiles han irrumpido con fuerza, en 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, apuntan con creciente interés a la implementación de las prácticas propuestas por las metodologías 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 37% en la productividad [83] y un decremento del 44% [84]. Por este motivo, desde las organizaciones que promueven el desarrollo ágil, solicitan la realización de estudios sobre las metodologías ágiles que permitan constatar o reprobar sus beneficios sobre todo cuando deben aplicarse a

proyectos complejos con equipos de tamaño mediano. Por ejemplo y aunque poco reportados en la literatura, E. Hadar y G. Silberman [6], hacen un esfuerzo, por averiguar la productividad en los equipos XP, para ello realizan su investigación en el marco de proceso de un desarrollo de software, encontrando que la productividad del equipo XP 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 [87], 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). Concluyen E. Hadar y G. Silberman, que mientras se mantengan las prácticas ágiles, es posible mantener también la productividad de los equipos aunque el equipo crezca.

La gestión del equipo es el otro reto que deben enfrentar las metodologías ágiles y entre las prácticas, la comunicación debido a que si crece el equipo, crecen también las interacciones y la información compartida por el equipo [21]. De ahí que los métodos ágiles deben proponer los canales efectivos que permitan establecer reglas para la gestión y la comunicación de los miembros del equipo considerando su tamaño. Al respecto, existen aproximaciones de autores como los descritos en párrafos posteriores que han trabajado la comunicación como factor importante en la gestión del equipo, cuando este crece y trata de mantener prácticas ágiles. Pero que, aunque no muestran resultados de medición que permitan vislumbrar la efectividad de sus resultados, son importantes dado que proponen estrategias para que la comunicación, como eje transversal al equipo, para que ésta fluya al rededor del proyecto.

2.3. Arquitectura en Métodos Ágiles

2.3.1 Enfoque CA o C3A

Ethan Hadar & Gabriel M Silberman [6] proponen una metodología basada en la arquitectura denominada Architecture Centric (CA) o C3A, Una arquitectura basada en 3 niveles que se fundamentan en el diseño de artefactos con un alto nivel de abstracción en un mínimo rango de detalles, para no redundar en la arquitectura ni demorar los tiempos en el desarrollo. CA usa artefactos de UML [73], recorre todo el ciclo de vida del producto hasta entregar resultados funcionales al usuario. Una explicación de cada nivel se expone en los siguientes párrafos:

El Nivel 0 considerado como el nivel más 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.

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 15)

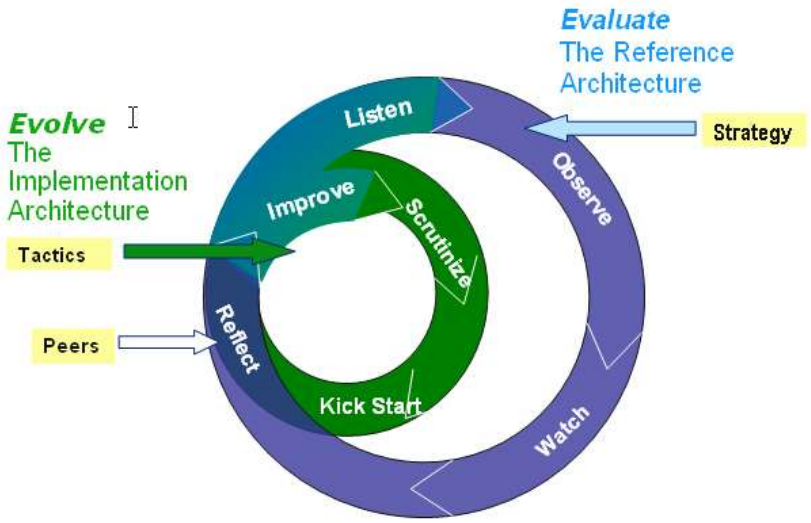


Figura 15 (Tomado de Agile Architecture Methodology: Long Term Strategy Interleaved)

Cada una estas etapas cumple 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.

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 [74] 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 [75].

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, ***AGATA*** 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.

D. Reifer et al. [19] 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 más

pequeños, aclarando que, dependiendo del tamaño del software, la comunicación y los otros principios del postulado ágil [76] se pueden hacer tan complejos que en la relación cliente y equipos XP pueden perderse los objetivos del sistema [77].

Larry L. Constantine & Lucy A. D. Lockwood [78] en su documento “Una guía práctica para modelos y métodos de diseño centrado en el uso”, comentan que "los desarrolladores”, necesitan ser empoderados con mejores herramientas y técnicas que les permitan establecer modelos de comunicación para entender de la mejor manera los requisitos de usuario y poder entregar software altamente utilizable. En su investigación, dan una mirada amplia a los procesos y cuestiones relacionadas con el diseño de los casos de uso, incluyendo la construcción de prototipos, ayuda en línea, acomodación de la progresión con el objeto de verificar los mensajes que trascienden el desarrollo del proyecto para entregar una solución basada en la usabilidad. Las ayudas en línea y las conversaciones a través de estos medios son utilizados para compartir la información en torno del sistema.

Aftab Iqbal et. al [30]. Afirman que muchos de los problemas del software cuando los equipos son grandes es el de la comunicación. Para medir su incidencia dentro los equipos de desarrollo, proponen una metodología basada en datos vinculados para relacionar e integrar es datos a través de repositorios software sin haya ambigüedad en la transmisión de la información. Además usan tecnologías de Web Semántica para representar los datos del repositorio de software utilizando RDF (Resource Description Framework) como el núcleo. Una vez modelado en RDF, los mensajes (datos) pueden ser fácilmente integrados, indexados y consultados mediante el SPARQL Query7 estándar y herramientas asociadas. Por último, los mensajes se pueden publicar en la Web utilizando principios de datos vinculados que permitan a terceros descubrir y posteriormente rastrear el conocimiento, y también permitir la interconexión con información de manera remota en la red. (ver Fig 16)

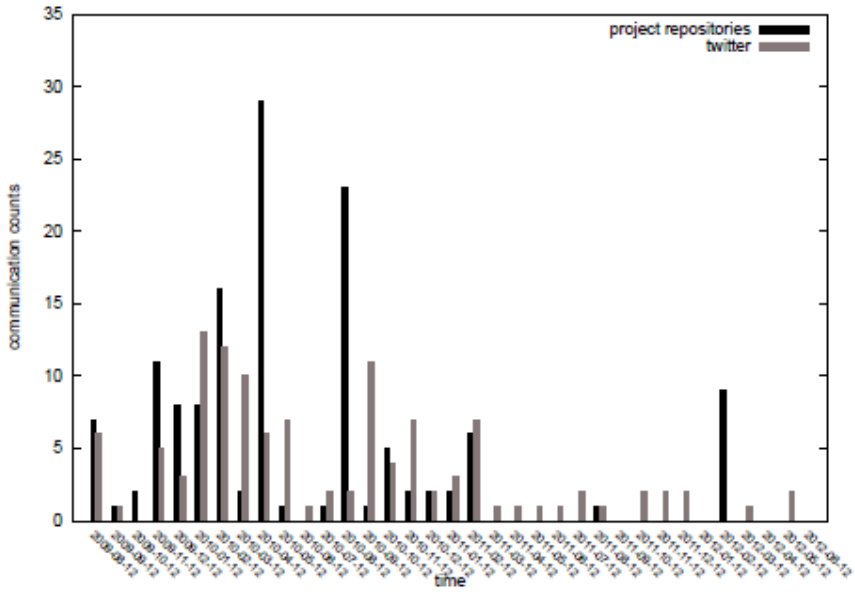


Figura 16 Comunicación social entre los desarrolladores de software En diferentes canales de comunicación. (Tomado de [30]

Como resultado, muestran que existe una correlación muy baja entre los desarrolladores de software a través de diferentes canales de comunicaciones (redes sociales como medio). Además, demuestran que la comunicación entre los miembros del equipo usando medios tecnológicos es relativamente baja respecto de la comunicación tradicional (es decir, listas de correo, repositorios de seguimiento de fallos, cara a cara. etc.).

Con base en esto, los investigadores proponen la centralización del proyecto en canales de comunicación que unan los miembros del equipo, integren los mensajes en artefactos relacionados con la aplicación. Estas herramientas abrirán nuevos desafíos que permitan analizar el impacto de la respuesta del usuario final sobre el éxito / fracaso de un proyecto.

Hace casi una década escritores de Manifiesto Agil declararon su objetivo como el descubrimiento de mejores maneras de desarrollar software a través de un cambio en los principios de desarrollo de software.

El manifiesto ágil cambió el sistema de valor de apreciar procesos y herramientas a valorar más a "Individuos e Interacciones".

El término "interacción" como se expresa en el diccionario de Oxford expresa una acción recíproca o mutua. La Interacción Humana ocurre para dos individuos o dos grupos de personas en presencia de una plataforma o contexto adecuado. Probablemente, la plataforma más importante que podemos señalar es una plataforma de comunicación, que a su vez asegura la suficiente frecuencia de comunicación, del equipo en el proyecto.

Seyed-Ali Marjaie y Urvashi Rathod [33] en su trabajo sobre la comunicación en proyectos de software ágil, afirman que, la comunicación es parte esencial en el proceso de colaboración, también explican que el término "Colaboración" sugiere un acto de trabajo conjunto. No se puede imaginar la colaboración entre individuos o equipos sin comunicación como una plataforma facilitadora común.

Basados en la flexibilidad propuesta por el manifiesto ágil, sostienen que la capacidad de respuesta al cambio necesita una plataforma de comunicación eficaz que además soporte la naturaleza recíproca o mutua de la interacción, la colaboración y la capacidad de respuesta al cambio junto a un mecanismo de retroalimentación adecuado. Debido a que un mecanismo de retroalimentación garantiza la mutua rendición de cuentas de los participantes para lograr los valores básicos de la agilidad en el desarrollo de software. Por lo tanto, cuando los equipos crecen, manifiestan que un mecanismo de comunicación y retroalimentación es parte esencial para soportar las prácticas de las metodologías ágiles.

Comentan Seyed-Ali Marjaie y Urvashi Rathod [33] que en la gestión de proyectos tradicionales, los modelos de comunicación no satisfacen las necesidades de interacción de los equipos. Aunque asignen diferentes métodos de comunicación para satisfacer los requisitos en la trasmisión de sus mensajes, el impacto de adoptar nuevos enfoques de comunicación es desconocido en la productividad y calidad del producto antes de experimentarlo en la realidad, más aun cuando la inestabilidad de los requisitos aumentan la ambigüedad del proceso de comprensión mutua; de ahí que la comunicación personal sea una reacción natural al alto nivel de ambigüedad en el desarrollo de software ágil.

Cuando los equipos son grandes muchos otros factores aparecen y hacen que la comunicación sea más crucial. Hay que entender que un producto de calidad resulta efectivo

cuando el conocimiento de la aplicación está ampliamente extendido a través del equipo, y que con equipos más pequeños es más fácil propagar este conocimiento.

Si bien es cierto en su estudio abordan la ambigüedad causada por la variabilidad de factores críticos en la comunicación en proyectos de desarrollo ágil, proponen construir un modelo dinámico para entender y capturar la variabilidad de los factores involucrados en el sistema de comunicación y los retos que persisten en los proyectos de desarrollo de software ágil en equipos de tamaño mediano; el trabajo solo se limita a la identificación de dichas variables pero no formulan una de política clara para acotar esos problemas y facilitar la productividad de los equipos ágiles.

AGATA, a través de sus canales establece políticas y criterios para mejorar la interacción entre los equipos de desarrollo ágil, de tal manera que los mensajes que se comparten entre ellos, tengan los significados entregados por el cliente y a través de los miembros del equipo hasta alcanzar los resultados esperados con el producto.

En un análisis sobre la comunicación y el ciclo de vida del proyecto, Vera Zaychik, William C. Regli [31] afirman que los medios electrónicos de comunicación como el correo electrónico, mensajería instantánea, herramientas cooperativas de trabajo, se han convertido elementos del día a día usado por equipos de tamaño mediano para mantener su comunicación, pero que carecen de apoyo para determinados factores del proceso de desarrollo de software y de la capacidad de apoyo basado en el contexto al proceso de trabajo colaborativo, relegándose la comunicación informal que contiene una gran cantidad de Información relacionada con el proyecto. Cuantos más desarrolladores se apoyen en las comunicaciones, mayor será la cantidad de datos que pueden estar disponibles en las comunicaciones para acceso futuro.

En este sentido, los investigadores introducen una nueva forma de integrar activamente herramientas de trabajo cooperativo apoyadas por computadora (CSCW) como correo electrónico, con el proceso de desarrollo de diseño de software. Para acceder a la información de contexto dentro del desarrollo se acoplo el cliente de correo electrónico con el software en el entorno de desarrollo y los mensajes siguen una arquitectura denominada Codelink . (ver figura 17)

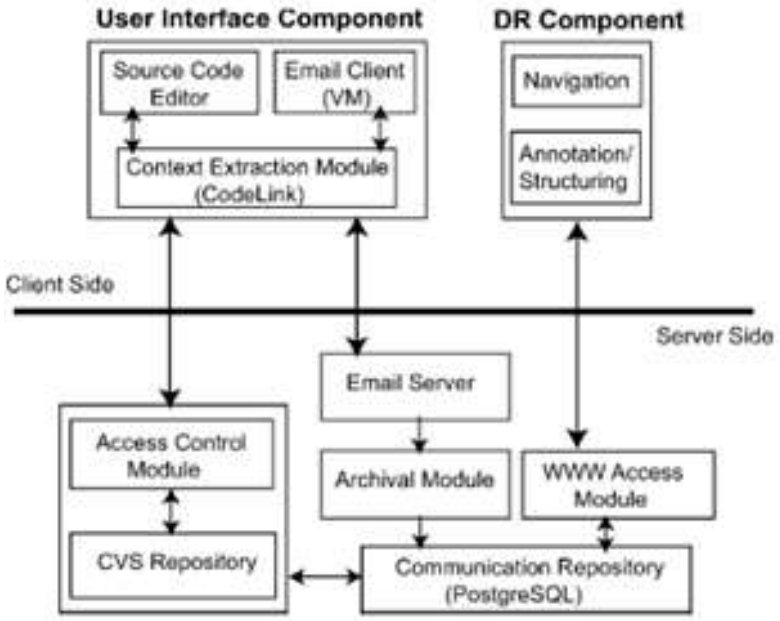


Figura 17 Arquitectura Codelink (Tomado de [31])

Codelink [31] consta de varios servicios y módulos que responden a las siguientes actividades:

- En el lado del usuario, un extractor de contexto y controlador denominado MIME, responsables de permitir el envío y la recepción de mensajes con referencias.
- Un navegador web permite el acceso a la navegación en línea / interfaz de búsqueda al repositorio de comunicaciones, y a la historia de los archivos fuente anotados con mensajes.
- En el lado del servidor, varios servicios permiten el archivado de los mensajes enviados, el almacenamiento de instantáneas. Todos los componentes del servidor excepto para la interfaz web pertenecen a un usuario Unix creado específicamente para este propósito.

Como resultado al usar esta arquitectura los investigadores afirman que la mayor parte del correo electrónico y de mensajes no fueron bien entendidos debido a que necesitaron una mayor explicación por parte del emisor; la mayoría de los problemas del usuario fueron causados por su desconocimiento.

Cabe aclarar que una debilidad importante de las aplicaciones existentes de Computer Suported Colaborative Work (CSCW) es que los usuarios tienen que deben involucrar nuevas herramientas, en lugar de usar las comunes; esto provoca una interrupción del proceso de diseño y posiblemente rechazo de la herramienta [33].

Además, las herramientas deben ser adoptadas por todos los miembros del grupo, de lo contrario, habría dificultades en la colaboración por parte de aquellos que no se acojan a esta dinámica, evento conocido comúnmente como masa crítica o el dilema del prisionero [29].

Para evitar estas dificultades AGATA a través de sus prácticas, dinamiza actividades como la intervención activa del cliente en cada una de las etapas del proceso; donde puede

aportar ideas y opinar sobre los resultados que se le van entregando progresivamente. La comunicación de esta manera fluye a través de los miembros del equipo y a su vez pueden priorizar las tareas es decir saben con certeza cuáles tienen un mayor peso y cuáles resultan secundarias o, incluso, innecesarias. Esta distinción ayuda a centralizar esfuerzos y a unificar criterios de actuación a través de los canales de comunicación propuestos por AGATA.

2.3.2 Propuestas para escalar métodos ágiles

Además de los trabajos anteriores, existen propuestas que han intentado a través de artefactos, medios y técnicas, escalar los métodos ágiles cuando los equipos crecen. Aquí describimos las más relevantes:

- **Scrum de scrums :**

Scrum of Scrums [98] es un equipo virtual compuesto por representantes de los equipos Scrum que colaboran para integrar y enviar un producto (ver Figura 2.18). Los Scrum Masters de los equipos Scrum generalmente funcionan como intercomunicadores con el equipo Scrum de Scrums cuya tarea es la controlar impedimentos, progreso, actividades de los equipos Scrum respondiendo siempre a las tres principales preguntas del Daily Scrum ¿Qué hiciste ayer?, ¿Qué harás hoy?, ¿Hay algún impedimento?. [98]

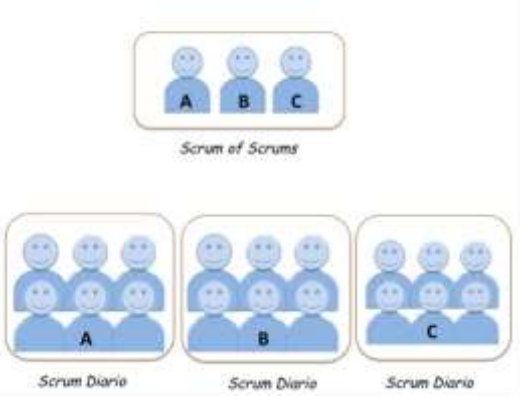


Figura 18 Scrum de scrums (Tomado de [98])

Para la gestión, Srucm de Scrums se apoya en un equipo formado por los ejecutivos de alto nivel (EAT). Quienes con los Srcum Master definen las funcionalidades de los equipos entre ellas:

- ✓ Reunirse una vez al día después de que todos los sub-equipos involucrados hayan llevado a cabo su Stand-up diario.
- ✓ Reflejar el nivel de equipo en los Stand-up diario en una conversación de seguimiento de 15 minutos (a menudo llamado el estacionamiento) para dar cuerpo a los impedimentos y / o posibles soluciones. Aquellos que no participan pueden volver al trabajo.
- ✓ Eliminar los impedimentos para equipos individuales si es posible, si no se les plantean a la siguiente Scrum de Scrums y así sucesivamente hasta que los obstáculos se eliminen o se deleguen a la EAT (equipo de acción empresarial).

- ✓ Trabajar de la mano con el EAT para ayudar a coordinar las mejores prácticas en toda la empresa. [99]

Aunque ha sido una buena aproximación para escalar los métodos ágiles, de todas formas este método de escalamiento manifiesta que el direccionamiento o implementación de un equipo Scrum de Scrums es considerado como algo crítico de llevar a cabo para cualquier equipo de desarrollo, porque "usualmente encontramos que limita el número de vías de comunicación necesarias para transmitir información relevante para el éxito de la empresa". Esto hace que no todos los mensajes compartidos entre los equipos obtengan los resultados esperados. También, hay funcionalidades que son ajenas al Scrum de scrums entre ellas:

- ✓ Desarrollar el backlog. Sin embargo, pueden agregar historias.
- ✓ Descomponer el backlog
- ✓ Coordinar historias
- ✓ Planear lanzamientos
- ✓ Manejo de las liberaciones

Lo que le hace difícil la gestión del equipo para atender los requisitos del cliente. [98], que a diferencia de AGATA, donde en sus prácticas incluye un par de canales por donde fluye autónomamente la información que necesita ser controlada. Los equipos necesitan trabajar holísticamente debe existir una relación inter-equipos, donde la misma auto-organización esta pendiente del trabajo desarrollado por todos, Un equipo de arquitectura se encarga de dinamizar y liberar tareas para ser desarrolladas. Los equipos giran a través de ese equipo de arquitectura lo que hace más fácil la gestión y el desarrollo del proceso.

- **Nexus**

Propuesta desarrollada por Ken Schwaber [100], y la define como una unidad de desarrollo que se compone de 3 a 9 equipos Scrum junto con un equipo de integración Nexus, trabajando en torno a un Product Backlog (ver Figura 19).

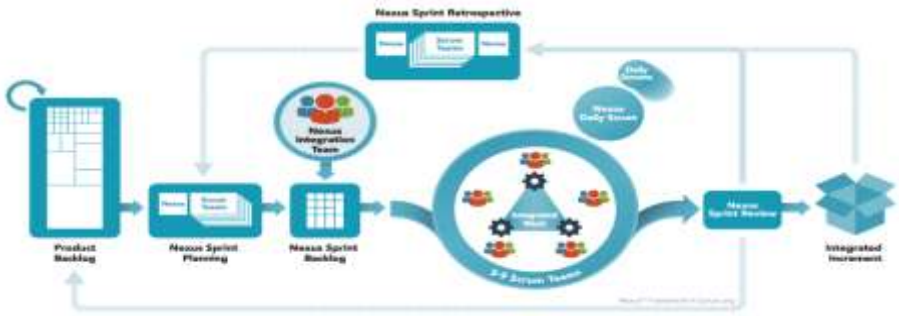


Figura 19 Nexus (Tomado de [100])

El equipo de integración Nexus sigue la misma filosofía de Scrum de Scrums, donde escalar mientras se entrega software debe ser un proceso en el que se eliminan impedimentos al mismo tiempo que se miden las limitaciones que permiten seguir creciendo; para ello aprovecha el conocimiento y experiencia de las personas que forman el sistema.

Los beneficios de Nexus son: Organizar equipos para maximizar la productividad, Organizar

a la gente en equipos adecuados para optimizar el esfuerzo, Enseñar a los Managers como organizar y gestionar un número muy grande de personas para construir software rápidamente, además ayudarles a detectar anomalías en la productividad, proporcionar prácticas para resolver las anomalías, presentar un patrón que permite auto-organización para un gran número de desarrolladores.

Como Nexus, proporciona unas reglas de uso que no tienen en cuenta muchas de las estructuras existentes en empresas ya establecidas, esas reglas han dado lugar a que muchas de las actividades no integran al equipo en torno del producto tal como lo requiere el cliente. Nexus sigue siendo Scrum a escala por tanto los problemas de comunicación no son claramente controlados y no se tienen claros los canales por los cuales fluirá la información para ser eficientemente controlada. [101]

En AGATA el cliente hace parte del equipo. Aunque el equipo es grande, sin embargo, tiene unas reuniones permanentes con un equipo de arquitectura que continuamente le está mostrando los resultados del proceso, quienes a su vez cuentan a todo el equipo los sentimientos del cliente frente al producto liberado. Esto hace que en cada iteración, los resultados de la retrospectiva, sea conocida por todos y las decisiones para continuar con el proyecto, sea también con la participación de todos. De esta forma, la comunicación que fluye a través de los canales: la arquitectura y el cara cara, es la que debe ser relevante al proyecto. De esta manera, es el equipo mismo quien se compromete a trabajar unido para alcanzar los objetivos propuestos desde las fases iniciales del proceso.

- **Disciplined Agile Delivery (DAD)**

DAD fue creada por Scott Ambler [89] un framework híbrido que se basa en los principios ágiles, y combina prácticas extraídas de Scrum, XP, Kanban, Lean y sobre todo conceptos de DevOps, entrega y despliegue continuo. [102]

DAD se compromete a ayudar a las empresas a implantar los valores y principios ágiles no solo en la fase del desarrollo, sino también desde la definición de requisitos, hasta el despliegue del software y contacto con el cliente. Para hacer esto, DAD parte de la premisa de que no todas las empresas son iguales y no todas las prácticas van a ser igual de útiles en todas las empresas. Por eso DAD se basa en establecer una serie de metas que hay que cumplir, ofreciendo varias estrategias para ello. Además, brinda unos consejos para decidir qué estrategia funcionarán mejor en la empresa. DAD propone también elegir entre varios ciclos de vida (basado en Scrum, o en Lean, o un ciclo de vida con entrega continua o exploratorio) (Ver Figura 20). [102]

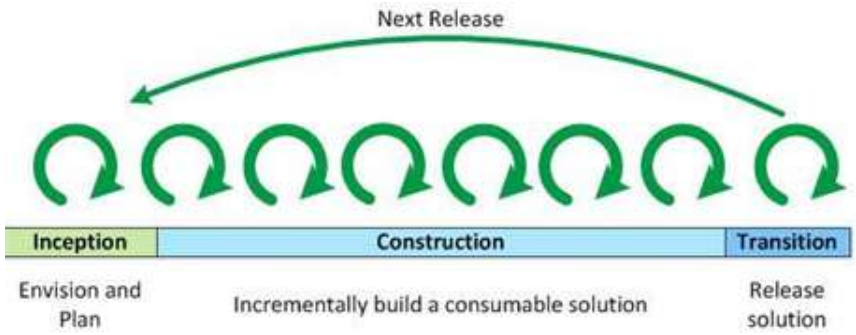


Figura 20 DAD (Tomado de [89])

Aunque DAD propone estrategias para cumplir con los parámetros ágiles dependiendo de la organización, no establece la forma de cómo se deben seguir estructuralmente, simplemente lo remite al enfoque metodológico en el que se apoya esa estrategia. Tampoco establece criterios claros para la gestión de los equipos ni el control de la comunicación entre ellos. DAD asume que un equipo grande se organizará en sub-equipos y continuará con las prácticas ágiles que más le convenga; pero seguir un método ágil del cual se sabe que no funciona en equipos de tamaño mediano, sino está debidamente gestionado, la entrega de producto puede fracasar tal como lo requieren las metodologías ágiles . [103]

- **SAFe: una framework ágil para organizar grandes empresas**

SAFe desarrollada por Vector ITC Group [104] es un framework que busca implementar la agilidad en grandes empresas: a nivel de organización y de equipo; está enfocado con Scrum y Lean Development.

Para su implantación distingue tres niveles de abstracción que hay que coordinar entre sí: el nivel de equipo, nivel de programa y nivel de portfolio. Cada uno de estos niveles posee un Backlog (team Backlog, program Backlog y portafolio Backlog), en los que se incluyen en función de ellos, "historias" con diferentes grados de abstracción (ver Fig. 2.21). [104]

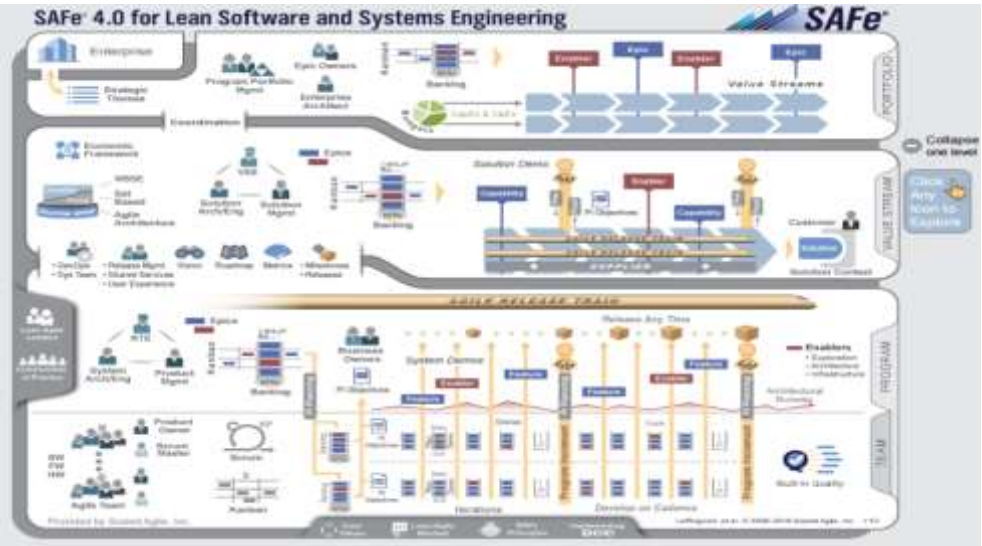


Figura 21 SAFe (Tomado de [104])

Todos los equipos SAFe son equipos ágiles, que se organizan dependiendo de su funcionalidad: equipo de Sistemas, equipo de arquitectura y equipos de desarrollo (que SAFe reconoce como "Equipos Agiles").

Un Equipo de Sistemas es un equipo especializado responsable de mantener el entorno de desarrollo utilizado por los equipos ágiles y de probar soluciones de extremo a extremo.[104]

Los equipos ágiles conformados por 5 a 9 personas trabajan en sprints de dos semanas utilizando XP (Extreme Programming) y tienen las habilidades que necesitan para definir, desarrollar, probar y entregar producto. El backlog de cada sub–equipo consta de elementos

retirados del backlog del sistema y la longitud de los sprints es sincronizada por todos los otros equipos en el mismo "tren de lanzamiento ágil". La idea de SAFe es basar las actividades de desarrollo de rutina en una cadencia rápida y sincrónica, un ritmo regular y predictivo de eventos importantes que ayudan a coordinar la variabilidad inherente en el desarrollo del sistema. [105]

Debido a que SAFe basa sus criterios en Scrum de scrums y además aprovecha prácticas de Lean y de XP, no presenta unas directrices que permitan una gestión clara del equipo, prácticas como la refactorización y la valentía no son permitidas dentro de SAFe, porque los equipos no funcionan de forma independiente y autónoma. Cada uno tiene que esperar la asignación de tareas, tiempos y responsabilidades dadas por el Director del proyecto y depender de la velocidad del resto de los equipos. [106].

Desde sus propósitos iniciales, la visión de AGATA es la de establecer herramientas para equipos de desarrollo que soporten la gestión y enfrenten los desafíos de la comunicación. AGATA, extiende valores como la autonomía, la auto-organización de los criterios ágiles a equipos de tamaño mediano; lo que permite que cada equipo trabaje independientemente, pero coordinado por un equipo conformado por un representante de cada sub-equipo, que continuamente está dinamizando las prácticas del modelo.

- **Xp/Architecture (XA)**

XA [41] 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].(ver Figura 22)

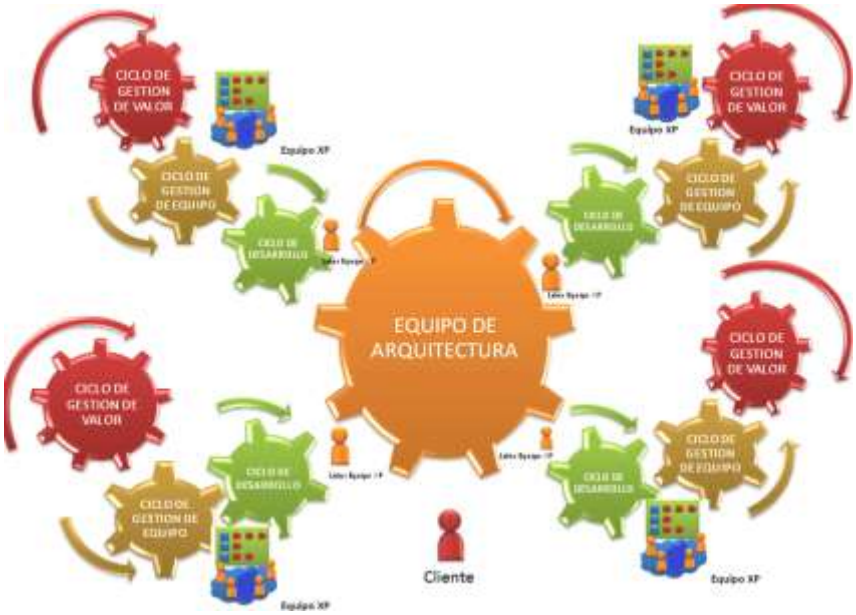


Figura 22 El ciclo de vida de Xp/Architecture. (Tomado de [41])

Al igual que los métodos ágiles (XP), XA propone valores y prácticas, extendiendo su definición para equipos de tamaño mediano [108]. La arquitectura se convierte en XA la herramienta más importante del equipo para resolver los problemas del equipo en cuanto a su tamaño, ya que los sub-equipos deben girar coordinadamente alrededor de los artefactos propuestos por la arquitectura para resolver las situaciones presentadas por la complejidad del proyecto.

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, 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 [107].

Teniendo en cuenta que todo modelo Holístico se basa en sus inter-relaciones, 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 prácticas 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 más 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 [109]; 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.

Si bien se ha definido el proceso XA, 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**,** y aunque en sus valores mantiene la comunicación como un criterio importante para la transmisión de la información del equipo, al igual que las metodologías ágiles, no presenta un modelo que permita la coordinación de la comunicación entre los miembros los equipos. Aspectos que se fortalecerían si el modelo mostrara actividades de gestión en la dinámica inter-grupal de los equipos. (el proceso completo se anexa al presente documento)

AGATA involucra prácticas de Scrum, un modelo para la gestión de equipos que ayuda sincronizar sus integrantes incluyendo al cliente, a través de prácticas que dinamizan el trabajo intergrupal. AGATA mejora la comunicación en el equipo sin importar su tamaño, dos canales de comunicación que se encargan de gestionar la información que debe ser conocida por todos de la mejor manera para que el proyecto avance.

A diferencia de los anteriores métodos, AGATA emplea de la mejor manera los metodologías ágiles: Scrum para la gestión y Extreme Programming para la producción. Usa de sus

prácticas que al fusionarse servirán para entregar producción de calidad. En la fase de ejecución, AGATA lleva a cabo una secuencia de mejoras evolutivas a través de prácticas eficientes. El flujo de trabajo sigue siendo el mismo de los métodos ágiles usados con la inclusión de algunos elementos propios de AGATA, que permiten alcanzar al equipo un desempeño holístico, mejorar la gestión y la comunicación, aunque el equipo crezca. La dinámica de AGATA ayuda a motivar al grupo de desarrollo y al equipo coordinador a realizar análisis retrospectivos para incorporar mejoras al proceso a medida que se avanza en el desarrollo del proyecto.

Para consolidar los criterios ágiles en equipos de tamaño mediano, AGATA se fundamenta en tres actividades que soportarán las actividades desarrolladas por el equipo:

AGATA = Comunicación (Scrum (XA, XP), Canales (Arquitectura, Modelo Holístico))

1. Arquitectura en Proyectos Ágiles (Arquitectura)
2. Modelos de Comunicación en Proyectos Grandes (Comunicación)
3. Composición de procesos a partir de procesos Ágiles (Gestión)

Cuando se trata de escalar los métodos ágiles, hay eventos que se deben tener en cuenta para mantener las prácticas ágiles sin importar el taño del equipo. Un equipo grande trae consigo una serie de variables que deben ser controladas para guardar las relaciones inter-grupales e intra-grupales que permitan al equipo trabajar sincronizadamente para alcanzar su objetivo. En la tabla 3, se resumen estas actividades que fueron tomadas del análisis de seis (6) propuestas anteriormente descritas para escalar las metodologías ágiles. . (Se anexa documento técnico sobre estudios para escalar los métodos ágiles basados en la arquitectura)

- **Resumen comparativo de AGATA con otros métodos que escalan los métodos ágiles**

En el siguiente cuadro, se muestra una comparación entre los métdos que propone la literatura para escalar los parámetros ágiles a equipos de tamaño mediano con los factores que basados en esta investigación permiten mejorar los resultados de la comunicación entre los miembros del equipo

Factores	Scrum de scrums	Nexus	DAD	SAFe	Xp/Architecture	AGATA
Gestión Ágil del equipo	X	X	X			X
Canales de comunicación						X
Sincronización del equipo ágil	X	X	X	X	X	X
Trabajo Holístico					X	X
Centrado en la Arquitectura					X	X
Modelo Empírico					X	X
Escala métodos ágiles	X	X	X	X	X	X

Tabla 3 Comparativa métodos que escalan

Capítulo 3

La función de un buen software es hacer que lo complejo aparente ser simple"
-- *Grady Booch*

AGile/ArchiTecture in Action (ÁGATA)

3.1 Descripción General

AGATA es un proceso software holístico y gestionado para el desarrollo de software ágil centrado en la arquitectura y orientado a equipos de tamaño mediano, el cual introduce prácticas y canales de comunicación con el fin de abordar el problema de la comunicación en equipos de tamaño mediano [68].

AGATA es un proceso que se basa en los valores y principios propios del desarrollo ágil, su estrategia de gestión se basa en el marco de procesos Scrum [55] y las prácticas de la programación extrema (XP) [5]. Con el fin de abordar la complejidad de la organización y la comunicación AGATA incluye prácticas de arquitectura de XP/Architecture (XA) [41], y establece unos canales cara a car especiales para facilitar la comunicación técnica y de gestión para todo el equipo de desarrollo. (ver Fig. 3.1).

Uno de esos canales fundamentales es la arquitectura, bajo la responsabilidad de un equipo central cuyo principal rol es la de dinamizar y motivar a la organización del equipo de desarrollo alrededor de una solución central (arquitectura).

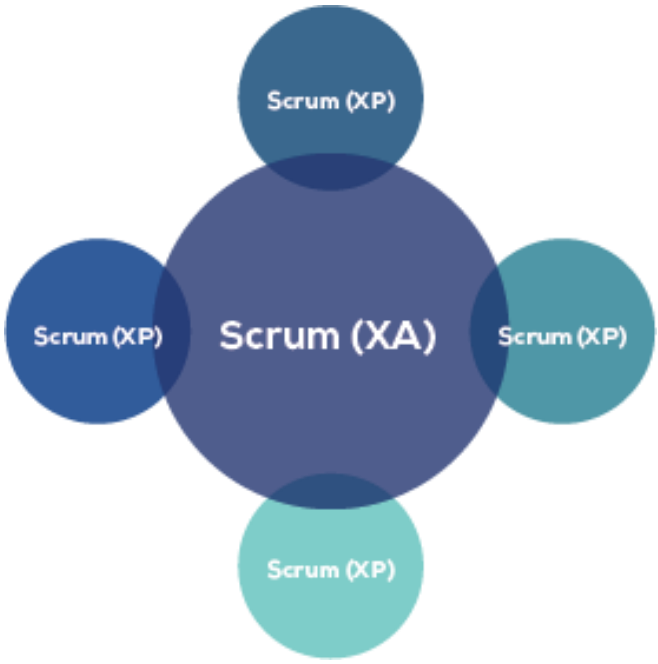


Figura 23 AGATA

3.2 La visión de AGATA

AGATA propone desde la perspectiva de la gestión, la comunicación intergrupal, el desarrollo centrado en la arquitectura; la adecuación de tareas que debe realizar el equipo de desarrollo (16±4) para alcanzar el objetivo del proyecto bajo enfoques centrados en valor propuestos por las metodologías ágiles, aplicando un conjunto de buenas prácticas para facilitar la comunicación y obtener los mejores resultados de un proyecto.

AGATA motiva a la auto-organización del equipo de trabajo debido a su tamaño, fragmentándolo en varios equipos interconectados a través de dos sistemas de comunicación que facilitan la redundancia, gestionando al equipo desde un centro de gestión y desarrollo central para que cumpla efectivamente con las demandas del proyecto.

AGATA se caracteriza por sus valores intrínsecos que se reflejan en cada iteración, los eventos que se realizan dentro de ellos y los resultados obtenidos en esas iteraciones

Figura 24 . Principios de AGATA

AGATA mantiene unos principios que posibilitan la gestión del equipo desde una perspectiva holística, para alcanzar el desarrollo de un producto d calidad (ver Figura 24) a saber:

Adaptación: porque tiene la capacidad de cambiar y aprender de la experiencia [55]. Entendido también como una red dinámica de varios equipos actuando en paralelo, constantemente y reaccionando con los resultados y/o la acción de todos los agentes involucrados por AGATA. El resultado total proviene de un gran número de decisiones hechas en algún momento por cada equipo de manera individual.

Control empírico: Basado en la inspección y adaptación continua en función de los resultados que se van obteniendo después de cada iteración y del contexto del proyecto. AGATA entiende que es más sencillo para el cliente comprender los resultados del producto a medida que se va desarrollando, puesto que le implicaría menos esfuerzo cada vez que debe tomar decisiones respecto del producto final.

Transparencia: Para AGATA es importante que los aspectos más relevantes sucedidos durante el proceso, sean visibles por todos los responsables del proyecto. De ahí que a través de los canales de comunicación propuestos por AGATA, se definan claramente dichos aspectos, bajo criterios comunes, significativos para todos los miembros del equipo (incluido el cliente). Esto hace que todos compartan el mismo lenguaje, los mismos significados y el mismo parecer al momento de liberar el producto.

Comunicación: Es trascendental durante todas las etapas del proceso, para ello AGATA propone como canales para la gestión de la comunicación: la arquitectura y la comunicación cara a cara (intra e intergrupal).

La arquitectura y la comunicación cara a cara extendida por AGATA, como canales de comunicación, buscan la inspiración que va ligada a la parte emocional de cada uno de los miembros del equipo, por otro lado, más que simplemente comunicarse, lo que se buscan es lograr conexión entre cada miembro y el proceso, es decir “hacer lo que se debe hacer, en los tiempos propuestos para lograrlo”.

Gestión del equipo: El equipo AGATA se auto- organiza en sub-equipos (2 a 9 Sub – equipos) [26], que giran en torno a un equipo central de Architecture, los Sub- equipos, se auto – gestionan realizando actividades tales como: seleccionar elementos (peticiones de cliente) de una lista priorizada. Acuerdan objetivos colectivos respecto al objetivodel sprint (iteración en términos de scrum), se adaptan a los cambios en cada iteración, realizan reuniones breves para inspeccionar su progreso y ajustan los pasos necesarios para completar con el trabajo pendiente. Revisan el sprint con los diferentes Stakeholders (interesados e involucrados en el producto) y realizan una demostración de lo que han desarrollado. Cuando estas actividades están bien entendidas por el grupo, las sinergias que se producen en el proceso, facilitan el alcance de los objetivos

3.3 El proceso de AGATA

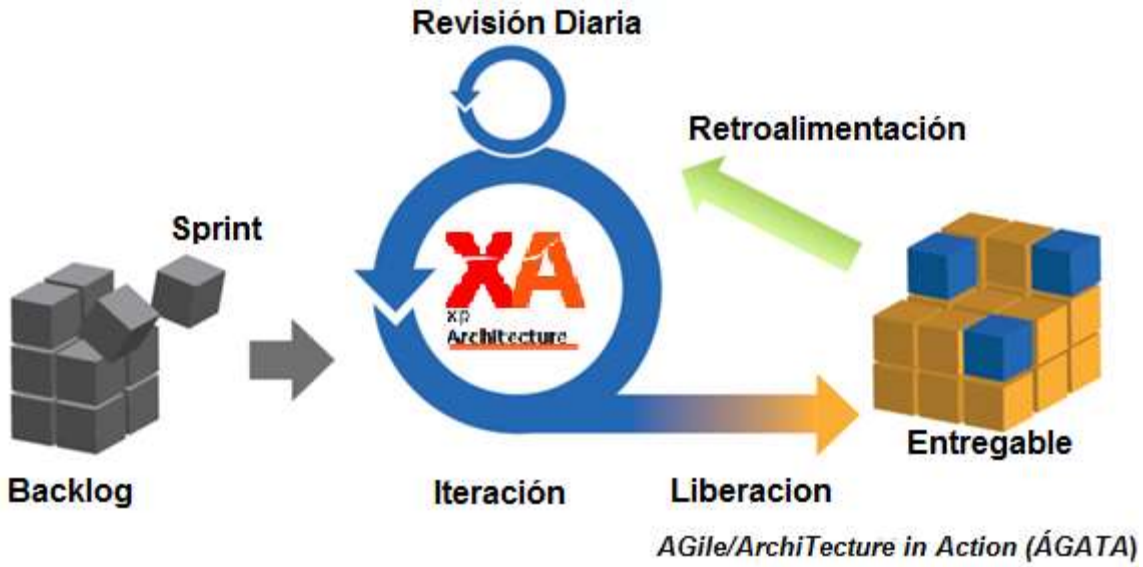


Figura 25 El Proceso de AGATA

AGATA como marco de trabajo posibilita que un equipo mayor a los permitidos por las metodologías ágiles (16±4), pueda desarrollar productos software de una forma ágil, iterativa e incremental. (ver Fig. 25). AGATA se estructura en ciclos de trabajo llamados Sprints o iteraciones, cuya duración no sobre pasa los estipulado por los métodos ágiles (no deben durar más de cuatro semanas cada una). Los Sprints para el equipo AGATA están orientados por el equipo central de arquitectura conocido como Scrum (XA) y están acotados en el tiempo; finalizan en una fecha determinada y difícilmente se prorrogan. AGATA propone que el equipo central con foco en la arquitectura extrema conocido como Scrum (XA) y los equipos de desarrollo XP gestionados o Scrum (XP), escojan una duración de Sprint y la mantengan para todos sus sprints, los cuales deben estar sincronizados con el ciclo general de valor. La idea es que los equipos estén alineados para lograr la generación de valor continuo hacia el cliente.

3.3.1 Los equipos de AGATA

Debido al tamaño del equipo (16 ± 4 personas), AGATA propone la estructuración en sub-equipos como se muestra en la Figura 3.1, es decir varios equipos satélites Scrum (XP) coordinados por un equipo central Scrum (XA), cada uno siguiendo desde la perspectiva técnica, las prácticas de XP para el desarrollo del producto y desde la perspectiva de gestión, las prácticas de Scrum.

EL EQUIPO SCRUM (XA) conformado por un equipo gestionado y centrado en la arquitectura del sistema es responsable de organizar:

- El Product Backlog: listado de requerimientos del cliente.
- Organizar el Backlog de ArquitecturaPriorizar los requisitos basados en el valor del negocio y de la arquietctura

- Los sprints a partir del product Backlog.
- El tablero de tareas de arquitectura por sprint.
- Desarrollar la Arquitectura del producto: especificar e implementar los aspectos arquitectónicos del producto, por ejemplo protocolos e interfaces entre componentes.
- Revisar los resultados del Sprint.
- Gestionar todo el proyecto basado en las políticas de Scrum.
- Evaluar el avance del proyecto usando los gráficos de avance de Scrum (Burndown y Sprint Burndown).

Los integrantes del equipo Scrum (XA) actúan como dueños del producto en cada equipo Scrum (XP). Por lo tanto, son los "vínculos técnicos sincronizados" entre los equipos Scrum (XA) y Scrum (XP).

EL EQUIPO SCRUM (XP): es un equipo de desarrollo gestionad que recibe del equipo SCRUM (XA) un sub product backlog (o subdivisión de los requisitos del cliente) con el objeto implementarlos en una aplicación software; para ello se encarga de:

- Revisar los resultados del sprint.
- Administrar el subproyecto teniendo a Scrum como soporte.
- Organizar el backlog de subproductos
- Priorizar los requisitos de acuerdo a la sincronización de gestión
- Definir sus sprints
- El tablero de tareas por sprint
- Rastrear el trabajo usando los gráficos de Scrum (Burndown y Sprint Burndown).
- Desarrollar el subproducto

Los equipos Scrum (XP) tienen entre sus miembros un Scrum Master quien se desempeña como líder del equipo, velando para que todos los participantes del proyecto sigan los valores y principios ágiles, las reglas y proceso de Scrum; además se coordina con los demás Scrum Máster para sincronizar la entrega de valor. Por lo tanto, los Scrum Masters son " los vínculos de gestión sincronizados" entre el equipo Scrum (XA) y los Scrum (XP).

Para AGATA es muy importante mantener los parámetros ágiles [89], también los pilares que soportan toda implementación del control de procesos empíricos [40] porque que serán los principios que asegurarán la gestión del equipo, los canales de comunicación y el desarrollo holístico del proyecto en equipos de tamaño mediano (2 a 9 sub – equipos) [145].

3.4 Prácticas en AGATA

Al igual que las metodologías ágiles [100], AGATA se fundamenta en prácticas trascendentales para obtener los mejores resultados esperados por el cliente y el equipo de desarrollo. De ahí que AGATA:

- Aprovechará las características del desarrollo incremental de los requisitos del product backlog organizado con el cliente y el product owner del equipo de arquitectura (Scrum (XA)) para convertirlo en un Architecture Product Backlog. El equipo Scrum (XA) organiza los requisitos funcionales en el sprint backlog, para luego distribuirlos a los equipos SCRUM (XP) quienes realizarán internamente las prácticas de desarrollo definidas por AGATA más adelante. Priorizará los requisitos basados en el valor para el cliente, pero

además tendrá en cuenta las variables dependientes al desarrollo del producto: estimación, alcance, importancia para la arquitectura y costo [74].

- Mostrará al cliente los resultados obtenidos a través del equipo Scrum (XA) al final de cada iteración, es decir mantendrá un control empírico del proyecto; de modo que el cliente pueda tomar las decisiones necesarias a partir de los resultados observados en el proyecto.
- Sincronizará y realizará adaptaciones diarias del equipo Scrum (XA) y los equipos Scrum (XP). Esto permitirá la coordinación y comunicación de los equipos comprometidos a entregar unos avances al cliente; para ello basado en su autogestión, tendrá la autoridad necesaria para organizar su trabajo.
- La Comunicación se convertirá en un elemento fundamental. Para ello AGATA dejará en los valores de Scrum (XA) y en la arquitectura parte esta responsabilidad. Complementada con las interacciones inter-grupales técnicas (arquitectura y product owners) y de gestión (scrum masters).Esto generará tanto en el cliente como en el equipo Scrum (XA) una dinámica que fortalecerá la toma de decisiones para conseguir los resultados esperados.

3.5 Alcances de AGATA

AGATA es conveniente para proyectos que involucren equipos de más de 10 personas, para proyectos de menor tamaño es recomendable seguir con una solución de tipo Scrum (XP), sin embargo si el tema de la arquitectura toma relevancia el desarrollo puede formularse con un equipo Scrum(XA).

AGATA posibilita la agilidad que ofrecen las metodologías ágiles [111], por tanto aunque el proyecto tenga cierto grado de complejidad, es posible que satisfaga la flexibilidad para moverse en un contexto con incertidumbre en los requerimientos. Es decir, se adhiere al ambiente cambiante que se presenta con las necesidades del cliente. De ahí que AGATA está encaminada hacia los desarrollos que requieren de cambios continuos en el transcurso de un proyecto.

Los proyectos realizados bajo este enfoque metodológico cumplen con lo estrictamente necesario en su funcionalidad en el 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 [112].

3.6 El ciclo de vida de AGATA

AGATA al igual que cualquier proceso de desarrollo de software [114], sigue unas fases e iteraciones como se muestra en las figuras 25 y 26.

Figura 26 Iteración conceptual en AGATA

Cada una de las fases del ciclo de vida en AGATA cumple su función y se describen así:

EXPLORACION: Los miembros del equipo de arquitectura de Scrum (XA) establecen los requisitos iniciales del proyecto, determinan el alcance y forman los backlog iniciales del producto y de la arquitectura. Se establecen inicialmente las reglas y prioridades con el cliente sobre los requisitos del sistema.

DESARROLLO: El equipo de arquitectura Scrum (XA) organizados autónomamente, se reúnen para establecer y resolver el SPRINT BACKLOG de arquitectura, derivar las necesidades hacia los equipos Scrum (XP), las cuales permitirán establecer el PRODUCT BACKLOG de cada equipo. Los equipos, siguiendo los criterios de Scrum (XA) realizan la arquitectura del sistema y los equipos Scrum (XP) realizan las componentes de esa arquitectura.

ENTREGA: Es la entrega del producto terminado, bajo la evaluación del cliente (aprobación de las pruebas de aceptación) y que estará listo para usar. (Ver Fig. 25)

SPRINT: Son bloques temporales cortos y fijos con una duración basada en las tareas del equipo (generalmente semanas, no exceden de un mes). Tiempo en el cual se da un incremento de producto potencialmente usable.

SINCRONIZACIÓN DE LOS SPRINTS: Los miembros Scrum (XA) actúan como Product Owner en cada equipo Scrum (XP). Por lo tanto, se convierten en los "vínculos técnicos" técnicos para la sincronización de los equipos. Los Scrum Masters Lideran los equipos para que sigan las prácticas y valores de AGATA. Por lo tanto, los Scrum Masters se encargan de la sincronización del equipo en torno al modelo. La arquitectura se convierte en el canal por donde transitan los mensajes a través de los cuales avanza el proyecto.

REVISIÓN DIARIA: Es el momento de la transferencia de la información y la colaboración entre los equipos, cada participante comenta sus avances, dificultades y proyección.

3.6 Elementos de AGATA

Al igual que Scrum [110] AGATA mantiene los mismos eventos, roles y artefactos, que trabajan de forma solapada y sincronizada.

3.7.1 Eventos de AGATA

Los eventos están basados en el tiempo, de tal forma que todos tienen una duración predefinida por los equipos de AGATA, que una vez concertados, no se deben cambiar (acortar o alargar). Aunque algunos de ellos están supeditados a los objetivos, pero de todas formas debe asegurar que la cantidad de tiempo utilizado para alcanzar ese objetivo no transgredirá a lo propuesto por el proceso.

- **Sprint Planning meeting:** Es un encuentro del equipo con el cliente (o su representante), aquí se comparten las inquietudes del sistema y las condiciones existentes para la realización del proyecto. Esta actividad no puede durar más de ocho (8) horas y se realiza en dos momentos:

    - ✓ **Conformación del Product Backlog y Architecture Backlog:** se aconseja que no dure más de cuatro (4) horas, El cliente presenta al equipo de arquitectura Scrum (XA) los requisitos del sistema con el que se conformará el Architecture product backlog. La comunicación debe ser muy clara y transparente, es aquí donde Scrum (XA) debe clarificar las dudas que surgen y priorizar los requisitos que se compromete a completar en la iteración, de manera que puedan ser presentados la cliente en el tiempo estipulado. De la misma manera los Product Owner (XA), presenta al equipo Scrum (XP), los requerimientos quienes armarán su product backlog y priorizarán las tareas.

    - ✓ **Planificación de la iteración** Es una reunión que realiza el equipo Scrum (XA) con el product owner, y el scrum master y no debe durar más de cuatro (4) horas. Una vez Scrum (XA) tiene claro el Architecture product backlog, elabora la lista de tareas para la iteración y que son aquellos requisitos a desarrollar de los que se ha comprometido. La estimación de esfuerzo se hace de manera conjunta. Los miembros de los equipos Scrum (XP) se auto-asignan las tareas en reuniones de planificación de la iteración que realizarán de forma solapada con las reuniones de Scrum (XA). Hay dos reglas fundamentales que deben trazarse los equipos para lograr el objetivo de esta fase: Primero tener muy claro lo que debe entregarse en el Incremento resultante del Sprint que comienza y segundo establecer la ruta con la que conseguirá hacer el trabajo necesario para entregar el Incremento.

Un **Sprint** es una Iteración de duración predefinida durante la cual los equipos scrum(XA) y scrum (XP) trabajan para convertir las historias del Architecture Product Backlog y el product backlog comprometidas, en una nueva versión de software totalmente operativo.

- **Daily sprint meeting**: Es la reunión diaria de máximo 15 minutos, en la que los equipos se sincronizan para trabajar de forma coordinada. Es una reunión que deben realizar todos los equipos de AGATA. Los equipos inspeccionan el trabajo que están realizando.

Esto permitirá verificar el progreso hacia el objetivo de la iteración, los obstáculos que pueden impedir alcanzar ese objetivo y las dependencias entre las tareas. Los resultados de esta reunión servirán para hacer las adaptaciones necesarias que conlleven a cumplir con el compromiso adquirido. Cada miembro comenta que hizo el día anterior, que hará hoy y cuáles son los impedimentos que no han dejado alcanzar el objetivo. Scrum(XA) hará sus daily sprint meeting solapadas respecto a los equipos Scrum (XP) con el objeto de poder llevar a estas reuniones todo el conocimiento de lo que va sucediendo con el proyecto. Los product Owner (XA) participarán en la reunión con el equipo Scrum(XA) para verificar los avances del producto respecto a los compromisos adquiridos. LO mismo harán los Product Owner (XP) en sus equipos Scrum (XP).

Durante la iteración Los Scrum Master se encargarán de verificar que los equipos puedan cumplir con su compromiso y que no disminuyan su productividad respecto al postulado ágil; para ello,

- ✓ Valida la sincronización de los equipos respecto de las prácticas de AGATA,
- ✓ Elimina los obstáculos que el equipo no pueda resolver por sí mismo y que le impidan cumplir con los compromisos de la iteración y
- ✓ Protege al equipo de interrupciones externas que puedan afectar su compromiso o su productividad.

- **Sprint Review :** El equipo de Arquitectura Scrum(XA) realiza una reunión muy informal al final del sprint donde presenta al cliente aquellos requisitos completados en la iteración, en forma de incremento de producto y haciendo un recorrido lo más cercano posible al objetivo que se pretende desarrollar en esta iteración.

  Esta reunión permitirá que el cliente pueda ver de manera objetiva cómo se van desarrollando los requisitos, compararlos con sus expectativas, y re-planificar el proyecto para tomar mejores decisiones. Por otro lado el equipo puede observar si realmente entendió cuáles eran los requisitos que solicitó el cliente, evaluar los canales usados para la comunicación y en qué puntos hay habrá que mejorar. Esta reunión se planifica para que no dure más de cuatro (4) horas.

- **Sprint retrospective:** Es una reunión que realizan todos los equipos de AGATA una vez haya finalizado el Sprint, donde se analiza qué se hizo bien, qué procesos serían mejorables y discuten acerca de cómo perfeccionarlos. Por otro lado el equipo guardará una bitácora de aquellas lecciones aprendidas y que le permiten incrementar la productividad. Los Scrum Master (XA) y (XP) se encargará de validar el trabajo respecto a las prácticas propuestas y quitar los obstáculos que no permiten alcanzar los objetivos. La reunión no debe durar más de cuatro (4) horas. La idea es hacer una buena planificación, trazar objetivos concretos y alcanzables. Por tanto, el tiempo debe ser limitado para no divagar en temas que no aportan al proyecto.

3.6.2 Los Artefactos de AGATA

Lista de Producto (Product Backlog): Son los requisitos del cliente quien los describe en un lenguaje no técnico y quien los prioriza según el valor de negocio, considerando su

beneficio y costo. Los requisitos y prioridades se revisan y ajustan durante el curso del proyecto en intervalos regulares. Esta labor se hace entre el cliente y el equipo Scrum (XA), quienes una vez conformado el product baclog, continúan con las prácticas propuestas por AGATA.

Architecture Product Backlog: El equipo de arquitectura (Scrum (XA)), convierte el Product Backlog del cliente en un Architecture Product Backlog, para formar las historias de arquitecto que luego serán compartidas a través del Product Owner con los equipos Scrum (XP) .

Artefactos de Arquitectura: Las historias de arquitecto se apoyan en patrones de arquitectura, que son modeladas en artefactos de diseño comprensibles para los equipos Scrum (XP). Estos diseños, son mostrados en un radiador de arquitectura, la intensión es que sean visibles para todos.

Lista de Sprint (Sprint Backlog): Es el subconjunto de requisitos que serán desarrollados durante el siguiente sprint. Al definir el sprint backlog, se describe cómo los equipos van a implementar los requisitos durante el sprint. Por lo general los requisitos se subdividen en tareas, que autónomamente se atribuyen los equipos Scrum (XP). Ninguna tarea debe tener una duración superior a 16 horas. En caso contrario, se divide en otras tareas menores. Esto lo hacen los miembros del equipo de la manera más adecuada.

Tablero de mandos (Burn down Chart): Es un diagrama para el seguimiento de los sprint en el que se representan el tiempo de duración de un sprint (o iteración) y la cantidad de trabajo comprometida con el cliente durante el sprint. Este artefacto es útil para AGATA porque ayuda a responder a la pregunta ¿cuánto falta para terminar?. También porque se entiende el avance del trabajo en el proyecto y al actualizarse diariamente permite verificar el ritmo sostenido por los equipos. Además obliga a cuantificar el trabajo, a dividir las historias en tareas y a estimar su duración.

3.6.3 Los Roles

La gestión de un proyecto en AGATA se centra en definir cuáles son las características que debe tener el producto a construir (qué construir, qué no y en qué orden) y en vencer cualquier obstáculo que pudiera entorpecer la tarea del equipo de desarrollo. De ahí que en el equipo se reconozcan los siguientes actores (Figura 27):

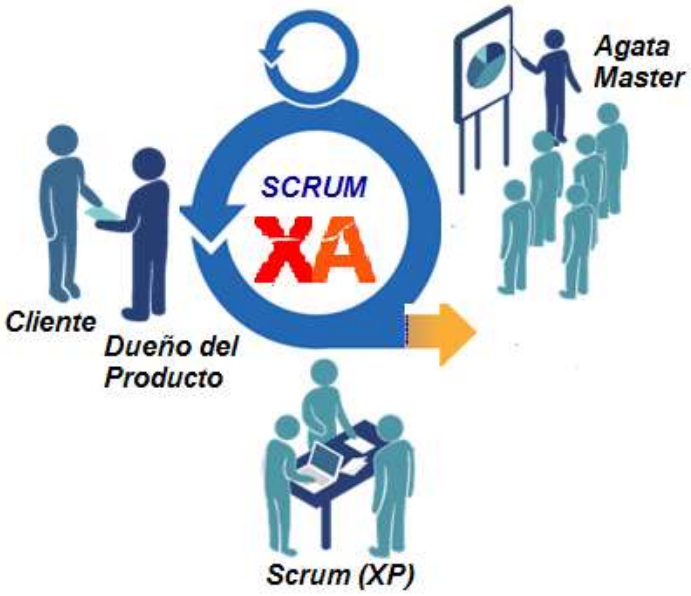


Figura 27 Actores de AGATA

El cliente: Es quien presenta una necesidad o lista de requisitos ante Scrum (XA) para ser sistematizados. Se entrevista permanentemente con el equipo de arquitectura, a quien entrega sus requerimientos y de quienes recibe producto terminado.

Scrum master (XA): Persona que lidera al equipo Scrum(XA) guiándolo para que cumpla las reglas y procesos de la metodología. Gestiona la reducción de impedimentos del proyecto y trabaja con el Product Owner para cumplir con los resultados proceso-producto.

Scrum Master (XP): Un representante del equipo Scrum (XP) encargado de dinamizar los valores y prácticas de XP gestionado.

Product Owner(XA): Es el representante del cliente ante el equipo Scrum (XA). Motiva al equipo para la entrega del producto con base en los requisitos del cliente. Se encarga de focalizar la visión del proyecto ante el equipo.

Product Owner (XP): Son los representantes de de Scrum (XP) ante el equipo Scrum (XA). Actúan como dueños de producto en ausencia del cliente ante los equipos Scrum (XP)

Los equipos: En AGATA se refiere a los equipos:

Scrum (XA): Conformado por un representante de los equipos Scrum (XA) encargados de la arquitectura del proyecto; y
Scrum (XP); Son los equipos encargados de la producción en el proyecto, seguirán las prácticas de XP gestionado.

Los miembros de los equipos, tienen los conocimientos técnicos necesarios para desarrollar el proyecto sincronizadamente.

3.7 Los canales de comunicación en AGATA

Como se mencionó en líneas anteriores AGATA propone dos canales de comunicación**: la Arquitectura** basada en *XP/Architecture (XA)* [41] y ***el cara a cara*** inter-equipo con el fin de

- Avanzar evolutivamente, en lugar de tener que acercarse a métodos predictivos o tradicionales.
- Evaluar periódicamente los resultados basados en criterios de calidad aprovechando el conocimiento tácito de las personas, y el aprendizaje que se genera en la gestión del equipo.
- Seguir los parámetros del desarrollo ágil: iniciando con la necesidad del cliente y la exposición de sus requisitos, luego el desarrollo del producto de forma incremental a través de cortas iteraciones que comprenden fases de exploración planificación construcción y entrega, que se repiten de forma continua hasta que el equipo Scrum (XA) entregue al cliente y a su vez manifieste satisfacción por el producto recibido.
- Entender los requisitos del cliente, compartirlos entre los equipos y proponer las estrategias de desarrollo basadas en incrementos a través de las iteraciones y revisiones propuestas por el marco de proceso.

Los canales de comunicación permiten de una manera holística la gestión [116] y la comunicación del equipo [115], para ello ayudan a:

- Transmitir información: Sea cual sea. La idea es que el mensaje obtenga una reacción del destinatario.
- Transferir ideas: que cambiarán la conducta del receptor, quien actuará a partir de una nueva reacción después de escuchar el mensaje.
- Acordar, crear significados: basados en las acciones del emisor, el receptor y los conocimientos fundamentales para crear cimientos compartidos.
- Dar y recibir retro-alimentación: posibilitan la corrección de errores, evalúa que se hizo bien o mal durante el proceso de producción.
- Co – Crear realidades: Influye en gran medida en las percepciones del emisor y el receptor.

Para que se cumplan estos objetivos dentro del proceso de comunicación los canales propuestos sugieren que se establezca sinergia en cada uno de los elementos que intervienen en la comunicación. [70]; además mantendrán control sobre los mensajes que circulan en torno al proyecto, intentan reducir el ruido buscando que los equipos compartan lo que es necesario para lograr un buen producto; para ello sugiere la transmisión de información se realizará en tres fases que actuarán recíprocamente:

En la primera fase los equipos Scrum (XA) y Scrum (XP) comparten los requisitos arquitectónicos del sistema, que luego codificarán en metáforas del sistema y vistas arquitectónicas, usando como canal Arquitectura del Sistema (Backlog de Arquitectura y el radiador de Información Arquitectónica, para luego ser decodificadas en metáforas del sistema, vistas arquitectónicas y funcionalidades arquitectónicas en esta momento es posible que generen ruido las ideas alternativas y/o elementos del modelo confusos.

En la segunda fase los equipos Scrum(XA) y Scrum (XP) comparten sus requisitos del sub-sistema que son codificados en Historias de Usuario; el Product Owner se convierte en el

canal en esta fase, lasm historias de usuario se decodifican en producto entregables. El ruido que puede afectar esta etapa puede estar dado por Software complejo y/o refactorizaciones incipientes.

En la tercera fase los equipos Scrum (Scrum(XA) y Scrum (XP) se comparten la Información de gestión de la entrega de valor que la codifican en el Backlog del Producto y el Backlog del Subproducto, además de los objetivos de los Sprints a sincronizar, el canal para esta fase es el Scrum Master y se decodifica el Backlog del Sprint Sincronizado, equipos alineados a la entrega de valor, para esta fase es posible que haga ruido Historias Incompletas

Capítulo 4

La mente que se abre a una nueva idea nunca vuelve a su tamaño original.

Albert Einstein

Evaluación del Método AGATA

En este capítulo se hace una descripción del método y su aplicación en dos estudios de caso; cuyo propósito fue evaluarlo desde diferentes perspectivas con el objeto de validar la pertinencia del modelo en equipos de tamaño mediano.

Un estudio de casos, es una indagación empírica, que analiza un fenómeno contemporáneo dentro de su contexto en el mundo real especialmente cuando los límites entre el fenómeno y el contexto son poco claros Yin [131]. Los estudios de casos permiten realizar investigaciones con alta validez, fiabilidad y establecer reglas bien definidas que dejan de lado lo que puede parecer un enfoque anecdótico. El Estudio de Caso, además permite la integración de métodos cualitativos y cuantitativos Sautu [130]. El desarrollo de los casos en esta tesis doctoral se llevó a cabo bajo las directrices de Runeson et al [47] y las guías "diseño del estudio de caso" (ver anexo), desarrolladas por los autores de este documento, que se puede encontrar en los anexos de esta investigación.

4.1. Estudio de Caso Preliminar

El primer estudio de caso consistió en un proyecto embebido, exploratorio, con el objeto de evaluar el desarrollo de software con equipos de tamaño mediano, siguiendo las prácticas de las metodologías ágiles (Extreme Programming (XP)), dinamizados por un equipo de arquitectura denominado XP/Architecture (XA) [108]. El caso embebe tres casos de desarrollo de aplicaciones dos de ellos en el mismo ambiente (seminario de desarrollo en la universidad FUP- Fundación Universitaria de Popayán) y la otra en un contexto diferente (empresa de software - DOBLE CLICK) y que corresponden a:

- La automatización de los proyectos de investigación del Sistema de Investigación e Innovación (SIDI) de la Fundación Universitaria de Popayán.
- Gestión del presupuesto de los proyectos de investigación avalados por le SID DE LA Fundación Universitaria de Popayán.
- Control de clientes de una empresa en servicios telemáticos de la ciudad de Popayán.

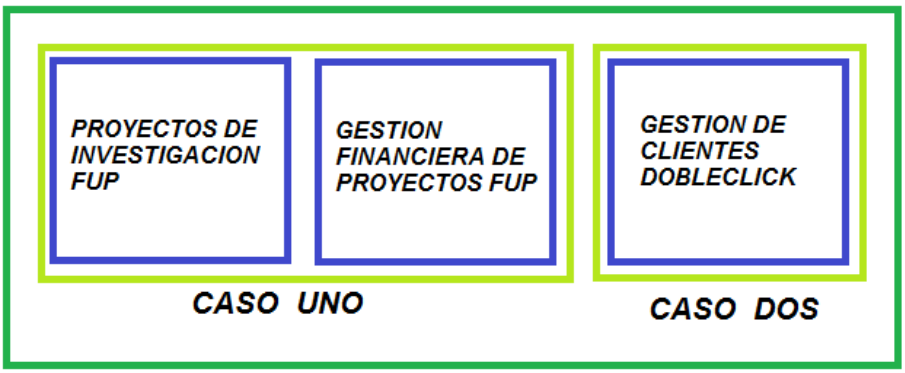


Figura 28 Primer estudio de caso

4.1.1. Diseño del estudio de caso

Para el diseño del estudio de caso, se tuvieron en cuenta los siguientes aspectos:

- **Objetivo del estudio de caso:**
    - Determinar los aspectos positivos y negativos de un equipo ágil con prácticas de arquitectura respecto al funcionamiento de la comunicacición inter-.grupal e intra-grupal.

Para resolver este objetivo, el diseño se basó en la siguiente pregunta orientadora:

- Qué problemas y oportunidades de comunicación se evidencian en equipos ágiles grandes organizados alrededor de las prácticas y un equipo de control focalizado en la solución arquitectónica?

- **Indicadores:**

La pregunta llevó a la investigación a buscar elementos de medición de dos variables: el problema de la comunicación y las posibilidades de la comunicación y se definieron como:

- Un problema de comunicación que se evidencia en los diferentes sub-equipos y que los reportan un número significativo de participantes. (> 30 %).
- Una posibilidad de comunicación es una solución emergente a un problema que los equipos adoptan como estrategia de mejoramiento.

- **Mediciones:**

Las métricas que soportaron los indicadores fueron:

- Complejidad de las dificultades de comunicación encontradas.
- Elementos claves que fueron usados en las estrategias de mejoramiento.

4.1.2 Descripción del contexto del estudio de caso:

Contexto del Caso 1 y 2:

Los dos primeros proyectos se desarrollaron en la Fundación Universitaria de Popayán una Institución de Educación Superior de carácter privado, con condiciones de calidad aprobadas por el Ministerio de Educación Nacional de Colombia.

Dentro de su estructura administrativa se encuentra el Sistema de Investigación desarrollo e Innovación (SIDI), área encargada de gestionar y direccionar los proyectos de investigación desarrollados por las 14 facultades que tiene la Universidad.

El SIDI debe gestionar los proyectos desde varias perspectivas: por un lado, da el aval para desarrollarlos cuando los proyectos se realizan en convenio con otras instituciones o son fruto de alguna convocatoria externa a la institución. Por otro lado, 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[3], 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 SIDI aporta los recursos directamente al proyecto y establece condiciones de seguimiento y cumplimiento de entrega a los grupos para verificar la inversión económica dada al proyecto [118]. 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 SIDI.

Contexto del Caso 3:

El tercer proyecto se desarrolló en la empresa Caucana DOBLECLICK dedicada al desarrollo de aplicaciones software y la intercomunicación empresarial. Es una organización proveedora de servicios de internet (ISP) colombiana, proveedora de servicios de valor agregado y telemáticos con la capacidad de diseñar, instalar y configurar redes de datos físicas e inalámbricas.

4.1.3 Los equipos de trabajo

Para el primer proyecto, se escogieron 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 (Extreme Programming) y XA (Arquitectura ágil). 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

[3] El Departamento Administrativo de Ciencia, Tecnología e Innovación (Colciencias) es la entidad encargada de promover las políticas públicas para fomentar la ciencia, la tecnología y la innovación en Colombia.

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 (tiempo de duración de su programa académico). Esto permitió durante el proyecto adherirse mejor al método, 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.

Para el segundo proyecto, se organizó un diplomado en desarrollo de aplicaciones WEB y se invitó a participar del proyecto a profesionales graduados en ingeniería de sistemas y afines, pero con alguna experiencia en desarrollo de software. Se contó con un grupo de 18 participantes, el 45% de ellos pertenecientes a diferentes empresas de desarrollo de software de la ciudad, el 35% estudiantes recién graduados en el área, y el resto, dedicados al desarrollo de aplicaciones con procesos de tercerización.

Para el tercer proyecto se aprovechó los 22 profesionales de la empresa dobleclick, quienes iniciaban un proyecto de desarrollo para la clientelización (información de sus clientes y mantenerlos informados de sus servicios y productos). La distribución de los equipos y del proyecto no fue difícil, la autonomía y la autogestión prevaleció debido a la dinámica holística del modelo propuesto. Voluntariamente cada quien según sus afinidades se organizó en un equipo XP, luego escogieron entre ellos quién los representaría y haría parte del equipo de arquitectura XA.

4.1.4 Diseño del Estudio de caso

Para desarrollar esta investigación, se siguió la propuesta de Runenson y Host [47]. Se partió de la pregunta de investigación inicial que permitiría validar la arquitectura como canal de integración y colaboración para equipos de tamaño mediano y que intentaba resolver: Cómo escalar un proceso ágil en proyectos con equipos de desarrollo grandes; que para dar respuesta, el estudio de caso definió los siguientes indicadores:

La Satisfacción: Una medida que buscó el modelo propuesto cumplió o superó las expectativas de los equipos: Se definió como el número de miembros los equipos, cuyo reporte de sus experiencias superaron las expectativas encontradas en la ejecución del modelo.

La calidad: entendida como las características del producto de software resultantes que satisface necesidades explícitas o implícitas del cliente (IS0/IEC 9126).

Productividad del equipo: Basada en el esfuerzo de los equipos para entregar producto al cliente.

Con los indicadores, se trazaron unas métricas e instrumentos para evaluar los datos como se presentan en la Tabla 4.

- **Indicadores estudio de caso embebido**

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 información se obtendrá 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 Scrum (XP) para conducir el desarrollo de software en una forma eficiente.	Productividad (P)	Gráfico de avance del proyecto	Burndown Chart

Tabla 4 indicadores, métricas, fuentes de información e instrumentos

En este estudio de caso 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 al equipo XP: Para conocer el grado de aceptación del modelo propuesto entre los miembros del equipo.

Encuesta al 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 qué criterios de calidad el resultado del proyecto satisface los requisitos

Lista de chequeo: Para verificar la calidad del 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”

4.1.5. Ejecución de los casos

Proyecto 1

Ilustración 1 Equipo primer proyecto

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: extreme programming (XP), con el objeto unificar términos de referencia. En la segunda parte se abordó el tema de arquitectura, particularmente conceptos fundamentales, vistas, estilos y los métodos de arquitectura propuestos por el Software Engineering Institute (SEI): Attribute-Driven Design (ADD) y Quality Attribute Workshops (QAW).

En la tercera parte, se les presentó XA con todos sus componentes, 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.

Los estudiantes se auto-organizaron en 3 equipos XP de 6 integrantes, cada equipo escogió su representante cliente/arquitecto. 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 iniciar la actividad, 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.

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 (Ver Ilustración 1). 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.

Proyecto 2.

XA fue aplicado por 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 SIDI y quien conocía la dinámica del proyecto y los requisitos para el desarrollo de la aplicación.

Como la mayoría de los participantes a diferencia del primer grupo, eran bastante heterogéneos debido a que no conocían la metodología para el proyecto ni 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. Luego se trabajaron temas de arquitectura, los métodos propuesto por el SEI (Software Engineering Institute) pero entre ellos se enfatizó en ADD (Attribute-Driven Design) y QAW (Quality Attribute Workshop); depues, se presentó el modelo con sus artefactos, roles principios y valores, para respaldar la capacitación y el desarrollo del proyecto se les entregó dos reportes técnicos desarrollados específicamente para introducir a las metodologías ágiles (Extreme Programming).El proyecto se desarrolló en 14 sesiones con una intensidad de 4 horas. El proyecto tuvo un total de 56 horas de desarrollo.

Ilustración 2 Un momento en la capacitación

Debido a la dinámica del modelo propuesto y del proyecto, el grupo se auto-organizó 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, 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.

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 organizar los elementos necesarios para aplicar extreme programming al módulo que deberían desarrollar.

Se establecieron también los mecanismos para determinar requisitos. Una vez diseñado la arquitectura en la primera iteración, el equipo XA, compartió con los equipos XP los requerimientos del subsistema elegidos por su Cliente (representante XP ante el equipo de arquitectura XA).

Los equipos XP durante la planificación mantuvieron un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente XP. 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.

Una vez obtenidas las “historias de usuario”, los equipos XP evaluaron rápidamente el tiempo de desarrollo de cada una de ellas, además se tuvo en cuenta en especificar el detalle mínimo de cada historia como para que los programadores realizaran una estimación poco riesgosa del tiempo que llevaría su desarrollo. Si alguna de ellas presentaba “riesgos” que no permitirían 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.

Proyecto 3.

Consistió en aplicar el modelo a un caso industrial donde se especificaría un software con todos los requerimientos y complejidad de un desarrollo comercial.

Fueron 22 los participantes del proyecto quienes autónomamente se organizaron para el desarrollo del proyecto planteado; se les dio una capacitación en XA de 16 horas. Dos días de trabajo en jornadas de 8 horas. Se conversó también sobre metodologías agiles entre ellas Extreme Programming (XP), con el objeto unificar términos de referencia. Se trabajó también temas de arquitectura, puesto que se necesitaba que el grupo conociera particularmente conceptos, vistas, estilos y métodos propuesto por el SEI (Software Engineering Institute) enfatizándose en ADD (Attribute-Driven Design) y QAW (Quality Attribute Workshop).

Ilustración 3 Equipo caso de estudio tres

Conforme a lo propuesto por el modelo, los 22 integrantes decidieron organizarse en 3 equipos XP de 6 integrantes cada uno. Cada equipo siguió los valores y principios de XP y los otros 4 conformaron el equipo Xp/Architecture XA.

Se establecieron reglas para la simplicidad que se aplicó a todos los aspectos de XA, desde la fase inicial con arquitecturas muy sencillas y basadas en prácticas básicas, incluyendo la refactorización de código.

La retroalimentación fue importante para el equipo de trabajo XA hacia el cliente, con el fin de brindarle avances sobre el desarrollo del sistema y conocer su apreciación, y también se realizó entre los representantes de XA hacia los equipos XP en los aportes al desarrollo del proyecto. Los integrantes demostraron valentía cuando se enfrentaron a los continuos

cambios que se presentaron en el transcurso de la actividad y hubo necesidad de re-hacer partes del proyecto desarrollados.
Para el desarrollo del proyecto se tuvo en cuenta los roles propuestos por el modelo con el objeto de organizar las actividades del sistema:

El cliente: en este caso fue una persona nombrada por la empresa para quien se desarrolló la aplicación y fue quien determinó los requisitos.

El Equipo XA: Nombrado por los integrantes de los equipos XP; lo hicieron durante su charla inicial cuando se comentaron sus experiencias en cada aspecto de la aplicación a desarrollar. Además de desarrollar la arquitectura, fueron líderes al interior del proyecto, encargados de vigilar el cumplimiento de los parámetros propuestos.

Los Equipos XP: Lo conformaron 18 de los integrantes del proyecto quienes se organizaron en 3 equipos XP

Resultados y Análisis

Las tablas siguientes muestran los resultados encontrados en cada uno de los proyectos desarrollados. (ver tablas 5,6,7)
.

DIFICULTAD	DESCRIPCION	ESTRATEGIAS DE SOLUCION	ELEMENTOS CLAVES INVOLUCRADOS
Requisitos incompletos	Los equipos Xp no entendieron bien los requisitos del cliente	Reuniones extras con los desarrolladores para hacer aclaraciones	Cliente, equipo XA, Modelo de arquitectura, Equipos XP
Arquitectura Incoherente	El equipo de arquitectura no completó algunos diseños por la falta de claridad en los requisitos	Reuniones de verificación con el cliente por fuera de las planeadas y hacer retrospectiva del trabajo realizado	Equipo XA, Modelo de arquitectura, Equipos XP
Historias de usuario no definidas	Los equipos Xp no entendieron algunas historias de arquitecto. Las historias de usuario no quedaron bien definidas	Apoyo de una herramienta para verificar los diseños	Historias de arquitecto, Historias de usuario, Equipos XA y XP
Baja satisfacción del producto	El producto entregado no cumplió completamente con los requisitos	Reuniones de planificación con el cliente para especificar mejor los requisitos	Cliente, Listado de Requisitos del cliente Historias de usuario y de arquitecto,

	del cliente		

Tabla 5 Hallazgos Proyecto 1

DIFICULTAD	DESCRIPCION	ESTRATEGIAS DE SOLUCION	ELEMENTOS CLAVES INVOLUCRADOS
Requisitos funcionales	El cliente no entregó completos los requisitos funcionales	Nombrar un dueño de producto, para que actúe como cliente en su ausencia.	Lista de requisitos del cliente, Cliente.
Restricciones en el diseño	No Hubo claridad en algunos servicios de la aplicación	En la arquitectura usar algunos elementos de configuración para aclarar servicios de cada componente	Restricciones, Arquitectura, Equipo XA
Comunicación	No hay condiciones claras para la transmisión de los mensajes	Establecer un método para que coordine la información compartida entre los equipos	Equipos XA, XP, cliente
Equipo	El equipo en algunas ocasiones actuó descoordinado	Proponer en el modelo un representante que este pendiente del trabajo del equipo y el modelo.	Equipo XA y XP
Prácticas del modelo	La adherencia al método fue baja.	Extender las horas de capacitación para el aprendizaje del método.	Modelo XA, Equipos XP y XA.

Tabla 6 Hallazgos Proyecto 2

DIFICULTAD	DESCRIPCION	ESTRATEGIAS DE SOLUCION	ELEMENTOS CLAVES INVOLUCRADOS
Descoordinación de los equipos	En algunos momentos los equipos no coordinaron sus tareas, retrasando tiempos de entrega y/o calidad en el producto	Realiza reuniones diarias para mirar avances del proyecto y/o dificultades presentadas	Listado de requisitos Equipos XA y XP,
Coordinación del equipo	Por las diferencias de los participantes en el desarrollo del proyecto no se aplicó completamente el modelo	En las reuniones diarias, verificar el cumplimiento del modelo y/o dificultados que impidieron cumplir con las prácticas	Equipos XA, XP Prácticas del modelo
Errores en la información compartida	Los mensajes compartidos no llegaban con la información requerida asu destino	Realizar a través de la arquitectura, mensajes mas sólidos en cuanto al desarrollo del proyecto	Arquitectura, equipo XA, Requisitos
Descoordinación de los planes de trabajo	En algún momento se hizo una planificación demasiada optimista, por parte del cliente	Organizar reuniones de planificación, revisión y reuniones diarias para controlar las actividades del proyecto	Cliente, Equipos XA, XP

Tabla 7 Hallazgos Proyecto 3

- **Lecciones aprendidas en el estudio de caso preliminar**

Debido a que las estrategias propuestas por Xp/Architecture (XA) para el desarrollo del proyecto ayudaron a la unificación de los equipos desde el inicio del proyecto, y motivaron a 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 miembro, 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.

Al evaluar tanto a los equipos XP como el equipo XA, afirman en un alto porcentaje (80%) satisfacción con los procedimientos del modelo en cuanto al desarrollo y están de acuerdo

con los resultados obtenidos. Sin embargo se notaron diferencias individuales en el tercer estudio de caso, al compartir la información entre los equipos. El equipo XA afirma que en estos casos es necesario proponer canales que fortalezcan sobre todo la comunicación; como se puede ver en los resultados, al compartir los requisitos del cliente, la curva en cuanto a la satisfacción de los requisitos fue bajando, debido a que los equipos eran cada vez más disimiles y no tenían criterios claros para la trasmisión de los mensajes. Es necesario establecer entonces, un canal que procure la unicidad del equipo en el manejo de la información, prácticas de diseño y programación. Esto hará, afirman ellos, que el proyecto empiece a madurarse muy temprano por muy complejo que sea.

Las prácticas demostraron adherencia al método propuesto por esta investigación, los seminarios de capacitación sirvieron para consolidar algunos conceptos sobre los métodos ágiles XP en nuestro caso. Pero al igual que con la variable sobre los requisitos, la curva de aceptación del método también fue bajando, Los participantes manifiestan que faltaron prácticas de gestión, que les permitieran trabajar holísticamente.

El nivel de conformidad del producto para el primer proyecto, fuer medianamente aceptable (76%), debido a que por el tiempo dado a los equipos no alcanzaron a entregar los resultados esperados, ni los requisitos desarrollados cumplieron con las expectativas del cliente. En la evaluación los participantes comentan que los mensajes que llevaban los requisitos eran poco entendidos, esto los obligó a pedir explicaciones y reuniones extras a los product ownwer para poder entender algunos requisitos. Actividades que demoraron el desarrollo del producto. En las siguientes experiencias (Segundo proyecto FUP y DOBLECLICK) mejoró el nivel de conformidad, aunque la hacer el análisis de los resultados, la curva nos demuestra que a medida que existen diferencias muy marcadas entre los miembros de los equipos, es decir, cada uno trabajando bajo sus propios esquemas, la conformidad es baja. Los equipos aducen también la falta de gestión y de comunicación para unificar criterios los imposibilitó al cumplimiento total del modelo.

Con los resultados de los tres proyectos en este estudio de caso embebido, se puede afirmar que la productividad del equipo XA mantiene el orden de productividad de los equipos XP reportados por la literatura, y en este sentido se puede afirmar que Xp/Architecture (XA) permite escalar XP a equipos de tamaño mediano que fue el propósito inicial de la investigación.[69][120]

Pero, con base en las inquietudes de los equipos y los resultados obtenidos a través de las experiencias se puede concluir que aunque el modelo escaló los métodos ágiles, es necesario adecuar la gestión del equipo, establecer canales de comunicación, para mejorar los resultados y las diferencias inter –grupales de sus miembros.

Para corroborar las inquietudes de la investigación respecto al modelo, se realizó una reunión muy informal (Una conversación sobre el proyecto), un representante de cada proyecto con el objeto de indagar sus percepciones respecto a la experiencia y sobre todo a XA, que debería tener el modelo para mejorar las curvas de desempeño.

La reunión tuvo una pregunta orientadora: Qué elementos técnicos necesita XA para lograr mejores resultados en grupos disimiles, que permitan alcanzar un mejor nivel de satisfacción de los requisitos del cliente y de conformidad del producto.

Con base en esta inquietud se corroboró que entre las estrategias más importantes para

mejorar la empatía del grupo es añadir elementos para la Gestión y la comunicación del equipo que permitieran orientarlo tanto al desarrollo del proyecto, como al seguimiento de la metodología y que además involucrara mejor al cliente desde las etapas iniciales y hasta que este finalice la actividad. La arquitectura debe ser uno de los canales que permita transmitir la información dada por el cliente a los equipos XP, realizar reuniones permanentes donde los involucrados en el proyecto puedan conversar cara a cara sobre sus inquietudes, esta actividad puede mejorar la unidad del equipo y los resultados del producto.

A partir de estas observaciones, se decidió involucrar a SCRUM entendido como un marco proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, gestionar el equipo, y obtener el mejor resultado posible de un proyecto [57].

Al fortalecer a XA con prácticas de Scrum, se decide mejorar el modelo en aspectos de gestión y de comunicación y denominarlo Agile/architecture in Action: **AGATA**. Es decir Xp/Architecture gestionado, y los equipos que hacen parte de AGATA se denominarían Scrum (XA) (XA gestionado) y Scrum (XP) (XP gestionado).

Con esta nueva visión se decide aplicar AGATA en otro estudio de caso confirmatorio, con el objeto de validar ya no la productividad del equipo, porque con las validaciones anteriores se comprobó que el modelo escala los métodos ágiles, sino observar el comportamiento de los equipos en cuanto a la gestión y la comunicación de sus participantes.

4.1.11. Estudio de Caso Confirmatorio

El caso confirmatorio, es un caso de estudio que involucra un proyecto desarrollado en la ciudad de Popayán: que consistió en el desarrollo de un tablero burndown Chart para AGATA. (Ver Tabla 7)

AGATA se propuso para trabajar con un equipo heterogéneo es decir, participantes de diferentes Instituciones y organizaciones algunos desarrolladores con experiencia y con algún conocimiento y practica en metodologías agiles.

- **Diseño del proyecto**

El diseñó se realizó teniendo en cuenta la pregunta de investigación ¿Cómo escalar los métodos ágiles en proyectos de software con equipos de grandes (2 a 9 sub-equipos) y qué prácticas podrían introducirse para superar a los desafíos de la comunicación?. Al igual que en el estudio de caso anterior, se tuvieron en cuenta las mismas hipótesis, canales de gestión y comunicación propuestos para AGATA.

La calidad de la comunicación entendida como la capacidad de una organización para adaptarse a los requerimientos de sus usuarios [121] y la eficiencia de los canales de comunicación que son los medios de difusión de los mensajes que procuran el mejor entendimiento entre los participantes de la conversación [122], fueron las variables para medir la comunicación al interior del proyecto.

Para medir los propósitos de AGATA, se definieron los siguientes indicadores:

- La calidad de la comunicación del equipo (Scrum/XP) y el Equipo (Scrum/XA) usando como canales de transmisión la conversación cara a cara y la descripción de la arquitectura para entender la lista de requisitos priorizada del negocio en los niveles estratégico, táctico y operativo. Para este indicador se tuvieron en cuenta las siguientes Items, que llevarían a complementar la medición:

  Distorsión: Este indicador permite medir el grado de Distorsión del mensaje al comunicar los requisitos del cliente**.** Para calcular la distorsión se tendrán en cuenta: el número de defectos encontrados, el número de mensajes transmitidos, mensajes y defectos con requisitos funcionales, mensajes y defectos con información de restricciones, mensajes y defectos con información Arquitectónica (Drivers, Tácticas y Estrategias). La información se obtuvo de una observación directa realizada en las reuniones de planificación, iteración y retrospectiva: Estos eventos se filmaron, para repasar las observaciones y calificar con mayor seguridad este aspecto del indicador.

  Compresión: Que es el porcentaje de la información entendida por el Equipo (Scrum/XP), usando los canales de comunicación establecidos, respecto a la información emitida por el equipo (Scrum/XA). Para medir los resultados, se tuvo en cuenta: El número de Mensajes comprendidos, el número de mensajes enviados; entendiendo que Una unidad de mensaje es la descripción de un requisito del cliente. Un mensaje comprendido es una funcionalidad del sistema documentada. La información se obtendrá de una observación directa realizada en las reuniones de planificación, iteración y retrospectiva y de las sesiones filmadas.

  Velocidad: Entendida como la cantidad de mensajes respondidos por el Equipo (Scrum/XP) al equipo (Scrum/XA) usando los canales de comunicación establecidos. Se calcula teniendo en cuenta los Mensajes Compartidos sobre el Tiempo de duración de una reunión. La información se obtuvo de una observación directa realizada en las reuniones de planificación, iteración y retrospectiva.

  Calidad Percibida: Aspectos positivos y negativos en la comunicación identificados por equipo. Para evaluar este ítem, se calculó el conjunto de apreciaciones clasificadas como positivas y negativas. Para obtener estos datos se realizaron encuestas a todos los miembros del equipo.

- El segundo Indicador se definió como la calidad de los canales de comunicación usados por los equipos (Scrum/XP) y el Equipo (Scrum/XA). El Emisario y la descripción de la arquitectura para entender la lista de requisitos priorizada del negocio en los niveles estratégico, táctico y operativo. Para asegurar su resultado, se trabajaron los siguientes ítems:

  Efectividad del canal Permite medir la efectividad del canal usado por los equipos Scrum (XA) y Scrum (XP). Para calcular este ítem se tuvieron en cuenta los Mensajes Incorrectamente Transmitidos Debido al Canal y Mensajes Totales Transmitidos. Estos resultados permitirán conocer la Funcionalidad entendida como la facilidad de uso de los canales propuestos. Se realizarán observaciones y encuesta a los miembros de los equipos Scrum (XA) y Scrum (XP) después de cada reunión.

Grado de aceptación del canal usado por los integrantes de los equipos Scrum (XA) y Scrum (XPPara ello se realizarán encuestas a los miembros de los equipos Scrum (XA) y Scrum (XP) después de cada reunión, para conocer su apreciación respecto del canal.

Confianza que tienen los miembros del equipo con los canales usados. Se realizarán encuestas a los miembros de los equipos Scrum (XA) y Scrum (XP) después de cada reunión.

INDICADORES						
CALIDAD DE LA COMUNICACIÓN				**CALIDAD DE LOS CANALES DE COMUNICACIÓN**		
Distorsión	Compresión	Velocidad	Calidad Percibida	Efectividad del canal	Grado de aceptación	Confianza

Tabla 8 Indicadores y métricas

- **Desarrollo del proyecto**

El equipo desarrolló un tablero Kanban (o burdown chart) que permitiría mejorar los flujos de trabajo del proceso productivo de los equipos ágiles de AGATA. Para ello se listaron los siguientes requisitos: El tablero permitiría Visualizar el flujo de trabajo, Limitar el trabajo en curso, gestionar y medir el flujo de trabajo, Implementar ciclos de feedback, clarificar políticas y procedimientos, verificar la evolución continua de forma colaborativa.

Un equipo de 24 participantes fue delegado para desarrollar la aplicación; con el objetivo de automatizar el seguimiento del trabajo de los equipos Scrum (XP) y scrum (XA). Los 24 integrantes del proyecto se organizaron autónomamente para el desarrollo del producto planteado.

Como los participantes no conocían la metodología *AGATA*, se les brindó una capacitación en una jornada extendida de 8 horas. En esta jornada se trabajaron también temas sobre metodologías ágiles entre ellas Scrum y Extreme Programming (XP) con el objeto unificar términos de referencia. Finalmente, se capacitó también en temas de arquitectura, puesto que se necesitaba que el grupo conociera particularmente los conceptos básicos, vistas, estilos, así como los métodos de arquitectura propuesto por el SEI (Software Engineering Institute), haciendo énfasis en ADD (Attribute-Driven Design) y QAW (Quality Attribute Workshop)

Ilustración 4 Miembros AGATA trabajando en QAW

Los desarrolladores se auto-organizaron en 4 equipos Scrum (XP) con 5 integrantes cada uno y un equipo Scrum (XA) de 4 miembros. Cada equipo escogió su representante cliente/arquitecto ante Scrum (XA).

En la dinámica del proyecto, mientras Scrum (XA) se reunió con el cliente, los equipos Scrum (XP) organizaron sus radiadores de información (carteleras) en un sitio visible para todos (ver Ilustración 4). En todas las sesiones (6) de 8 horas cada una, se desarrolló al comienzo una reunión de 15 minutos entre el cliente y el equipo Scrum(XA) para feedback sobre el desarrollo del proyecto. Lo mismo hicieron los equipos Scrum (XP), que además de seguir las prácticas de extreme Programming, cumplieron con los eventos de Scrum dinamizados por AGATA, para la gestión del equipo.

En cada reunión, los equipos aprovecharon los canales de comunicación designados por AGATA; esto facilitó la trasmisión de la información inter- equipos, además de compartir experiencias en las conversaciones cara a cara en cada evento. Por las observaciones que se realizaron al material digital recopilado durante el desarrollo del proyecto, se notó una dinámica positiva entre los miembros de los equipos, una adherencia al método que se pudo notar en los resultados obtenidos. (Ver Ilustración 5)

Ilustración 5 Miembros equipos Scrum (XP)

El cliente tuvo un compromiso muy alto con el proyecto, estuvo disponible y participó activamente en el desarrollo de los requisitos, bajos los esquemas de las propuestas ágiles [123]. Además los equipos Scrum (XA) y Scrum (XP) siguieron las prácticas de planificación, de iteración y del trabajo diario de acuerdo a lo establecido por AGATA. Durante el desarrollo del proyecto se recolectó la información de acuerdo a lo planificado; además de los instrumentos propuestos por el diseño, se filmó cada jornada con el fin de obtener un archivo digital que permitiera repasar las actividades ejecutadas en cada sesión. La dinámica correspondió 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 y sucesos del proyecto durante el tiempo de su ejecución.
.

- **El ciclo de vida de AGATA**

Para el desarrollo del proyecto, el equipo siguió las fases del ciclo de vida de AGATA:

PRIMERA FASE: Exploración: Los miembros participantes del proyecto autónomamente se organizaron, basados en la gestión de AGATA, se dividieron en equipos Scrum (XP), luego ellos mismos, teniendo en cuenta los criterios del modelo eligieron un representante o líder de equipo, que además de velar por el trabajo de su equipo Scrum (XP), formó parte del equipo Scrum(XA).

En esta fase los miembros del equipo Scrum(XA), y el cliente realizaron su primera reunión: **Sprint Planning meeting**, un primer encuentro donde se comparten las inquietudes, necesidades respecto del proyecto a desarrollar y las condiciones fundamentales para la entrega del mismo. Como resultado de esta reunión, se obtiene **el product Backlog** del proyecto**;** una lista de los requisitos del sistema. Se enfatiza en el uso adecuado de los canales de comunicación propuestos por AGATA, debido a que en este estudio de caso fue uno de los puntos focales de la investigación. Con el cliente se priorizan los requisitos, los que Scrum(XA) se compromete a entregar en cada iteración para cumplir con el cronograma

del proyecto. En esta fase cada equipo Scrum (XP) organizó los artefactos dispuestos por AGATA para el arranque del sistema.

SEGUNDA FASE DESARROLO: En esta fase, se reunieron los miembros del equipo SCRUM (XA), eligen su product owner, quien se encargó de vigilar el cumplimiento de los requisitos, fue dueño del producto en ausencia del cliente y eligen también su scrum master encargado de observar la ejecución de las prácticas de AGATA. Una vez Scrum(XA) tiene claro el product backlog, elaboró la lista de tareas para la iteración, aquellos requisitos comprometidos a desarrollar. La estimación del esfuerzo se hizo de manera conjunta. Cada uno de los miembros de Scrum(XA) se convirtieron luego en los Product owner de sus equipos Scrum (XP), quienes también eligen su Scrum master encargado de motivar al equipo para el seguimiento de las prácticas de Extreme Programming gestionado. Los equipos tuvieron en cuenta dos reglas fundamentales para lograr el objetivo de esta fase:

- Tener muy claro lo que se debe entregar en el Incremento resultante del Sprint que comienza y
- Establecer la ruta con la que conseguirá hacer el trabajo necesario para entregar el Incremento [125].

Scrum(XA), estableció los objetivos de la fase, construyó la arquitectura del sistema, para ello se apoyó en las historias de arquitecto y aprovechó los aportes de ADD y QAW (Ver Ilustración 6 y 7) con el objeto de establecer atributos y escenarios de calidad propuestos por estos métodos de arquitectura y que fortalecerían al equipo AGATA.

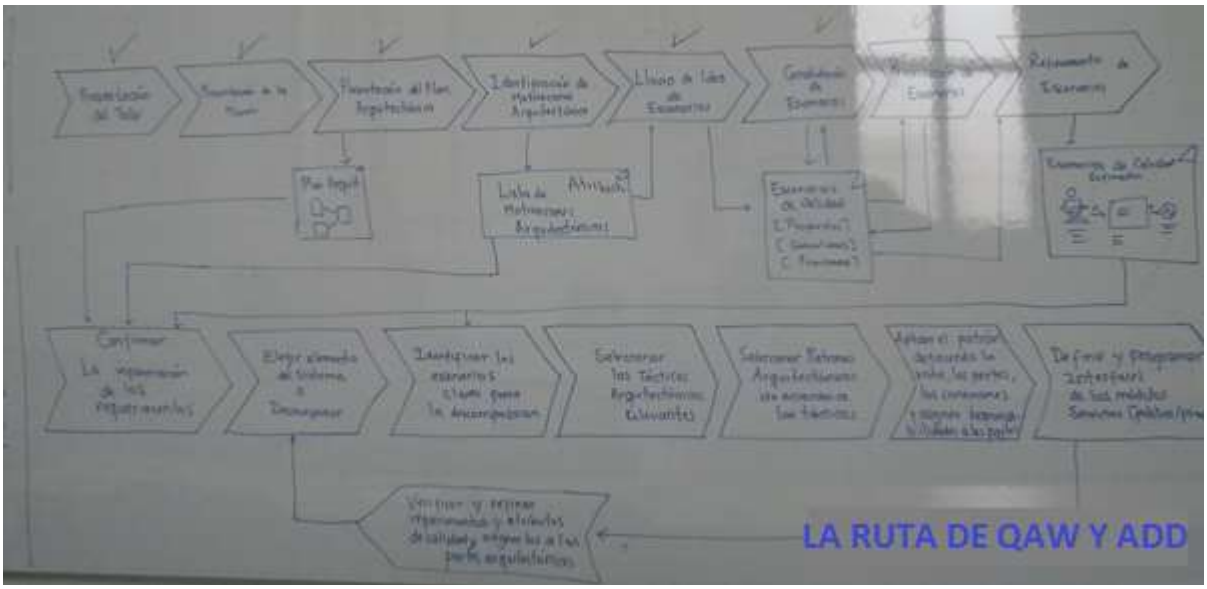


Ilustración 6 aplicando ADD

Ilustración 7 aplicando QAW

Cada representante de Scrum(XA) después de cada reunión, volvió a su equipo Scrum (XP) con las historias de arquitecto, donde cada equipo Scrum (XP) las convirtió en historias de usuario, con las que armaron sus product backlog y continuar con las prácticas de Extreme Programming gestionado: priorizar los requisitos, definir las tareas, y los tiempos de entrega. Luego, basados en la fase de planificación, se priorizaron los requisitos del sistema y se desarrolló un sketch de la arquitectura planteada por el equipo de arquitectura (Scrum (XA)). Para facilitar esta actividad, AGATA se apoyó en ArchAssitent (ver Ilustración 8), un mecanismo integrado proceso-producto para facilitar la incorporación de los métodos de arquitectura QAW y ADD [127] en el ciclo de vida durante el desarrollo del proyecto. En forma disciplinada Scrum(XA), tuvo en cuenta aspectos fundamentales de los atributos de calidad [117]. Entre ellos la escalabilidad y la modificabilidad que estaban relacionados con la evolución del software y la duración del ciclo de vida [126]. Esto llevó también a que se priorizaran tanto los atributos visibles al usuario final como los atributos que facilitarían la evolución del software. ArchAssitent ayudó a mantener un adecuado balance de todos estos atributos de calidad. En este caso la arquitectura se convirtió en el canal de comunicación y la estructura del sistema que comprende elementos software, las propiedades de esos elementos visibles externamente y las relaciones entre ellos [119]

Ilustración 8 ArchAssitent

Entrega Final: El equipo Scrum(XA) presentó al cliente los resultados de cada iteración, quién validó lo ejecutado y dio paso al siguiente sprint. AGATA entiende que un **Sprint** es una Iteración con una duración predefinida durante la cual los equipos Scrum(XA) y Scrum (XP) trabajan para convertir las historias de arquitecto y de usuario del ProductBacklog, a las que se comprometieron en cada sprint, en una nueva versión de software totalmente operativo.

Para cumplir con los objetivos del proyecto los equipos realizaron las siguientes reuniones previamente programadas:

Daily sprint meeting: Una reunión diaria de máximo 15 minutos, en la que los equipos se sincronizaron para trabajar de forma coordinada. Los equipos inspeccionaron el trabajo que estaban realizando. Esto permitió verificar el progreso hacia el objetivo de la iteración, los obstáculos que podrían impedir alcanzar este objetivo y las dependencias entre las tareas de cada uno. Los resultados de esta reunión sirvieron para hacer las adaptaciones necesarias que conllevaron a cumplir con el compromiso adquirido. Scrum(XA) hizo sus daily sprint meeting solapadas respecto a los equipos Scrum (XP) con el objeto de poder llevar a estas

reuniones todo el conocimiento de lo que iba sucediendo con el proyecto. El product Owner participó en la reunión con el equipo Scrum(XA) para verificar los avances del producto respecto a los compromisos adquiridos. El scrum master estuvo atento al cumplimiento de las prácticas de AGATA, y motivó al equipo a volver al camino cuando encontró alguna falencia sobre la aplicación del modelo

Sprint Review : El equipo de Arquitectura Scrum(XA) realizó una reunión al final de cada sprint (tres en total), donde presentó al cliente aquellos requisitos completados en la iteración en forma de incremento de producto y haciendo un recorrido lo más cercano posible al objetivo que se había comprometido a desarrollar en la reunión de planificación.

En esta reunión el cliente puedo ver de manera objetiva cómo se desarrollaron los requisitos del sistema; los comparó con sus expectativas; le permitió también re-planificar el proyecto para tomar mejores decisiones. Por otro lado, a través del canal de comunicación la arquitectura, el equipo pudo observar si realmente se habían comprendido y compartido los requisitos que solicitó el cliente y en qué puntos hubo que mejorar la comunicación entre ambos. (ver Ilustración 9,10, 11)

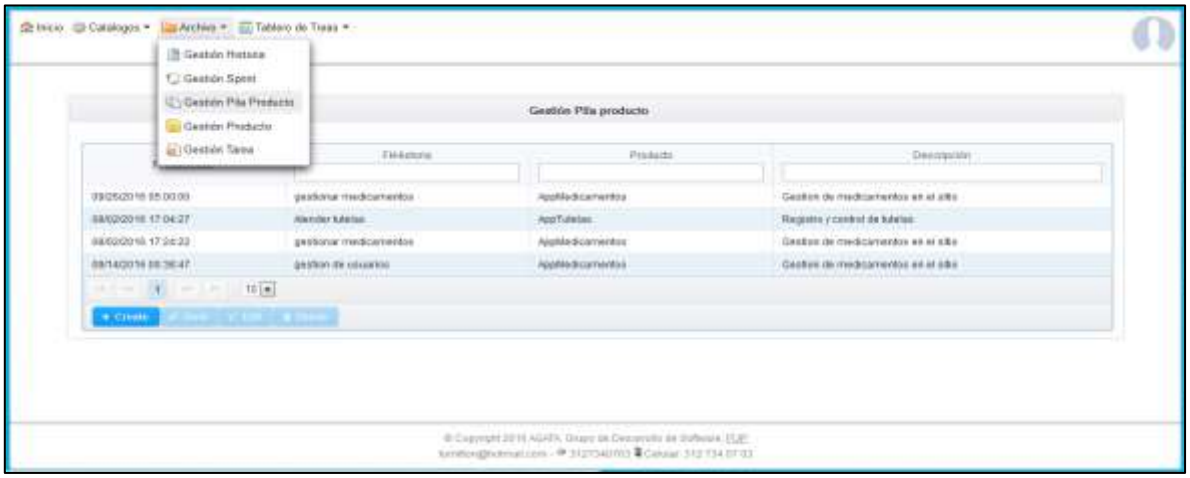


Ilustración 9 Interfaz de usuario

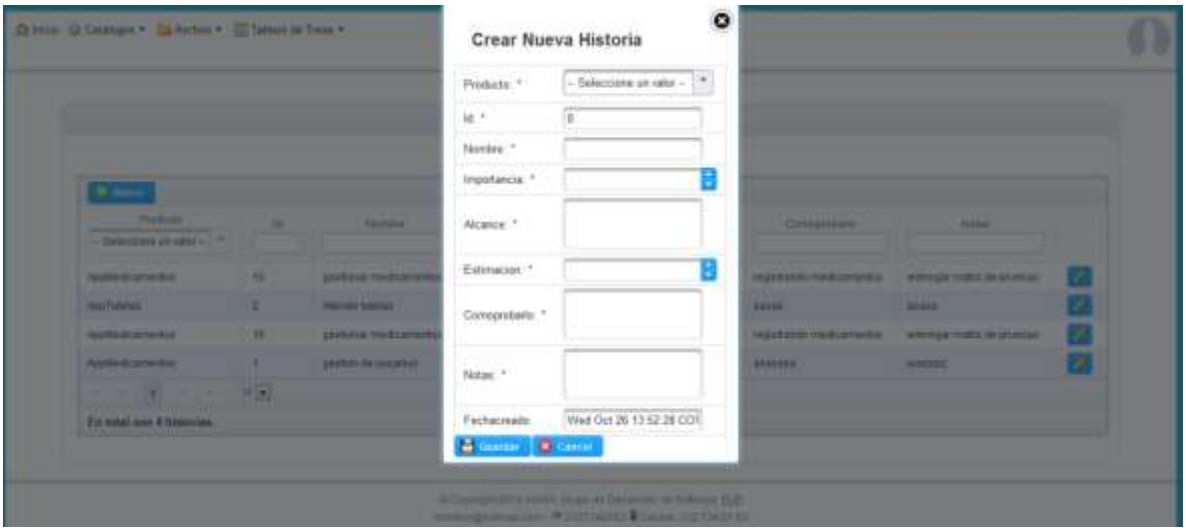


Ilustración 10 Historia de usuario

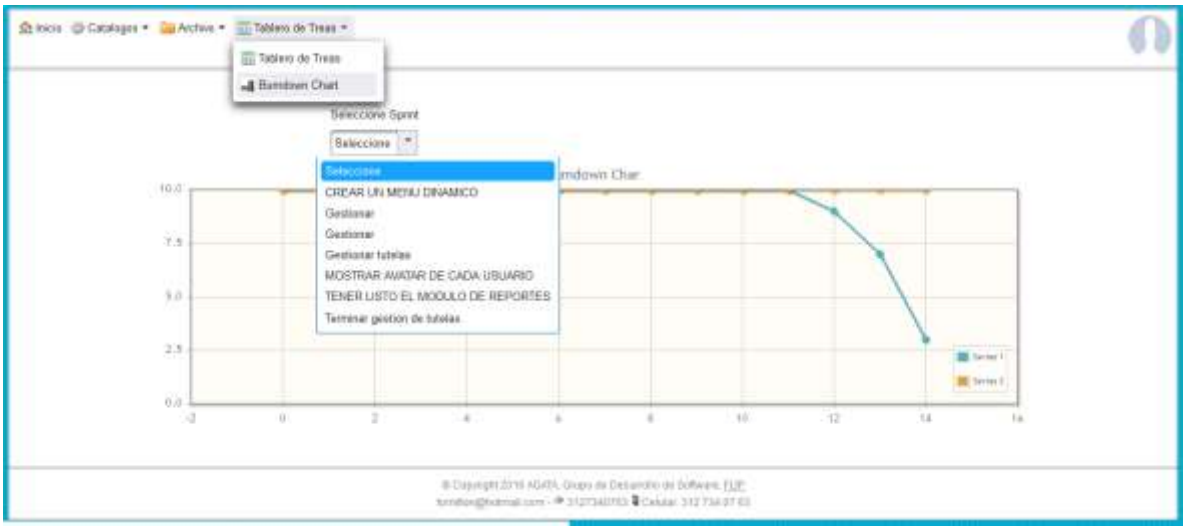


Ilustración 11 burndown chart

Sprint retrospective: Una reunión realizada por todos los equipos de AGATA una vez finalizado el Sprint; se analizó qué se hizo bien, qué procesos serían mejorables y se discutió acerca de cómo mejorarlos. Aprovechando los indicadores para la evaluación del proceso (AGATA), se guardó una bitácora de aquellas lecciones aprendidas que le permitirían a AGATA incrementar la productividad. El Web master se encargó de validar el trabajo respecto a las prácticas propuestas por AGATA y quitar los obstáculos que no permitirían alcanzar los objetivos. Aprovechó para recordar algunas prácticas referentes a la comunicación de los mensajes que surgieron de la arquitectura. (ver Ilustración 12)

Ilustración 12 Reunión de retrospectiva

- **Resultados del caso de estudio:**

Como se expresó en la tabla 5 para evaluar este estudio de caso se propusieron dos indicadores basados en la calidad de la comunicación [128] y en la eficiencia del canal usado para transmisión de la información y cada uno con sus ítems que explicaran el indicador.

- **Calidad de la comunicación**

La calidad de la comunicación del equipo (Scrum/XP) y el Equipo (Scrum/XA) usando como canales de transmisión la conversación cara a cara y la descripción de la arquitectura para entender la lista de requisitos priorizada del negocio en los niveles estratégico, táctico y operativo [129]

Para este indicador se establecieron tres categorías:

- **La Distorsión (D):** Este indicador permite medir el grado de Distorsión del mensaje al comunicar los requisitos del cliente [120].

Para medir la Distorsión se propuso la siguiente fórmula:

$$D = \{(NDRF/NMERF) + (NDR/NMER) + (NDA/NMEA)\}$$

Donde:

ND es el número de defectos encontrados.
NME es el número de mensajes transmitidos.
RF hace referencia a mensajes y defectos con requisitos funcionales
R hace referencia a mensajes y defectos con información de restricciones

A hace referencia a mensajes y defectos con información Arquitectónica (Drivers, Tácticas y Estrategias)
Para obtener la información, se realizaron observaciones directas en las reuniones de planificación, iteración y retrospectiva; también se hicieron grabaciones de estas reuniones y algunas preguntas aleatorias a los participantes en cada evento.

Se evaluaron los mensajes compartidos en cada una de las reuniones entre el cliente y el equipo Scrum (XA) y los equipos Scrum (XP). En AGATA un mensaje es la descripción de un requisito del cliente o de Scrum (XA), que puede afectar al producto.

Para este caso se analizaron 37 mensajes compartidos entre el cliente y el equipo AGATA de los cuales se notó que el 26% de ellos no fueron claramente entendidos por los equipos y que necesitaron de alguna aclaración. (ver Figura 29)

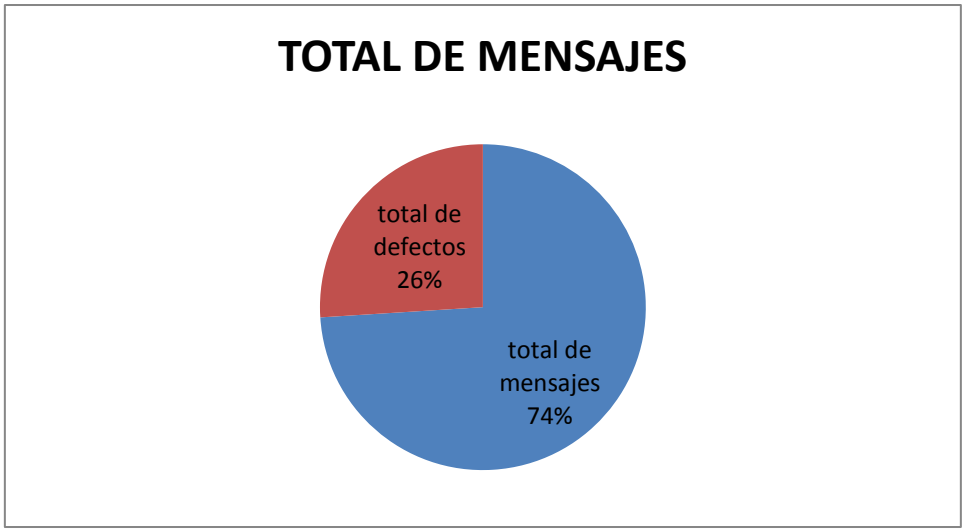


Figura 29 Defectos

Los defectos encontrados, se refirieron específicamente a los requisitos funcionales, las restricciones y la arquitectura cuyos resultados fueron:

Distorsión requisitos funcionales: 0.9
Distorsión restricciones: 0,3
Distorsión Arquitectura: 0,4

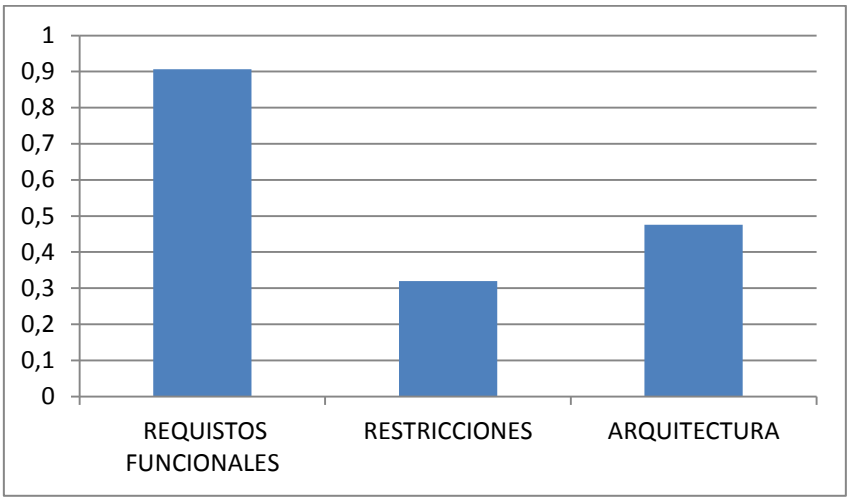


Figura 30 Distorsión

Con base en los resultados, se pudo comprobar que la medianamente alta, es decir los miembros de los equipos no usaron los canales, adecuadamente, contra un pero el 74% que lo usaron adecuadamente y compartieron los mensajes emitidos por el cliente. Sin embargo, se pudo notar (como lo muestra la figura 30) que la distorsión más alta fue causada por los requisitos funcionales lo que llevó a analizar que algunos mensajes no llegaron a su destino con la misma fidelidad con la cual fueron enviados. Se pudo comprobar que los receptores de la información no captaron correctamente el mensaje, debido a que necesitaron pedir explicaciones por más de una vez por alguna especificación en el sistema; le dieron distintos significados a los artefactos usados en la arquitectura, además como la información debía atravesar varios nodos en la comunicación, en algún momento pudo alterarse, la sobrecarga por el tamaño del equipo que necesitaba intercomunicase fue bastante grande.

- **La convergencia:** entendida como el porcentaje de la información comprendida por el Equipo (Scrum/XP), usando los canales de comunicación establecidos, respecto a la información emitida por el equipo (Scrum/XA).[121].

  Para medir la convergencia se propuso la siguiente fórmula:

$$S = (NMC/NME) * 100 \text{ (Comprendidos/Enviados)}$$

Donde:

NMC es el número de Mensajes comprendidos.
NME es el número de mensajes enviados.

Cabe aclarar que para AGATA una unidad de mensaje es la descripción de un requisito del cliente; un mensaje comprendido es una funcionalidad del sistema documentada.

Para recoger estos datos, se realizó una observación directa en las reuniones de planificación, iteración y retrospectiva, al mismo tiempo, se hicieron algunas preguntas aleatoriamente a los miembros que participaban en cada una de las reuniones.

Con base en el análisis que se hizo de las observaciones y los videos respecto a los

mensajes intercambiados en cada reunión, se pudo concluir que el nivel de comprensión fue del 76% (ver Figura 4.4 y 4.5) de un total de 54 mensajes compartidos (ver Figura 4.6)

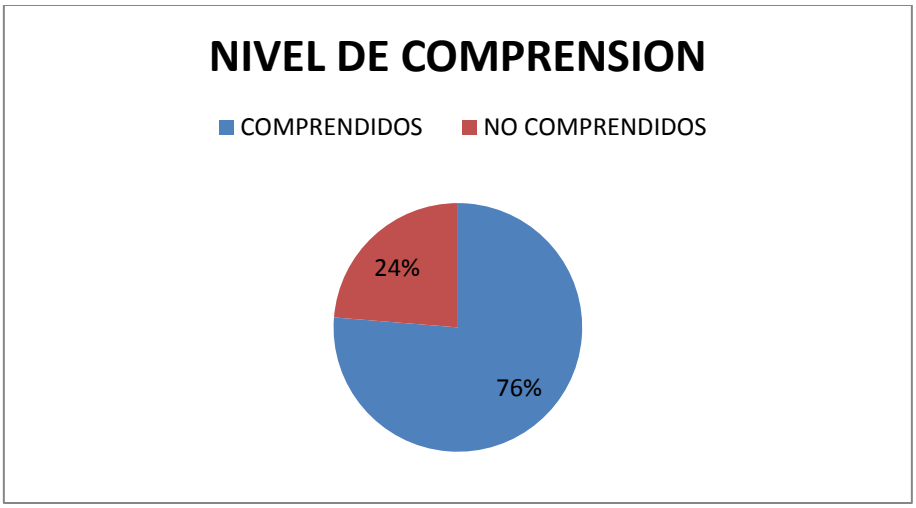


Figura 31 Nivel de compresión

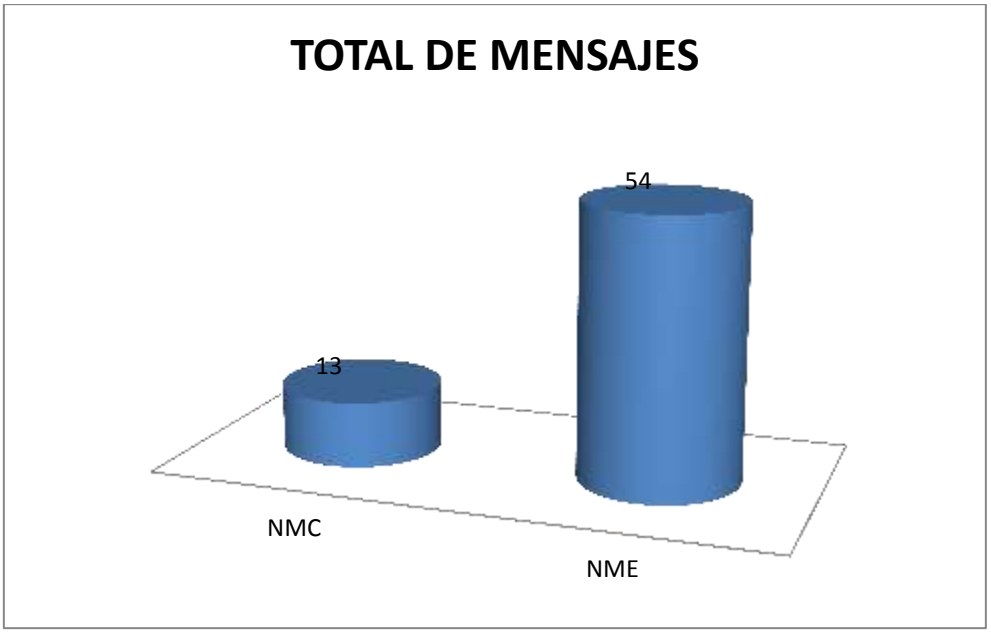


Figura 32 Total de mensajes

Por los resultados en esta validación, se pudo concluir que el grado de comprensión fue aceptable, los canales usados en este aspecto la arquitectura y el cara a cara lograron modestamente su propósito, sin embargo al analizar el 23% de mensajes no comprendidos, se pudo observar que el equipo una vez recibida la información le dio su propia interpretación a información ambigua. Se identificó que las fuentes que emiten los mensajes en algunos momentos no fueron suficientemente claras y comprensibles en la emisión de sus mensajes para ser entendidos por sus receptores. La idea es que el receptor debe hacer lo que el mensaje diga que haga. Queda pendiente en el modelo, trazar estrategias para

evitar la compresión subjetiva es decir la determinación de cómo los participantes en el proyecto ven un mensaje en particular influenciados por variables obtenidas de sus propias experiencias.

- **Velocidad del Canal (V):** Entendida como la cantidad de mensajes respondidos por el Equipo (Scrum/XP) al equipo (Scrum/XA) usando los canales de comunicación establecidos. Es también el promedio de la cantidad de mensajes transmitidos durante un periodo de tiempo

Para calcular la velocidad se usó la fórmula:

$$V = MC/\ T$$

Donde

V= Tiempo de duración de una reunión
MC= Numero de mensajes compartidos

Para obtener la información se realizó una observación directa en las reuniones de planificación, iteración y retrospectiva y se tomó el tiempo de cada reunión. Además se filmaron estas actividades para tener certeza de los datos encontrados.

Para medir el indicador, se aprovecharon las tres reuniones que se realizaron con el cliente, con Scrum (XA) y con Scrum (XP). Cada reunión tuvo una duración de 15 minutos (ver Figura 4.6) Tiempos de reunión). Un total de 45 minutos en las tres reuniones y donde se compartieron 37 mensajes.

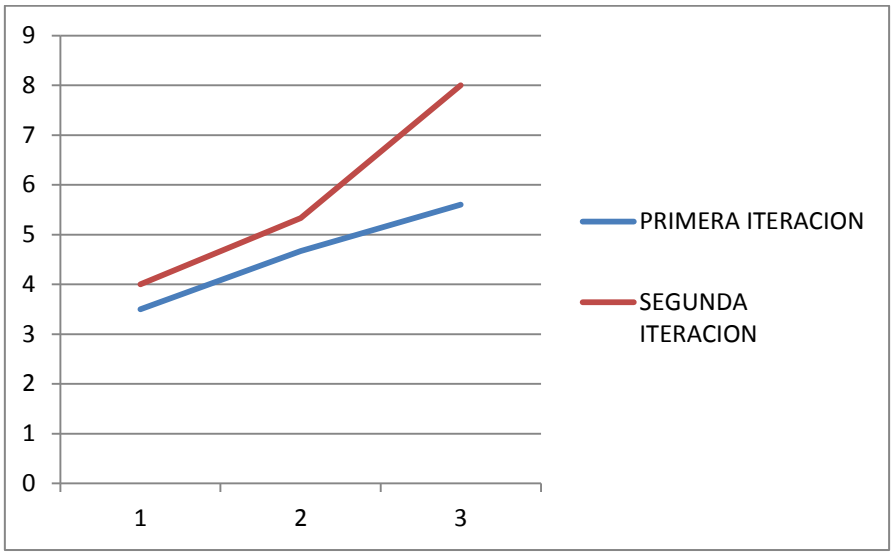


Figura 33 tiempos de reunión

Con base en la experiencia de esta categoría, se pudo observar que la velocidad de los equipos respecto a los mensajes compartidos y comprendidos en todas las reuniones fue constante (0.6); 6 mensajes por cada 10 minutos. Aspecto que se pudo declarar como positivo, porque los equipos no demoraron mucho compartiendo la información.

En conversación con los equipos, manifiestan, que el interés por los equipos de desarrollar lo más pronto sus tareas, hizo que las reuniones fueran muy dinámicas y rápidas. Además porque la arquitectura planteada por el equipo Scrum (XA) fue bastante clara para entender relativamente lo que se debía desarrollar. Por otro lado, como hubo una continua interacción entre los miembros del equipo y con el cliente, los asuntos del proyecto se estaban compartiendo continuamente, de modo que cuando se llegaba a las reuniones sobre todo a las de revisión y retrospectiva, habían mensajes que ya habían sido tratados al interior del equipo y los product owner llegaban a las reuniones con estos informes muy claros y concisos. Pero se debe aclara que la dinámica implantada por Scrum (XA) no validó completamente como positivo los resultados del proyecto; esto sucedió, porque por fuera de los encuentros de equipo deberon reunirse algunos su-equipos para aclara dudas, lo que retrasó la entrega de producto funcional. Es necesario entonces plantear otras estrategias que brinden la posible superar la velocidad expuesta en esta validación.

- **Calidad Percibida por el equipo:** Este categoría permite al indicador medir la calidad de la comunicación, los aspectos positivos y negativos en la comunicación identificados por los equipos. Para ello se tuvieron en cuenta, el conjunto de apreciaciones del equipo frente al intercambio de los mensajes y los resultados obtenidos por el proyecto. Para realizar esta medición, se hicieron unas encuestas (ver anexo encuesta Calidad Percibida CPE) a los miembros del equipo durante la reunión de Retrospectiva.
  La encuesta tuvo 10 preguntas orientadoras y se les realizó a los 20 participantes del proyecto. Para un total de 200 apreciaciones (ver Figura 4.7)

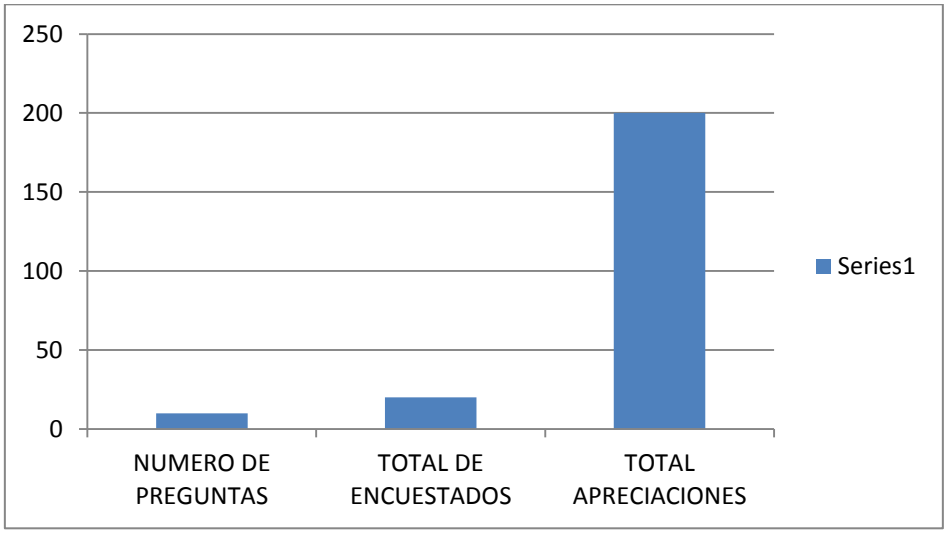


Figura 34 Calidad Percibida

Al analizar estos resultados, se pudo comprobar que el 89% de los miembros del equipo manifiestan que la calidad de la comunicación es alta; el 11% comenta que la calidad es baja debido a algunas incoherencias en los mensajes (ver Figura 35)..

Figura 35 Calidad de la comunicación

Los resultados obtenidos en cuanto a la calidad percibida demostró que los canales usados para la transmisión de los mensajes fueron fáciles de usar y suficientes para enviar y entender la información que se compartía. Sin embargo, se pudo observar también que hubo información que no llegó tal como se había transmitido y que fueron necesarios usar otros medios para que esa información llegara a su destino. Por la cantidad de información que se movía al interior del grupo, algunos mensajes se retrasaron en llegar para cumplir con su tarea de información.

Al promediar la calidad de la comunicación con base en las categorías de Distorsión, Comprensión, Velocidad y calidad percibida por los equipos, se pudo notar que existe un alto grado de confianza en los equipos al transmitir sus mensajes para el desarrollo de las actividades requeridas por el proyecto que se estaba desarrollando. Pero y como se puede notar también en los resultados, existieron todavía hay falencias que se analizaron para saber cuál de los canales propuestos por AGATA estaban generando esta dificultades que impedían que la comunicación alcanzara unos niveles más altos de aceptación y calidad.

Para ellos se propone entonces trabajar un segundo indicador basados en el estudio de los canales de comunicación.

- **Segundo indicador: Calidad de los canales de comunicación**

Al igual que con el anterior indicador, se establecieron tres categorías, que permitieron entregar mejores resultados respecto de la evaluación. Esas categorías fueron: la efectividad, la aceptación, y la confiabilidad de los canales.

Recordemos que para AGATA el canal es el medio utilizado por los equipos para la transmisión de los mensajes. AGATA propone dos canales: LA ARQUITECTURA y EL CARA A CARA, que fueron evaluados con los siguientes ítems:

- **Efectividad del Canal (EC):** Que consistió en la eficacia del canal para comprender y entregar el mensaje entre los equipos Scrum (XA) y Scrum (XP).

Para su medición, se propuso la siguiente fórmula:

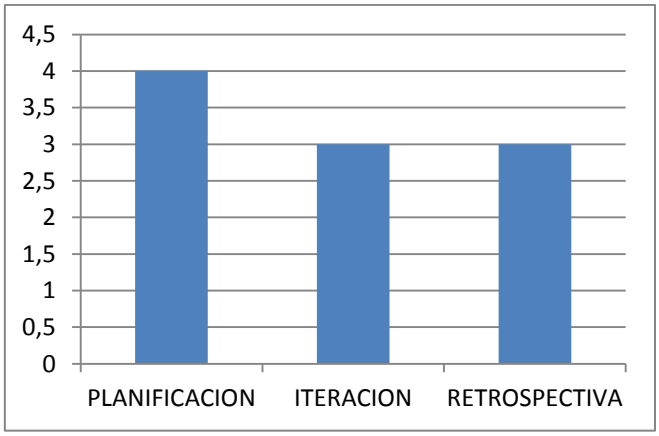
$$EC = (MC/MT)*100$$

Donde

MC = (1 - Mensajes Incorrectamente Transmitidos debido al Canal
MT = Mensajes Totales Transmitidos

Para medir esta categoría, se realizaron observaciones y se filmaron diferentes eventos en las reuniones y actividades de los equipos, además se realizaron encuesta a los miembros de los equipos Scrum (XA) y Scrum (XP) después de cada reunión.

Estas observaciones y encuestas se hicieron en las tres reuniones de los equipos. Es así que para medir la efectividad de los enlaces técnicos (cara a cara) (ver figura 36)



Se encontró una efectividad del 73% (ver Figura 37)

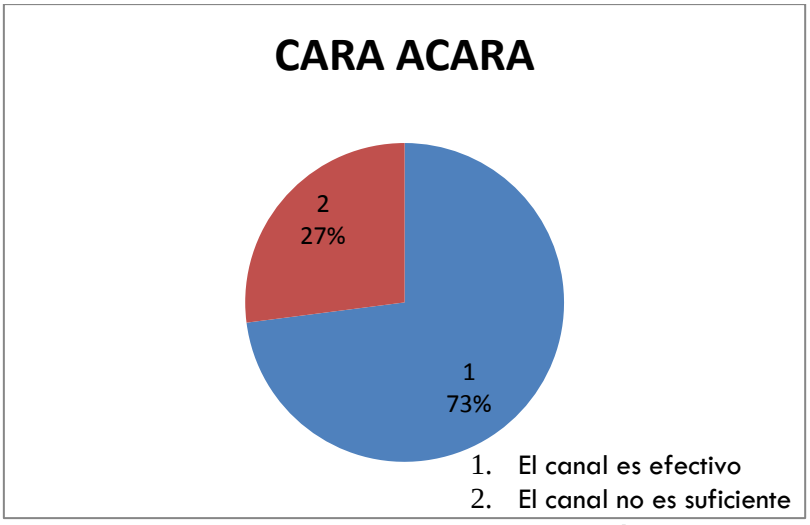


Figura 37 Efectividad de los enlaces técnicos

Las mismas condiciones se tuvieron en cuenta para medir la efectividad de la arquitectura como canal (ver Figura 38)

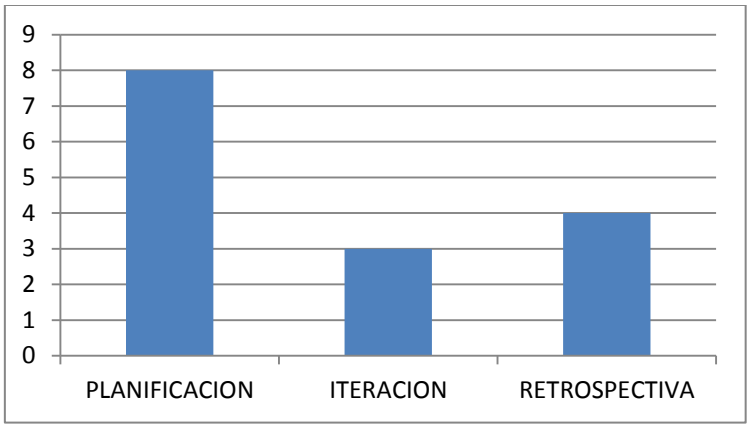


Figura 38 Medición de la arquitectura

Dando como resultado una efectividad del 59%; menor que la efectividad de los enlaces técnicos (ver figura 39)

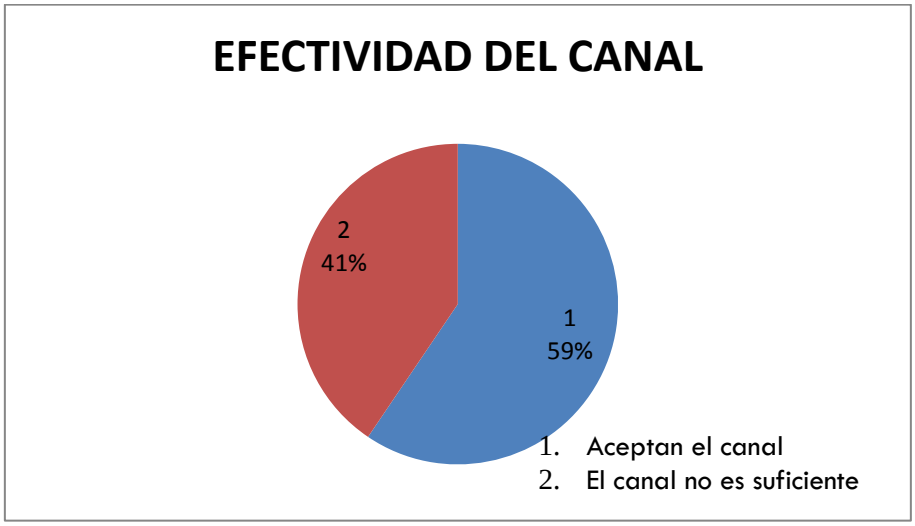


Figura 39 efectividad de la arquitectura

Al corroborar estos resultados entre los mensajes que presentaron defectos entre el total de mensajes compartidos por los equipos, se comprobó que el 64% de los mensajes cumplieron con su objetivo. Es decir que, además de seguir el flujo propuesto por AGATA, los resultados que logrados por el mensaje fueron los esperados. Sin embargo, el 26% de los mensajes presentaron defectos; en el camino recibieron algún tipo de ruido que no les permitió llegar claramente a su destino. Tratándose de software, es un porcentaje bastante alto debido a la responsabilidad que lleva esta actividad para cumplir con los requisitos del cliente y mas aun si se trata de software de alta complejidad.

Aceptación del canal: Es el Grado de aceptación del canal usado por los integrantes de los equipos Scrum (XA) y Scrum (XP)

AC= (suma(RE_i)/TP)*100

RE= sumatoria de las respuestas contestadas por los encuestados.
TP: Total de preguntas realizadas a los encuestados

Para medir este indicador se realizaron 20 encuestas a los miembros de los equipos Scrum (XA) y Scrum (XP) un total de 200 preguntas; obteniendo que para el 84% de los encuestados, la aceptación del canal fue alta, y para el 16% la aceptación del canal fue baja. (Totalizando los resultados de cada canal)(ver Figura 40 y 41)

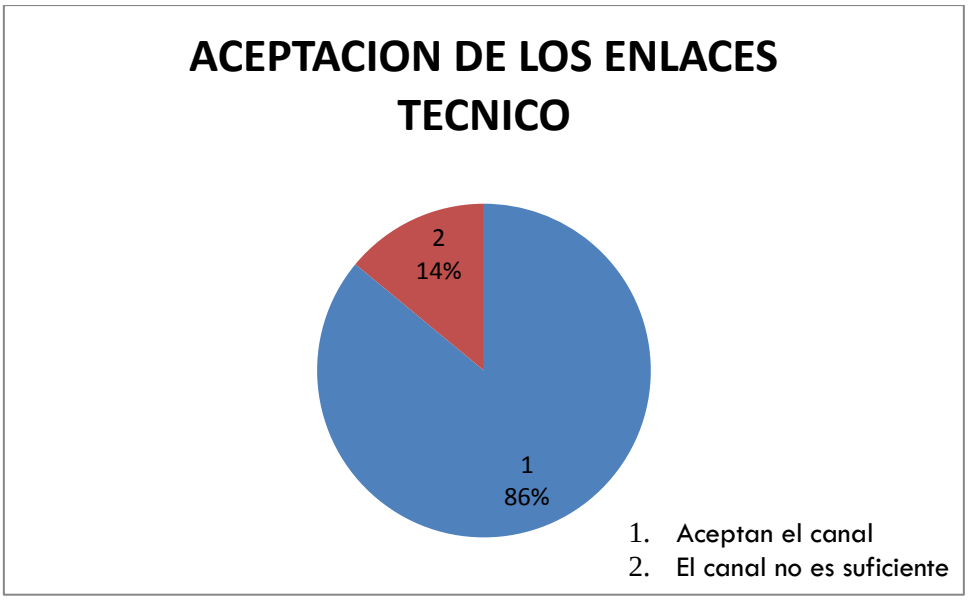


Figura 40 Enlaces técnicos

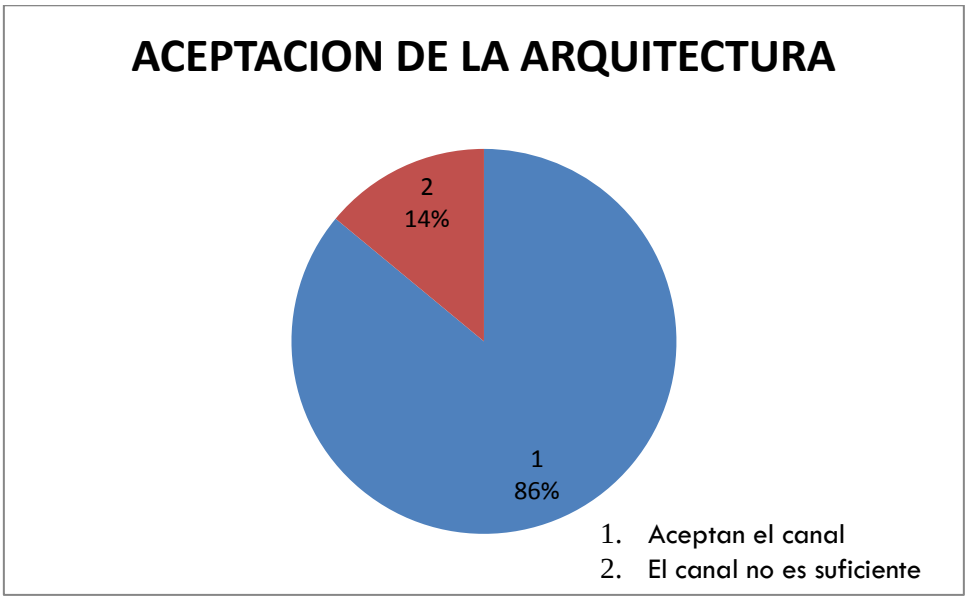


Figura 41 Arquitectura

En este indicador se tuvo en cuenta aquellos factores que permitieron establecer la dimensión de los atributos más importantes de cada uno de los canales utilizados por los equipos; además se observó el impacto del canal en cuanto a su demanda y la calidad de mensajes compartidos a través de ellos, que con los resultados obtenidos, los canales propuestos por AGATA tuvieron un nivel de aceptación bastante alta.

En conversaciones informales con algunos miembros del equipo, para analizar el 16% sobre la aceptación baja del canal, se pudo observar que se presentaron algunos factores de dominancia de los hábitos comunicacionales por algunos miembros del equipo en algunos momentos en que los canales no fueron suficientes para compartir la cantidad de información que se compartía en torno al sistema. A veces como que los hábitos individuales pesaron sobre la incorporación de un modelo nuevo que proponía canales de comunicación para equipos de tamaño mediano. AGATA debe en estos casos establecer estrategias de culturización y adecuación de los equipos hacia los artefactos usados por el modelo, para lograr niveles más altos de aceptación.

- **Confiabilidad del canal:** Es el grado de confianza que tienen los miembros del equipo con los canales usados por los integrantes de los equipos Scrum (XA) y Scrum (XP)

AC= C= Suma(F_i)/TP*100

Fi= Evaluación del factor i
TP= Total de preguntas

Para corroborar el nivel de aceptación del canal, se decidió hacer otra encuesta a los miembros de los equipos con el objeto de medir el grado de confianza que percibían ellos en los canales propuestos por AGATA. La intensión, tener una visión imparcial sobre estos medios, dado que fueron propuestos por el modelo para escalar los métodos ágiles y atender de la mejor manera los desafíos de la comunicación. Se necesitó saber que aunque no son los únicos canales para trasmisión de mensajes en características iguales. De todas maneras era necesario validar la hipótesis y solidificar la aceptación de los canales a través de la confianza en ellos por parta de los miembros de los equipos.

Se encuestaron 20 integrantes de los equipos un total de 200 preguntas de las cuales el 88% de los miembros reportaron un nivel de confianza alto y el 12 % reportaron un nivel de confianza bajo (Totalizando los resultados de los dos canales) (Ver Figura 42 y 43). Al contrastar estos resultados con el nivel de aceptación del equipo quienes reportaron 88 % y 16 % respectivamente, se pudo concluir que las mediciones y sus resultados fueron simétricos, por lo tanto, se pudo decir que los canales propuestos por AGATA para enfrentar los desafíos de la comunicación son aceptables y confiables en equipos de tamaño mediano.

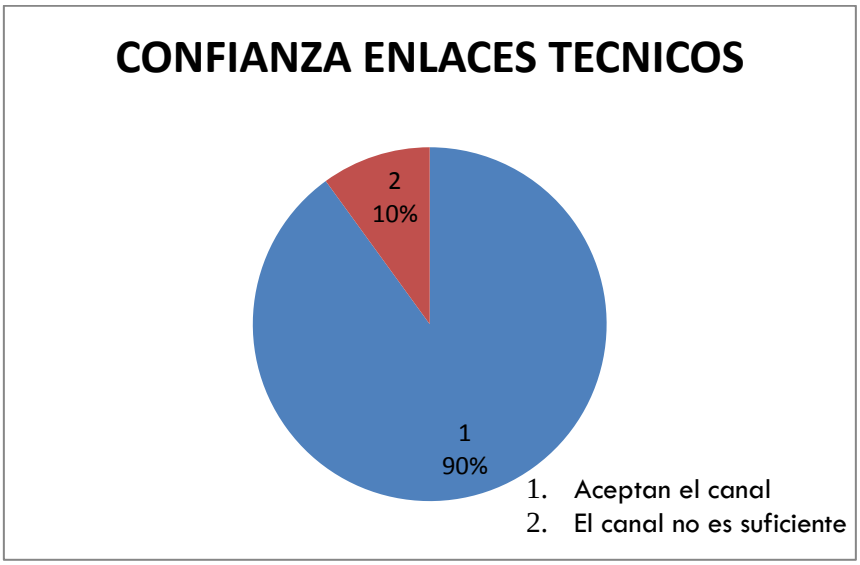


Figura 42 Enlaces técnicos

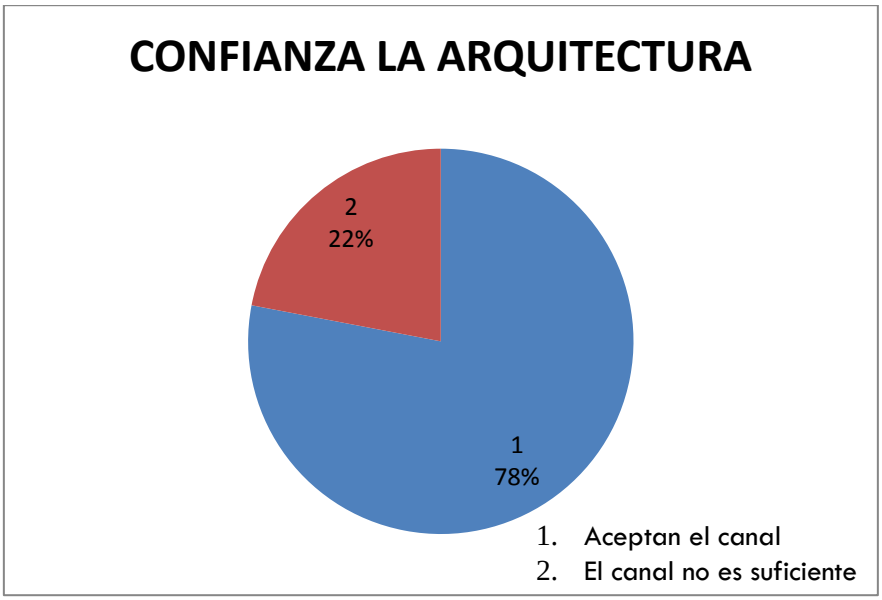


Figura 43 Arquitectura

- **Análisis cualitativo del desempeño de los canales de comunicación**

Con la validación de los canales propuestos por AGATA se pudo verificar el impacto que causa la comunicación al interior de los equipos de desarrollo de software; no es solo un intercambio de opiniones entre personas, sino una actividad cuyo propósito es generar acción, que exige además de transferir ideas, el intercambio de sentimientos, actitudes, emociones y deseos. La comunicación requiere ciertas habilidades de los actores que participan en ella. Una especie de percepción y de comprensión del comportamiento del otro que conduce a una sensibilidad por sus necesidades y a una flexibilidad para saberlas manejar con justicia y objetividad. Significa comprender al otro, aunque no se esté de acuerdo con él o no se acepten totalmente sus puntos de vista. Cuando los equipos crecen

la comunicación crece dramáticamente y con ella crecen las subjetividades, individualidades que se deben controlar estableciendo métodos y canales claros por donde deben transitar todas las ideas y mensajes compartidos; donde los canales ayuden a:

- La gestión en el discurrir de los mensajes y elaborar estrategias comunicativas más adecuadas.
- Establecer límites de la información que se debe compartir, sirviendo de enlace entre los equipos y el cliente.
- Apoyar a los equipos a adaptarse aportando nuevas ideas procedentes del entorno.
- Comunicar las ideas que fluyen en el intercambio de mensajes a todos los participantes del proyecto.

La planificación de la comunicación en AGATA se convirtió en un escenario teórico-descriptivo que relató todo aquello que había que hacer y con qué se contaba para ello. Incluyó metas, objetivos, modos y estrategias para conseguir lo que se pretendía, además se desarrollaron las tácticas, acciones y herramientas que soportaron las intenciones y mensajes que fluyeron al interior del grupo.

Los canales se convirtieron en el medio más eficaz para seguir objetivamente las directrices planteadas por el modelo. Con los resultados obtenidos, se pudo verificar que fueron altamente aceptados por los actores del proyecto.

Sin embargo, se pudo constatar también, que cuando los equipos son grandes y disimiles es decir conformado por personas con diferencias individuales muy notables en cuanto al trabajo en equipo, la incorporación a nuevos modelos, métodos y desarrollo de trabajo colaborativo, el modelo falla y los canales propuestos en este caso por AGATA, no son suficientes para lograr una completa unicidad del grupo; de ahí que es necesario establecer estrategias de culturización del modelo, para involucrar activamente a todos los miembros del equipo, de tal manera que usen cabalmente los canales y el modelo de comunicación para lograr con mayor efectividad los objetivos propuestos en el proyecto.

Capítulo 5

El conocimiento no es una vasija que se llena, sino un fuego que se enciende. Plutarco

Conclusiones y trabajo futuro

En este capítulo se describen las conclusiones de la investigación alrededor de **AGATA**, en particular, la posibilidad de escalar el método Extreme Programming – XP con prácticas que enfrentan los desafíos del comunicación 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

Con esta investigación, se pudo concluir que las metodologías ágiles, Extreme Porgramming (XP) para este caso, ha demostrado trabajar para proyectos de baja complejidad y equipos pequeños (5±2 personas) [49]. También, que intentar escalarlas a equipos más grandes se requiere de ciertas actividades complementarias que soporten los parámetros ágiles, con herramientas y patrones que mantengan la agilidad en la dinámica del equipo.

Esta tesis ha abordado una propuesta que permite escalar el método XP a través de un modelo holístico centrado en la arquitectura de software, la gestión y la comunicación. La propuesta ha sido denominada **Agile Architecture in Action (AGATA)**, aplicada a DOS estudios de caso embebidos (Cinco proyectos en total) en el contexto de cursos de último año del programa de ingeniería de sistemas de la Fundación Universitaria de Popayán, y empresas del sector industrial, en los que participaron 103 personas en total para el desarrollo de los proyectos. En los estudios, AGATA demostró que escala los métodos ágiles, que controla la comunicación a través de unos canales propuestos por el método, mientras gestiona holísticamente al equipo en el marco de proyectos con equipos de tamaño mediano (2 a 9 sub-equipos).

En equipos de tamaño mediano, es necesario establecer elementos de gestión que permitan controlar y organizar el equipo en torno al proyecto software: Para ello AGATA se apoyó de Scrum un marco de proceso para la gestión de equipos [55].

Las prácticas de gestión y de generación de producto, a través de AGATA, pudieron conducir al equipo para alcanzar las conductas esperadas, basadas en la auto – organización, la autonomía y el auto – control durante los eventos y fases del proyecto.

AGATA incorporó guías técnicas para la socialización del modelo estableciendo sus propios valores que sirvieron de guía para el cumplimiento de las prácticas ágiles y obtener producto dentro de los parámetros fijados por Extreme Programming (XP)

Debido a que la comunicación es un factor trascendental en los equipos ágiles, que por cierto es poco reportada en estos ambientes; cuando los equipos crecen, AGATA comprobó que se distorsiona y los resultados finales no son los esperados por los actores del proyecto.

Para ello, AGATA incorpora dos canales de comunicación en su modelo: La arquitectura y el cara a cara, que lograron satisfacer y controlar en un alto porcentaje la cantidad de información que fluía a través de los equipos. Planificar la comunicación a través de los canales, es decir que se debe comunicar y que tanta información debe circular a través los mismos, sirvió para evitar en un alto grado de confiabilidad la distorsión causada por el exceso y/o la ambigüedad.

Los canales permitieron establecer una comunicación asertiva, con cierto grado de autonomía y de confianza que dinamizó la creatividad del equipo con base en sus interacciones.

La arquitectura y el cara a cara, mejoraron la comunicación en cuanto a la calidad y la eficiencia de los medios para la transmisión de los mensajes, minimizaron la distorsión, mejoraron la calidad percibida por los miembros de los equipos, demostraron velocidad al transmitir los mensajes necesarios para el desarrollo del proyecto y generaron un alto nivel de confianza por los resultados obtenidos.

A partir de las prácticas de comunicación y de los canales, las conversaciones inter- grupo y con el cliente se notaron más abiertas y dinámicas, los mensajes emitían lo "que se necesitaba hacer" evitando redundancias y contratiempos en el desarrollo del proyecto. Se eliminaron las barreras y la burocracia derivada de una comunicación horizontal.

La planificación con el cliente y entre los miembros del equipo, sirvió para solidificar los objetivos del proyecto, establecer ambientes de proactividad, valentía, flexibilidad y respuesta al cambio.

Los miembros de los equipos y el cliente ante los eventos generados por el modelo demostraron un alta capacidad de auto - aprendizaje, entendieron que como equipo se aprende todos los días y en cada práctica, así mismo que con sus aportes se validaba y re-construía el modelo.

AGATA generó valor para el cliente, el proyecto, el producto y el equipo. Con los resultados obtenidos, cada una de estas instancias pudo comprobar que se pueden mantener los criterios ágiles con equipos de tamaño mediano a través a través de prácticas gestionadas y adecuando canales emergentes que controlen la comunicación.

El modelo involucró a todos los miembros del equipo y al cliente, AGATA abordó estrategias de gestión a nivel organizacional que permitieron generar responsabilidades en cada uno de los participantes midiendo sus resultados y motivándolo a mejorarlos.

Específicamente desde la definición de AGATA y su aplicación en los estudios de caso se pudieron observar los siguientes aspectos:

- La comunicación entre los miembros del equipo AGATA, debe ser clara y coherente a través de los canales propuestos por el modelo. Los miembros del equipo AGATA deben conocer claramente los objetivos del proyecto y los requisitos de negocio del cliente. La participación de los miembros del equipo Scrum (XA) como clientes de cada equipo Scrum (XP) fue decisoria para que estos aspectos se lograran en la práctica.

- Los representantes del equipo Scrum (XA) en AGATA, además de ser los elementos de conexión entre el equipo de arquitectura y los equipos Scrum (XP), también deben ser los dinamizadores del equipo. Desde una perspectiva técnica, tener un equipo responsable y orientador de la infraestructura del sistema causa confiabilidad en todos los equipos Scrum (XP) y Scrum (XA) para alcanzar las metas del proyecto.

- Los equipos deben preferir usar los canales de propuestos por AGATA antes que sus propias individualidades. Esto hará que el que equipo equilibre sus criterios y se mantenga unido en torno al proyecto.

- Al final de cada iteración, a través de la reflexión, los miembros del equipo comprendieron su papel para lograr la integridad del modelo holístico, su responsabilidad como arquitectos y lograron enfrentar los desafíos de la comunicación.

5.2 Limitaciones

Desde la perspectiva del investigador, es necesaria una mayor aplicación a equipos heterogéneos en prácticas industriales. En estos ambientes como se pudo comprobar en un estudio de caso aparecen variables que son difíciles de controlar máxime cuando los equipos son grandes. Se trata de medir la eficacia del método propuesto de una forma más generalizada. Estas experiencias aportarían para solidificar más aún a AGATA.

.Aunque AGATA 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.

Debido al tamaño del equipo, existen diferencias individuales que afectaron el desarrollo del proyecto, algunos miembros de los equipos no se adhirieron completamente al método, en ciertos momentos prefirieron usar su experiencia antes que las prácticas de AGATA. Esto hizo que los canales usados para la comunicación no alcanzaran la eficiencia esperada.

El arranque del proyecto, se notó pérdida de tiempo debido a que ni los equipos ni las herramientas ni la arquitectura estaban listas para iniciar como los demandaba el modelo. Esto limitó en cierta medida los tiempos de desarrollo y afectó en algún momento la producción en el proyecto

5.3 Trabajo futuro

Uno de los principales trabajos futuros es extender la evaluación a más estudios de caso en contextos diferentes. Particularmente, desarrollar nuevos casos en contextos industriales donde las variables que se presentan en este tipo de ejercicios son difíciles de controlar. La aplicación del modelo a estudios de caso académicos ha permitido establecer que para corroborar los resultados obtenidos en esta investigación es necesario considerar otras variables de contexto como las establecidas por [11][108] que permitan hacer una comparación teniendo en cuenta por ejemplo, la experticia de los participantes en los diferentes aspectos: procesos, métodos, tecnologías y arquitecturas de software, los métodos y canales de comunicación entre otros. Todas estas consideraciones serán adoptadas para mejorar el diseño de los nuevos estudios de caso para ser aplicados a otros casos industriales que permitan solidificar más la propuesta.

Adicionalmente AGATA, debe ser extendido con prácticas de evaluación teniendo en cuenta otros valores y prácticas de los métodos ágiles con el fin de agregar otros aspectos de validación al producto en el contexto ágil y que su vez sea un instrumento valioso para evaluar la calidad de AGATA.

Es necesario establecer estrategias para crear una cultura educativa para la adherencia al método, sobre todo en equipos con personas experimentadas, debido a que por su experiencia estas personas, en algunos momentos intentan razonar sobre su experiencia negándose al aprendizaje de nuevas prácticas que pueden incorporar en su trabajo para lograr una mayor eficiencia.

Un Sprint 0 o Sprint de arranque debido al tamaño del equipo es necesario que se involucre

en el modelo, cuyo objetivo es atender a los preparativos previos antes de comenzar el desarrollo. Donde se realizarán tareas como: dejar listos los entornos de desarrollo, el encuentro inicial con el cliente por parte del equipo de arquitectura para estudiar los requisitos iniciales, conceptualizaciones arquitectónicas iniciales, la planificación inicial, y todo lo que se necesita para iniciar el proyecto.

Referencias bibliográficas

[1] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, “Agile software development methods review and analysis”, *VTT. Publications*, vol. 478, pp. 107, Sep 2002.

[2] R. Costello, and D. Liu, “Metrics for requirements engineering”, *ACM*, vol. 29, pp. 39-63, Abril 1995.

[3] K. Beck, M. Fowler, “Planning extreme programming”, *ACM,* pp. 160, Oct 2000.

[4] A. Qumer, B. Henderson, “An evaluation of the degree of agility in six agile methods and its applicability for method engineering”, *ScienceDirect* , vol. 50, 4, pp. 280-295, Mar 2008.

[5] K. Beck, C. Andres, “Extreme programming explained: embrace change”, *Addison-Wesley*, pp. 224, Enero 2012.

[6] E. Hadar, G. Silberman, “Agile architecture methodology: long term strategy interleaved with short term tactics”, *ACM,* pp. 641- 652, Oct 2008.

[7] H. Koontz, H. Weihrich, “Management: A global perspective”, *McGraw - Hill,* vol. 10, pp. 600, Enero 2005

[8] G. Earl Graves Ltd, “I.T. Jobs For Non-I.T. People”, *Black Enterprise*, vol. 31, 2 2000. Disponible en: www.blackenterprise.com.

[9] B. Calandra, “Toward a silver-tongued scientist”, *The scientist,* Sep 2002. Disponible en: www.the-scientist.com/?articles.view/articleNo/14243/title/Toward-a-Silver-Tongued-Scientist/.

[10] M.S. Peterson, “Personnel interviewers perception of the importance and adequacy of applicants communication skills”, *Communication Education*, vol 46, pp. 287-291, May 2009.

[11] P. Restrepo, “Crecimiento de industria del software y TI sería del 27%”, Agosto 2013. Disponible en :www.elheraldo.co.

[12] La industria de software ‘criolla’ dio un salto de calidad para conquistar el mercado, Revista Dinero, Octubre de 2015. Disponible en http://www.dinero.com/pais/articulo/progreso-industria-del-software-colombiana/215210.

[13] E. Yourdon, “Análisis estructurado moderno”, *Prentice hall hispanoamericana S.A,* pp. 735, Agosto 1993.

[14] G. Booch, “Object solutions:managing the object-oriented project (OBT)”. *Addison-Wesley*, pp. 336, Nov1996.

[15] M. Marchesi, G. Succi, D. Wells, and L. Williams, “Extreme programming perspectives”, *Addison Wesley*, pp. 624, Agosto 2002.

[16] A.Wellington, T. Briggs, and C. Girard, “Comparison of student experiences with plan-driven and agile methodologies”, *IEEE*, Oct 2005.

[17] J. Chanjan, M. Pister, and B. Rumpe, “Testing agile requeriments models”, *Jzus*, pp 73-79, May 2004. SIN REFERENCIAR ARRIBA

[18] G. Booch, I. Jacobson and J. Rumbaugh, “The unified modelling language for object-oriented development”, *Rational Software Corporation*, pp. 35, Sep 1996. SIN REFERENCIAR ARRIBA

[19] D. Reifer, F. Maurer, H. Erdogmus, “Scaling agile methods”. *IEEE,* vol 20, 4, pp. 12-14, Jul 2003.

[20] P. Pendharkar, and J. Rodger, “The relationship Between Software Development Team Size and Software Development Cost”, *ACM*, vol 52, 1, pp. 141-144, Enero 2009.

[21] M. Hoegl, “Smaller teams - better teamwork: How to keep project teams small”, Elsevier, pp. 209–214. Disponible en: http://dx.doi.org/10.1016/j.bushor.2004.10.013.

[22] M. Paasivaara, C. Lassenius, y J. Pyysiäinen, “Communication Patterns and Practices in Software Development Networks”, Helsinki University of Technology, 2003. pp. 783-798. Disponible en: http://www.soberit.hut.fi/veto/english/Julkaisut/ComPatterns.pdf.

[23] J. Stahl, S. Killich, y H. Luczak, “Co-ordination, Communication, and Co-operation in Locally Distributed Product Development”, Lawrence Erlbaum Associates, 2001, pp. 947-959.

[24 F. Rudolph, K. Verderber, y Otros. “¡Comunicate!”. Cengage Learning Editores, 2016, pp. 192. Disponible en: https://issuu.com/cengagelatam/docs/verderver.

[25] D. Damian y D. Zowghi, "Requirements Engineering Challenges in Multi-site Software Development Organizations," Requirements Engineering, 2003, vol. 8, pp. 149-160.

[26] P. Scerri, X. Yang, L. Elizabeth, y Otros, “Scaling Teamwork to Very Large Teams”, Carnegie Mellon University, 2004, vol 2. Disponible en: https://www.cs.cmu.edu/~pscerri/papers/aamas04.pdf.

[27] P. Kruchten, “Software architecture and agile software development: a clash of two cultures?”, IEEE, 2011. DOI: 10.1145/1810295.1810448.

[28] Z. Xiaobo, H. Ying, y Otros, "University Dormitory Management System Based on Agile Development Architecture", IEEE, 2011, pp. 1-4. DOI: 10.1109/ICMSS.2011.5998992.

[29] R. Axelrod, y W.Hamilton, “The evolution of cooperation”, Basic Books, 2006, pp.

1390-1396.

[30] A. Iqbal, M. Karnstedt, M. Hausenblas, “Analyzing social behavior of software developers across different communication channels”, 2013. Disponible en: https://aran.library.nuigalway.ie/bitstream/handle/10379/4476/iqbal_a_et_al_june_2013.pdf?sequence=1&isAllowed=y.

[31] V. Zaychik, and William C, “Capturing communication and context in the software project lifecycle”, *Springer*, 2003, vol 14, pp 75 – 88.

[32] I. McChesneya, and S.Gallagher, “Communication and coordination practices in software engineering projects”, *Information and Software Technology*, pp. 473 – 489, 2004.

[33] S. Marjaie, U. Rathod, “Communication in agile software projects: qualitative analysis using grounded theory in system dynamics”, 2011. Disponible en: http://www.systemdynamics.org/conferences/2011/proceed/papers/P1353.pdf.

[34] W Reinhardt, “Communication is the key support durable knowledge sharing in software engineering by microblogging”, 2009. Disponible en: http://www.systemdynamics.org/conferences/2011/proceed/papers/P1353.pdf.

[35] M. Paasivaara, C. Lassenius, J. Pyysiäinen, “Communication patterns and practices in software development networks”, 2003. Disponible en: http://www.soberit.hut.fi/veto/english/Julkaisut/ComPatterns.pdf.

[36] A. Sillitti and G. Succ, “Requirements engineering for agile methods”, *Springer*, pp. 478, 2005. Disponible en: www.springer.com/987-3-540-25043-2.

[37] P. Kruchten, “Agility and architecture – can they coexist?”, *IEEE*, vol 27, pp. 16-22, Abril 2010.

[38] K. Beck, and C. Andres, “Extreme programming explained: embrace change”, *Addison-Wesley*, pp. 26, 2005.

[39] C. Larman, and B. Vodde, “Scaling lean & agile development: thinking and organizational tools for large-scale scrum”, *Addison Wesley*, Dic 2008.

[40] R. Greening, “Enterprise scrum: scaling scrum to the executive level”, *IEEE*, pp 1-10, Enero 2010.

[41] L. Munoz, J. Alegría, " XA: An XP extension for supporting architecture practices", *7 CCC - IEEE,* Oct 2012.

[42] M. Bunge, “La ciencia y su método y su filosofía”, *LAETOLI*, pp. 144, 2013.

[43] M. Shaw, and P. Clements, “The golden age of software architecture”, *IEEE* , vol 23, 2, pp. 31-39, Marzo 2006.

[44] R. Maranzato, M. Neubert, P. Herculano. “Moving back to scrum and scaling to scrum

of scrums in less than one year". *ACM*, pp. 125 – 130, 2011.

[45] J. Alegría, "Método científico en ingeniería de software", Universidad del Cauca, 2004. Disponible en: www.dcc.uchile.cl/~jhurtado/mcis.pdf.

[46] M. Paasivaara, S. Durasiewicz, and C. Lassenius, "Distributed agile development: using scrum in a large project", *IEEE*, pp. 527-544, 2008

[47] P. Runeson, M. Höst, "Guidelines for conducting and reporting case study research in software engineering", *Springer* ,14, pp. 131–164, 2009.

[48] M. Light, "The first key to project success is collaborative requirements definition and manegement", Agosto 2008. Disponible en: www.gartner.com/doc/739919/key-project-success-collaborative-requirements.

[49] K. Méndescalo, E. Estevez, P. Fillotrani, "Un framework para evaluación de metodologías ágiles", 2008. Disponible en: sedici.unlp.edu.ar/bitstream/handle/10915/21086/Documento_completo.pdf?sequence=1.

[50] A. Begel, "Usage and perceptions of agile software development in an industrial context: an exploratory study", pp. 255 – 264, Sep 2007.

[51] J. Highsmith, "Agile software development ecosystems", *Addison-Wesley*, pp. 448, Abril 2007.

[52] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "The Agile Manifesto", 2001.

[53] T.Dibá, T.Dingsoyr, "Empirical studies of agile software development: A systematic review", *ScienceDirect*, vol 50, pp. 833-859, Agosto 2008

[54] J. Sutherland, D. Patel, C. Casanave, G. Hollowell and J. Miller, "business object design and implementation". *Springer,* 1995.

[55] K. Schwaber, J. Sutherland, "The definitive guide to scrum: the rules of the game", Jul 2013. Disponible en: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf.

[56] K. Schwaber, "Agile project management with scrum (Microsoft Professional)", *Microsoft Pres*, pp. 188, Feb 2004

[57] A. Pham, "Business driven it-wide agile (scrum) and kanban (lean) implementation: an action guide for business and it leaders". *CRC Press*, pp. 178, 2012.

[58] E. Woodward, S Surdek, M. Ganis, "A practical guide to distributed scrum", *IBM press*, pp. 146, 2010

[59] J. Palacio, K. Ruata, “Gestión de proyectos scrum manager”, *Scrum Manager*, Abril 2014. Disponible en: www.safecreative.org/work/1012268137397.

[60] R. Kazman, M. Klein, R. Nord. “Tailorable architecture methods”, *IEEE*, pp. 152-156, Marzo 2003.

[61] I. Jacobson, G. Booch, J. Rumbaugh, “The unified software development process”, *Addison Wesley*, pp. 512, 1999.

[62] P. Pendharkar, and J. Rodger, “The relationship between software development team size and software development cost”, *ACM,* vol 52, 1, pp. 141-144, Enero 2009.

[63] A. Qumer, B. Henderson, “An evaluation of the degree of agility in six agile methods and its applicability for method engineering”, *ScienceDirect,* vol 50, 4, pp. 280 – 295, 2007.

[64] A. Begel, N. Nagappan,” Usage and perceptions of agile software development in an industrial context: an exploratory study”, *IEEE*, pp. 255-264, Sep 2007.

[65] J. Pop, Al. Pop, “Stakeholder communication in software project management. modelling of communication features”, 2007. Disponible en: www.wseas.us/e-library/conferences/2013/brasov/acmos/acmos-57.pdf

[66] M. Peterson. “Personnel interviewers perception of the importance and adequacy of applicants communication skills education”, *Communication Education*, 46, pp. 287 – 91, 1997

[67] “Meriam webster dictionary”, *An Encyclopedia Britannica Company*. Disponible en: www.merriam-webster.com/.

[68] P. Flensburg, “An enhanced communication model”, *The international journal of digital accounting research*, vol 9, 2009. Disponible en : www.uhu.es/ijdar/10.4192/1577-8517-v9_2.pdf.

[69] S. Estrada, and S. Restrepo De Ocampo, “Model of communication for changing organizations”, *Universidad Tecnologica de Pereira - Scientia et technica*, 44, 2010.

[70] S. González and J. M. Serna, “Tema 1: Lenguaje y Comunicación”, Agosto 2015. Disponible en http://www.auladeletras.net/material/comunica.pdf

[71] L. Marín, “La nueva comunicación”, *Trotta*, pp. 448, 2009.

[72] G. Melnik, and F. Maurer, “Direct verbal communication as a catalyst of agile knowledge sharing”, *IEEE*, Jun 2004.

[73] C. Larkman, “UML y Patrones”, *Prentice Hall*, Enero 2008

[74] “Agile software development: A gentle introduction”, Dic 2014. Disponible en www.agile-process.org/.

[75] J. Highsmith, K. Orr, “Adaptive Software Development: A Collaborative Approach to Managing Complex Systems”, *Dorset House*, pp. 624, 2006

[76] D. Hix, H. Hartson, “Developing User Interfaces: Ensuring Usability Through Product & Process”, *Wiley*, pp. 416, 2003

[77] P. Letelier, M. Penadés, “Metodologías ágiles para el desarrollo de software: Extreme Programming (XP)”, 2006. Disponible en: www.cyta.com.ar/ta0502/v5n2a1.htm

[78] L. Constantine, and A Lockwood, “Software for use: a practical guide to the models and methods of usage centered design”, *ACM*, 2004

[79] A. Kornstadt, J. Sauer, “Tackling offshore communication challenges with agile architecture-centric development”, *IEEE,* 2007

[80] Rick Kazman, Mark Klein, Robert L. Nord. Tailorable Architecture Methods. Software Engineering Workshop Proceedings. 28th Annual NASA Goddard CMU Institute. 2003

[81] P. Kruchten, “Architectural blueprints the “4+1” view model of software architecture”, *IEEE*, pp. 42-50, 1995.

[82] J. Rolf Njor, M. Thomas, S. Peter, and T. Gitte, “Architecture and design in extreme programming; introducing developer stories”, *ACM*, pp 133-142, 2007.

[83] D. Dalcher, O. Benediktsson, and H. Thorbergsson, “Development life cycle management: a multiproject experiment”, *IEEE,* pp. 289 – 296, 2005.

[84] A. Wellington, T. Briggs, and C. Girard, “Comparison of student experiences with plan-driven and agile methodologies”, *IEEE*, Abril 2007.

[85] B. Bohem, “Get ready for agile methods, with care”, 2002. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.224.6128&rep=rep1&type=pdf.

[86] D. Shahane, “Factors Influencing the Agile Methods in Practice - Literature Survey & Review”. *IEEE*, pp. 556 – 560. 2014

[87] L. Layman, L. Williams, and L. Cunningham, “Exploring extreme programming in context: an industrial case study”, *IEEE*, pp. 32–41. 2004.

[88] D. Wells, “Extreme Programming a Gentle Introduction,” 2012: Disponible en: http://www.extremeprogramming.org.

[89] S. Ambler ,”Quality in an agile world”, 2005. Disponible en: http://www.ambysoft.com/downloads/agileQuality.pdf.

[90] G. Melnik, and F. Maurer, “Direct verbal communication as a catalyst of agile

knowledge sharing", *IEEE,* Nov 2004.

[91] P. Abrahamsson, J. Warsta, M. Siponen, and J. Ronkainen, "New directions on agile methods: a comparative analysis", *IEEE*, 2003.

[92] A. Cockburn, "Agile software development", pp. 467, 2001. Disponible en: http://alistair.cockburn.us/get/1885

[93] P. Schulz, P. Cobley, "The handbook of communication science", pp 389 – 419, 1987.

[94] J. Spillan, M. Mino, S. Rowles, "Sharing organizational messages trough effective lateral communication", *Communication Quarterly*, pp. 96 – 104, 2002

[95] A. Kornstädt y J. Sauer, "Mastering Dual-Shore Development – The Tools and Materials Approach Adapted to Agile Offshoring", *Springer,* pp. 83–95, 2007. Disponible en: https://www.nordakademie.de/fileadmin/downloads/Forschung/Sauer-Mastering_Dual-Shore_Development_Kornstaedt_Sauer.pdf.

[96] W Reinhardt, G Beham, M Sillaots, "Microblogging in technology enhanced learning: A use-case inspection of ppe summer school", *Proceedings of the 2nd SIRTEL,* 2008.

[97] A. Bartoli, "Comunicación y organización: la organización comunicante y la comunicación organizada", *Paidos Ibérica*, pp. 222, 2009.

[98] Portal Integrators, "Scrum of Scrums", *Scrum Inc*, 2017. Disponible en: https://www.scruminc.com/scrum-of-scrums/.

[99] J. R. Guerrero, "Scrum A Fondo: Scrum De Scrums, Agile En Grandes Proyectos", WordPress.com, 2014. Disponible en: https://jesusramirezguerrero.com/2014/02/12/scrum-a-fondo-scrum-de-scrum-agile-en-grandes-proyectos/.

[100] K. schwaber, "La Guía Nexus - La Guía Definitiva a Nexus: El exoesqueleto de desarrollo a escala con Scrum", 2015. Disponible en: https://www.scrum.org/resources/nexus-guide.

[101] J. Palacios, "Nexus: Que Es Y Cómo Usarlo Para Escalar Agile", 2015. Disponilble en: https://jeronimopalacios.com/2015/08/nexus-que-es-y-como-usarlo-para-escalar-agile/.

[102] J. Garzas, "Otra estrategia para escalar Scrum: Disciplined Agile Delivery (DAD)", 2015. Disponible en: http://www.javiergarzas.com/2015/03/disciplined-agile-delivery-dad.html.

[103] A. Lozano, "Disciplined Agile Delivery (DAD). Introducción", 2015. Disponible en: http://www.angellozano.com/disciplined-agile-delivery-dad-la-introduccion/.

[104] J. Garzas, "Scaled Agile Framework (SAFe), una metodología ágil para grandes empresas", 2013. Disponible en: http://www.javiergarzas.com/2013/09/scaled-agil.html.

[105] I. Mitchell, “Method Wars: Scrum vs SAFe”, 2013. Disponible en: https://dzone.com/articles/method-wars-scrum-vs-safe.

[106] A. Elssamadisy, "Has SAFe Cracked the Large Agile Adoption Nut?", 2013. Disponible en: https://www.infoq.com/news/2013/08/safe.

[107] L. Robert, and E. James, “Software architecture centric methods and agile development”, *IEEE*, 2, pp. 47 – 53. 2006.

[108] L. Munoz, J. Alegría, " Agile Communication (AgileCom): Un Mecanismo de Comunicación para equipos de software ágiles cuando el equipo crece”.

[109] R. Jeffries, A. Anderson, C. Hendrickson, “Extreme programming installed”, *Addison-Wesley*, pp. 328, 2006.

[110] Schwaber Ken. Agile Project Management with Scrum. Microsoft press. USA. pp 192. 2004.

[111] L. Craig, “Agile and iterative development: a manager's guide”, *Addison-Wesley*, pp. 342, 2004.

[112] Agile Software Development: A gentle introduction. Disponible en www.extremeprogramming.org. Revisado en Julio de 2014.

[113] Unit Tests. Disponible en www.extremeprogramming.org/rules/unittests.html. Revisado en Agosto de 2014

[114] Cohn Mike. User Stories Applied: For Agile Software Development. Editorial: Addison-Wesley Professional. Pp 280. 2004.

[115] González-Serna S., J. M. Tema 1: Lenguaje y Comunicación. Disponible en http://www.auladeletras.net/material/comunica.PDF. Revisado agosto de 2015.

[116] Robbins SP, Mary C. Management. Ed. Prentice Hall. Pp 258. New York. 2003.

[117] Estrada Mejía Sandra, Restrepo De Ocampo Luz Stella. Model of communication for changing organizations. Scientia et Technica Año XVI, No 44, 2010.

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

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

[120] Layman L. Williams L., Cunningham L. “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

[121] Ishikawa kaouru. What is total quality control, the japanese way. Grupo editorial Norma. Colombia. 1986.

[122] J. Escobar and F. I. Bonilla-Jimenez, Grupos focales : una guía conceptual y metodológica. vol. 9, no. 1, pp. 51– 67, 2009.

[123] Plata, L. Comparación del Proceso de Elicitación de Requerimientos en el desarrollo de Software a Medida y Empaquetado. 2009.

[124] Sould, E. Simple Linear Regression — Formulas & Theory, 1–6. 2008

[125] Bellomo, S.; Nord, R.L.; Ozkaya, I., "A study of enabling factors for rapid fielding combined practices to balance speed and stability," Software Engineering (ICSE), 2013 35th International Conference on , vol., no., pp.982,991, 18-26 May 2013.

[126] Mary Shaw and David Garlan. Software Architecture: Perspectives on an Emerging Discipline. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 1996.

[127] P. Clemens, L. Bass. 2010. Using Business Goals to Inform a Software Architecture. In Proceedings of the 2010 18th IEEE International Requeriments Engineering Conference (RE ‘10). IEEE Computer Society, Washington, DC, USA, 69-78.

[128] Chen Qing-Lan, Wei Chiou-Shuei, Huang Mei-Yao, Chiu-ChiWei. A model for project communication medium evaluation and selection. Concurrent Engineering: Research and Applications 21(4) 237–251. 2013

[129] Alice m. Johnson, Albert l. Lederer. The Effect of Communication Frequency and Channel Richness on the Convergence Between Chief Executive and Chief Information Officers.. pages 227-252. 08 Diciembre 2014.

[130] Sautu, R. Práctica de la Investigación Cuantitativa y Cualitativa. Ediciones Lumiere. STRAUSS. 2007

[131] YIN, R. K. Case Study Research: Design and Methods. Sage Publications. 3rd Edition. 2003.

[132] Mary Beth Chrissis, Michael D. Konrad, Sandra Shrum CMMI for Development: Guidelines for Process Integration and Product Improvement, Third Edition, Addison-Wesley Professional. 2011.

[133] ISO Family. International Organization for Standardization. Revisado mayo de 2016. Disponible en www.iso.org/standard/

[134] Takeuchi Hirotaka, Nonaka Ikujiro. The New New Product Developement Game. Hitotsubashi University, Harvard Business Review, Enero-Febrero de 1986.