UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS

MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

REDUCCIÓN DE LA VAPORIZACIÓN DEL CONOCIMIENTO ARQUITECTÓNICO EN EL DESARROLLO ÁGIL Y GLOBAL DE SOFTWARE

TESIS
que presenta para obtener el grado de DOCTOR EN CIENCIAS

GILBERTO BORREGO SOTO

DIRECTOR DE TESIS:
DR. ALBERTO LEOPOLDO MORÁN Y SOLARES

CO-DIRECTOR DE TESIS:
DR. RAMÓN RENÉ PALACIO CINCO

ENSENADA, B. C. **AGOSTO DE 2018**

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS

Maestría y Doctorado en Ciencias e Ingeniería

Reducción de la vaporización del conocimiento arquitectónico en el desarrollo ágil y global de software

Tesis que para obtener el grado de Doctor en Ciencias presenta

Gilberto Borrego Soto

Aprobada por el siguiente comité

Dr. Alberto Leopoldo Morán y Solares
Director del Comité

Dr. Ramón René Palacio Cinco
Codirector del Comité

Dra. Marcela Deyanira Rodríguez Urrea
Miembro del Comité

Dra. Eloísa del Carmen García Canseco
Miembro del Comité

Dr. Oscar Mario Rodríguez Elías
Miembro del Comité

Ensenada, Baja California, 28 de Junio de 2018

RESUMEN de la Tesis de **Gilberto Borrego Soto**, presentada como requisito parcial para la obtención del grado de **DOCTOR EN CIENCIAS**. Ensenada, Baja California, México. Junio 2018.

REDUCCIÓN DE LA VAPORIZACIÓN DEL CONOCIMIENTO ARQUITECTÓNICO EN EL DESARROLLO ÁGIL Y GLOBAL DE SOFTWARE

Resumen aprobado por:

Dr. Alberto Leopoldo Morán y Solares
Director de tesis

Dr. Ramón René Palacio Cinco
Co-Director de tesis

Cada vez es más común que las empresas practiquen el desarrollo global de software (GSD por sus siglas en inglés), en donde las distancias físicas, temporales, lingüísticas y culturales se solventan intercambiando documentos basados en estándares, entre las partes locales y foráneas. También es común que las empresas practicantes del GSD adopten metodologías ágiles debido a la reducción de tiempos y costos que se pueden obtener al implementar el Desarrollo Ágil y Global de Software (AGSD por sus siglas en inglés). Sin embargo, estas metodologías están diseñadas para equipos co-localizados, que basan la comunicación en interacciones personales y en documentos informales. Lo anterior provoca que se maneje poca y pobre documentación, particularmente referente al conocimiento arquitectónico (AK por sus siglas en inglés). Esto a su vez conduce a la vaporización del AK (pérdida del conocimiento al paso del tiempo), provocando defectos en la evolución y el mantenimiento del software, y erosión de relaciones del equipo por constantes preguntas.

La literatura reporta que los desarrolladores practicantes del AGSD prefieren comunicarse por medios electrónicos textuales no estructurados (UTEM por sus siglas en inglés), como: correo electrónico y mensajería instantánea, ya que es una manera de disminuir la brecha del lenguaje. Los UTEM dejan rastros de interacciones (bitácoras), donde se encuentra AK valioso para los desarrolladores, el cual solventa un poco la deuda de documentación. Sin embargo, estas bitácoras se encuentran sin estructura y dispersas, lo que dificulta la recuperación de AK.

La contribución principal de esta tesis es el concepto de condensación del AK, como una manera de reducir la vaporización del AK en equipos de AGSD, aprovechando el conocimiento residente en las bitácoras de los UTEM. Este concepto nació a partir de estudios con desarrolladores de AGSD y a partir de revisiones de literatura. Para probar la factibilidad de este concepto se desarrolló de forma iterativa un prototipo tecnológico, y el resultado de cada iteración fue evaluado de forma controlada por desarrolladores de AGSD. Los resultados de las evaluaciones indicaron que el concepto de condensación de AK puede ser factible a implementarse en ambientes de AGSD, lo cual da pie a mejorar el prototipo y ejecutar evaluaciones a largo plazo en empresas de AGSD.

Palabras clave: desarrollo ágil y global de software, manejo del conocimiento arquitectónico, vaporización del conocimiento

ABSTRACT of the Thesis of **Gilberto Borrego Soto**, presented as a partial requirement for obtaining the degree of **DOCTOR IN SCIENCE**. Ensenada, Baja California, Mexico. June 2018.

REDUCTION OF ARCHITECTURAL KNOWLEDGE VAPORIZATION IN AGILE AND GLOBAL SOFTWARE DEVELOPMENT

Approved by:

Dr. Alberto Leopoldo Morán y Solares
Thesis advisor

Dr. Ramón René Palacio Cinco
Thesis co-advisor

Nowadays, software development companies commonly adopt a global manner to work, i.e., they practice global software development (GSD). In GSD physical, temporal, linguistic and cultural distances are resolved by sharing standards-based documents between local and remote parties. GSD practitioners also commonly adopt agile methodologies due to the reduction in time and costs by implementing Agile and Global Software Development (AGSD). However, these methodologies are designed for co-located teams, which base their communication on personal interactions and informal documents. These characteristics cause few and poor documentation, particularly concerning architectural knowledge (AK). Consecuently, these situations leads to AK vaporization (loss of knowledge over time), causing defects in software evolution and maintenance, and erosion of team relationships by constant questioning.

The literature reports that AGSD practitioners prefer to communicate through unstructured textual electronic media (UTEM), such as email and instant messaging, as it is a way to bridge the language gap. UTEM leave traces of interactions (logs), where there exists AK valuable to developers, which solves a bit of the documentation debt. However, these logs are unstructured and scattered, making it difficult for AK to recover.

The main contribution of this thesis is the concept of AK condensation as a way to reduce AK vaporization in AGSD teams, taking advantage of the knowledge located in the UTEM logs. This concept was conceived from studies with AGSD developers and literature reviews. To test the feasibility of this concept, a technological prototype was iteratively developed, and AGSD developers evaluated the result of each iteration through controlled experiments. The obtained results indicated that the concept of AK condensation could be feasible to implement in AGSD environments, which led to improve the prototype and to conduct long-term evaluations in AGSD companies in the future.

Keywords: agile and global software development, architectural knowledge management, knowledge vaporization

Dedicatoria

A mi esposa Raquel, mis hijos: Rossana, Celeste y Gael, y a mis padres.

No lo hubiera logrado sin su apoyo.

Agradecimientos

Quiero agradecer especialmente a mi esposa, hijos, padres y todos mis familiares que me apoyaron para concretar este proyecto desde el inicio hasta el final.

A mis dos directores de tesis: Dr. Alberto Leopoldo Morán y Solares y el Dr. Ramón René Palacio Cinco, por su invaluable apoyo tanto en la parte académica como en la parte personal, para facilitar el camino durante mis estudios.

A los miembros de mi comité de tesis Dra. Marcela Deyanira Rodríguez Urrea, Dra. Eloísa del Carmen García Canseco, y al Dr. Oscar Mario Rodríguez Elías, cuya retroalimentación contribuyó al desarrollo de esta tesis y a los resultados asociados.

Al grupo ALARCOS de la Escuela Superior de Informática de la Universidad de Castilla-La Mancha, en Ciudad Real, España, por darme la oportunidad de realizar una estancia académica durante mis estudios. En especial a la Dra. Aurora Vizcaíno Barceló quien contribuyó de gran forma en la recta final de la investigación realizada en esta tesis, y en que la estancia mía y de mi familia fuera más agradable.

Al Dr. Miguel Bernardo Romero Téllez, al Dr. Manuel Escobar Armenta y a Luis Alonso Osorio, por el apoyo en todos los preparativos para realizar mi estancia académica en la Universidad Castilla-La Mancha.

A las empresas que me dieron la oportunidad realizar las distintas evaluaciones requeridas para esta tesis: EMCOR, Softtek, Nearsoft, Tiempo Development, Sahuaro Labs, Repobox, Trapichar, Grupo SMI, Ubilogix y TUFESA.

Al Ing. José Roberto Corona Muñoz de la empresa EMCOR y al M.C. Eduardo Gasca Figueroa de la empresa Gasca Tecnologías por sus apoyos económicos para asistir a diversos congresos internacionales.

Al Consejo Nacional de Ciencia y Tecnología por darme los recursos para realizar mis estudios de doctorado

Índice

Índice de figuras

Índice de tablas

Listado de acrónimos

A

ADL: Architecture Definition Language

AGSD: Agile and Global Software Development

AK: Architectural Knowledge

AKM: Architectural Knowledge Management

ASD: Agile Software Development

G

GSD: Global Software Development

GSE: Global Software Engineering

I

IDE: Integrated Development Environment

R

RUP: Rational Unified Process

S

SUS: System Usability Scale

T

TAM: Technology Acceptance Model

U

UTEM: Unstructured Textual Electronic Media

UML: Unified Modelig Language

X

XP: eXtreme Programming

Capítulo 1.
Introducción

La ingeniería de software desde su aparición en 1968 ha ido evolucionando como respuesta a los cambios del entorno en donde se practica (Sommerville, 2010). En sus inicios, la introducción de nuevas computadoras con más capacidad y velocidad de procesamiento provocó que programas informáticos que antes eran irrealizables, ahora fueran propuestas factibles. Esto a su vez provocó que las técnicas y procesos para desarrollar software evolucionaran para poder producir sistemas más grandes y complejos, en tiempo razonable y con alta calidad. De la misma manera nuevas tecnologías en comunicación, nuevos dispositivos de cómputo y mejores interfaces gráficas de usuario (por mencionar algunas causas) han demandado de nuevas técnicas y métodos en la ingeniería de software.

Los factores tecnológicos no son los únicos que han impulsado la evolución de la ingeniería de software, los factores socioeconómicos han tenido también un papel importante. Por ejemplo, la competitividad de los mercados en donde se exigen productos y servicios de calidad cada vez en un menor tiempo, o bien, el fenómeno de la globalización, también han provocado que la manera de desarrollar software cambie y se adapte a dichas condiciones.

Debido a la globalización, cada vez se amplía el campo competitivo de las empresas más allá de las fronteras del país en donde se localiza. Este fenómeno ha alcanzado al ámbito del desarrollo de software, de tal manera que hoy en día una empresa de este giro puede ofrecer sus servicios a clientes que se encuentren en otros países, o bien, colaborar con empresas extranjeras de desarrollo de software para atender en conjunto distintos proyectos. A este enfoque se le conoce generalmente como Desarrollo Global de Software (GSD, por sus siglas en inglés) o como Ingeniería Global de Software (GSE, por siglas en inglés) (Herbsleb and Moitra, 2001). Una de las características principales del GSD es que entran en juego cuatro distancias: la física, la temporal, la lingüística y la cultural, lo cual ha provocado la evolución de la ingeniería de software introduciendo técnicas, prácticas, herramientas, etc. para solventar los retos de comunicación, control y coordinación (Vizcaíno et al., 2014) que las cuatro distancias introducen.

Otro de los aspectos que ha provocado la evolución de la ingeniería de software es el dinamismo de los mercados actuales. Al respecto, en la década de los noventas, las necesidades de mercado crecían y cambiaban tan rápido que cuando se terminaba el desarrollo del software que cubría tales necesidades, era muy probable que las condiciones y necesidades ya hubieran cambiado (Shore and Warden, 2007). En aquel entonces el método de desarrollo de "cascada" era el estándar de facto en la ingeniería de software. En este método los requerimientos de software deben estar completos antes de pasar al diseño funcional, y este último completo antes del diseño detallado, y así sucesivamente a hasta llegar a la implantación del software. En respuesta, surgió una propuesta de desarrollo en la cual la retroalimentación rápida y la voluntad de cambio son las bases para conducirse, de tal manera que, si el equipo de software no está seguro de entender lo que el usuario necesita, proporciona una primera aproximación (entrega de software) y luego escucha los comentarios del cliente para refinarla de manera iterativa hasta terminar con el producto deseado. A esta propuesta se le llamó el desarrollo ágil de software (Augustine, 2005).

La misma dinámica de mercados y la globalización han llevado a muchas compañías que aplican el GSD a explorar el desarrollo ágil (Ramesh et al., 2006), es decir, practican el desarrollo ágil y global de software (AGSD por sus siglas en inglés). En principio pareciera que existe una brecha entre el GSD y el desarrollo ágil que los hace incompatibles, ya que el desarrollo ágil está diseñado para equipos co-localizados, mientras en el GSD los equipos están distribuidos. Además, en GSD el desarrollo se basa en grandes planes, en una estricta división de tareas y roles, y una comunicación entre equipos basada en documentación exhaustiva (Šmite et al., 2010a) para disminuir el efecto de las cuatro distancias inherentes en el GSD. Este tipo de comunicación es contrario a los principios ágiles (Beck et al., 2001), que marcan que las interacciones de las personas y el software funcionando son preferidas sobre el seguimiento de procesos, herramientas y una documentación completa, e interacciones cara a cara como el mejor método para transmitir información dentro de un equipo de desarrollo.

Estas diferencias son el origen de muchos de los retos del AGSD, siendo uno de estos el manejo del conocimiento (Dorairaj et al., 2012; Jiménez et al., 2009; Yanzer Cabral et al., 2014a), ya que la transferencia del mismo se ve limitada en este tipo de ambientes (Kamaruddin et al., 2012), por la distribución geográfica y temporal, por la reducción del

número de documentos y por la poca formalidad de los mismos (son ad hoc) (Turk et al., 2002), generalmente sin apegarse a algún estándar como UML[1]. Un tipo de conocimiento relevante en cualquier paradigma de desarrollo de software es el relacionado con la arquitectura de software o conocimiento arquitectónico (AK por sus siglas en inglés), ya que es la parte sustantiva del trabajo para construir un producto de software. De hecho, los métodos ágiles más populares: Scrum y XP (Ghani and Bello, 2015; Hamed and Abushama, 2013) (muy usados también en AGSD[2]), tienen maneras muy laxas de especificar el AK, lo que produce documentos con notación informal (Hoda et al., 2010). Incluso en otros casos la documentación del AK no existe, ya que las presiones de tiempo inherentes al desarrollo ágil y la preferencia del código trabajando sobre la documentación, causa que los desarrolladores dejen de lado esta última (Sneed, 2014), afectando principalmente a las fases de codificación (Noordeloos and Manteli, 2012). A esto último se le llama la deuda de documentación, que eventualmente provoca la vaporización del conocimiento, es decir, la pérdida o tergiversación del mismo al paso del tiempo (Bosch, 2004). Por lo tanto, el manejo del AK se ve afectado por las condiciones del AGSD que provocan que exista más conocimiento en forma tácita que en forma explícita (Clerc et al., 2011; Yanzer Cabral et al., 2014a), lo que finalmente afecta severamente al manejo del conocimiento dado que en esta disciplina uno de los principales objetivos es convertir el conocimiento tácito a explícito (Duhon, 1998).

Esta tesis se desarrolla dentro del ámbito del manejo del AK en equipos de AGSD, y particularmente se trata el problema de la vaporización del AK. La solución propuesta a este problema se aborda a través del concepto de condensación del AK, también producto de esta tesis. La propuesta de este concepto surgió a través de una amplia revisión de literatura y un estudio empírico realizado con empresas de desarrollo software mexicanas. Además, a través de un proceso iterativo apoyado por desarrolladores de software, se diseñó e implementó una solución tecnológica que instancia el concepto de condensación de AK, la cual fue evaluada por estudiantes y desarrolladores de software para determinar la factibilidad del concepto propuesto para reducir la vaporización del AK en el AGSD.

En este capítulo introductorio se describen la motivación y justificación, y el planteamiento del problema, para después definir los objetivos a alcanzar en esta tesis.

[1] https://www.omg.org/spec/UML/2.0/

[2] https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2

Posteriormente se describe la metodología de trabajo utilizada para el cumplimiento de los objetivos definidos, se presentan las principales contribuciones de esta tesis, y se finaliza con la descripción del contenido del resto del documento.

1.5. Motivación y justificación

Cada vez es más común que las compañías de GSD adopten el desarrollo ágil, sin embargo, la deuda de documentación del AK y la eventual vaporización del mismo provocan en las empresas practicantes del AGSD problemas como: (1) pérdida de tiempo respondiendo las mismas preguntas y tratando de encontrar soluciones a problemas que habían sido resueltos anteriormente, (2) pérdida de conocimiento cuando un miembro del proyecto se retira, (3) defectos en la evolución y el mantenimiento del software, (4) falta de visibilidad para el seguimiento del proyecto y las soluciones técnicas, (5) y requisitos mal entendidos (Holz and Maurer, 2003; Uikey et al., 2011). Estos problemas podrían parecer triviales, pero se ha demostrado que un buen manejo del conocimiento es un factor importante para el éxito de cualquier proyecto de desarrollo de software (Levy and Hazzan, 2009). Además, los problemas citados pueden impactar negativamente a la efectividad de equipo (Hackman, 1987), particularmente en los aspectos de resultados y de procesos sociales.

En cuanto a las afectaciones a los aspectos de resultados, cuando los desarrolladores no tienen claro el AK de un proyecto, se requiere un mayor esfuerzo para desarrollarlo y mantenerlo, ya que se ignoran las decisiones de diseño que llevaron a la estructura actual del software (Galster and Babar, 2014). Incluso, entre el 40% y el 60% del tiempo de mantenimiento se dedica a entender un software cuando la documentación del AK no es adecuada o no existe (Pfleeger, 2001), lo cual puede retrasar los proyectos y por ende afectar los resultados.

Los efectos de la vaporización de AK no son exclusivos al mantenimiento del software. Por ejemplo, cuando una persona se retira de un proyecto en curso, y no hay documentación, el AK se evapora por la ausencia de un miembro del equipo. Entonces la única alternativa para recuperar el AK es analizar el código fuente, sin embargo, esta tarea consume mucho tiempo, y el AK obtenido de esta manera es generalmente desestructurado, incompleto e inconsistente (Selic, 2009), lo que no garantiza que el software evolucione como fue planeado en el momento del diseño, lo cual invariablemente afecta al resultado de un proyecto de software en cuanto al tiempo de término y a la calidad del producto.

Se podría argumentar que el AK no se vaporiza del todo al retirarse un miembro del equipo, gracias a la propiedad compartida del código que se practica en el desarrollo ágil, es decir, que todos en cierta manera tienen conocimiento de las partes del código fuente (Shore and Warden, 2007). Sin embargo, las cuatro distancias inherentes al GSD causan menos eficiencia al compartir AK, ya que se pierden oportunidades para la interacción casual (Isaacs et al., 1997), así como y la conciencia informal (Gutwin et al., 1996). Por lo tanto la propiedad compartida de código no tiene el mismo efecto en una situación con equipos distribuidos geográficamente.

Ahora respecto al impacto en la efectividad de equipo en sus aspectos sociales, para los desarrolladores más experimentados de un equipo de software, el estar respondiendo las mismas preguntas, y el estar tratando de encontrar soluciones a problemas que han sido resueltos anteriormente, puede llegar a ser muy molesto. Esto podría conducir a una erosión en las relaciones sociales entre los miembros del equipo, y afectar el flujo de conocimiento en equipos ágiles (Ryan and O'Connor, 2013). Además, la erosión de las relaciones interpersonales es crítica para el AGSD dado que éstas son fundamentales para el buen funcionamiento de un equipo ágil (Beck et al., 2001), así como la confianza entre los miembros del equipo (Sriram and Mathew, 2012; Xu et al., 2006), la cual también se puede ver afectada al erosionarse las relaciones interpersonales.

Por todo lo anterior, al mejorar el manejo del conocimiento arquitectónico en AGSD, también se mejoraría la efectividad de equipo, en sus aspectos de resultados, y en sus aspectos sociales.

1.6. Planteamiento del problema

El predominio del AK en su estado tácito sobre el AK en su estado explícito en las empresas practicantes del AGSD convierte al manejo del AK en un reto. Además, este predominio, también visto como deuda de documentación, conduce eventualmente a la vaporización del AK, lo que conduce a problemas en la evolución y mantenimiento del software, e incluso conduce a problemas referentes a los aspectos sociales del equipo.

Existen muy pocos trabajos en la literatura que abordan el manejo del AK en empresas de AGSD (M. a. Babar, 2009; Clerc et al., 2011; Eloranta and Koskimies, 2013; Hadar et al., 2013). Se han tratado de usar prácticas de manejo de manejo de conocimiento arquitectónico para GSD en proyectos de AGSD, encontrando que las técnicas de personalización

predominan sobre la centralización (Clerc et al., 2011), es decir, termina predominando el conocimiento tácito sobre el explícito. También se ha intentado el uso de repositorios para compartir AK, pero con problemas de búsqueda por las maneras informales de codificar el conocimiento, y con problemas de adopción por el tiempo que implica codificar el AK (M. a. Babar, 2009). También se está trabajando en el diseño de métodos ligeros de codificación de AK para AGSD (Eloranta and Koskimies, 2013; Hadar et al., 2013), sin embargo no se han reportado pruebas de campo para estos métodos.

Con lo anterior se concluye que la literatura muestra que hay una falta de soluciones para reducir la deuda de documentación y la vaporización del AK en entornos ágiles y distribuidos, que faciliten la evolución y el mantenimiento del software, y que impacten lo menos posible los aspectos benéficos del desarrollo ágil y del GSD.

1.6.1 Escenario tipo del problema

Con el fin de ser más claro con el planteamiento del problema a abordar en esta tesis, se presenta el escenario esquematizado en la Figura 1.1. Se parte de una situación en la que una compañía de desarrollo de software tiene una sucursal en México y su matriz en Estados Unidos. A continuación, se describe paso a paso el escenario:

1. El desarrollador de Estados Unidos (EE.UU.) inicia un proyecto de software y, basado en los requerimientos, diseña la arquitectura del mismo, la cual plasma en un cuaderno, pizarrón o algún programa de cómputo usando notación informal, con el fin de diseñarla

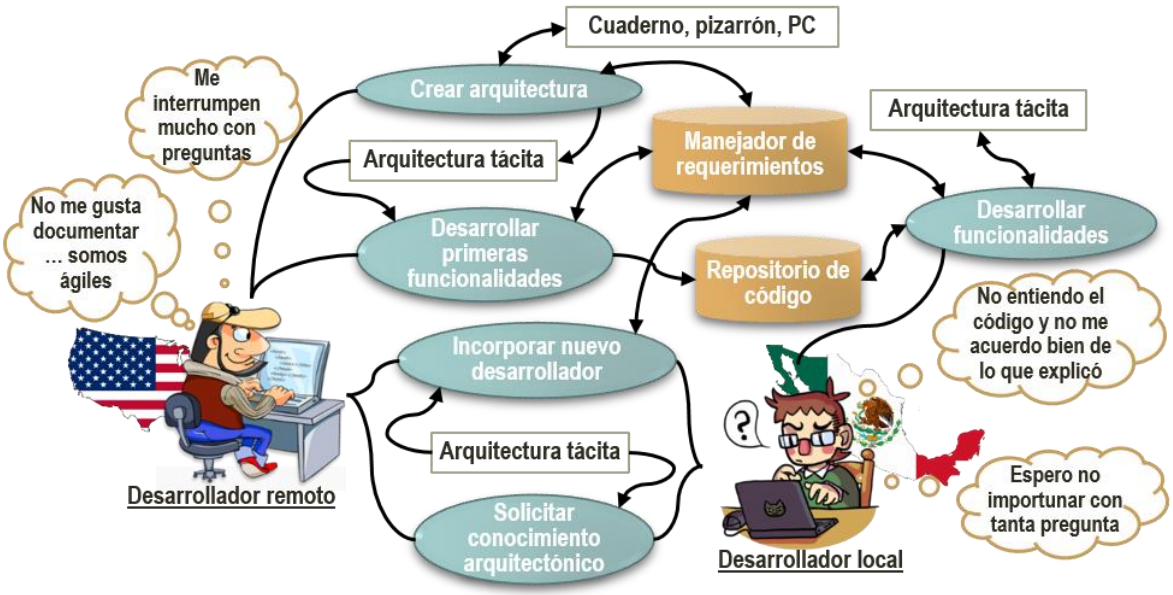


Figura 1.1. Gráfica rica de escenario tipo del problema.

lo más rápido posible. Por lo tanto, parte del conocimiento de dicha arquitectura queda de manera tácita.

2. El desarrollador de EE.UU, basado en la arquitectura diseñada, codifica los primeros conjuntos de funcionalidad, dejando el código resultante en un repositorio.
3. Posteriormente, se decide incorporar a un nuevo desarrollador (de la sucursal de México) para acelerar el proyecto. Para esto, el desarrollador de EE.UU. le explica al desarrollador de México la generalidad del proyecto, la arquitectura de software y los requerimientos por hacer. Esta explicación se puede hacer por llamada, videoconferencia o por mensajero instantáneo. En caso de que exista alguna gráfica para explicar la arquitectura, esta es compartida con su contraparte.
4. El desarrollador de México inicia con la codificación de la funcionalidad que le corresponde, pero le surgen dudas sobre aspectos de la arquitectura, ya sea porque no las recuerda bien (se empieza a vaporizar el AK), o porque su contraparte no tuvo el cuidado de explicar algunos detalles. Entonces, el desarrollador trata de solventar dichas dudas a través de la revisión del código, sin tener mucho éxito. Es importante que las dudas referentes a la arquitectura se resuelvan ya que el código que se introduzca debe respetar los lineamientos de la misma, para garantizar que el software evolucione como se tenía previsto al diseñarse.
5. Al no tener éxito tratando de resolver sus dudas a través de la revisión de código, el desarrollador de México decide consultar a su contraparte a través de una llamada, videoconferencia o mensajero instantáneo. Sin embargo, depende de la disponibilidad de su compañero, por lo que siente cierta preocupación de no ser inoportuno. También depende que a su contraparte recuerde bien el AK que se le solicita, es decir, que no esté vaporizado el AK. Por otro lado su contraparte de EE.UU. puede llegar a molestarse por ser interrumpido durante su trabajo.

Dado el escenario descrito, una solución sería que se incorpore un proceso para documentar de manera formal los aspectos arquitectónicos, pero se requeriría tiempo y esfuerzo extra, lo cual diezmaría los tiempos cortos de entrega de resultados palpables, lo que es una de las ventajas del enfoque ágil sobre los enfoques tradicionales. Por lo anterior, es evidente que falta balancear la informalidad provocada por el desarrollo ágil al registrar

aspectos arquitectónicos, y la formalidad que exige el GSD para solventar sus cuatro distancias mencionadas.

1.7. Objetivos de la investigación

En esta sección se presentan los objetivos planteados para esta tesis, divididos en objetivo general y objetivos específicos.

1.7.1 Objetivo general

Caracterizar y modelar el uso y diseminación de conocimiento arquitectónico en un ambiente de AGSD, con el fin de implementar y evaluar una solución de software para reducir la vaporización de este tipo de conocimiento, sin perder las características ágiles y del GSD.

1.7.2 Objetivos específicos

1. Caracterizar la forma de usar y de diseminar el conocimiento arquitectónico en el AGSD.
2. Determinar las métricas de desarrollo de software y los aspectos relacionados con la efectividad de equipo que son afectados al implementar una solución que reduzca la vaporización del conocimiento arquitectónico en AGSD.
3. Establecer los elementos que debe contemplar una solución para reducir la vaporización el conocimiento arquitectónico en el AGSD, sin perder las características ágiles y del GSD.
4. Diseñar e implementar una solución de software que permita la reducción de la vaporización del conocimiento arquitectónico en el AGSD, sin perder las características ágiles y del GSD.
5. Determinar el impacto en las métricas y la efectividad del equipo al implementar la solución de software para reducir la vaporización del conocimiento arquitectónico en el AGSD.

1.8. Metodología de investigación

En la Figura 1.2 se esquematiza la metodología definida para el desarrollo de este trabajo de tesis. La metodología consta de tres grandes partes: (1) entendimiento, que es la fase en donde se ejecutan las actividades necesarias para contextualizarse del ambiente de trabajo del AGSD y del estado del arte respecto al manejo AK en ambientes ágiles y distribuidos; (2) concepción de solución, que es la fase donde emergen y son evaluadas las primeras ideas

para atender el problema planteado a nivel conceptual; (3) validación de solución, donde se diseñan, implementan y evalúan soluciones concretas de software en dos iteraciones. A continuación, se describe cada uno de los pasos de los que se compone cada fase.

1. **Revisión de literatura**. Se realizó un análisis bibliográfico con la finalidad de identificar los trabajos previos realizados en el tema propuesto y fundamentar un marco teórico para el presente trabajo (ver Capítulo 2). Como resultado se obtuvieron los enfoques que se reportan en la literatura para el manejo AK en AGSD. Además, gracias a esta revisión se obtuvo que una posible vía de solución para el problema planteado sería aprovechando el AK que se comparten los desarrolladores en medios electrónicos textuales no estructurados (UTEM – por sus siglas en inglés).
2. **Estudio de observación**. Este estudio se realizó en una empresa de desarrollo de software practicante de AGSD, con el fin de lograr el entendimiento sobre el AK que se comparte mediante los UTEM, cara a cara y mediante documentación (ver Capítulo 3). De este estudio se obtuvo un modelo que representa los aspectos de AK que se comparten los miembros de equipos de AGSD mediante UTEM, así como los aspectos de comunicación involucrados al compartir dicho conocimiento. También, se identificaron los temas arquitectónicos comunes compartidos cara a cara, mediante UTEM y mediante documentación, en términos del modelo de 4+1 vistas (Kruchten, 1995). Finalmente, se obtuvo que el AK compartido en UTEM es considerado importante el valioso para los equipos.
3. **Validación de resultados**. En este paso se mostraron los resultados a otras tres empresas practicantes del AGSD, con el fin de darle mayor validez a los hallazgos obtenidos.

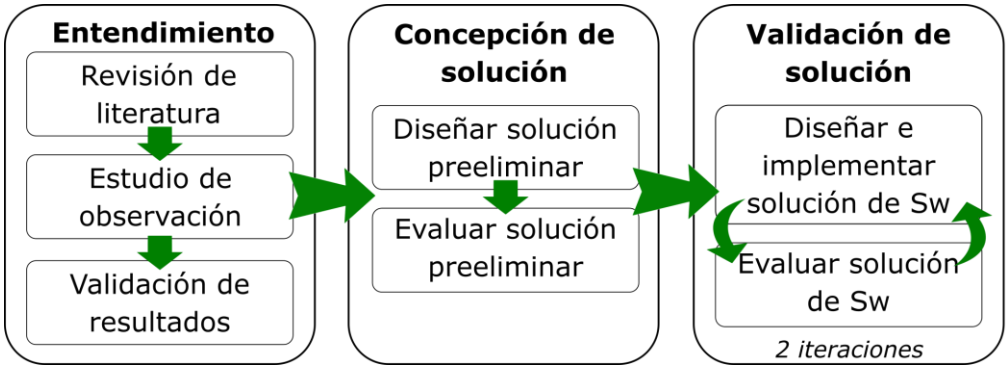


Figura 1.2. Metodología del trabajo de investigación.

Además, se confirmó que el AK que se comparte en UTEM es importante para los miembros de los equipos virtuales (ver Capítulo 3).

4. **Diseño de solución preliminar**. Con base en los resultados obtenidos del estudio de observación y su validación, así como en revisión de literatura se propone el concepto de condensación de AK como una solución a la vaporización del mismo, provocada por las condiciones de AGSD (ver Capítulo 4). Este concepto se basa en el aprovechamiento del AK compartido en UTEM, para ser clasificado y puesto a disposición de los miembros de equipos de AGSD. Además, se proponen los elementos iniciales que debería contemplar una solución de software que implemente la condensación de AK. Finalmente, se diseñó un prototipo no funcional de software que implementa este concepto con el fin de ser evaluado (ver Capítulo 5).
5. **Evaluación de solución preliminar**. El prototipo no funcional que implementa el concepto de condensación de AK, fue evaluado por desarrolladores de software practicantes del AGSD. De esta evaluación se obtuvo un refinamiento de los elementos a contemplar para implementar la condensación de AK, resaltando que la clasificación de AK compartida en UTEM basada en etiquetado social[3] obtuvo buena recepción en los participantes (ver Capítulo 5).
6. **Diseño e implementación de solución de software**. Esta parte se ejecutó en dos ocasiones debido a las dos iteraciones de la fase de validación de la solución. A continuación, se presentan las actividades ejecutadas en cada iteración.
    a. Primera iteración. Con base en los resultados de la evaluación del prototipo no funcional presentado en la etapa anterior, se diseñó e implementó un prototipo de mecanismo de clasificación de AK compartido en UTEM basado en etiquetado social basado en Web, el cual se llamó Chatagger (ver Capítulo 6).
    b. Segunda iteración. Con base en los resultados de la evaluación del prototipo de mecanismo de clasificación de AK, se diseñó e implementó un prototipo que implementa todos los elementos que contempla la condensación de AK, el cual se nombró ArchiKCo (ver Capítulo 7). En esta implementación se incluyó una

[3] https://www.igi-global.com/dictionary/social-tagging/27505

versión del mecanismo de clasificación de AK usado en Chatagger, ahora implementado sobre un mensajero instantáneo comercial (Skype).

7. **Evaluación de solución de software**. Esta parte se ejecutó en dos ocasiones debido a las dos iteraciones de la fase de validación de la solución. A continuación, se presentan las actividades ejecutadas en cada iteración.
    a. Primera iteración. Se condujo una evaluación de Chatagger con desarrolladores practicantes de AGSD de diferentes empresas, con la cual se confirmó que el etiquetado social puede ser un medio eficiente para la clasificación de AK compartido en UTEM (ver Capítulo 6). Además, se obtuvieron varias recomendaciones para la mejora de dicho mecanismo.
    b. Segunda iteración. Se condujo una evaluación de ArchiKCo con desarrolladores practicantes de AGSD de diferentes empresas mexicanas, y con estudiantes de licenciatura y posgrado de la Escuela Superior de Informática (ESI) de la Universidad Castilla-La Mancha (UCLM) de España (ver Capítulo 8). De esta evaluación se obtuvo que el concepto de condensación de AK es factible a ser implementado en un ambiente de AGSD, sin tener gran afectación en la agilidad y en los aspectos inherentes al GSD. Además, se obtuvieron recomendaciones para la mejora del prototipo, así como nuevas posibilidades que explorar como trabajo futuro.

1.9. Contribuciones principales

La principal aportación de esta tesis es el concepto de condensación de AK (definido en el Capítulo 4) y la determinación de su viabilidad a través de la evaluación de un prototipo (ver Capítulo 8). Este concepto representa una opción viable para disminuir la vaporización del AK a través del manejo del AK en ambientes de AGSD, que como se estableció anteriormente, es uno de los retos establecidos en la literatura al practicar este paradigma de desarrollo de software.

Al introducir el concepto de condensación de AK, se hizo un trabajo de definición del concepto de vaporización de conocimiento basado en las transformaciones de conocimiento tácito a explícito y viceversa, ya que no existe un consenso en la literatura al respecto de dicho concepto (ver Capítulo 4). Además, una aportación consecuente de este trabajo de

definición fue la introducción de dos nuevos sub-estados al estado explícito documentado del conocimiento (organizado y no organizado), así como un nuevo estado del conocimiento llamado vaporizado.

Finalmente, una aportación previa a la definición de la condensación de AK es la identificación de los enfoques para el manejo de AK en AGSD que reporta la literatura, y su clasificación en tres grandes áreas (ver Capítulo 2). Además, como consecuencia de seguir un enfoque para el manejo de AK en AGSD surge un modelo que representa los aspectos de comunicación y propios del AK que intervienen al compartir este conocimiento en UTEM en ambientes de AGSD, lo cual se puede considerar como otra aportación.

1.10. Alcances y limitaciones

Como parte del proceso de este trabajo de tesis se desarrolló un prototipo de software, el cual contempla las funciones precisas para poder evaluar la propuesta para la reducción de la vaporización del AK, es decir, se trata de un prototipo con fines de investigación. Por lo tanto, este prototipo no representa una solución de software completa y lista para ser implementada en una empresa de desarrollo de software que practique el AGSD.

Por otro lado, es sabido que la mayoría de los efectos al aplicar técnicas de manejo de conocimiento en general en alguna empresa no son evidentes en un corto plazo, por ello en cuanto a la determinación de viabilidad del manejo de AK propuesto en esta tesis, se enfocó sólo en variables en las que se pudiera ver un cambio en un corto plazo y a través de un experimento controlado, a partir de las cuales se hacen proyecciones de los efectos a largo plazo que podría tener dicha propuesta.

Además, para determinar la manera de manejar AK en dichos ambientes, se realizó una revisión de literatura y un estudio en empresas practicantes del AGSD, de donde surgen varias oportunidades interesantes de investigación, pero que desvían la atención a la problemática principal planteada en esta tesis.

Una de las limitaciones más grandes de este trabajo lo constituye el perfil tan específico de personas las cuales se verían directamente beneficiadas por los resultados obtenidos (desarrolladores de software practicantes del AGSD), lo cual limitó el número de participantes en las evaluaciones conducidas durante esta investigación. Sin embargo, esto no invalida los resultados obtenidos, ya que los análisis estadísticos realizados no se

centraron en la cantidad de participantes, sino en la cantidad de interacciones, etiquetas, resultados de búsqueda, etc., es decir, datos producidos por los participantes a partir de las actividades de evaluación.

1.11. Contenido del documento

El resto del documento está dividido en ocho capítulos los cuales se describen brevemente a continuación. En el Capítulo 2 se describen los paradigmas de desarrollo de software que intervienen en esta tesis, así como sus beneficios y retos actuales. También en el mismo capítulo se describen las disciplinas del manejo del conocimiento y del manejo del AK. Finalmente, en el mismo capítulo se presenta un mapeo sistemático de literatura sobre los enfoques para el manejo del AK en el AGSD.

En el Capítulo 3 se presenta el entendimiento de la manera de compartir AK en AGSD, incluyendo su compartición por medio de UTEM, cara a cara, mediante documento, u obtenido de un estudio en sitio. En el Capítulo 4 se presenta el concepto de condensación de AK, los elementos por los cuales se compone, así como la repercusión de este concepto en el modelo SECI, donde se describen las transformaciones entre conocimiento tácito a explícito y viceversa. Además, también se presentan las métricas que se verían afectadas al implementar la condensación de AK en AGSD y su relación con la efectividad de equipo.

En el Capítulo 5 se describe el prototipo preliminar (no funcional) que implementa la condensación de AK, pasando por la presentación del etiquetado social como mecanismo de clasificación de AK durante las interacciones en UTEM. Además, se presenta la justificación de usar el etiquetado social, y los trabajos que en la literatura y herramientas comerciales reportan el uso de esta técnica en la ingeniería de software. Finalmente, en este mismo capítulo se presenta la evaluación conducida con desarrolladores practicantes del AGSD, así como sus impresiones sobre el prototipo y las recomendaciones de diseño resultantes.

En el Capítulo 6 se presenta el diseño e implementación de un prototipo de mecanismo de clasificación de AK basado en etiquetado social, así como los resultados de su evaluación realizada por desarrolladores de software practicantes del AGSD, donde se incluyen percepciones de los participantes, perfiles que describen el comportamiento de los participantes de etiquetado, y recomendaciones para la mejora del mecanismo evaluado.

En el Capítulo 7 se presenta el diseño e implementación de un prototipo de condensador de AK para ambientes de AGSD llamado ArchiKCo, el cual se basó en los

resultados obtenidos tanto en el Capítulo 5 como en el Capítulo 6. Además, se presenta una clara relación entre los elementos diseñados e implementados y los elementos que conforman el concepto de condensación de conocimiento. Finalmente en este capítulo se presenta el escenario proyectado de uso de ArchiKCo y los beneficios potenciales en ambientes de AGSD.

En el Capítulo 8 se presenta la evaluación del prototipo ArchiKCo conducida con desarrolladores de distintas empresas mexicanas practicantes del AGSD, y con estudiantes de licenciatura y posgrado de la ESI de la UCLM. Los resultados de esta evaluación se presentan en términos del comportamiento de etiquetado, del desempeño de recuperación de AK y de las percepciones generales del prototipo. Finalmente en este capítulo se discuten los resultados y se determina la viabilidad del concepto de condensación de AK.

En el Capítulo 9 se presentan las conclusiones y reflexiones derivadas de este trabajo de tesis. Además, se presentan las principales aportaciones, incluyendo las publicaciones generadas a partir de esta tesis y otras colaboraciones derivadas del mismo trabajo. Finalmente, se presentan las oportunidades de trabajo futuro identificadas.

Capítulo 2.
Marco teórico

En este capítulo se describen los paradigmas de desarrollo de software que tienen injerencia en el desarrollo de la investigación que se plantea en esta tesis, es decir, el desarrollo global de software, el desarrollo ágil de software y su combinación llamada el desarrollo ágil y global de software. De estos tres paradigmas se describen sus beneficios y retos actuales, así como las diferencias intrínsecas que se originan en el desarrollo ágil y global de software. También, se describe la disciplina del manejo del conocimiento, junto con la definición de los tipos de conocimiento (tácito y explícito) y el ciclo integrado del manejo del conocimiento, previamente estableciendo la definición de conocimiento que se tomará como base en esta investigación. Adicionalmente, se describe la disciplina del manejo del AK, discutiendo previamente sobre las definiciones existentes de arquitectura de software, lo que se toma como base para describir de manera más clara el concepto de AK.

Además se presenta una revisión de literatura acerca del manejo del AK en empresas que practiquen AGSD, con el fin de conocer diferentes perspectivas que guíen los trabajos de esta investigación. Esta revisión se realizó siguiendo los lineamientos de un mapeo sistemático de literatura, ya que este tipo de revisión se enfoca en una exploración en amplitud, más que en una exploración en profundidad (como lo hace una revisión sistemática (Grant and Booth, 2009)), lo que permite conocer la diversidad de maneras reportadas en la literatura para el manejo del AK en AGSD.

2.1. Desarrollo global de software

El fenómeno de la globalización ha ampliado el campo competitivo de las empresas, incluso más allá de las fronteras del país en donde se localizan. Este fenómeno ha alcanzado a diversos ámbitos, siendo uno de ellos el del desarrollo de software, de tal manera que hoy en día una empresa de este giro puede ofrecer sus servicios a clientes que se encuentren en otros países. A este enfoque se le conoce como: Desarrollo Global de Software (GSD, por sus siglas en inglés) (Herbsleb and Moitra, 2001).

Se puede definir el GSD como el desarrollo de software que se realiza con equipos en múltiples ubicaciones geográficas, donde estos equipos pueden ser de la misma

organización, o bien con colaboraciones o subcontrataciones que involucren a diferentes organizaciones (Sangwan et al., 2006). Estos equipos podrían estar dentro de un país o en distintas partes del mundo; de hecho, la evidencia sugiere que una vez que los equipos están separados por más de 50 metros, una distancia mayor es irrelevante (Allen, 1984). Dada su naturaleza distribuida del GSD, se requiere la creación de equipos virtuales, es decir, un grupo de trabajo comunicado electrónicamente con diferencias geográficas, temporales y/o culturales (Lipnack and Stamps, 1997).

2.1.1 Beneficios y retos del desarrollo global de software

Algunos de los beneficios potenciales o motivadores más comunes para adoptar el GSD en el mundo, son los siguientes (Carmel, 1999; Herbsleb et al., 2005; Vizcaíno et al., 2014):

- **Acceso a un amplio catálogo de mano de obra capacitada**. El GSD al ser de carácter global, se cuenta con un número mayor de trabajadores potenciales de los países involucrados o zonas involucradas, con lo que se podría evitar cancelar proyectos por limitada experiencia en las tecnologías que se requieren para construir los complejos sistemas de hoy en día. También el acceso a más diversidad de mano de obra fomenta la innovación y creatividad en los proyectos.
- **Diferencias en los costos de desarrollo que favorecen la dispersión geográfica del equipo**. Esto se basa en la diferencia salarial entre los desarrolladores en distintos países, por ejemplo entre EE.UU. y la India el salario es aproximadamente ocho veces mayor
- **Aprovechamiento de las diferentes zonas horarias**. Gracias a la diferencia horaria, se puede implementar un sistema de trabajo basado en turnos llamado “siguiendo al sol”. Este sistema de trabajo consiste en que haya trabajo continuo empalmando los horarios de trabajo de distintas ubicaciones, es decir, cuando en un sitio se esté terminando la jornada laboral, en otro estaría iniciando. Esto a su vez permite que se reduzcan los tiempos de ingreso al mercado de los productos de software.
- **Avances en infraestructura**. Este punto se refiere más a un motivador para adoptar el GSD, ya que cada vez se tienen mejores anchos de banda de Internet y mejores herramientas de desarrollo e integración de software, que disminuyen la dificultad de trabajar a distancia.

- **Mayor proximidad al mercado y al cliente**. Una empresa de un país lejano puede estar atendiendo a clientes de otro país a través de la colaboración con otra empresa, o simplemente a través de una sucursal de ella misma.

Estos beneficios potenciales del GSD fueron estudiados por (Conchúir et al., 2009) encontrando algunas diferencias entre lo que potencialmente se espera y lo que realmente ha pasado en la práctica. Al respecto de la reducción de costos por la diferencia salarial en otros países, se ha visto que hay gastos significativos en comunicación, coordinación y control, por lo tanto el beneficio económico no es como se preveía. En cuanto a la reducción de los tiempos de desarrollo al aplicar la técnica de "seguir al sol", no ha sido tan efectiva como se esperaba ya que esta técnica se termina aplicando sólo para tareas de pruebas, es decir, mientras unos terminan de desarrollar en un sitio en el otro inician las pruebas. Esto se ha debido también a que el beneficio asociado a tener una mayor diversidad de mano de obra realmente no ha cumplido las expectativas, debido a que no todos los perfiles de trabajadores que se desean están disponibles, lo cual también ha causado que sólo las tareas menos complejas sean las que se asignen a los trabajadores remotos.

Además de las distancias geográficas y temporales, obvias en este paradigma de desarrollo, también existen las distancias lingüísticas y las culturales, lo cual se traduce en retos en la comunicación, control y coordinación (Holmstrom et al., 2006). Un ejemplo de problema causado por el lenguaje es la renuencia a participar activamente en teleconferencias, porque se prefiere interactuar por correo electrónico. Sin embargo, la interacción por este medio puede llegar a ser lenta y frustrante, lo que ocasiona problemas en la comunicación y coordinación. Del mismo modo, la cultura puede causar confusión y frustración. Un ejemplo es el de personas de una cultura que tiende a ser muy directa a la hora de interactuar con personas de otra cultura que no lo es tanto. Por un lado, los que son más directos pueden sentirse frustrados de que no se vaya al grano en las interacciones, y por otro lado los que son menos directos pueden sentirse ofendidos por la manera de actuar de sus contrapartes (Sangwan et al., 2006).

A pesar de los retos y problemas expuestos anteriormente el GSD sigue siendo adoptado por muchas empresas a nivel mundial, a tal grado que ha sido reportado como una tendencia fuerte en el Reporte Mundial de Inversiones de las Organización de las Naciones

Unidas del año 2017[4]. Por ello también el GSD y los retos que siguen abiertos son una tendencia en la investigación.

2.2. Desarrollo ágil de software

A partir de la primera década del presente siglo se ha popularizando un paradigma en el ámbito del desarrollo de software llamado el desarrollo ágil de software. Este paradigma fue una respuesta a la necesidad de responder a los cambios constantes que son comunes en los desarrollos de software, y que el paradigma tradicional de desarrollo (basado en planes estrictos) no es capaz de atender, ya que considera los cambios como un "enemigo" (Cockburn, 2006). Todo desarrollo de software que se considere ágil se basa en el manifiesto ágil (Beck et al., 2001), el cual se compone de cuatro valores y 12 principios a través de los cuales se entiende la esencia de este paradigma. Los cuatro valores ágiles son los siguientes: (1) Los individuos y sus interacciones sobre procesos y herramientas, (2) Software trabajando sobre una documentación extensiva, (3) Colaboración con el cliente sobre negociación contractual, y (4) Responder a los cambios sobre el seguimiento de un plan. A continuación, se explica cada uno de ellos.

"Los individuos y sus interacciones sobre procesos y herramientas". Este valor se refiere a dar más atención a las personas del equipo que a roles especificados en un diagrama del procesos. También se refiere a dar más atención a las interacciones entre los individuos, ya que la calidad de éstas es importante, tanto que cualquier tipo de proceso es beneficiado por la mejora de la comunidad. Es tan importante la interacción de los individuos que se recomienda que los equipos de trabajo se encuentren en un mismo espacio físico, donde puedan interactuar cara a cara sin tener que trasladase si quiera entre edificios (Smith, G., Sidky, 2009). En resumen este valor expresa la preferencia de un proceso no documentado con buenas interacciones sobre un proceso documentado con interacciones hostiles (Cockburn, 2006).

"Software trabajando sobre una documentación extensiva". El software funcionando es lo único que mide realmente el avance de un equipo de desarrollo. Los documentos de requerimientos, el análisis, el diseño, los flujos de pantalla, etc. son útiles como una ayuda,

[4] http://unctad.org/en/PublicationsLibrary/wir2017_en.pdf

pero en el paradigma ágil deben ser usados con moderación, siempre cuidando usar sólo los documentos suficientes y necesarios, para no agregar una sobrecarga de trabajo que al final no tenga un impacto importante en el trabajo sustantivo del equipo, es decir, en el software a desarrollar (Cockburn, 2006). De hecho, existen algunas metodologías ágiles que se centran casi exclusivamente en la fase de construcción o programación del software, sin tener una vista integral e ingenieril del mismo (Gotterbarn, 2004).

"Colaboración con el cliente sobre negociación contractual". Aquí la colaboración se refiere a la comunidad, la amabilidad, la toma conjunta de decisiones, la rapidez de la comunicación y las conexiones de los individuos. La colaboración con el cliente también se refiere a una relación amistosa entre diversos roles y límites organizacionales. Una buena colaboración puede salvar una situación contractual cuando está en peligro (Cockburn, 2006).

"Responder a los cambios sobre el seguimiento de un plan". Si bien la elaboración de un plan es útil, este pierde su utilidad cuando lo planteado se aleja demasiado de la situación actual, y hasta se convierte en perjudicial aferrarse a un plan obsoleto. Esto no significa que en el desarrollo ágil no se planee, de hecho cada una de las metodologías ágiles actuales contiene actividades de planificación, pero también contienen mecanismos para hacer frente a la evolución de las prioridades (Cockburn, 2006).

2.2.1 Beneficios y retos del desarrollo ágil de software

La popularidad del desarrollo ágil se ha debido a los beneficios reportados por las empresas que han adoptado este paradigma de desarrollo. A continuación, se presentan los beneficios más relevantes del desarrollo ágil (Cho, 2009; Papatheocharous and Andreou, 2013; Solinski and Petersen, 2016).

- **Alta adaptabilidad**. El equipo de desarrollo debe comprometerse a entregar un subconjunto de características acordado del producto en cada ciclo de desarrollo. Sin embargo, existe la oportunidad de refinar y redefinir constantemente las características del producto, de tal manera que en una siguiente iteración se comprometan elementos nuevos o modificados, lo que ofrece la oportunidad de introducir cambios en un plazo de pocas semanas.

- **Incremento en la visibilidad y predictibilidad del proyecto**. Cada miembro del equipo puede saber cómo va el proyecto en cualquier momento. Las reuniones y revisiones diarias del ciclo de desarrollo ofrecen maneras concretas de ver el progreso. Esto aumenta la predictibilidad del proyecto, además de manejar tiempos cortos de entrega de subproductos funcionales, lo cual a su vez disminuye el riesgo.
- **Alta productividad, calidad y velocidad de desarrollo**. Al tener entregas de producto en tiempos cortos, inherentemente se aumenta la productividad, pero sin descuidar la calidad. La calidad es alta ya que uno de los principios ágiles es la atención a la excelencia técnica en cada fase (Beck et al., 2001). Además, al dividir el proyecto en unidades manejables, el equipo puede concentrarse en el desarrollo, las pruebas y la colaboración de alta calidad. También, al tener entregas frecuentes y hacer revisiones durante cada iteración, la calidad mejora al encontrar y reparar oportunamente defectos e identificar desajustes en las expectativas planteadas en un inicio.
- **Incremento en la moral del equipo**. Uno de los principios ágiles refiere que los equipos de trabajo deben de ser auto-organizados (Beck et al., 2001), es decir, no hay un líder que organice el equipo, el mismo equipo se organiza como una comunidad. Formar parte de un equipo de esta naturaleza permite a las personas ser creativas, innovadoras y reconocidas por su experiencia. Además, el trabajo multifuncional permite a los miembros del equipo de desarrollo aprender nuevas habilidades y crecer enseñando a otros. Todo esto lleva a que los procesos cognitivos, conductuales y motivacionales/afectivos del equipo, se mejoren y por lo tanto sea más fácil lograr la efectividad de equipo (Kozlowski and Ilgen, 2006).

Si bien estos beneficios son muy atractivos para las empresas para decidir adoptar el desarrollo ágil, también existen retos abiertos en este paradigma en los que hay que pensar antes de dicha adopción. A continuación, se describen los principales (Fitriani et al., 2016; Hochmüller and Mittermeir, 2008; Solinski and Petersen, 2016).

- **Manejo del equipo**. Se refiere a retos relacionados con el diseño, coordinación y comportamiento de los miembros del equipo, incluyendo desacuerdos dentro del equipo, dificultad para introducir nuevos miembros, resistencia al cambio. la búsqueda de personas adecuadas para el paradigma ágil, la falta de orientación del equipo y el no enfrentarse entre sí.

- **Priorización de requerimientos**. La idea es que el requerimiento de máxima importancia tiene que ser implementado antes que los otros. Este proceso es se complica particularmente cuando se involucra a múltiples partes interesadas (incluyendo el cliente) con múltiples requisitos y prioridades que compiten entre sí,
- **Documentación**. La documentación mínima es un reto significativo en el paradigma ágil, ya que poca o nula documentación conduce a una falta de comprensión del proceso por parte de los desarrolladores recién incorporados, sobretodo en relación a los requerimientos y cambios a los mismos, y en la construcción de pruebas al código. Otra perspectiva de este reto es la tendencia a utilizar el pensamiento tradicional en cascada que conduce a la creación de extensos documentos para la especificación de requerimientos, lo que provoca que el sistema no sea bien comprendido, porque es difícil encontrar información en documentos grandes.
- **Filosofía o cultura de la compañía en desacuerdo con los valores ágiles centrales**. No siempre es fácil para la parte gerencial de una empresa comprometerse con el paradigma ágil, sin un "jefe" visible en el equipo. Esto ocasiona que la gestión ágil de proyectos no pueda adaptarse a todas las organizaciones, ya que se necesita de factores ambientales empresariales maleables para aplicar un enfoque ágil con éxito.
- **Escalabilidad**. Dado que el desarrollo ágil está planteado para equipos co-localizados donde la interacción personal es preferida sobre el seguimiento de flujos de proceso, aplicar este paradigma en equipos grandes (más de 10 personas) y distribuidos se ha convertido en un reto importante que implica una adaptación entre las metodologías tradicionales que son aptas para este tipo de equipos, y los valores y principios ágiles.

Estos retos no han frenado la adopción del paradigma ágil a nivel mundial, a tal grado que la metodología ágil Scrum[5] es considerada como un estándar de facto en muchos países[6] (Paper et al., 2014), lo que genera que la investigación en estos temas sea de actualidad.

2.3. Desarrollo ágil y global de software

En años recientes ha habido compañías de desarrollo de software que han adoptado el desarrollo ágil en combinación con el paradigma del GSD, con lo cual se ha formado un

[5] https://www.scrum.org/
[6] https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2

nuevo concepto llamado: el desarrollo ágil global de software (AGSD, por sus siglas en inglés) (Dumitriu et al., 2011). Una de las razones principales del creciente interés en adoptar el desarrollo ágil en el GSD es el fuerte enfoque en la colaboración, coordinación y comunicación que el desarrollo ágil aportaría a un ambiente distribuido (Šmite et al., 2010a), además del resto de los beneficios de este paradigma como la alta productividad, el incremento de velocidad de desarrollo y la alta adaptabilidad.

En principio ambos paradigmas parecieran que van en sentido contrario (como se muestra en la Tabla 2.1), ya que por un lado, en el GSD los equipos de desarrollo de software se encuentran distribuidos, lo cual dificulta la interacción cara a cara que el desarrollo ágil establece en sus principios y valores; y si se toman en cuenta los factores culturales y lingüísticos, esta dificultad se incrementa. Además, se tiene que en el GSD se suele realizar una documentación extensiva y clara para solventar las cuatro distancias inherentes a este paradigma, y por otro lado en el desarrollo ágil la documentación del software queda en un segundo plano, ya que se prefieren las interacciones entre los individuos y el software funcionando.

2.3.1 Beneficios y retos del desarrollo ágil y global de software

Las diferencias entre el desarrollo ágil y el GSD hacen que surjan muchos retos en la implementación del AGSD en la industria. Estos retos han sido clasificados en cuatro factores que se describen a continuación (Awar et al., 2017; Jain and Suman, 2016; Ramesh et al., 2006; Richter et al., 2016).

- **Factores humanos**. Donde se incluyen aspectos de construcción de buenas relaciones entre los involucrados, aspectos de liderazgo, de comportamiento y de comunicación, sobre todo motivado por la distancia lingüística.

Tabla 2.1. Comparación de las características del desarrollo global de software y del desarrollo ágil de software.

Desarrollo Global de Software		Desarrollo Ágil de Software
Equipos geográficamente distribuidos	Versus	Equipos co-localizados
Muchos documentos formales y basados en estándares		Pocos documentos ad hoc o informales
Basado en el seguimiento de procesos		Basado preferentemente en la interacción de las personas
Diferente cultura y zona horaria		Misma cultura y zona horaria

- **Factores de administración**. Estos factores incluyen la estimación de esfuerzo, el manejo del conocimiento, el entendimiento compartido y factores asociados a la distancia física y temporal, además de encontrar un balance en los acuerdos contractuales que tome en cuenta la formalidad del GSD y la informalidad de los acuerdos ágiles.
- **Factores de procesos**. Estos factores se refieren a la coordinación de tareas, la mejora de procesos, la aplicación de las prácticas ágiles y los procesos de comunicación, es decir, alcanzar un balance entre la comunicación informal que se estable en el desarrollo ágil y la comunicación formal que establece el GSD.
- **Factores de organización**. Se refiere a encontrar una forma de controlar los proyectos que balancee los controles basados en las personas, que establece el desarrollo ágil, y los controles basados en procesos, establecidos por el GSD

A pesar del evidente antagonismo entre los dos paradigmas que componen el AGSD, se han reportado algunos beneficios obtenidos con su adopción, como los siguientes (Holmström et al., 2006; Jalali and Wohlin, 2010; Li and Maedche, 2012; Shrivastava and Date, 2010).

- **Fortalecimiento de la comunicación y la colaboración en equipo**. Debido a la comunicación diaria y a los métodos menos formales del desarrollo ágil éstos aspectos han sido reportados como una mejora, además del fortalecimiento de las relaciones de confianza.
- **Reducción del tiempo de desarrollo**. Este es uno los beneficios obvios que se esperan en la implementación del desarrollo ágil y que efectivamente ha sido reportado en la industria, ya que la reducción en la documentación y la simplificación del proceso de desarrollo, enfoca a los involucrados a las actividades que más aportan a la construcción del producto de software. Además, el desarrollo se puede acelerar aprovechando que la jornada laboral pueda ser más amplia, por la diferencia de zonas horarias en las que se localicen los equipos de desarrollo.
- **Reducción de los costos de desarrollo**. Aunado a lo anterior, al simplificarse el proceso de desarrollo, también se simplifican los costos asociados, lo que aumenta la posibilidad de incrementar los ingresos. También, la reducción de costos se da por las diferencias salariales que puedan existir en distintos países.

- **Mejora en el manejo de requerimientos tardíos**. Dada la dinámica actual de los mercados es muy común que haya un constante cambio de requerimientos en los proyectos de software; el desarrollo ágil ha beneficiado al GSD en el manejo de estos cambios, ya que los cortos ciclos de desarrollo permiten la incorporación nuevas características a los productos de software entre el final y el inicio de estos ciclos.

Estos beneficios han sido más significativos para las empresas que los retos que el AGSD presenta, a tal grado que cada vez más empresas practicantes del GSD adoptan el desarrollo ágil (Ramesh et al., 2006), De hecho, el 11º informe anual del estado de agilidad elaborado por VersionOne[7] (compañía que produce herramientas de apoyo para el desarrollo ágil desde el 2002) indica que el 86% de los encuestados contaban con equipos distribuidos practicando algún tipo de metodología ágil.

El hecho que haya una gran adopción del AGSD en la industria no significa que ya exista un consenso o comprensión profunda, teóricamente fundamentada, de la aplicabilidad del desarrollo ágil en el GSD para obtener los beneficios potenciales (Šmite et al., 2010a). Es por ello que la investigación acerca del AGSD sigue siendo un tema abierto.

2.4. Manejo del conocimiento

Uno de los retos más significativos del AGDS es el manejo del conocimiento (Awar et al., 2017; Dorairaj et al., 2012; Jiménez et al., 2009), dado que las distancias físicas, temporales, lingüísticas y culturales limitan la trasferencia de conocimiento entre los involucrados (Kamaruddin et al., 2012). Con el fin de comprender mejor porque el manejo del conocimiento en AGSD es un reto, se establece una definición de conocimiento y a que se refiere el manejo del mismo.

Según Davenport y Prusak (Davenport and Prusak, 2000, p. 3), desde una perspectiva empresarial el conocimiento es "una mezcla fluida de experiencia, valores, información contextual y conocimientos de expertos que proporciona un marco para evaluar e incorporar nuevas experiencias e información… que se origina y se aplica en la mente de los conocedores". Además, agregan que "en las organizaciones, el conocimiento queda

[7] https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2

embebido en documentos y repositorios, pero también en rutinas, procesos, prácticas y normas".

Por otro lado, el manejo del conocimiento se considera un campo de estudio multidisciplinario, por lo que existe una multitud de definiciones dependiendo de la perspectiva que se tome, ya sea un punto de vista de negocios, o como activo de intelectual, o desde el punto de vista de las ciencias cognitivas o ciencias de la información, entre varias perspectivas (Dalkir, 2011). Sin embargo, para el propósito de este trabajo consideramos la siguiente definición: "el manejo del conocimiento es la disciplina que promueve un enfoque integrado para identificar, capturar, evaluar, recuperar y compartir todos los activos de información de una empresa" (Duhon, 1998). Al mencionar activos de información se refiere a los documentos, rutinas, procesos y prácticas en donde se embeben las experiencias, valores, información contextual y conocimientos de expertos.

Contextualizando todo lo anterior al ambiente del AGSD, es claro que el manejo del conocimiento es un reto, ya que en este ambiente existe un claro antagonismo entre la manera de trasmitir conocimiento: el paradigma ágil indica la preferencia de la interacción cara a cara y el GSD requiere trasmitir conocimiento a través de documentos para disminuir los efectos de las distancias físicas, temporales, lingüísticas y culturales. Es decir, en términos de tipos de conocimiento se refiere a la preferencia del conocimiento tácito, por el lado ágil, y a la preferencia por el conocimiento explícito, por parte del GSD.

El conocimiento explícito puede expresarse en un lenguaje formal y sistemático, puede compartirse fácilmente ya que puede ser capturado en forma de fórmulas científicas, documentos, grabaciones de audio o imágenes, etc., ya sea en forma física o virtual (en algún medio electrónico) (Dalkir, 2011; Nonaka et al., 2000); por lo tanto, se puede procesar, transmitir y almacenar con relativa facilidad. Por el contrario, el conocimiento tácito tiende a residir en las cabezas de los conocedores, y está profundamente arraigado en la acción, los procedimientos, las rutinas, etc. (Dalkir, 2011); por lo tanto, generalmente es difícil (pero no imposible) poner este conocimiento en palabras, texto o dibujos (Nonaka et al., 2000), es decir, se requiere un proceso (en ocasiones no trivial) para convertirlo en explícito. En términos del desarrollo de software en general, el conocimiento explícito representaría manuales, contratos, definición de proyectos, requerimientos, diagramas de especificación

técnica, etc., pero en el desarrollo ágil, el conocimiento relacionado con diseño, codificación e incluso parte de los requerimiento se queda en su forma tácita (Bider and Söderberg, 2016).

2.4.1 Ciclo integrado del manejo del conocimiento

El ciclo de manejo del conocimiento implica una serie de actividades embebidas en su propia definición, es decir: identificar, capturar, evaluar, recuperar y compartir conocimiento. Al igual que existe una multitud de definiciones de manejo de conocimiento, también existen varias versiones de ciclos de actividades para realizar dicho manejo. Dalkir propone un ciclo integrado para el manejo del conocimiento que consiste en tres fases (Dalkir, 2011): (1) Crear/Capturar, (2) Compartir/Diseminar y (3) Adquirir/Aplicar, las cuales se describen a continuación.

1. **Crear/Capturar**. La creación de conocimiento es el desarrollo de nuevos conocimientos y *know-how* (el saber hacer que generalmente es tácito), es decir innovaciones dentro de la empresa. La captura del conocimiento se refiere a la identificación y posterior codificación del conocimiento y *know-how* dentro de una empresa. Esta fase generalmente implica la transformación del conocimiento tácito a explícito.
2. **Compartir/Diseminar**. El paso posterior a la creación y captura del conocimiento debe ser alguna forma de evaluación de este conocimiento capturado, basada en las siguientes preguntas: ¿Es válido este contenido? ¿tiene suficiente valor para la organización que debería ser añadido a la reserva de capital intelectual? Una vez evaluado el conocimiento que ya se encuentra capturado, se disemina o comparte usando los medios disponibles de la empresa, que de manera muy tradicional sería haciendo llegar copias físicas de dichos documentos a los interesados, o bien, de una forma más moderna sería compartiendo documentos electrónicos en repositorios que permitan el acceso a los distintos interesados.
3. **Adquirir/Aplicar**. Adquirir conocimiento puede implicar un proceso previo de encontrar lo que se está buscando, o bien, que dicho conocimiento llegó al interesado a través de un curso, presentación o capacitación. Una vez adquirido el conocimiento, el interesado lo aplica a su trabajo diario teniendo la oportunidad de refinarlo y hacer nuevas aportaciones a la base de conocimiento empresarial, lo que reiniciaría el ciclo nuevamente.

Contextualizando este ciclo de manejo del conocimiento al desarrollo de software, los conocimientos principales que se crean, capturan, comparten, adquieren y aplican son los directamente involucrados en el desarrollo del producto, es decir, requerimientos, diseño de software, aspectos técnicos de codificación e implantación, etc. Las empresas de desarrollo de software tratan que los ingenieros recopilen y capturen sus experiencias, y que las distribuyan usando medio tecnológicos, para facilitar el flujo de dicho conocimiento en la organización (Dingsøyr and Conradi, 2002). Los medios tecnológicos preferidos para respaldar los flujos del conocimiento son los repositorios, y en el caso del desarrollo ágil el "saber quién sabe qué" es más importante dado el predominio del conocimiento tácito en este paradigma (Bjørnson and Dingsøyr, 2008).

2.5. Manejo del conocimiento arquitectónico

Una buena práctica durante cualquier ciclo de desarrollo de software es la de capturar y compartir el conocimiento de los desarrolladores para mantener las mismas ideas de diseño y garantizar la evolución esperada del proyecto en términos de arquitectura de software. En este punto ya se tiene un problema, ya que existen muchas definiciones de arquitectura de software en la literatura. La Tabla 2.2 presenta tres de las definiciones de este concepto que se encuentran con más frecuencia en la literatura, de donde se obtiene una definición unificada.

Definición 2.1. Arquitectura de software. Se refiere a la definición de estructuras de componentes y la descripción de su organización y relaciones de gran importancia para guiar la evolución de un software.

Tabla 2.2. Definiciones comunes del concepto de arquitectura de software.

Autor	Definición de arquitectura de software
Estándar IEEE 1471-2000 (IEEE, 2000)	"La organización fundamental de un sistema encarnada en sus componentes, las relaciones entre sí y con el medio ambiente, así como los principios que guían su diseño y evolución"
(Bass et al., 2003)	"La estructura o estructuras del sistema, que comprenden los elementos de software, las propiedades visibles externamente de esos elementos, y las relaciones entre ellos"
(Fowler, 2002)	"La descomposición de más alto nivel de un sistema en sus partes; las decisiones que son difíciles de cambiar; hay múltiples arquitecturas en un sistema; lo que es arquitectónicamente significativo puede cambiar a lo largo de la vida útil de un sistema; y, al final, la arquitectura se reduce a lo que sea importante"

Esto significa que la arquitectura de software se refiere a cuestiones generales y de alto nivel de abstracción en la definición de un software, sin embargo Kruchten en su definición de arquitectura menciona que (Kruchten et al., 2006): “la arquitectura de software también implica funcionalidad, usabilidad, resistencia, rendimiento, reutilización, comprensibilidad, limitaciones económicas y tecnológicas, compensaciones y preocupaciones estéticas”. Por lo anterior, incluso los requerimientos de software y diseño detallado son parte de una arquitectura de software.

Una vez teniendo una visión más amplia de lo que se define como arquitectura de software es pertinente establecer lo que se conoce como AK. Una cantidad considerable de conocimiento es usada y producida durante el ciclo de vida de la arquitectura de cualquier proyecto de software. La mayor parte de estos conocimientos se refieren a la solución arquitectónica en términos de: componentes, conexiones, protocolos de comunicación, aspectos físicos, etc. Además, se toman muchas decisiones de diseño para lograr una solución arquitectónica, y es importante registrar esas decisiones para poder evolucionar la arquitectura en el futuro. Al conjunto de diseños y decisiones arquitectónicas se le denomina AK (Philippe et al., 2006).

La idea principal detrás del manejo de AK es capturar y compartir el AK entre los desarrolladores para mantener las mismas ideas de diseño y garantizar la evolución esperada del proyecto en términos de arquitectura. Al momento de escribir estas líneas, no se encontró en la literatura una definición concisa y consensuada de manejo del AK. Sin embargo, extrapolando la definición de manejo del conocimiento de Duhon (Duhon, 1998) se conformó la siguiente definición.

Definición 2.2. Manejo del AK. Es una disciplina que promueve un enfoque integrado para identificar, capturar, evaluar, recuperar y compartir todos los elementos de una arquitectura de software de un proyecto, tomando en cuenta las decisiones y razones para haber llegado al estado actual de la arquitectura.

Los beneficios de manejar el AK generalmente se pueden obtener a mediano y largo plazo, ya que la documentación de la lógica del diseño evita procesos de recuperación de la arquitectura en etapas de mantenimiento, que se utilizan principalmente para recuperar las decisiones cuando el diseño, la documentación o incluso los creadores de la arquitectura ya no están disponibles (Kruchten, 2009). Otro de los beneficios es que se mejora el proceso de

arquitectura de software, al proporcionar un mejor apoyo para la toma de decisiones, y para la reutilización de artefactos arquitectónicos, con el fin de llegar a una arquitectura eficaz y eficiente, con un sencillo mantenimiento y fluida evolución (M. A. Babar, 2009).

Para obtener los beneficios del manejo del AK se requiere una atención continua para mantener los cambios en el código y el diseño alineados con las decisiones. Sin embargo, este trabajo generalmente es demasiado tedioso y no trae ningún beneficio inmediato a la persona que realiza el manejo (Kruchten, 2009). Es por ello que aún es un reto contar con medios o procesos que faciliten las actividades asociadas al manejo del AK, aún más pensando en cualquier desarrollo ágil de software, debido a la preferencia a disminuir el número de documentos a generar. Ahora, si además de trabajar de manera ágil los equipos de desarrollo están geográficamente distribuidos el reto es aún mayor, ya que entran en juego los factores temporales, culturales y lingüísticos propios del GSD.

2.5.1 Manejo del AK en el desarrollo global de software

En las compañías de GSD se requieren estrategias de coordinación para diseminar las decisiones de diseño entre los equipos distribuidos, particularmente cuando estos equipos desarrollan interfaces para comunicar componentes de software (N. Ali et al., 2010). Se han propuesto estrategias y prácticas arquitectónicas para gestionar el AK, tales como las siguientes (N. Ali et al., 2010; Beecham et al., 2010):

- Alineación de la arquitectura y aspectos organizacionales, para que los componentes de software coincidan con el organigrama de la empresa para reducir la necesidad de conocimiento entre departamentos durante el desarrollo. Además, se sugiere tener un equipo de arquitectos para facilitar la comunicación entre las diferentes ubicaciones.
- Prácticas de manejo de conocimiento para crear y diseminar AK, como los siguientes: (1) Interacciones frecuentes entre sitios (preferentemente en cara a cara); (2) reuniones de inicio de proyecto llevadas a cabo cara a cara para garantizar que todos los sitios tengan las mismas expectativas, y en donde se pueda desarrollar arquitectura de alto nivel de abstracción; (3) Una "red de voluntarios" que pueda responder preguntas sobre alternativas arquitectónicas, modelos de diseño, etc; (3) delegación en sitios cruzados, donde un sitio asigna personal para reubicarse temporalmente en sitios remotos, para establecer una comprensión compartida de la arquitectura del sistema y para crear relaciones personales con el personal local; (4) implementar una estructura

organizacional clara para proporcionar líneas claras de comunicación entre las partes interesadas en sitios locales y remotos; (5) crear un repositorio de artefactos, almacenar decisiones y su racionalización, así como los diseños arquitectónicos reales.

- Infraestructura para administrar AK, por ejemplo el uso de Wikis para almacenar y compartir este conocimiento. Un Wiki es una herramienta colaborativa e informativa que en el desarrollo de software es utilizada para compartir conocimiento (Klobas and Beesley, 2006).

También se han presentado desafíos no resueltos sobre el manejo del AK, tales como: la necesidad de herramientas para administrar este conocimiento en GSD, y la necesidad de comunicar decisiones de diseño entre equipos geográficamente dispersos y culturalmente diferentes (N. Ali et al., 2010).

2.5.2 Manejo del AK en el desarrollo ágil de software

Respecto al manejo del AK en el desarrollo ágil de software, se han reportado problemas en la documentación de los proyectos, sobre todo entre los desarrolladores, ya que prefieren la comunicación cara a cara. Esto lleva a los desarrolladores a obtener AK directamente del código, lo que no garantiza la adquisición correcta del mismo (Yanzer Cabral et al., 2014a). Se han identificado un conjunto de herramientas de manejo de conocimiento, tales como (Yang et al., 2016; Yanzer Cabral et al., 2014a): Wikis, Wikis semánticos y diferentes herramientas tipo groupware, la mayoría de las cuales está enfocada a desarrolladores, pero sólo unas pocas se han probado en entornos reales. Un groupware es un sistema informático que apoya a un grupo de personas comprometidas en una tarea común, y que proporciona una interfaz para un entorno compartido (Ellis et al., 1991). Además, los Wikis (tradicionales y semánticos) y las herramientas de groupware se han tomado de otros campos del desarrollo de software, por lo que ninguna de ellas está dedicada a soportar el manejo de la arquitectura de software en ambientes ágiles (Yanzer Cabral et al., 2014a).

También se ha visto que la captura del AK es la actividad que se discute con más frecuencia cuando los equipos ágiles aplican prácticas de arquitectura de software (Yang et al., 2016). Y precisamente la mayoría de los desafíos reportados están relacionados con la descripción de la arquitectura, así como con la forma tácita de manejar AK.

2.6. Manejo del conocimiento arquitectónico en el AGSD

Considerando que el manejo del AK en AGSD es central en el desarrollo de esta tesis, se realizó una revisión de literatura siguiendo los lineamientos de un mapeo sistemático de literatura, debido a que este tipo de revisión se enfoca en una exploración en amplitud, más que en una exploración en profundidad (como lo hace una revisión sistemática (Grant and Booth, 2009)). Con el resultado de esta revisión se busca conocer la diversidad de maneras reportadas en la literatura para el manejo del AK en AGSD, y así contar con diferentes perspectivas que guíen los trabajos de esta tesis. Los detalles del planteamiento del mapeo sistemático realizado se pueden consultar en (Gilberto Borrego et al., 2017).

2.6.1 Enfoques para el manejo del AK en AGSD

Después de ejecutar el protocolo de búsqueda especificado para el mapeo sistemático, se analizaron los artículos resultantes y se obtuvo una ontología, donde se reflejan los enfoques utilizados para manejar el AK en AGSD (ver Figura 2.1). En la ontología se identifican tres áreas principales: enfoques basados en metodología, enfoques basados en documentación basada en artefactos y enfoques basados en la comunicación, los cuales se describen a continuación. (1) Enfoques basados en artefactos de documentación, que se refieren al software de soporte al proceso de desarrollo de software, por ejemplo: ambientes integrados de desarrollo (IDE por sus siglas en inglés), Wikis, repositorios, etc., que es usado para

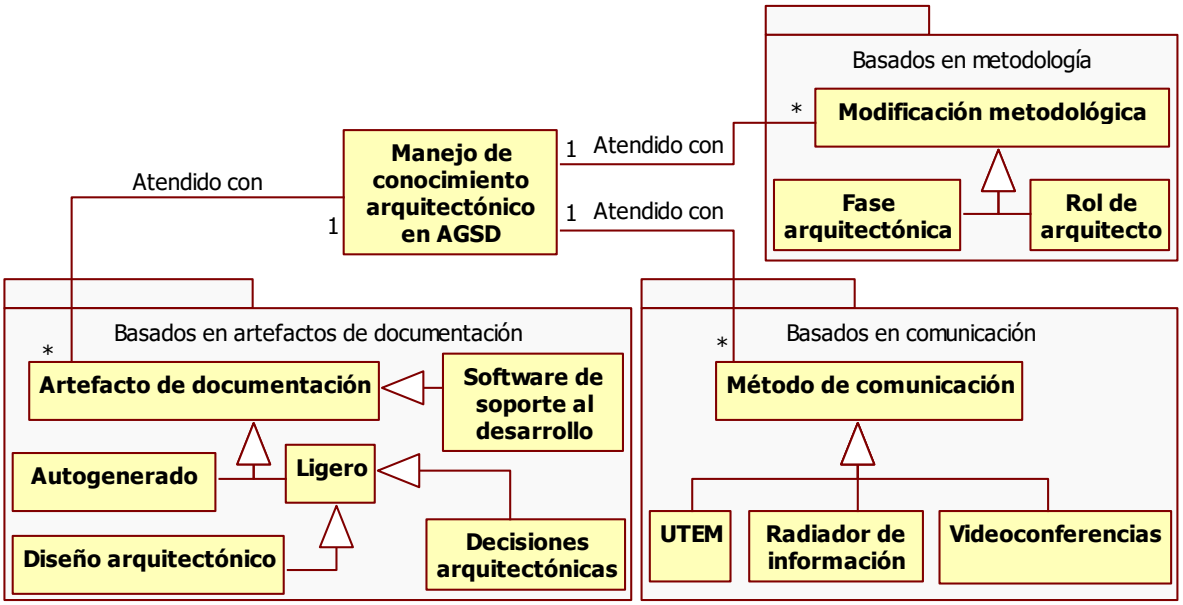


Figura 2.1. Representación ontológica de los enfoques utilizados para solventar los desafíos del manejo de AK en AGSD.

registrar el diseño arquitectónico o las decisiones arquitectónicas. También se refiere a la documentación generada por un proceso automático o basada en una notación ligera. (2) Enfoques basados en metodología, que se refieren a la inclusión de fases o roles para abordar problemas arquitectónicos. (3) Enfoques basados en la comunicación, referentes a compartir o publicar AK usando videoconferencias, y medios textuales electrónicos no estructurados (UTEM por sus siglas en inglés), como el correo electrónico, mensajería instantánea, foros, etc.), así como radiadores de información, por ejemplo, pantallas públicas para diseminar conocimiento.

Además, se relacionaron los artículos obtenidos con este esquema de clasificación (ver Tabla 2.3). Se obtuvo que los enfoques basados en la metodología representa el 20% de los artículos, los enfoques basados en artefactos de documentación representan el 45% y los enfoques basados en la comunicación representan el 35% de todos los artículos resultantes. En las siguientes secciones se describe cada una de las tres grandes áreas de enfoques identificados.

2.6.1.1 Enfoques basados en artefactos de documentación

En esta sección se describen los enfoques de documentación basados en artefactos ligeros y autogenerados, así como en el software de soporte al desarrollo, es decir, repositorios, wikis y herramientas de groupware.

2.6.1.1.1 Software de apoyo al desarrollo

Existen compañías de AGSD con políticas para que los artefactos de AK se publiquen en wikis (M. a. Babar, 2009; Lloyd et al., 2017; Razzak and Smite, 2015; Van Hillegersberg et al., 2011; Young and Terashima, 2008). Sin embargo, los desarrolladores sienten que es una carga capturar conocimiento constantemente para mantener el proyecto documentado (Sohan et al., 2010) y se termina por no usarlos (Dingsøyr, 2009). Además, los desarrolladores de AGSD reportan problemas con la recuperación de conocimiento en Wikis, tales como: una estructura ineficaz para la consulta de conocimiento (Capilla et al., 2016) y un mecanismo ineficiente para operar un wiki (Stol et al., 2010)

Tabla 2.3. Distribución de los artículos por entidad del modelo ontológico.

Entidades del modelo ontológico			Artículos relacionados	Porcentaje
Artefactos de documentación	Software de soporte al desarrollo		(M. a. Babar, 2009; Bruun and Hansen, 2014; Capilla et al., 2016; Clerc et al., 2011; Kim et al., 2014; Lloyd et al., 2017; Nowak and Pautasso, 2010; Razzak and Smite, 2015; Read and Maurer, 2003; Rost et al., 2013; Stettina and Heijstek, 2011; Stol et al., 2010; Urrego et al., 2014; Van Hillegersberg et al., 2011)	30%
	Autogenerado		(Antonino et al., 2014; Gayer et al., 2016; O'Reilly et al., 2005; Sohan et al., 2010)	9%
	Ligero	Diseño arquitectónico	(de Graaf et al., 2014; Pérez et al., 2010)	4%
		Decisiones arquitectónicas	(Díaz et al., 2014; Nowak and Pautasso, 2010; Zimmermann et al., 2015)	7%
Métodos de comunicación	Videoconferencia		(Esbensen et al., 2015; Holz and Maurer, 2003; Korkala and Maurer, 2014; Paasivaara and Lassenius, 2014; Schmidt and Meures, 2016; Sharp et al., 2012; Šmite et al., 2010b; Van Hillegersberg et al., 2011; Young and Terashima, 2008)	20%
	UTEM		(Clerc et al., 2011; Harrison and Veerappa, 2014a; Sharp et al., 2012; Van Hillegersberg et al., 2011; Young and Terashima, 2008)	11%
	Radiadores de información		(Esbensen et al., 2015; Sharp et al., 2012; Yagüe et al., 2016)	7%
Modificación Metodológica	Rol de arquitecto		(M. a. Babar, 2009; Bass, 2016; Clerc et al., 2011; Eeles, 2013; Lloyd et al., 2017; Martini and Bosch, 2014; Moe et al., 2014; Paasivaara, 2017; Paasivaara et al., 2018; Phalnikar et al., 2009; Razzak and Smite, 2015; Shen et al., 2016; Šmite et al., 2017; Van Hillegersberg et al., 2011)	30%
	Fase arquitectónica		(Avritzer et al., 2010; Boehm et al., 2010; Costa et al., 2014; Del Nuevo et al., 2011; Díaz et al., 2014; Lloyd et al., 2017; Moe et al., 2014; Paasivaara, 2017; Paasivaara et al., 2018; Šmite et al., 2017; Tanveer, 2015)	24%

Por otro lado, se ha reportado el uso de herramientas de groupwares enfocados en compartir o representar AK, como COACH-IT (Read and Maurer, 2003), que utiliza el Lenguaje de Definición de Arquitectura (ADL por sus siglas en inglés) como estándar para representar una arquitectura, sin embargo, el mismo ADL es su principal desventaja, ya que este lenguaje no ha sido adoptado en el desarrollo de software. Se tiene también el caso de Archinotes (Urrego et al., 2014), que ofrece una forma más informal de crear arquitecturas

colaborativamente. Sin embargo, es importante que estas herramientas sean evaluadas en entornos industriales para considerarlos como soluciones confiables.

En resumen, los trabajos presentados dependen en gran medida de la disciplina de los interesados para gestionar el conocimiento de manera eficiente. Desde el estudio de (Dingsøyr and Conradi, 2002), se ha enfatizado que los desarrolladores deberían recopilar y distribuir conocimiento. Sin embargo, en el desarrollo ágil esto es difícil debido a las presiones de tiempo que son comunes en este paradigma de desarrollo (Sneed, 2014).

2.6.1.1.2 Artefactos ligeros

Los artefactos ligeros pretenden ser una alternativa a la informalidad al representar arquitecturas en AGSD (Bruun and Hansen, 2014; Urrego et al., 2014), como lo muestra también la tendencia en la literatura respecto al desarrollo ágil (Yang et al., 2016; Yanzer Cabral et al., 2014a). Estos artefactos están diseñados para consumir menos tiempo que las formas tradicionales de representación de AK, además de ser comprensibles, independientemente del idioma del interesado. Es decir, sin causar pérdida de significado cuando los artefactos son consultados fuera del contexto original (Paredes et al., 2014). Un ejemplo de la pérdida de significado se presenta, cuando se hace un boceto arquitectónico en una reunión de diseño y un miembro del equipo lo consulta tiempo después, por lo general no se interpreta de la misma manera en la que se hizo durante la reunión.

Los artefactos ligeros se pueden dividir en dos categorías: (1) los que capturan diseño arquitectónico y (2) los que capturan decisiones arquitectónicas. Los primeros incluyen notaciones gráficas alternas al UML, para representar componentes, aspectos o variaciones y sus características (e.g. Plastic Partial Components – PPC (Pérez et al., 2010)); y una documentación donde el equipo de desarrollo debe crear una ontología para representar sus necesidades de AK para luego instanciar (o "poblar") la ontología con el conocimiento de diferentes proyectos (de Graaf et al., 2014). Los artefactos ligeros para la captura de decisiones arquitectónicas**,** incluyen extensiones de modeladores de UML para capturar decisiones sobre los mismos modelos (Zimmermann et al., 2015); notaciones gráficas para ayudar a descartar alternativas en un espacio de diseño (Nowak and Pautasso, 2010); y una extensión de PPC que considera las decisiones de arquitectura y el diseño en un mismo modelo (Díaz et al., 2014). Así, los desarrolladores tienen la capacidad de rastrear (1) por

qué se eligió un elemento, (2) cuál es el costo y el riesgo de usar tal elemento, (3) la descripción de la decisión, (4) suposiciones y (5) restricciones.

2.6.1.1.3 Artefactos autogenerados

En respuesta a las presiones de tiempo inherentes en entornos ágiles, los artefactos autogenerados parecen ser una buena opción para codificar AK. Se encontraron cuatro trabajos sobre este tema, donde se presentan soluciones para (1) mostrar las estructuras gráficas del sistema en desarrollo (WRCC - War Room Command Console) (O'Reilly et al., 2005); (2) generar directorios de expertos en ciertos temas (como en la revisión de (Bjørnson and Dingsøyr, 2008)), basados en las comunicaciones por correo electrónico y los historiales de repositorios de código (Moraes et al., 2010); (3) rastrear artefactos (Traceman) (Antonino et al., 2014; Gayer et al., 2016), y (4) relacionar mensajes de correo electrónico con historias de usuarios (Sohan et al., 2010), con un 70% de precisión en las relaciones, por lo que requeriría de mejoras para una implementación industrial.

En el caso particular de Traceman (Antonino et al., 2014; Gayer et al., 2016) se requiere (1) la existencia de documentos de requerimientos, historias de usuarios, diseño de arquitectura y casos de prueba, los cuales no siempre existen de manera formal en un entorno ágil; y (2) la existencia de roles de ingeniero de requerimientos, arquitecto de software e ingeniero de pruebas, para establecer relaciones en Enterprise Architect[8] (un modelador UML[9]) entre historias de usuario - requerimientos, historias de usuario - casos de prueba, y diseño de arquitectura - historias de usuario. Estos roles no siempre son parte de un equipo ágil, por lo tanto, las actividades asociadas ellos (establecer las relaciones) tendrían que ser adoptadas por otros roles (e.g. un desarrollador), con lo cual se corre el riesgo que éstas no se ejecuten por no ser significativas para el rol que las adopte.

2.6.1.2 Modificación metodológica

Los métodos ágiles no tienen una fase para definir una arquitectura de software. El manifiesto ágil (Beck et al., 2001) establece que "El software funcionando sobre una documentación completa" y "Las mejores arquitecturas, requerimientos y diseños surgen de equipos auto-organizados". Esto significa que la codificación es preferible al diseño de una arquitectura

[8] http://www.sparxsystems.com/products/ea/
[9] http://www.omg.org/spec/UML

(la arquitectura surge a medida que avanza el proyecto), por lo tanto el rol del arquitecto, como tal, no existe, ya que todos los miembros del equipo participan en el diseño de la arquitectura. Sin embargo, cuando los equipos de desarrollo están distribuidos geográficamente y los proyectos son grandes, se debe usar al menos una definición mínima de arquitectura para coordinar los esfuerzos de desarrollo. Para solventar lo anterior, se encontraron dos enfoques en AGSD para incluir aspectos arquitectónicos en metodologías ágiles: incluir un rol de arquitecto e incluir una fase de arquitectura. Estas modificaciones metodológicas deben hacerse con cuidado, ya que se podría introducir una sobrecarga que afecte la agilidad y los tiempos de entrega de los proyectos

2.6.1.2.1 Rol de arquitecto

La inclusión de un rol de arquitecto se puede referir a incorporar a una sola persona o un equipo de arquitectos (M. a. Babar, 2009; Bass, 2016; Clerc et al., 2011; Eeles, 2013; Martini and Bosch, 2014; Moe et al., 2014; Phalnikar et al., 2009; Razzak and Smite, 2015; Shen et al., 2016; Van Hillegersberg et al., 2011). Al incluir este rol, los equipos de AGSD tienen una entidad que define, refina, comunica, codifica (en artefactos arquitectónicos) y hace un seguimiento de la arquitectura de software de los proyectos, así como de los principios arquitectónicos, patrones, convenciones de codificación, etc.

Un equipo de arquitectos puede organizarse jerárquicamente (con un arquitecto en jefe y arquitectos subordinados) (Bass, 2016; Clerc et al., 2011; Martini and Bosch, 2014; Moe et al., 2014) o por disciplina (Eeles, 2013), por ejemplo, arquitecto de aplicaciones, arquitecto de seguridad, arquitecto de datos, etc. Los arquitectos también interactúan con los clientes, por lo que tienen más elementos para priorizar las historias de los usuarios y para diseñar una solución general (M. a. Babar, 2009). En otros casos, un Scrum Master (un facilitador de equipo y un mediador en la metodología Scrum (Schwaber and Sutherland, 2011)) desempeña el papel de arquitecto o mentor técnico (M. a. Babar, 2009).

Por último, se observó que todos los estudios de AGSD en los que se introdujo un rol de arquitecto se realizaron en grandes empresas (+50 empleados). Esto significa que agregar un arquitecto en pequeñas empresas de AGSD podría ser inconveniente debido a problemas de estructura y presupuesto.

2.6.1.2.2 Fase arquitectónica

La necesidad de coordinación arquitectónica en AGSD ha llevado a agregar una fase de arquitectura en algunos proyectos. Además, esta modificación metodológica puede ser más conveniente para las pequeñas empresas, en vez de modificar su estructura para contratar un arquitecto de software.

Al respecto, se reporta la inclusión de una fase llamada *Sprint Zero* (Avritzer et al., 2010; Boehm et al., 2010; Costa et al., 2014; Díaz et al., 2014; Eloranta and Koskimies, 2012; Moe et al., 2014), también llamada fase de pre-desarrollo (Avritzer et al., 2010) o ingeniería de dominio (Díaz et al., 2014), para desarrollar una comprensión común de los requerimientos y proponer una primera propuesta arquitectónica en conjunto.

Además, se encontraron propuestas de metodologías híbridas basadas en Scrum y RUP (Del Nuevo et al., 2011; Tanveer, 2015), para utilizar el marco de administración de proyectos ágiles de Scrum y la disciplina de ingeniería de software de RUP. Sin embargo, esta hibridación puede ser demasiado pesada para equipos pequeños, ya que requiere muchas actividades y artefactos (Del Nuevo et al., 2011). De hecho, no se reporta algún estudio de caso donde se haya implementado la metodología híbrida (Del Nuevo et al., 2011; Tanveer, 2015).

2.6.1.3 Enfoques basados en la comunicación

La comunicación cara a cara y la colaboración entre los miembros de equipo es la forma natural de compartir AK en el desarrollo ágil; sin embargo, en AGSD, este método de comunicación es complicado debido a las distancias temporales y físicas.

2.6.1.3.1 Medios electrónicos textuales no estructurados

Los UTEM (correo electrónico, foros, mensajería instantánea, etc.) emergen como una alternativa para compartir este conocimiento. Estos medios son algunos de los preferidos por los desarrolladores en compañías de software (Clerc et al., 2011; Harrison and Veerappa, 2014a; Sharp et al., 2012; Van Hillegersberg et al., 2011). Particularmente, el correo electrónico se usa para compartir y solicitar AK, porque los desarrolladores pueden verificar su escritura antes de enviar el mensaje (Clerc et al., 2011).

En general, las redes sociales son populares en desarrollo ágil (co-localizado o distribuido), particularmente para aclarar los requerimientos arquitectónicos (Harrison and Veerappa, 2014a; Van Hillegersberg et al., 2011), y son usados generalmente durante el inicio de los proyectos y en las etapas de planificación de cada ciclo de desarrollo de los mismos. Los desarrolladores ágiles aprovechan los mensajeros instantáneos, ya que mantienen registros de ellos y usan esos registros para aclarar problemas con los clientes (Harrison and Veerappa, 2014a). Sin embargo, algunas discusiones sobre el código realizadas a través del correo electrónico rápidamente se convierten en una tarea laboriosa y lenta (Young and Terashima, 2008).

2.6.1.3.2 Videoconferencias

También se identificaron las videoconferencias como una práctica popular para compartir AK en AGSD (Esbensen et al., 2015; Korkala and Maurer, 2014; Paasivaara and Lassenius, 2014; Schmidt and Meures, 2016; Sharp et al., 2012; Šmite et al., 2010b; Van Hillegersberg et al., 2011; Young and Terashima, 2008), ya que se considera que es la forma más rica de compartir AK (cuando las zonas horarias lo permiten) debido a la comunicación fluida y las funciones que ofrecen las herramientas de videoconferencia (e.g. compartir pantalla, tableros virtuales, etc.). Esta práctica se utiliza para celebrar diferentes tipos de reuniones, incluyendo: reuniones de inicio, diarias, de diseño de arquitectura, de entrenamiento, sesiones de comunidades de práctica y reuniones de aclaración sobre cualquier tema. Sin embargo, las video conferencias no son preferidas por los desarrolladores ya que pueden resaltarse las distancias culturales y lingüísticas con sus contrapartes. Un ejemplo de esta actitud de los desarrolladores fue la poca adopción que tuvieron aplicaciones de ventanas virtuales en empresas de AGSD (Esbensen et al., 2015)

Cabe enfatizar una reunión basada en videoconferencia, es decir, las sesiones de comunidades de práctica (CoP por sus siglas en inglés) o comunidades de conocimiento. Una CoP es un "grupo de personas que comparten una preocupación, una serie de problemas o una pasión sobre un tema, y que profundizan sus conocimientos y experiencia en esta área interactuando de forma continua" (Wenger et al., 2002). Las CoP se han identificado como una práctica creciente para compartir AK en AGSD entre equipos locales y remotos (Korkala and Maurer, 2014; Paasivaara and Lassenius, 2014; Schmidt and Meures, 2016). Se han identificado cuatro propósitos diferentes de las CoP (Paasivaara and Lassenius, 2014): (1)

intercambio de conocimiento y aprendizaje (conocido también como "enseñanza por expertos" (Schmidt and Meures, 2016)), (2) coordinación, (3) trabajo técnico (conocido también como "llamadas técnicas" (Korkala and Maurer, 2014)) y (4) desarrollo organizacional. Las CoP centradas en el trabajo técnico, el intercambio de conocimientos y el aprendizaje son las más comunes (Dingsøyr, 2009; Wenger et al., 2002), y son prominentes en el intercambio de AK y para tomar decisiones de diseño entre miembros locales y remotos (Paasivaara and Lassenius, 2014). Sin embargo, las distancias temporales y físicas limitan la implementación de las CoP (Moe et al., 2014) cuando no existe un soporte tecnológico adecuado, ni los equipos con horarios que coincidan.

2.6.1.3.3 Radiadores de información

Otro método de comunicación son los radiadores de información (concepto introducido por Cockburn (Cockburn, 2002)), es cual es un gráfico (físico o virtual) que los desarrolladores utilizan para conocer el proyecto de un vistazo. Debe ser accesible para mostrar información valiosa a los miembros del equipo, y debe ser fácil de actualizar y de entender (Paredes et al., 2014). Un radiador de información se puede implementar en un tablero inteligente o en una pantalla electrónica, y es útil para apoyar reuniones técnicas con contrapartes remotas (Esbensen et al., 2015; Sharp et al., 2012; Yagüe et al., 2016). Después de la reunión técnica, el conocimiento generado en la reunión puede mantenerse en el radiador, de modo que los desarrolladores pudieran consultarlo para conocer los problemas técnicos tratados.

2.6.2 Impactos de los enfoques de manejo de AK en el AGSD

Al realizar la revisión de literatura se observó que solo ocho de los artículos seleccionados reportan los impactos o beneficios que se obtuvieron al aplicar un enfoque determinado para el manejo del AK. A continuación, se muestran los beneficios y/o impactos que reportan los trabajos seleccionados, organizados por enfoque.

- **Software de apoyo al desarrollo**. Los usuarios estaban más dispuestos a tener reuniones diarias para revisar la arquitectura general del proyecto respaldado por Archinotes (Urrego et al., 2014).
- **Artefactos auto-generados**. La precisión de coincidencia entre el correo electrónico y las historias de usuario del mecanismo de etiquetado automático fue aproximadamente del 70%, sin embargo se requiere mejorar para un ámbito industrial (Sohan et al., 2010).

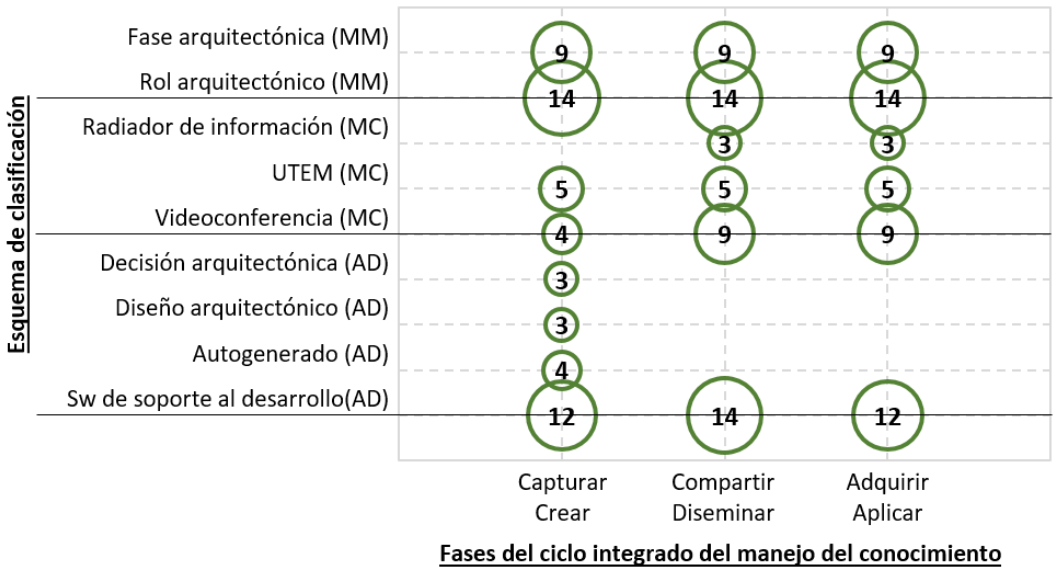
WRCC fue útil para agilizar la trasferencia de conocimiento de un producto de software a los nuevos miembros del equipo, para visualizar la complejidad del mismo y comprender el impacto de los cambios arquitectónicos (O'Reilly et al., 2005).

- **Artefactos ligeros para diseño arquitectónico**. La necesidad de refactorización de código se redujo, ya que la arquitectura se diseñó al inicio del proyecto y no en el transcurso, lo cual permitió codificar bajo una guía mejor establecida y por lo tanto ahorrar tiempo de desarrollo (Pérez et al., 2010). La ontología fue útil para razonar acerca de qué AK sería conveniente incluir en la documentación del diseño arquitectónico (de Graaf et al., 2014).
- **Artefactos ligeros para las decisiones arquitectónicas**. Fue fácil y rápido localizar el impacto de las decisiones de diseño así como las razones que llevaron a tomar dichas decisiones (Díaz et al., 2014).
- **Videoconferencia**. Las comunidades de práctica mantuvieron informados a los desarrolladores sobre lo que sucede en la organización en general en términos de AK (Paasivaara and Lassenius, 2014).
- **Fase arquitectónica**. Las historias de usuario se derivaron con base en una arquitectura lógica, lo que permitió alinearlos adecuadamente en un corto tiempo (Costa et al., 2014).

Como se puede observar, los impactos y beneficios reportados se pueden resumir en un aumento de la claridad acerca de aspectos arquitectónicos de los proyectos, lo cual naturalmente conduce a una disminución del tiempo de desarrollo, que a su vez se puede traducir en menores egresos para la empresa de desarrollo, así como en mayores ingresos por lanzar más rápido los productos de software al mercado.

2.6.3 Fases del manejo del conocimiento soportadas por los enfoques de manejo de AK en el AGSD

Como parte del trabajo de análisis de los resultados de esta revisión de literatura, se relacionó cada enfoque obtenido con una fase del ciclo de manejo de conocimiento integrado, obteniendo la Figura 2.2. A primera vista se observa una distribución casi uniforme en cuanto al soporte a cada fase del ciclo del manejo del conocimiento: Capturar/Crear - 34%, Compartir/Diseminar - 34%, y Adquirir/Aplicar - 32%. También se observa que en la documentación predominan los artefactos usados para registrar el AK generado durante el desarrollo (ver Figura 2.2, fase de Capturar/Crear). En el caso del software de soporte al



**(MM = Modificación metodológica, MC = Métodos de Comunicación, AD = Artefactos de Documentación)*
Todos los valores representan cantidad de artículos.

Figura 2.2. Distribución de enfoques de trabajos por fase del ciclo del manejo del conocimiento y por categoría.

desarrollo, las fases de Compartir/Diseminar y Adquirir/Aplicar están presentes ya que las herramientas como wikis, repositorios y herramientas de groupware también soportan dichas fases (el mismo caso que para los UTEM). El enfoque de videoconferencias tiene cuatro artículos que soportan la fase de Capturar/Crear, ya que estos documentos especifican que los diseños arquitectónicos se crearon en video-llamadas. Finalmente, los enfoques donde se agrega al desarrollo ágil un rol de arquitecto y una fase de arquitectura tienen el objetivo de crear, capturar, compartir y aplicar el AK. Cabe señalar que en la Tabla 2.3, un solo documento puede estar relacionado con más de un enfoque, por lo tanto, los números en la Figura 2.2 no coinciden con la cantidad de trabajos seleccionados.

Parecería que el soporte al manejo del AK en AGSD está completo, debido a la distribución uniforme de los artículos seleccionados en las fases del ciclo integrado de administración del conocimiento; sin embargo, existen detalles ocultos, particularmente en la fase de Capturar/Crear. Casi todos los enfoques encontrados contemplan la fase de Capturar/Crear, pero hay casos en los que solo se crea AK y otros en los que sólo se captura, sin especificar cómo se codifica el conocimiento, es decir, la acción de crear conocimiento no implica que este será capturado. Con el fin de analizar con detalle los resultados de la fase Capturar/Crear, se clasificó la manera de codificar conocimiento que reporta cada uno de los

artículos que soportan esta fase de acuerdo con los tres estados propuestos por (Nonaka and Takeuchi, 1995):

- **Tácito**: El conocimiento está en la mente de las partes interesadas y generalmente es difícil de hacerlo explícito en algún documento.
- **Documentado**: El conocimiento se encuentra codificado de manera informal o improvisada en algún medio físico o virtual.
- **Formalizado**: El conocimiento se encuentra codificado en una estructura formal, es decir, apegada en algún estándar, ya sea de la empresa o internacional.

Si se observa en la Figura 2.3, solo el 6% de los artículos reportan una manera formal de codificación de AK. Los artículos que forman parte de este 6% corresponden a los siguientes enfoques: artefactos ligeros para las decisiones arquitectónicas, para el diseño de arquitecturas, artefactos de documentación auto-generado y el software de soporte al desarrollo. Además, en la misma Figura 2.3 se observa que el 13% de los artículos reporta una manera volátil de mantener el AK, es decir, de manera tácita (4%) o de manera documentada (9%). Se considera volátil tener este conocimiento de manera documentada porque este puede perder significado a lo largo del tiempo o en otro contexto al no estar apegado a un estándar con una semántica fija, es decir, el conocimiento podría vaporizarse. El 15% restante de los artículos correspondientes a la fase Capturar/Crear no especifican cómo se codifica el AK.

Con base en esta evidencia, no se puede asegurar que la fase de Capturar/Crear del ciclo integrado del manejo del conocimiento está completamente soportada por los enfoques obtenidos de esta revisión. Incluso, tampoco se pudiera garantizar que el resto de las fases del ciclo estén totalmente soportadas, debido a la volatilidad del conocimiento que presentan algunos de los enfoques para capturar el AK, lo cual podría causar su vaporización al compartirlo sucesivamente en un ambiente AGSD. Finalmente, para cerrar esta sección, se puede afirmar que el manejo del AK en AGSD todavía es un campo abierto para explorar e investigar mejores formas de trabajar con este tipo de conocimiento en entornos ágiles y distribuidos.

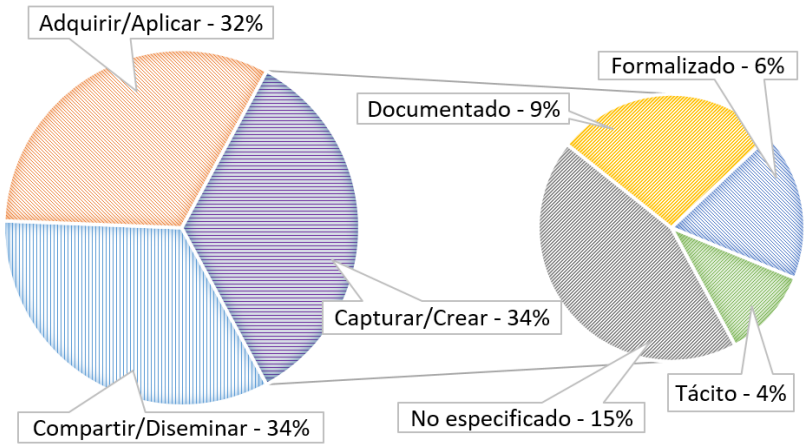


Figura 2.3. Clasificación de artículos según la fase del ciclo integrado del manejo del conocimiento que soportan.

2.6.4 Reflexiones sobre el manejo del AK en el AGSD

Al realizar esta revisión de literatura se resaltó que los enfoques basados en modificaciones metodológicas son los más usados para el manejo del AK en AGSD (54% de los trabajos seleccionados). Sin embargo, el enfoque de agregar el rol de arquitecto sólo se encontró aplicado en empresas medianas o grandes (>100 empleados según el Glosario de TI de Gartner[10]), lo cual significa que para el resto de empresas podría ser prohibitivo en cuestiones de estructura y presupuesto.

Además, destaca que sólo el 6% de los artículos seleccionados reporta una manera formal de codificar AK, es decir, estructurada o apegada a algún estándar. Por lo tanto, se puede afirmar que la mayoría del conocimiento que se comparte y adquiere en ambientes de AGSD está estado tácito o en estado documentado, lo cual lo hace propenso a la vaporización. A pesar de ello se destaca que los beneficios reportados al implementar el manejo del AK en AGSD se relacionan directamente con reducción de tiempos de desarrollo y por lo tanto en el aumento de ingresos a las empresas.

A partir del análisis realizado a los resultados del mapeo de literatura presentado, se identificó que las bitácoras de UTEM podría ser un nicho interesante a explorar, ya que estos son herramientas de uso cotidiano para los desarrolladores, mediante las cuales se comparten AK. Por lo tanto, sería una manera discreta aprovechar el conocimiento articulado en estos medios.

[10] https://www.gartner.com/it-glossary/smbs-small-and-midsize-businesses

Analizando los nueve enfoques encontrados en esta revisión se pueden destacar los puntos críticos para implementar un manejo de AK eficiente en AGSD, que son los siguientes: (1) agregar un rol o una fase para abordar los problemas arquitectónicos, (2) usar artefactos de documentación ligeros para codificar AK, (3) aprovechar el AK tácito que ser articula en los UTEM, ya sea para autogenerar artefactos de documentación o simplemente haciéndolo disponible para los interesados, (4) diseminar artefactos autogenerados y/o ligeros a través de wikis, repositorios o herramientas de groupware, son opciones de búsqueda adaptadas a las necesidades del AK.

Por otro lado, se encontró que las estrategias usadas en ingeniería de software para el manejo del conocimiento en general y las estrategias para el manejo del AK son similares. Sin embargo cada estrategia enfatiza diferentes áreas dependiendo de paradigma de desarrollo, es decir, ya sea AGSD, GSD o desarrollo ágil. Por ejemplo, en AGSD, las estrategias de AK deben enfatizar las maneras automáticas o livianas de codificar el conocimiento, a fin de dedicar menos tiempo a las actividades de documentación.

Finalmente, con la evidencia presentada en este capítulo y particularmente en el mapeo de literatura, se puede afirmar que el manejo del AK sigue siendo un tema abierto en ambientes AGSD.

2.7. Resumen del capítulo

En este capítulo se presentó una descripción de los principales conceptos teóricos que intervienen en el desarrollo de esta tesis y se discutieron algunos conceptos para los cuales la literatura ofrece más de una definición o bien no se cuenta con una definición concreta y acordada. Además, se presentó una revisión de literatura donde se identificaron y se describieron nueve enfoques para administrar AK en equipos de AGSD. A continuación, se presentan los principales resultados de este capítulo.

- Se presentaron los paradigmas de desarrollo de software global y el ágil, sus beneficios potenciales y los retos que están siendo atendidos en la literatura, para dar pie a la definición del concepto de desarrollo ágil y global de software; en donde se hizo hincapié en el antagonismo intrínseco que ocurre al combinar el paradigma ágil y el global, y también se describieron los beneficios y retos actuales de este paradigma.

- Se describió la disciplina del manejo de conocimiento, pasando por la descripción de los conocimientos tácito y explícito, y del ciclo integrado del conocimiento. Todo ello para dar pie a la descripción del manejo del AK, donde se elaboró una definición general del concepto de arquitectura de software, el cual fue la base para describir el concepto de AK. Además, se presentaron los beneficios potenciales al implementar el manejo del AK y los retos a los que se enfrentaría cualquier empresa durante su implementación.
- Se presentaron las estrategias que se aplican tanto en el GSD como en el desarrollo ágil de software para implementar el manejo del AK, así como los retos reportados en la literatura al hacer dicha implementación.
- Se presentó un mapeo sistemático de literatura donde se identificaron y se describieron nueve enfoques para administrar AK en equipos de AGSD, los cuales se pueden agrupar como: documentación basada en artefactos, enfoques basados en la comunicación y enfoques basados en metodología; donde los enfoques de documentación eran prominentes (45% de los artículos seleccionados).
- Se encontró que estos enfoques respaldan uniformemente las tres fases del ciclo integrado del manejo del conocimiento: Capturar/Crear (34%), Compartir/Diseminar (34%) y Adquirir/Aplicar (32%), pero sólo el 6% de los artículos reporta una manera formal de codificar AK, es decir, estructurada o apegada a algún estándar.

En el siguiente capítulo se presenta un estudio empírico para tener un mejor entendimiento sobre la compartición y articulación de AK particularmente mediante UTEM por parte de desarrolladores practicantes del AGSD. De esta manera se busca determinar cómo explotar el conocimiento que podría encontrarse en las bitácoras de estos medios.

Capítulo 3.
Articulación del conocimiento arquitectónico en AGSD

Según (Clerc and Lago, 2011; H.-Christian Estler et al., 2012), en AGSD se prefiere transmitir el AK mediante interacciones frecuentes entre sitios a través de UTEM (mensajeros instantáneos, correos electrónicos, foros, etc.), es decir, este tipo de conocimiento se articula en estos medios de comunicación. Además, en el mapeo sistemático de literatura presentado en el capítulo anterior (Gilberto Borrego et al., 2017), el manejo de AK basado en la comunicación mediante UTEM es parte de los nueve enfoques identificados en dicha revisión. Este enfoque realmente ha sido poco explotado, sólo un 11% de los artículos seleccionados tocan el tema, a pesar del potencial que presenta. El potencial de este enfoque se debe a que los UTEM dejan un registro de texto de la información transmitida, es decir un registro no estructurado de AK compartido del proyecto. Sin embargo la literatura no reporta específicamente que aspectos del AK se articulan en los UTEM, ni como sucede exactamente la acción de compartir este tipo de conocimiento en UTEM. En este capítulo se presenta un estudio empírico para entender la articulación de AK mediante UTEM en equipos AGSD. Esto con el fin de tener un mejor entendimiento del ambiente de trabajo en AGSD con respecto al AK y así poder proponer eventualmente una solución que ayude a sobrellevar las diferencias del GSD y el desarrollo ágil que afectan al manejo de este tipo de conocimiento.

3.5. Planteamiento de estudio de observación en sitio.

En esta sección se describe un estudio empírico en el que participaron cuatro empresas mexicanas de AGSD. El estudio consistió en una fase de exploración donde se analizaron los hallazgos de la fase de observación con los participantes de las cuatro compañías.

3.5.1 Objetivo y preguntas de investigación

El objetivo planteado para este estudio fue comprender la articulación de AK en UTEM entre miembros locales y remotos en equipos de AGSD, para tener elementos que respalden tecnológicamente el manejo de AK en equipos distribuidos y ágiles. De este objetivo planteado se derivaron las siguientes preguntas de investigación:

- **RQ1**: ¿Qué elementos están involucrados en la articulación de AK en los equipos de AGSD?
- **RQ2**: ¿Cuáles son los temas de AK más frecuentes en las interacciones de UTEM en los equipos de AGSD según la percepción de los miembros del equipo?
- **RQ3**: ¿Cómo se relacionan la frecuencia percibida y la importancia percibida de los temas de AK en las interacciones de UTEM entre los miembros del equipo de AGSD?
- **RQ4**: ¿Cómo están relacionan los temas de AK en UTEM y las interacciones cara a cara?

3.5.2 Participantes

Los participantes del estudio fueron 20 ingenieros de software que trabajan en cuatro diferentes empresas mexicanas que practican el AGSD (A = EMCOR[11] , B = Tiempo Development[12], C = Nearsoft[13], D = Softtek[14] - ver Tabla 3.1). Los participantes se mantienen en contacto con contrapartes remotos a través de UTEM.

Las empresas participantes tienen proyectos con otros en los EE.UU., España, Ecuador, Argentina, Colombia, Perú, Guatemala y Honduras. Todas las compañías aplican Scrum como base metodológica para administrar proyectos. Sin embargo, con respecto a la gestión de requisitos, la empresa B y la empresa D sólo utilizan historias de usuarios; la compañía A usa una combinación de historias y casos de uso; y la empresa C recibe documentos de requerimientos de sus clientes, pero deben ser refinados antes de iniciar el proceso de desarrollo. En cuanto al diseño y desarrollo de software, las cuatro compañías comparten las siguientes prácticas: desarrollo de prototipos y modelado ágil. La compañía B

Tabla 3.1. Características de los participantes.

Compañía	Número de participantes	Proyectos en los que participan	Edad		Experiencia			
					Ingeniería de software		AGSD	
			Prom	*Med*	*Prom*	*Med*	*Prom*	*Med*
A. EMCOR	11	6	23.7	23	1.5	1.5	0.9	0.4
B. Tiempo Development	4	3	36.3	35.5	13.5	13	5.3	5.5
C. Nearsoft	3	3	26.3	27	3	3	1.6	1
D. Softtek	2	2	28.5	28.5	7	7	4.5	4.5

[11] http://emcor.mx/
[12] https://www.tiempodev.com
[13] https://nearsoft.com/
[14] https://www.softtek.com/

y la compañía C también aplican la programación por pares y prácticas ágiles de diseño participativo.

Con respecto a la distribución del trabajo, los participantes de la empresa B y la empresa D forman parte de equipos de desarrollo en los que participan desarrolladores remotos. Por lo tanto, estas empresas comparten la responsabilidad en todo el ciclo de desarrollo entre los miembros locales y remotos. Además, sólo tienen reuniones de diseño cara a cara en caso de un proyecto complejo, es decir, hacen visitas a la ubicación remota. En el caso de la empresa A, los participantes refinan los requerimientos con sus compañeros de equipo remotos, y tienen el control de producto y su desarrollo. Además, los participantes de la empresa A realizan pruebas de sistema y desarrollan documentos de implementación. Los compañeros remotos son los responsables de las pruebas de usuario y la implementación del software. Finalmente, los participantes de la empresa C reciben documentos de requerimientos para luego desarrollar, probar y desplegar el producto de software.

3.5.3 Recopilación y análisis de datos

La recolección de datos se dividió en dos fases: observación y exploración. La fase de observación se realizó solo en la empresa A de manera local, dado que fue la única compañía que permitió monitorear y analizar sus interacciones UTEM. Se realizaron observaciones durante tres semanas, cinco días a la semana y ocho horas al día. En esta fase se tuvieron once participantes quienes trabajaban en seis proyectos en total, donde se interactuaba con miembros de España y América Latina.

La intención principal de la fase de observación fue registrar y analizar las interacciones UTEM entre sitios. Para este propósito, se solicitó a los participantes de la compañía A el UTEM más común que utilizan para interactuar entre miembros locales y remotos. Se incluyeron los siguientes medios: mensajero electrónico (Microsoft Lync[15]), correo electrónico (Microsoft Exchange[16]) y foros de discusión (parte de la herramienta Target Process[17]). Las interacciones a través de estos medios fueron capturadas con una herramienta de software desarrollada a medida expresamente para ello.

[15] https://products.office.com/en-us/microsoft-lync-2013
[16] https://products.office.com/es-mx/exchange/email
[17] https://www.targetprocess.com/

Se decidió observar las interacciones cara a cara entre los miembros co-localizados, y analizar los artefactos de documentación relacionados con aspectos de arquitectura de software, para encontrar una relación entre ellos y las interacciones mediante UTEM. Las interacciones cara a cara se registraron en un formato, que incluye: hora de inicio y finalización, medios de apoyo (tableta, cuaderno, computadora portátil, pizarra, etc.), nombre del participante y una breve descripción del tema de interacción. Cabe aclarar que no se interactuó con los participantes durante la observación.

Con el fin de tener un modelo de referencia para identificar cuándo se discutió un tema arquitectónico, ya sea en interacciones o documentos, se utilizó el modelo arquitectónico 4 + 1 vistas (Kruchten, 1995). Este modelo establece cinco vistas arquitectónicas, que se describen a continuación.

- **Vista lógica**. Se refiere principalmente a la funcionalidad y su descomposición en un conjunto de abstracciones de claves, tomadas del dominio del problema, en forma de clases aprovechando los principios de abstracción, encapsulación y herencia. Esta descomposición no es sólo para el análisis funcional, sino que también sirve para identificar mecanismos y elementos de diseño comunes en las distintas partes del sistema. Los diagramas UML[18] más comunes utilizados para representar esta vista son los diagramas de clase y diagramas de secuencia, ambos en un alto nivel de abstracción.
- **Vista de implementación**: Se centra en la organización real de componentes, subsistemas, módulos de software en el entorno de desarrollo. Los diagramas UML más comunes usados para representar esta vista son los diagramas de componentes, los de paquetes y los de clases (a bajo nivel de abstracción).
- **Vista de procesos**. Tiene como base requisitos no funcionales, como el rendimiento y la disponibilidad. Se incluyen temas de concurrencia y distribución, de integridad del sistema, de tolerancia a fallas, y cómo las principales abstracciones de la vista lógica encajan dentro de la arquitectura del proceso. El diagrama de UML comúnmente usado en esta vista es el diagrama de actividad, para representar las interacciones de componentes y clases al nivel del proceso de negocios.

[18] http://www.omg.org/spec/UML

- **Vista de despliegue**: Se centra principalmente en requisitos no funcionales como disponibilidad, fiabilidad (tolerancia a fallos), rendimiento (*throughput*) y escalabilidad. Además, esta vista representa la topología de componentes de software en la capa física y las conexiones físicas entre estos componentes. El diagrama de UML comúnmente usado en esta vista es el diagrama de despliegue, con el que se representan los aspectos de hardware y configuración que ella conlleva.
- **Vista de escenarios**. Los escenarios son en cierto sentido una abstracción de los requisitos más importantes, descrito utilizando un conjunto de casos de uso, que se utilizan para identificar y validar el diseño de arquitectura. Los escenarios y casos de uso describen secuencias de interacciones entre objetos, y entre procesos, y son un punto de partida para ejecutar pruebas de un prototipo arquitectónico. Esta vista es también considera requerimientos e historias de usuarios.

Pasando a la fase de exploración, se analizaron los hallazgos de la fase de observación en la compañía A junto con los participantes de las cuatro compañías. Se realizaron cuatro sesiones de grupos focales (una por cada compañía), donde se presentaron y discutieron los hallazgos, los cuales se pueden resumir en (1) interacciones UTEM, (2) una lista de artefactos utilizados para registrar asuntos arquitectónicos, y (3) las intenciones principales de las interacciones cara a cara en relación con la arquitectura de software. Todas las sesiones fueron grabadas en audio para su posterior análisis.

Para analizar las interacciones de UTEM capturadas, se solicitó a los participantes de la compañía A una lista de palabras clave usadas en cada proyecto para referirse a cuestiones arquitectónicas, tanto formal como informalmente. Esta lista se usó para filtrar las interacciones UTEM a fin de mantener sólo las interacciones relacionadas con AK. Una vez filtradas, las interacciones se leyeron una por una para identificar los principales temas arquitectónicos tomando como base el modelo 4+1 vistas antes descrito.

Para analizar las interacciones cara a cara, se examinó cada entrada en el registro para obtener las intenciones de contacto principal entre los miembros del equipo. Finalmente, se transcribieron los audios de los grupos focales para aplicar teoría fundamentada (Martin and Turner, 1986) para comprender la articulación de AK a través de UTEM en los equipos de AGSD.

3.6. Articulación del AK en UTEM

Los resultados de este estudio se presentan en tres subsecciones: Entendimiento en general de la articulación del AK, categorización de las interacciones mediante UTEM y categorización de las interacciones cara a cara.

3.6.1 Generalidades de la articulación del AK

Durante la fase de observación se registraron 45 interacciones cara a cara sobre diferentes aspectos de arquitectura de software. Además, se registraron y analizaron 216 interacciones UTEM, donde 145 fueron de conversaciones de mensajería instantánea, 51 de mensajes de correo electrónico y 20 de comentarios de foros de discusión. Cabe señalar que MS Lync define una conversación como un grupo de mensajes enviados mientras que una ventana de IM permanece abierta.

Con base en estas interacciones registradas en la compañía A, y en las sesiones grupales con todas las compañías participantes, se desarrolló una ontología (ver Figura 3.1) que representa el entendimiento obtenido sobre la articulación del AK mediante UTEM en equipos de AGSD, junto con todos los elementos relacionados que fue posible identificar (cumpliendo con esto la pregunta de investigación RQ1). El modelo resultante tiene dos grandes partes: aspectos de comunicación y aspectos de conocimiento articulado, los cuales se describe en las siguientes secciones.

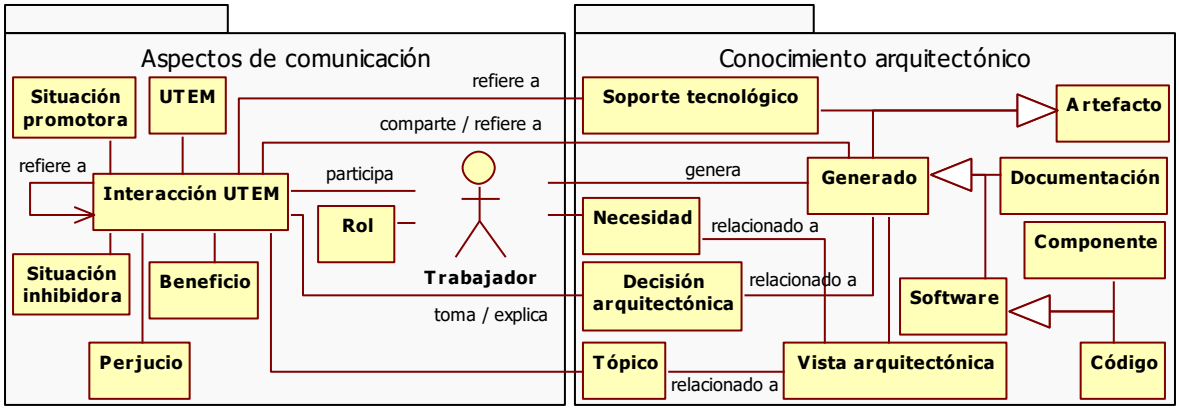


Figura 3.1. Ontología representando los aspectos involucrados en la articulación del AK mediante UTEM en equipos AGSD.

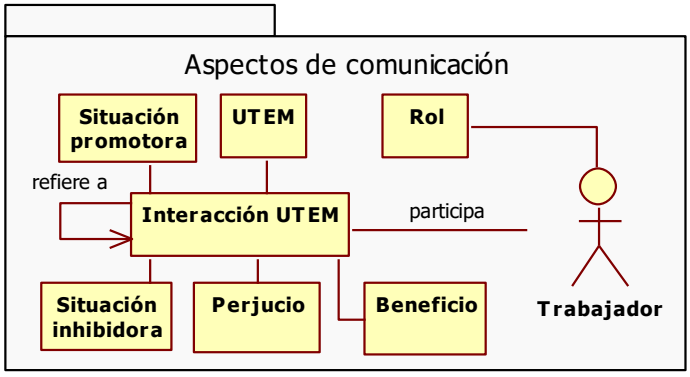


Figura 3.2. Aspectos de comunicación de ontología de los aspectos involucrados en la articulación del AK a través de UTEM en equipos AGSD.

3.6.1.1 Aspectos de comunicación cuando se comparte AK por UTEM

En cuanto a los aspectos de comunicación (ver Figura 3.2), se considera que el elemento principal es la interacción mediante UTEM la cual está relacionada con el intercambio de AK entre los trabajadores de AGSD, quienes desempeñan un rol específico en el equipo (desarrollador, probador, analista, propietario del producto, cliente, etc.). Se observó que los trabajadores de AGSD con frecuencia se refieren a una interacción pasada (conversación de mensajería instantánea, mensaje de correo electrónico, entrada al foro, etc.), con el fin de argumentar o dar contexto a la discusión actual.

También se identificó que las interacciones mediante UTEM relacionadas con AK son promovidas o desencadenadas por diferentes situaciones. Algunas de estas situaciones promotoras son las siguientes: (1) cuanto se pide a alguien (a través de UTEM) información específica para evitar perder tiempo buscando documentos que contengan AK; (2) cuando un equipo recibe un nuevo miembro y el equipo remoto lo pone en contexto en cuanto a AK; (3) cuando se tiene documentación arquitectónica pobre o simplemente ésta es inexistente, el AK se solicita a través de UTEM); (4) cuando existen dudas sobre sistemas heredados y/o componentes con códigos complejos, los expertos en el tema son consultados a través de UTEM.

En contraste, se encontraron situaciones que inhiben las interacciones referentes al AK. Algunas de estas situaciones son las siguientes: (1) referir a una persona a los repositorios de AK, en vez de extender una interacción por UTEM para compartir dicho conocimiento; (2) cuando los trabajadores de AGSD tienen una estricta adherencia a metodologías bien definidas, lo cual disminuye la posibilidad de malentendidos y reduce la

frecuencia de preguntas; (3) el seguimiento de las mejores prácticas de documentación como documentos concisos y convenciones de códigos, lo cual lleva a la auto-documentación del código al respetar dichas convenciones; (4) cuando se ha preguntado repetidamente por UTEM a una misma persona provoca que se eviten futuras interacciones con dicha persona ya que en ocasiones el trabajador que pregunta puede sentirse avergonzado.

Además, se identificaron los beneficios y los perjuicios de establecer interacciones relacionadas con AK a través de UTEM. Algunos de los beneficios más relevantes que se identificados fueron:

- El refinamiento de los requerimientos, ya que al estar aclarando dudas se pueden encontrar errores de definición de requerimientos, lo cual impacta directamente en el tiempo de desarrollo y en la calidad del producto final;
- La transferencia de conocimiento entre los miembros del equipo, el simple hecho de interactuar compartiendo AK hace que éste se disperse en todo el equipo, con lo cual se disminuye la necesidad de consultar repositorios de conocimiento o documentos;
- La disminución del número de errores de software, ya que el conocimiento amplio de cuestiones arquitectónicas propicia que se desarrollen productos de mejor calidad.

En cuanto a los perjuicios más relevantes que se identificaron se tienen los siguientes:

- Pérdida de tiempo, al estar contestando preguntas que se evitaría si existiera documentación arquitectónica, es decir, si no hubiera tanta deuda de documentación;
- Sentido de falta de fiabilidad del AK, ya que el conocimiento proviene de medios informales, como las entradas de UTEM y no de documentos consensados.

3.6.1.2 Aspectos del AK que se comparte por UTEM

Con respecto a los aspectos compartidos de AK (ver Figura 3.3), se observó que diversos artefactos están involucrados durante las interacciones mediante UTEM, ya sean generados como resultado del mismo ciclo de desarrollo de software, o artefactos de soporte tecnológico (ambientes integrados de desarrollo, rastreadores de errores, administradores de tareas, repositorios, etc.). Con respecto a los artefactos generados, estos son compartidos o referidos en las interacciones de UTEM por los trabajadores de AGSD. Este tipo de artefactos incluyen: documentación (diagramas formales o ad-hoc, especificaciones, programaciones, etc.) y software, ya sean componentes (ejecutables, bibliotecas, marcos de trabajo, etc.) o

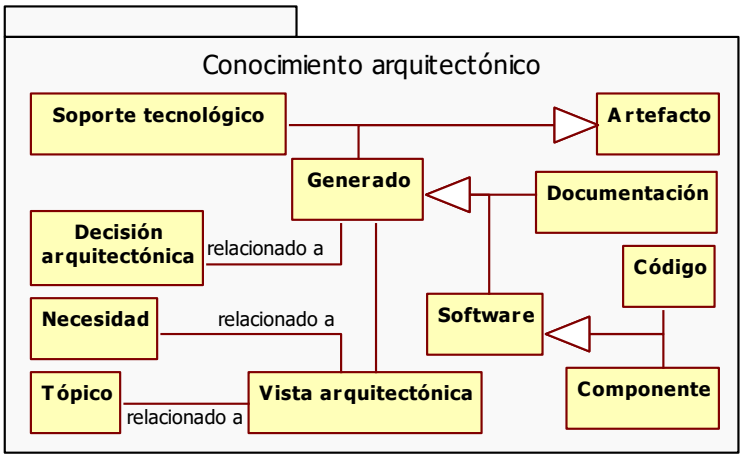


Figura 3.3. Aspectos de AK compartidos mediante UTEM por equipos de AGSD.

fragmentos de código fuente y/o archivos. Todos estos artefactos generados están relacionados con una o varias vistas arquitectónicas en particular del modelo de 4+1 vistas.

Cabe destacar que en las empresas participantes, la mayoría de los documentos arquitectónicos que se encontraron estaban relacionados con la especificación de requerimientos (en forma de historias de usuario) y la documentación técnica estaba casi ausente. Estos documentos generalmente se realizan únicamente bajo demanda.

Además, se observó que cuando un artefacto generado se comparte o es referido, se le asocia una explicación, donde ocasionalmente decisiones arquitectónicas y las razones que llevaron a ellas están presentes en dicha explicación. Por ejemplo: cuando se comparte un artefacto de software, el remitente normalmente explica cómo funciona y por qué tiene que funcionar de esa manera. Relacionado con lo anterior, las decisiones arquitectónicas podrían tomarse a través de UTEM, dependiendo de la distribución del trabajo del equipo de AGSD. Por ejemplo, cuando todos los miembros de un equipo virtual son desarrolladores es más probable que este tipo de decisiones se tomen utilizando UTEM (decisiones sobre patrones y estilos arquitectónicos, tecnologías, protocolos, etc.), ya que esta es la vía preferida para interactuar.

Aspectos adicionales de AK que se identificaron fueron necesidades de conocimiento sobre una vista arquitectónica particular. Por ejemplo, los participantes manifestaron las siguientes necesidades de conocimiento: acceso al diseño arquitectónico, en forma de diagramas y documentos (los que existan); acceso al correo electrónico y al historial de mensajería instantánea para buscar AK, y saber a quién preguntar cuestiones arquitectónicas

específicas. Finalmente, se identificaron diferentes temas arquitectónicos en las interacciones mediante UTEM (e.g. codificación, requerimientos, bases de datos, pruebas, etc.) que también están relacionados con diferentes vistas arquitectónicas.

3.6.2 Categorización de las interacciones mediante UTEM referentes a AK

Una vez que se logró una comprensión preliminar de la articulación del AK en UTEM, y con el objetivo de responder con la pregunta de investigación RQ2, se analizaron las interacciones UTEM capturadas para categorizar los temas arquitectónicos que se discuten en estos medios (los mismos temas identificados como parte del modelo presentado en la Figura 3.3). Se obtuvieron 11 categorías de interacción en UTEM (ver Tabla 3.2), las cuales se relacionaron con diferentes vistas arquitectónicas del modelo de 4 + 1 vistas. Por ejemplo, la categoría 3 ("Solicitud de información sobre el flujo interno del sistema") se relacionó con las vistas de implementación y procesos porque sus interacciones se referían a temas de codificación y componentes. Cabe aclarar la relación de la categoría 8 ("Propuestas de proyectos nuevos") y la categoría 10 ("Aclaración de temas de bases de datos") con las vistas arquitectónicas. La categoría 8 se relacionó con la vista de escenarios ya que las interacciones eran sobre la funcionalidad general para nuevos proyectos. La categoría 10 se relacionó con las vistas de

Tabla 3.2. Categorías de interacciones en UTEM relacionadas con las vistas arquitectónicas del modelo arquitectónico 4 + 1.

Categorías	**Vistas Arquitectónicas***				
	E	***L***	***I***	***P***	***D***
C1. Segmentos de código compartido			●		
C2. Configuración de entornos de prueba/despliegue					●
C3. Solicitud de información sobre el flujo interno del sistema			●	●	
C4. Información sobre clases (sin código compartido)			●		
C5. Referencia a reglas de negocio u operaciones internas explicadas por un tercero	●		●	●	
C6. Explicación técnica sobre la solución de errores			●		
C7. Aclaración de reglas de negocio, características o historias de usuarios	●				
C8. Propuestas de proyectos nuevos	●				
C9. Dudas a nivel de usuario sobre el funcionamiento de un producto terminado	●				
C10. Aclaración de temas de base de datos		●	●		●
C11. Solicitud de información sobre la estructuración de datos para una prueba	●				

**(E = Escenarios, L = Lógica, I = implementación, p = procesos, d = despliegue)*

implementación y de despliegue, porque sus interacciones se centraron en códigos de consultas, configuración e instalación de bases de datos.

Uno de los aspectos más destacados de las interacciones fue que los participantes comentaron que la mayoría de ellos repetía información, ya sea por equivocación o porque el mensaje no fue bien entendido desde el principio, provocando que se volvieran a discutir algunos aspectos a través de UTEM. En este sentido, uno de los participantes en los grupos focales mencionó: *"... puede causar una mala impresión a la otra persona si continuamente haces preguntas..."*, refiriéndose a que pudiera pensarse que si se pregunta mucho es porque la persona no tiene una buena capacidad de entendimiento.

Otro aspecto importante es que la vista escenarios y la vista de implementación comprenden muchas de las categorías mostradas en la Tabla 3.2, es decir, predominan los temas relacionados con requerimientos y con cuestiones de codificación. También es de destacar que sólo la categoría 10 está relacionada con la vista lógica. Esto puede significar que se debaten pocas cuestiones sobre elementos conceptuales de los sistemas en las interacciones UTEM; en este caso el tema predominante fue sobre esquemas de bases de datos.

Con el fin de identificar cuán frecuentes y cuán importantes son las categorías identificadas, se realizó una evaluación en la que todos los participantes expresaron que tan frecuente es que se comparta información acerca de cada categoría, así como la importancia de que ello suceda. Se utilizó una escala Likert estructurada como se indica en la Tabla 3.3 para que los participantes expresaran sus percepciones. Al final se agruparon los resultados de cada categoría por vista arquitectónica y por compañía, y se obtuvieron las medianas para cada grupo.

Tabla 3.3. Estructuración de valores Likert para evaluar frecuencia e importancia percibida de las categorías de AK compartido en UTEM.

Aspectos a evaluar	**Valores Likert**				
	1	***2***	***3***	***4***	***5***
Frecuencia	Muy poco frecuente	Poco frecuente	Neutral	Frecuente	Muy frecuente
Importancia	Muy poco importante	Poco importante	Neutral	Importante	Muy importante

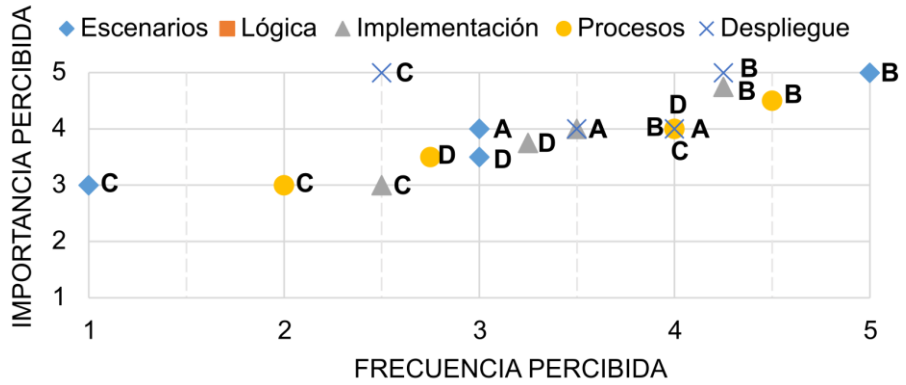
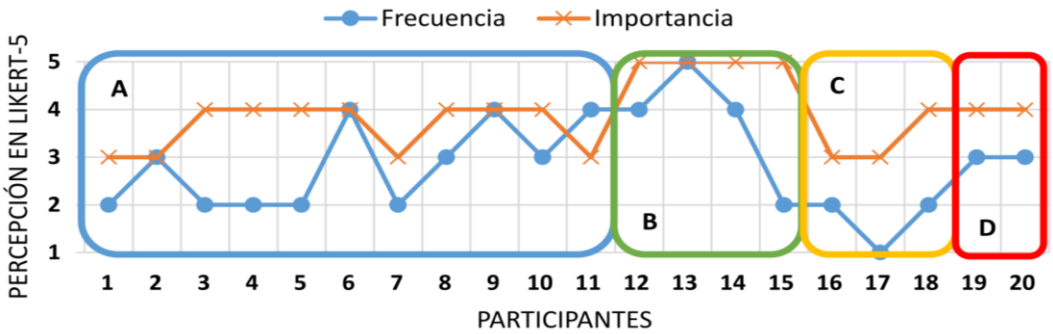


Figura 3.4. Gráfica de dispersión sobre la importancia y frecuencia percibida (Likert-5) de las interacciones UTEM, agrupadas por vistas arquitectónicas y empresa (A, B, C, D).

Con el fin de hacer una comparación entre frecuencia e importancia percibida por los participantes, pero agrupado por vistas arquitectónicas, se elaboró una gráfica de dispersión, donde el eje X corresponde a los valores frecuencia y el eje Y corresponde a los valores de importancia (ver Figura 3.4). Se observó que la mayoría de las vistas arquitectónicas se percibían al menos como "importantes" (> 3 en escala Likert). En términos de frecuencia, aproximadamente la mitad de las percepciones fueron al menos "frecuente" (> 3 en escala Likert).

Visualmente se aprecia una diferencia en la Figura 3.5 entre la frecuencia y la importancia de las interacciones UTEM. Hay más dispersión en las percepciones de frecuencia que en las percepciones de importancia. Para confirmar esta diferencia de manera analítica, todos los participantes fueron tomados como una muestra única de trabajadores de AGSD, es decir, se conjuntaron los resultados de los participantes de las cuatro compañías. Posteriormente, se obtuvo la mediana de la importancia percibida y de la frecuencia percibida



**(empresa A = caja azul, empresa B = caja verde, empresa C = caja amarilla, empresa D = caja roja)*

Figura 3.5. Medianas de frecuencia e importancia percibida (Likert-5) obtenida de cada participante sobre las categorías de interacciones de UTEM.

Tabla 3.5. Detalles de la prueba de rangos con signo de Wilcoxon.

Valores del estadístico W		Valores del estadístico Z	
Valor W	5.5	Valor Z	-3.2318
Diferencia de medias	-0.19	Media (W)	68
Suma de rangos positivos	5.5	Desviación estándar (W)	19.34
Suma de rangos negativos	130.5	p-value	0.00062
$W_{p\leq 0.01}$ (n = 16) [a]	23		

de cada participante sobre las categorías. Por lo tanto, se obtuvieron dos conjuntos de datos de 20 elementos, es decir, el número total de participantes en este estudio (gráficamente vistos en la Figura 3.5).

Dado el tamaño de la muestra y la naturaleza de los datos (no apegados a la distribución normal), se decidió aplicar una prueba de rangos con signo de Wilcoxon. Se definió una hipótesis nula (H_0) de la siguiente manera: *No hay diferencia entre las percepciones de frecuencia e importancia de las interacciones UTEM relacionadas con AK*. La hipótesis alternativa (H_1) fue la siguiente: *La importancia percibida es significativamente mayor que la percepción de frecuencia de las interacciones UTEM relacionadas con AK*. Después de aplicar la prueba estadística, con un nivel de significancia de 0.01 (ver detalles en la Tabla 3.5), la hipótesis nula fue rechazada (valor $p = 0.00062 \leq 0.01$, $W \leq Wp \leq 0.01$). Por lo tanto, podemos establecer que la percepción de importancia de las interacciones UTEM es mayor que la percepción de frecuencia para los participantes de este estudio, con lo que se contesta la pregunta de investigación RQ3. Cabe aclarar que el tamaño efectivo de la muestra presentando en la Figura 3.4 es de 16 porque hay cuatro pares de percepciones de frecuencia e importancia que son iguales (Figura 3.5).

3.6.3 Categorización de las interacciones cara a cara referentes a AK

Se observaron interacciones cara a cara para tener el escenario completo de las interacciones relacionadas con AK en un entorno de AGSD. Como se mencionó anteriormente, se

Tabla 3.4. Temas arquitectónicos identificados en las interacciones cara a cara.

Temas arquitectónicos	Frecuencia	Vista arquitectónica
Aclaraciones de funcionalidad actual	39%	Escenarios
Aclaraciones de funcionalidad técnica actual	29%	Implementación
Aclaraciones de nueva funcionalidad	16%	Escenarios
Aclaración de programación de base de datos	11%	Implementación
Configuraciones de aplicaciones y ambientes	5%	Despliegue

registraron 45 interacciones cara a cara en la empresa A. Es destacable que el 39% de estas interacciones se relacionaron con aclaraciones sobre funcionalidad actual de proyectos y el 29% relacionadas con el funcionamiento técnica actual (ver Tabla 3.4). Esto significa que los temas de programación y requerimientos son prominentes en las interacciones cara a cara.

Además, se agruparon las interacciones por vista arquitectónica (ver Figura 3.6) considerando el modelo arquitectónico de 4+1 vistas, y se observó que el 55% de ellas estaban relacionadas con la vista de escenarios, y el resto estaban relacionadas con la vista de implementación (40%) y con la vista de despliegue (5%). Desde esta perspectiva, se puede observar una coincidencia entre las interacciones mediante UTEM y las interacciones cara a cara, ya que estas tres vistas arquitectónicas (escenarios, implementación y despliegue) fueron frecuentes en las interacciones mediante UTEM.

Además, dado que la observación se realizó sólo en la compañía A, durante las sesiones de los grupos focales se les preguntó a los participantes de las otras compañías sobre sus temas arquitectónicos más frecuentes en las interacciones cara a cara. Los temas que se obtuvieron de ello fueron: aclaración de la funcionalidad, funcionamiento técnico de cualquier aplicación (tanto de manera general como de manera detallada), y soporte técnico en temas de desarrollo los cuales fueron consistentes con los observados en la compañía A.

3.7. Reflexiones sobre la articulación del AK

La discusión de los resultados está organizada en dos partes: en primer lugar, se discute sobre la documentación del AK, las interacciones mediante UTEM y las interacciones cara a cara; y finalmente, se discute sobre los problemas relacionados con el intercambio de AK. Al final de esta sección se presentan las conclusiones obtenidas de este estudio.

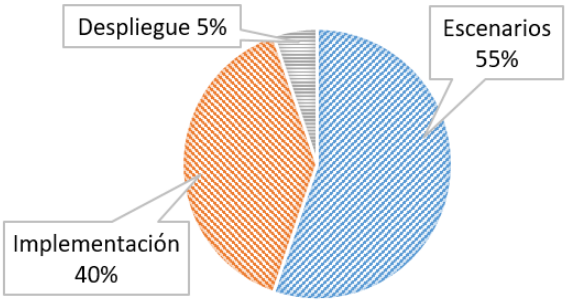


Figura 3.6. Frecuencia de interacciones cara a cara agrupadas por vista arquitectónica.

3.7.1 Documentación de AK e interacciones mediante UTEM y cara a cara

Durante la fase de observación, sólo se encontraron documentos relacionados con la vista de escenarios, es decir, documentos relacionados con la especificación de funciones y las historias de los usuarios, tal como también lo reporta (Gervigny and Nagowah, 2017). Los participantes comentaron que los documentos de requerimientos generalmente son la base única de todo el ciclo de desarrollo. Además, comentaron que escriben documentos técnicos sólo por demanda, tal como lo reporta (Dorairaj et al., 2012). Con respecto a esta situación, un participante dijo que: "*...las únicas veces que tengo que escribir algún documento es porque habrá una reunión con el cliente...*".

Además, los participantes comentaron que rara vez escriben documentos técnicos porque este tipo de documento rara vez se consulta: "*...generamos diagramas UML, pero rara vez los vemos, a menos que no entendamos el código...*". En particular, la empresa D depende en gran medida de la disciplina y las habilidades técnicas de sus desarrolladores. Mencionaron que el código auto-documentado (es decir, la asignación de nombres descriptivos a variables, métodos, clases, etc.) es una práctica común, para evitar hacer documentación técnica y ayudarse a entender el código de una mejor manera.

Por otro lado, los participantes de la compañía B dijeron que escriben diagramas técnicos, pero solo como una ayuda para comprender problemas. Estos diagramas suelen ser ad-hoc y "desechables", es decir, no se guardan como documentación del proyecto. En la compañía A sucede algo similar con la documentación del AK. Esto puede estar sesgado porque ninguno de los seis proyectos observados estaba en las primeras etapas, y por lo tanto los problemas conceptuales ya han sido abordados. Sin embargo, la empresa A informó que trabajan solo en base a la metodología de Scrum, la cual que no incluye una fase para identificar elementos conceptuales para el desarrollo de software (Schwaber and Sutherland, 2011). Podría decirse que este comportamiento refleja la naturaleza del paradigma ágil (Beck et al., 2001), en el que el tiempo de diseño tiende a reducirse o incluso desaparecer.

Teniendo en cuenta las categorías de interacción mediante UTEM, se observa que la mayoría de las categorías relacionadas con los requerimientos (C5, C7 y C11) se percibieron como las más altas en frecuencia e importancia en las cuatro empresas participantes. Además, el 55% de las interacciones cara a cara se relacionó también con la vista de escenarios. Aunque los requerimientos fueron el único elemento arquitectónico documentado, este

comportamiento muestra que estos documentos no están claros. En resumen, esto significa que cuando una vista arquitectónica no está documentada adecuadamente, aumenta la interacción entre los miembros del equipo, tanto cara a cara como a través de UTEM; con esto se contesta la pregunta de investigación RQ4. Además, refleja la aplicación de dos valores del manifiesto ágil (Beck et al., 2001): "*Individuos e interacciones sobre procesos y herramientas*" y "*Software funcionando sobre documentación integral*". Lo anterior da como resultado la característica de poca o nula documentación en equipos ágiles, como también se evidencia en la literatura (Rost et al., 2013; Stettina and Heijstek, 2011).

3.7.2 Problemas relacionados con la compartición de AK

Teniendo en cuenta el punto de los documentos arquitectónicos, se encontró que la ausencia o falta de claridad en ellos, también fomenta el aumento de interacciones para consulta (Martini et al., 2013). En este estudio en particular, esto se reflejó en las principales causas de las interacciones mediante UTEM. En principio, preferir el software funcionando y la interacción de los individuos es beneficioso para reducir los tiempos de entrega (Smith, G., Sidky, 2009). De hecho, la mayoría de los participantes mencionaron que consultar dudas a través de UTEM es un camino rápido hacia el AK, lo cual se percibió como un beneficio. Sin embargo, debe existir una alta disponibilidad y voluntad de las personas para que este comportamiento no se convierta en un obstáculo (Young and Terashima, 2008). En otras palabras, la comunicación en las organizaciones tiene una gran influencia en la velocidad de desarrollo (Martini et al., 2013).

En el caso de los equipos de AGSD, los miembros remotos están disponibles sólo unas pocas horas al día, tomando en cuenta la diferencia de huso horario y las actividades diarias de trabajo. Esto reduce el tiempo disponible para aclarar problemas entre los miembros locales y remotos (al menos sincrónicamente). Por ejemplo, uno de los participantes informó lo siguiente con respecto a la disponibilidad de sus contrapartes: "...*a veces tengo días de retraso... porque el analista está ausente*...". Por otro lado, aunque los miembros remotos estaban completamente disponibles, algunos participantes consideran que deben pasar hasta tres horas al día haciendo actividades de aclaración de dudas, ya sea por llamada telefónica o por UTEM, que son parte de los perjuicios identificados como resultados del estudio.

Otra consecuencia negativa del intercambio de AK a través de interacciones personales (por medio de UTEM o cara a cara) es la vaporización del mismo (Bosch, 2004). Esto significa que el AK puede ser alterado o perdido a través del tiempo, cuando no está codificado adecuadamente. En las empresas participantes, la vaporización de AK afecta cuando los nuevos miembros se integran en los equipos, o cuando se retoman proyectos previos; como no existe suficiente conocimiento documentado se pierde tiempo en contextualizar a los nuevos miembros o contextualizándose a sí mismo tratando de recordar aspectos del proyecto retomado. Lo que sucede también es que los pocos documentos arquitectónicos que existen se complementan con interacciones entre los compañeros del equipo (locales o remotas). Sin embargo, las interacciones entre pares no garantizan la adquisición total del AK necesario, ya que depende de la memoria y el conocimiento de los colegas.

Respecto a lo anterior, uno de los participantes comentó: "... *tuve que navegar mucho en el código... batallar mucho para entender cómo hacer un pequeño cambio en la aplicación y creo que aún no lo conozco suficientemente bien todavía...*". Esto es, la falta de AK provocó que tareas simples llevaran más tiempo de lo planeado. Esto confirma la importancia de conocer al menos la arquitectura de referencia en los equipos AGSD (Martini et al., 2013).

Relacionado con el factor tiempo, uno de los participantes dijo: "... *Pasé demasiado tiempo arreglando errores... porque tenía tantas preguntas para mi contraparte que prefería parar...*". Una de las situaciones que inhibe las interacciones en UTEM es que sería embarazoso pedir demasiado a través estos medios, o podría ser molesto recibir demasiadas preguntas. Esto conduciría a un perjuicio de las relaciones sociales entre los miembros del equipo, y afectaría el flujo de conocimiento (especialmente tácito) en equipos ágiles (Ryan and O'Connor, 2013).

En el caso anterior, el participante se mostró más preocupado con el conocimiento de la vista de implementación, es decir, aspectos muy relacionados con la programación de los proyectos. Si bien las categorías relacionadas con la vista de implementación (C1, C3, C4, C6 y C10) no se percibieron con tanta frecuencia como las relacionadas con la vista de escenarios, la importancia de estas interacciones es alta, ya que pueden afectar negativamente al desarrollo de software.

Por otro lado, los resultados muestran que hay un problema de accesibilidad de AK en entornos de AGSD. Este problema es causado por la preferencia de compartir este conocimiento a través de interacciones interpersonales, y por las distancias físicas y temporales. Este estudio estuvo centrado en las interacciones mediante UTEM, que dejan rastros de información donde se puede extraer AK (Harrison and Veerappa, 2014b; Yanzer Cabral et al., 2014b). En este sentido, algunos participantes informaron de esfuerzos para recuperar AK de los registros de correo electrónico y de las conversaciones de mensajería instantánea (como también se informa en (Harrison and Veerappa, 2014b)). Sin embargo, no cuentan con los mecanismos para explotar de manera efectiva estas fuentes, es decir, actualmente este proceso les consume tanto tiempo y esfuerzo, lo cual afecta sus tareas sustantivas.

3.8. Resumen del capítulo

En este capítulo se presentaron los resultados de un estudio empírico realizado con cuatro empresas mexicanas de AGSD. Se observaron y analizaron diferentes aspectos relacionados con el AK en un entorno AGSD: artefactos, interacciones por medios electrónicos de texto no estructurados (UTEM) e interacciones cara a cara. Se utilizó como referencia el modelo arquitectónico de 4+1 vistas para comprender y caracterizar los artefactos arquitectónicos y las interacciones en UTEM y cara a cara relacionadas con AK. Los principales resultados de este capítulo incluyen:

- Documentos arquitectónicos elaborados por equipos de AGSD, donde la mayoría eran referentes a la vista de escenarios, sin embargo estos documentos generalmente no eran muy claros. También se encontraron documentos referentes a la vista de implementación, la mayoría de ellos informales o ad-hoc; los más formales que se encontraron había sido elaborados sólo bajo demanda. Esta falta de documentos arquitectónicos provoca aumento del tiempo de desarrollo por tratar de obtener AK directamente del código fuente, por esperar a que la persona que posee el conocimiento se encuentre disponible y/o responda las dudas o por tratar de resolver las dudas buscando la respuesta en historiales de UTEM.
- Interacciones referentes al AK en equipos de AGSD, agrupadas en 11 categorías de interacciones en UTEM y 5 acerca de interacciones cara a cara, donde la mayoría de ellas

estaban relacionadas con la vista de escenarios (requerimientos) y la vista de implementación (codificación, componentes, etc.), lo cual refleja dos valores ágiles en los equipos de AGSD: "*Individuos e interacciones sobre procesos y herramientas*" y "*Software funcionando sobre documentación integral*". Se identificó que el AK compartido a través de UTEM se considera importante por parte de los equipos AGSD, independientemente de la frecuencia con que se haga. Esto indica que existe conocimiento valioso en las interacciones de UTEM, independientemente de la distribución del trabajo entre los miembros locales y remotos en los equipos AGSD. Además, se modeló la compartición del AK mediante UTEM en equipos de AGSD. El modelo cuenta con dos grandes partes, una que describe los aspectos de comunicación que están involucrados en la compartición de conocimiento, y la otra los aspectos de AK que se comparten a través de UTEM.

- A pesar de que la comunicación basada en documentos ayuda a minimizar los efectos de las cuatro distancias del GSD, las prácticas ágiles tienden a ser prominentes en un entorno AGSD. De hecho, se ha determinado que los impulsores más importantes para AGSD se relacionan con los principios ágiles y el resto de los impulsores se relacionan con las perspectivas comerciales globales (Kamaruddin et al., 2012).

Una vez teniendo un entendimiento más amplio de la manera de articular AK en equipos de AGSD, en el siguiente capítulo se presenta una propuesta conceptual para disminuir uno de los problemas más relevantes provocado por un mal manejo del AK en AGSD, es decir, el problema de la vaporización del conocimiento. Esta propuesta se basa tanto en los resultados obtenidos en este capítulo como en el anterior.

Capítulo 4.
Condensación del conocimiento arquitectónico

En los capítulos anteriores se presentó una revisión de literatura para conocer la manera en la que se reporta el manejo del AK en empresas de AGSD (Gilberto Borrego et al., 2017), y además se presentó un estudio en sitio para comprender a detalle la compartición de conocimiento en dicho ambiente (Borrego et al., 2016). De la revisión de literatura se obtuvo que un enfoque popular para el manejo del AK es a través de los UTEM, los cuales a su vez son los medios de comunicación preferidos de los desarrolladores de AGSD, debido a que permiten reducir los efectos que provocan las diferencias lingüísticas (H.-Christian Estler et al., 2012). Del estudio en sitio se obtuvo que los desarrolladores de AGSD comparten conocimiento en UTEM sobre requerimientos y cuestiones de codificación, además este conocimiento es valioso para ellos. Además, los desarrolladores reportaron dificultad para recuperar dicho conocimiento de las bitácoras de los UTEM, dado que éstos no están diseñados para ello y que generalmente no recuerdan quien, ni cuando, ni por qué medio les compartieron el conocimiento. Esto al pasar el tiempo provoca lo que se conoce en la literatura como *vaporización del conocimiento*, es decir, que el conocimiento se pierda.

Todo lo anterior condujo a concebir una manera de reducir la vaporización del AK en AGSD, aprovechando que el conocimiento no se ha perdido totalmente, sino que se encuentra en las bitácoras de los UTEM. Sin embargo, este conocimiento se encuentra disperso en las bitácoras de distintos UTEM y no cuenta con alguna estructura que facilite su recuperación. A esta manera de reducir la vaporización del AK se le llamó análogamente *condensación del conocimiento*, que en términos de transiciones de la materia, vaporización y condensación son transiciones que se pueden llamar contrarias, porque representan el paso del estado líquido al gaseoso y viceversa. En este capítulo se presenta la definición de este concepto, su contextualización para el AGSD y las características que debe cumplir una implementación del mismo.

4.1. Definición de vaporización y condensación de AK

Dado que la condensación de AK tiene una estrecha relación con la vaporización, sería lógico partir de la definición de este último concepto para construir la definición del primero. Al

momento de escribir estas líneas, el concepto de vaporización no cuenta con alguna definición clara y apegada al ámbito del manejo del conocimiento, sólo algunas definiciones poco formales apegadas al ámbito de la ingeniería de software. Dado que uno de los detonantes de la concepción de la condensación del AK es la transformación de conocimiento tácito en explícito (llamado articulación (Nonaka and Takeuchi, 1995)) lo cual sucede frecuentemente entre los desarrolladores de AGSD al estar interactuando por UTEM, se considera importante que tanto el concepto de vaporización y el concepto de condensación se definan apegados a los términos del manejo del conocimiento. Por lo tanto en esta sección se definirán ambos conceptos en estos términos.

Como se mencionó anteriormente, la vaporización del conocimiento no cuenta con una definición formal. Sin embargo se encontró en la literatura tres citas destacadas donde se expone este concepto, desde la perspectiva de la ingeniería de software:

- Cuando una cierta parte del conocimiento sobre algún artefacto arquitectónico desaparece (M. A. Babar, 2009).
- La pérdida de conocimiento referente a: análisis de dominio, estilos arquitectónicos utilizados en el sistema, patrones de diseño seleccionados y todas las demás decisiones de diseño tomadas durante el diseño arquitectónico, así como la arquitectura de software resultante, por no tener una representación de primera clase (Bosch, 2004).
- La vaporización del AK sucede porque que la mayoría de las decisiones tomadas durante el diseño de arquitecturas de software permanecen implícitas, es decir, no documentadas (Zernadji et al., 2014).

Conjuntando estas tres nociones la vaporización del AK se obetiene la siguiente definición.

Definición 4.1. Vaporización del AK. La desaparición de artefactos y documentos arquitectónicos (diseño de software y a las decisiones tomadas para llegar a su versión actual), debido a una pobre o nula documentación.

Sin embargo, cuando el AK no se encuentra documentado, generalmente este se puede recuperar (de forma parcial) analizando código fuente, lo cual es muy tardado, y el conocimiento adquirido es desestructurado, incompleto e inconsistente, lo que no garantiza que el software evolucione como se planeó en el momento del diseño (Selic, 2009).

Con base en la definición anterior se observa que el concepto de vaporización del conocimiento aún tiene que ser clarificado, ya que en la literatura se interpreta que el conocimiento vaporizado es conocimiento perdido. Por otro lado, se dice que cuando no se cuenta con AK este puede ser recuperado (por lo menos parcialmente) por medio del análisis de código fuente. Por ello, con el fin de definir el concepto de condensación de conocimiento, es necesario refinar la definición de vaporización de AK. Para ello se tomará como base el modelo SECI (Nonaka and Takeuchi, 1995), debido a que ambos conceptos (vaporización y condensación) están estrechamente relacionados con las transformaciones entre el conocimiento tácito y explícito (y viceversa), como se mencionó anteriormente. A continuación, se presenta el modelo SECI (ver Figura 4.1) y se describen las cuatro transformaciones:

- **Socialización**. Es la creación o adquisición de conocimiento tácito por medio de experiencias compartidas, como pasar tiempo juntos o vivir en el mismo entorno. A través de este tipo de experiencias es como el conocimiento tácito es más fácilmente adquirido, ya que este tipo de conocimiento es difícil de formalizar y es a menudo dependiente del tiempo y el espacio.
- **Externalización**. Es el proceso en el cual se articula el conocimiento tácito en conocimiento explícito, es decir, se traduce a dibujos, grabaciones de audio y/o video, notas de texto, etc. Esto permite que el conocimiento se comparta por más personas y que se convierta en la base de nuevo conocimiento. De hecho, una conversión exitosa del conocimiento tácito en explícito depende del uso de metáforas, analogías y modelos.

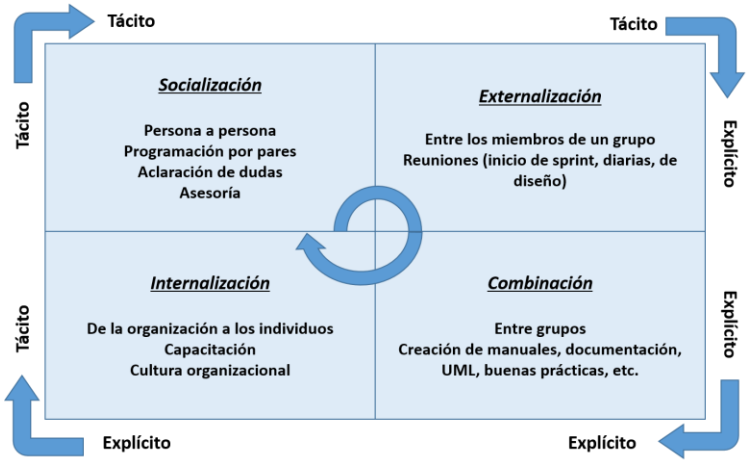


Figura 4.1. Modelo SECI de Nonaka y Takeuchi.

- **Combinación**. Es cuando el conocimiento explícito se expresa en conjuntos más complejos y sistemáticos que la forma inicial de expresión de este tipo de conocimiento. En el proceso de combinación, se recolecta conocimiento explícito interno o externo a la organización y luego se combina, edita o procesa para formar nuevos conocimientos.
- **Internalización**. Se trata de la incorporación del conocimiento explícito al conocimiento tácito. El conocimiento explícito creado a través del proceso de combinación se comparte en toda una organización, con el fin de que los individuos lo conviertan en tácito al comprender y aplicar el conocimiento.

Contextualizando las transiciones del modelo SECI hacia el desarrollo de software ágil y el manejo de AK, la socialización del conocimiento ocurre todos los días cuando los desarrolladores interactúan cara a cara, por ejemplo: en las dinámicas de programación o revisión por pares, aclarando algún requerimiento, preguntando algún detalle técnico o del diseño arquitectónico del proyecto. En un ambiente distribuido, los desarrolladores ágiles tienen una limitada oportunidad de socialización, ya que, en un equipo virtual, los miembros podrían estar totalmente distribuidos (cada uno en una localización diferente), o bien, que existan grupos de trabajo que se encuentren en el mismo sitio y que interactúen con otros miembros o grupos que se encuentren en sitios distintos.

En este punto existe riesgo de vaporización, ya que si el AK que se crea permanece en su forma tácita, y cuando un miembro del equipo se retira definitivamente, todo su conocimiento se retira también, sólo permanece el conocimiento arquitectónico que haya sido compartido y que pueda ser recordado por el resto de sus compañeros; es decir, un factor de vaporización del conocimiento tácito es el simple olvido. Con el fin de ilustrar la relación de la vaporización del AK con el modelo SECI, en la Figura 4.2 se expresan las cuatro transformaciones como transiciones en un diagrama de estados de UML, así como los estados del conocimiento, i.e. tácito o explícito. Se puede observar también una transición del estado tácito al estado vaporizado cuando ocurre la vaporización. Además, se puede observar una transición entre el estado vaporizado y el estado tácito, lo que significa que el AK vaporizado pudiera ser recuperado, como lo establece (Selic, 2009). Concretamente para este caso, el conocimiento tácito que se encuentra vaporizado sólo se podría recuperar recordando o si la persona que se retiró regresa. También se observa una transición del estado vaporizado al

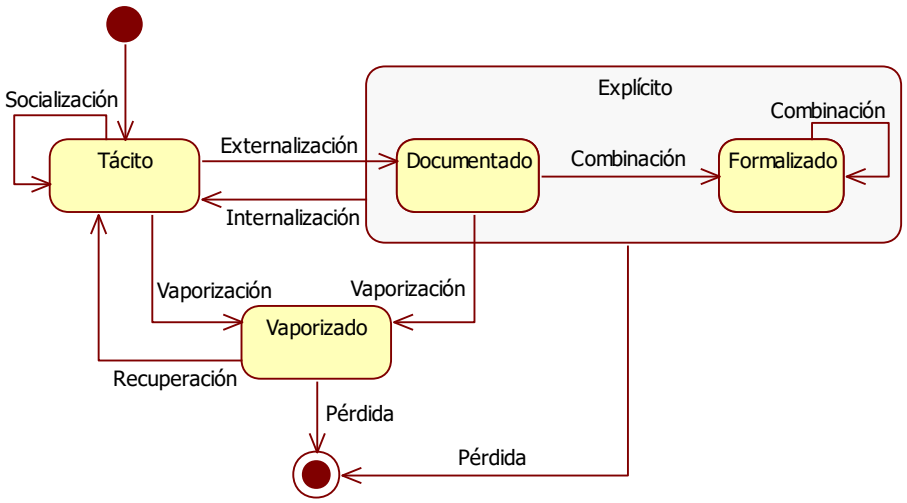


Figura 4.2. Representación del Modelo SECI como un diagrama de estados incluyendo la vaporización del conocimiento.

estado final del diagrama lo que simboliza el olvido total del conocimiento tácito que se encuentra en dicho estado.

Continuando con los procesos del modelo SECI, la externalización en el desarrollo ágil de software puede suceder en alguna reunión técnica donde se realicen diagramas o esquemas que apoyen en la trasmisión y creación del AK. En el caso del AGSD, la externalización también sucede en este tipo de reuniones, tanto de manera física como de manera virtual. En el caso de las reuniones virtuales, el conocimiento generalmente se articula en audio y video (si la sesión es grabada) o en el texto de las bitácoras de los medios usados para comunicarse. De lo anterior, surge una discrepancia con el modelo SECI, ya que éste establece que la externalización sucede cara a cara (Nonaka et al., 2000), y en el AGSD esto sólo puede ser posible si se interactuara únicamente entre el personal que se encuentra co-localizado. Sin embargo, las interacciones con personal remoto son frecuentes, y por lo tanto también se articula conocimiento tácito a través de medios virtuales.

Si bien mediante la externalización se convierte el conocimiento tácito en explícito, en el caso del AGSD este conocimiento generalmente no está expresado formalmente (Clear, 2003), de tal manera que se pueda almacenar en algún repositorio para ser consultado por todos en cualquier momento. Se requeriría pasar ese conocimiento a algún formato estandarizado para asegurar que este pueda preservarse. En el trabajo de (Nonaka and Takeuchi, 1995) se definen dos tipos de conocimiento explícito: el conocimiento explícito documentado y el conocimiento explícito formalizado. El primero se refiere a maneras

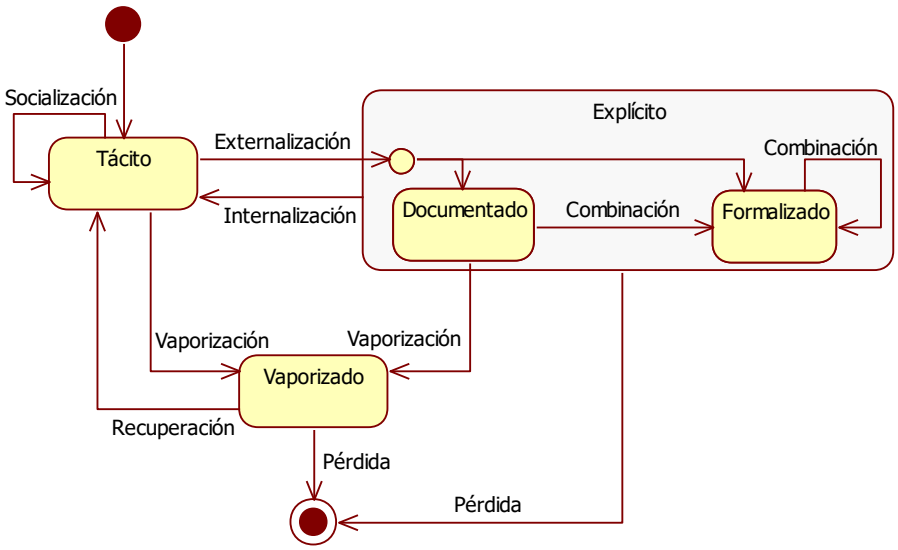


Figura 4.3. Representación del Modelo SECI como un diagrama de estados incluyendo la vaporización del conocimiento y los tipos de conocimiento explícito.

informales de articular el conocimiento, como: notas de texto, dibujos, audio, video, etc. El conocimiento explícito formalizado se refiere al conocimiento que se expresa siguiendo algún estándar que pueda ser interpretado por cualquier persona e incluso por computadoras (e.g. ADL, UML, etc.). Por lo tanto, se podría decir que en el AGSD la externalización del AK generalmente se queda en su forma documentada, sin embargo, no se podría descartar la posibilidad de que se externalice el conocimiento directamente al estado formalizado, cuando quien lo hace es alguien disciplinado y/o con experiencia. Con el fin de analizar las transiciones entre los tipos de conocimiento explícito y su relación con la vaporización, se agregan estos tipos de conocimiento al diagrama de estados presentado anteriormente (ver Figura 4.3).

En la Figura 4.3 se muestra que la transición de externalización llega a un punto de opción en el cual se puede llegar al estado documentado o al estado formalizado dependiendo de las circunstancias, tal como se especificó en el párrafo anterior. También se muestra la transición hacia el estado final del diagrama representando la pérdida del conocimiento explícito. En el caso concreto del desarrollo de software, la pérdida del conocimiento explícito documentado sucedería cuando se extravían definitivamente notas de texto, dibujos, modelos, etc. que hayan sido útiles para una explicación, es decir, cuando el medio en el que se articuló el AK se pierde. Sin embargo, el conocimiento tácito seguiría presente en las mentes del personal. Además, en la Figura 4.3 se muestra la transición del sub-estado

documentado al estado vaporizado. Para el caso del AGSD, la vaporización sucedería cuando el AK que se articuló en algún medio virtual, no es fácilmente localizable al pasar del tiempo, generalmente debido a que el conocimiento no se encuentra organizado y los medios en donde se articula no están preparados para la búsqueda de conocimiento. En el caso de conocimiento articulado a medios físicos, la vaporización podría suceder cuando notas y esquemas se van guardando, y al pasar el tiempo no son fácilmente localizables.

El concepto de condensación del AK se introduce como una transición entre conocimiento vaporizado y conocimiento explícito documentado. En términos físicos la condensación representa el cambio del estado gaseoso al estado líquido, lo que a su vez representa que los átomos del material condensado estarán más contiguos entre ellos, es decir, menos dispersos, y además con cierto orden distinto al estado vaporizado. A su vez, en términos del conocimiento, el concepto de condensación en general consistiría en conjuntar y ordenar el conocimiento articulado, ya sea en medios físicos (p. ej. notas y dibujos en papel, esquemas en pizarrones, etc.) o en medios virtuales (p. ej. audio y video de reuniones, correos electrónicos, mensajeros instantáneos, foros, etc.). De esta manera existiría la posibilidad de pasar del estado vaporizado al sub-estado documentado. Sin embargo, el concepto de condensación implica que el conocimiento se debe estructurar u organizar, para luego almacenarse, y el sub-estado documentado sólo implica que el conocimiento está articulado de manera informal. Esto nos conduce a introducir dos sub-estados al conocimiento explícito documentado: organizado y no organizado (ver Figura 4.4). El primero se refiere a

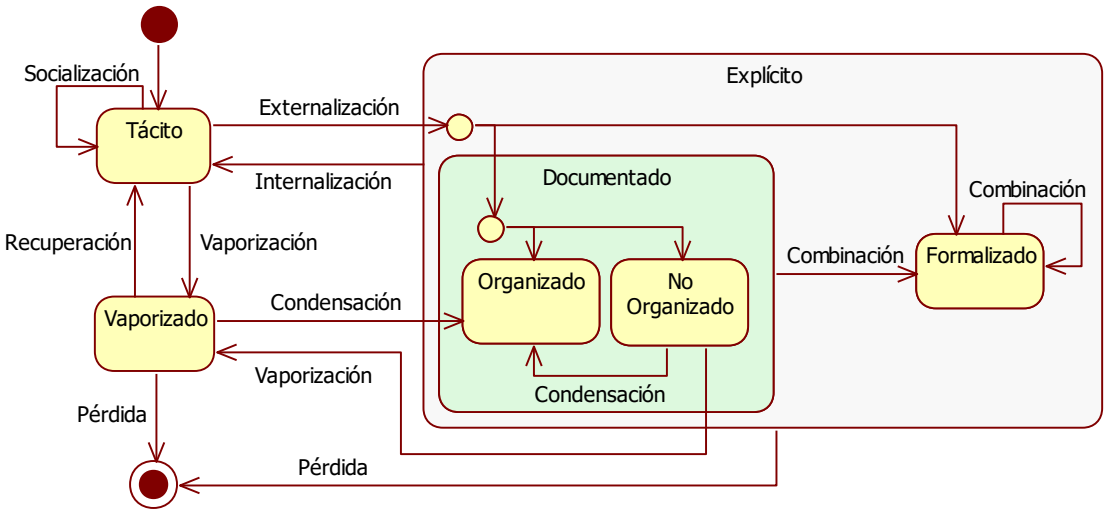


Figura 4.4. Representación del Modelo SECI como un diagrama de estados incluyendo la vaporización del conocimiento y los tipos de conocimiento documentado propuestos.

conocimiento expresado de manera informal, pero con cierta organización que facilita su localización y comprensión. El conocimiento no organizado sería el estado común al que se transformaría el conocimiento tácito a través de la externalización, aunque cabría la posibilidad de que el conocimiento tácito se articule directamente al estado organizado si se cuenta con un mecanismo que ayude a organizar el AK al momento que se está externalizando. Si se observa la Figura 4.4, ahora el conocimiento no organizado es el que tiene una transición hacia el estado vaporizado, ya que al pasar el tiempo éste tendería a vaporizarse por no estar organizado. También se observa que a través de la condensación se organiza el conocimiento no organizado, lo cual podría ser una manera de evitar que este conocimiento se vaporice.

Ahora, pasando al conocimiento explícito formalizado, la transición que lleva hacia éste sería la combinación, ya sea desde el sub-estado documentado o desde el mismo sub-estado formalizado. Esto se debe a que en el modelo SECI el proceso de combinación sucede cuando el conocimiento articulado de manera informal se transforma a notaciones formales y se integra al conocimiento formalizado de la empresa o equipo de trabajo; o bien, cuando se integran varias fuentes de conocimiento formalizado para crear nuevo conocimiento.

Pasando al proceso de internalización, que es cuando el conocimiento explícito es trasmitido a través de documentos a los individuos, el modelo SECI no especifica claramente qué tipo de conocimiento explícito sería el que se le entrega a los individuos. Sin embargo, la mayoría de los ejemplos de internalización se refieren al proceso de capacitación en donde intervienen manuales y documentos empresariales, los cuales representan conocimiento explícito formalizado. En el caso del AGSD, donde mucho del conocimiento externalizado se queda en el sub-estado documentado, podría caber la posibilidad de que ocurra la internalización desde dicho sub-estado. Por ejemplo, cuando a una persona nueva en el equipo le entregan notas y esquemas informales para que entienda la arquitectura del proyecto en el cual trabajará. Por lo tanto, la transición de internalización sale desde el super-estado explícito al estado tácito, como se aprecia en la Figura 4.4.

Finalmente, la última transición por describir es la existente entre el conocimiento explícito en general (incluyendo todos los tipos) y el estado final del diagrama, lo cual simboliza la pérdida total del conocimiento expresado ya sea de manera formal o informal (organizada o no). Esto podría suceder cuando se pierden definitivamente los registros de

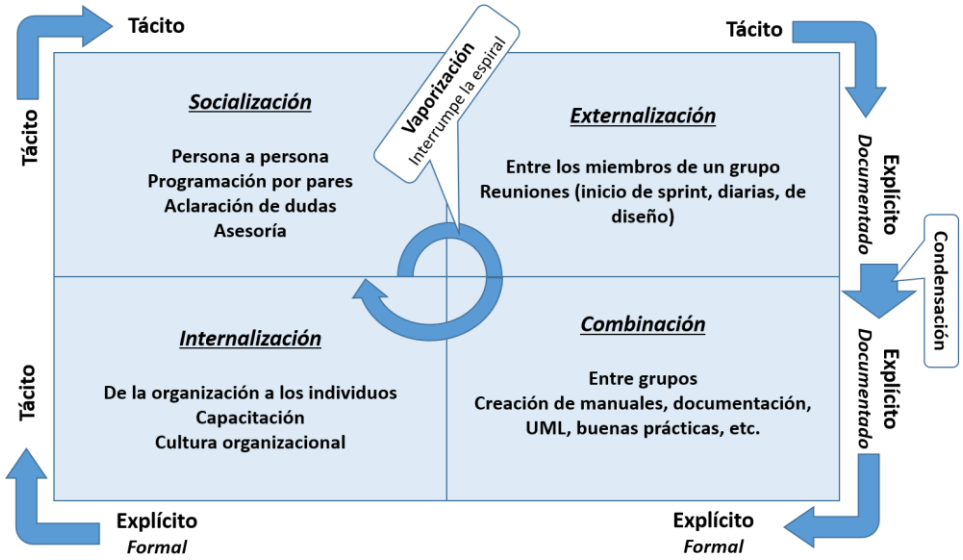


Figura 4.5. Modelo SECI con la inclusión de vaporización y condensación del conocimiento.

dicho conocimiento. Recapitulando, una definición de vaporización del conocimiento, tomando en cuenta el modelo SECI sería la siguiente:

Definición 4.2. Vaporización de conocimiento. Se refiere a la transición hacia un estado intermedio (vaporizado) entre el conocimiento tácito o explícito y la pérdida total del conocimiento, teniendo aun la posibilidad de regresar al estado tácito del conocimiento a través de la recuperación y al explícito a través de la condensación.

Si bien la vaporización no representa la pérdida total del conocimiento, ésta si representa una interrupción o desaceleración de la espiral de creación del conocimiento expresada en el modelo SECI (ver Figura 4.5), ya que el conocimiento vaporizado no es fácilmente localizable. Por otro lado, una definición de condensación del conocimiento basada en el concepto de vaporización es la siguiente:

Definición 4.3. Condensación de conocimiento. Una manera de preservar la espiral de creación del conocimiento, reduciendo la posibilidad de pérdida de conocimiento explícito documentado en su forma no organizada, ya sea organizando el conocimiento antes de que se vaporice o incluso rescatando el conocimiento vaporizado antes de que se pierda de manera definitiva.

4.2. Condensación de conocimiento arquitectónico para AGSD

Una vez establecidas las definiciones de los conceptos de vaporización y condensación del conocimiento que se manejarán en este trabajo, se contextualizará a continuación éste último al ámbito del desarrollo global y ágil de software.

En la Figura 4.4 se observa que la condensación del conocimiento es la transición entre el estado vaporizado al estado explícito – documentado - ordenado, y también la transición entre el estado explícito - documentado - no ordenado al explícito, documentado y ordenado, sin hacer distinción sobre el tipo de medio en el cual se haya articulado el conocimiento (físico o virtual). Para el caso específico de este trabajo enfocado al desarrollo global y ágil de software, la condensación de conocimiento se concentra sólo en el conocimiento articulado en medios virtuales, particularmente en medios electrónicos textuales no estructurados (UTEM por sus siglas en inglés), tales como: mensajeros instantáneos, correos electrónicos, foros, etc. Esto debido a que estos medios son los preferidos por los desarrolladores ágiles para comunicarse con sus pares remotos (H.-C. Estler et al., 2012). Además, en un estudio anterior (Borrego et al., 2016) se encontró que las bitácoras de los UTEM contienen conocimiento arquitectónico valioso, particularmente sobre requerimientos y cuestiones relacionadas a código. En este mismo estudio los desarrolladores expresaron la necesidad de recuperar el conocimiento articulado en los UTEM, así como las dificultades que tienen para ello al no estar ordenado el conocimiento, ni contar con las herramientas adecuadas para encontrarlo. Establecido lo anterior, se define la condensación de AK en el ámbito del AGSD.

Definición 4.4. Condensación de AK en AGSD. Recopilar AK del estado vaporizado o del sub-estado no ordenado, que haya sido articulado en UTEM, y estructurarlo para facilitar su recuperación a través de herramientas de búsqueda, con el fin de reducir su vaporización.

Para explicar este concepto más claramente se utilizará una analogía. Supongamos un paisaje lacustre, donde el agua representa el AK articulado en bitácoras de UTEM durante el desarrollo de software. En condiciones de clima seco, el agua del lago tiende a evaporarse y formar nubes. El clima seco representa las condiciones del entorno ágil y global que favorecen la vaporización del conocimiento. El clima seco afecta negativamente al ambiente ya que habría menos agua al pasar el tiempo y tendería secarse la vegetación (ver Figura 4.6),

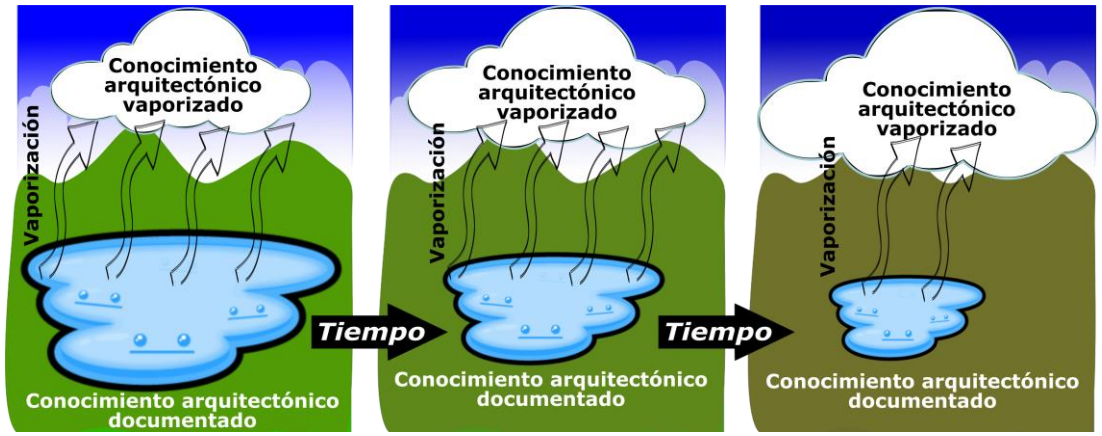


Figura 4.6. Representación metafórica de la vaporización del AK.

lo cual representa que la vaporización del AK afecta al trabajo diario de los desarrolladores. A menos de que haya un cambio en las condiciones del ambiente, de tal manera que deje de ser seco (lo que representaría cambiar las condiciones ágiles y globales), sería difícil recuperar el agua vaporizada o reducir su vaporización, es decir, recuperar conocimiento vaporizado de las bitácoras de los UTEM. Por lo tanto, la condensación del AK es comparable a proporcionar los medios para provocar la lluvia siempre que quiera reducir la vaporización del conocimiento, de tal manera que se mantenga el agua del lago en un nivel aceptable (ver Figura 4.7), que no permita que seque la vegetación. En términos de desarrollo de software esto significaría contar con AK suficiente para poder trabajar con menos contratiempos.

Para lograr la condensación del AK se requiere cumplir los siguientes elementos:

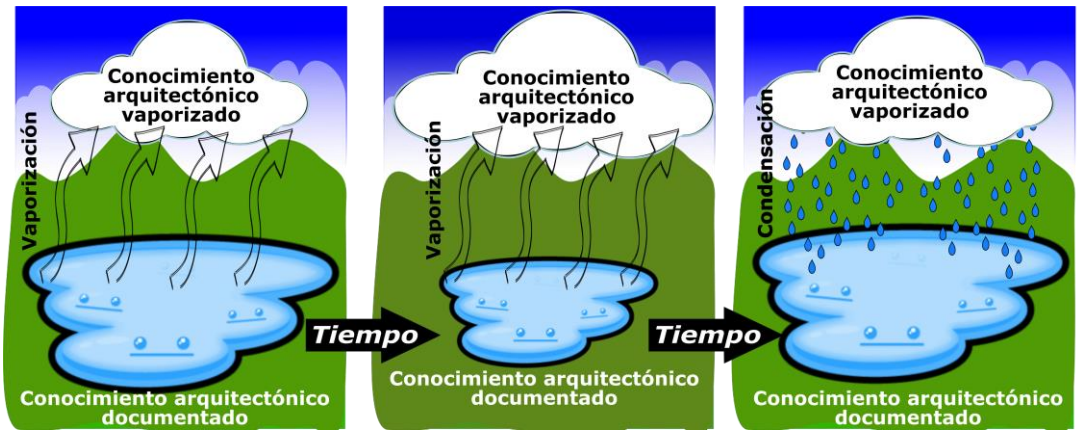


Figura 4.7. Representación metafórica de la condensación del AK.

1. **Información de las bitácoras de UTEM accesible**. Todas las partes interesadas deben poder acceder a toda la información de las bitácoras de UTEM y así poder acceder al AK que se comparte entre los mismos interesados.
2. **Mecanismo de clasificación de bitácoras de UTEM**. Debe haber un mecanismo de clasificación con el que se estructure la información de las bitácoras de UTEM para facilitar la recuperación de AK.
3. **Mecanismo de búsqueda AK**. Con base en el esquema de clasificación, todos los interesados podrían encontrar AK valioso en los registros UTEM ya estructurados y con menos esfuerzo.

4.3. Impacto en el desarrollo global y ágil de software

Una vez definidos los elementos que constituyen la condensación de AK, es posible delinear cuales sería los impactos al implementar este concepto en ambientes de AGSD. A continuación, se describen tres métricas concretas que se verían afectadas al implementar dicho concepto.

- **Tiempo para encontrar AK**. Teniendo acceso al conocimiento compartido en diversos UTEM a través de un mecanismo de recuperación, es muy probable que los involucrados en el proceso de desarrollo de software dediquen menos tiempo en encontrar conocimiento. Esto debido a la integración de bitácoras de UTEM y al esquema de clasificación en el que se basa el mecanismo de recuperación de AK.
- **Cantidad de interrupciones**. Es coherente pensar que el número de interrupciones entre los compañeros de trabajo se reduzca, ya que al contar con un medio de consulta de AK que integre lo compartido en diversos UTEM, se prefiera consultar primero este medio antes de consultar a un compañero de trabajo.
- **Tiempo en concluir tareas de desarrollo de software**. La combinación de los dos puntos anteriores lleva naturalmente a pensar que el tiempo para terminar tareas de desarrollo se reduzca, ya que según el estudio reportado en el capítulo anterior (y en (Borrego et al., 2016)), los desarrolladores de AGSD suelen perder mucho tiempo buscando AK en los UTEM o esperando a que algún compañero les resuelva alguna cuestión relacionada con la arquitectura, lo que naturalmente hace que las tareas asignadas lleven más tiempo de lo planeado.

Estos beneficios inciden indirectamente en lo que se conoce como efectividad de equipo, la cual en el trabajo de (Hackman, 1987) se define con base en tres elementos: (1) los resultados, ya que se deben cumplir o exceder los estándares establecidos en la organización; (2) los procesos sociales, porque se debe mejorar, o por lo menos mantenerse, la capacidad del grupo para trabajar juntos en el futuro; (3) el aprendizaje, debido a que la experiencia de trabajar en equipo debe fomentar el aprendizaje de los miembros del equipo.

Según (Kozlowski and Ilgen, 2006) la efectividad de equipo puede lograrse centrándose en los procesos cognitivos, conductuales y motivacionales/afectivos de los miembros del equipo, lo cual coincide con la definición de Hackman. La manera en la que se relaciona la efectividad de equipo con los beneficios antes mencionados se describe a continuación (ver Tabla 4.1).

Los resultados se afectarían por las tres métricas definidas, ya que si se tiene AK localizable, y si es poco el tiempo invertido en localizarlo, habría más tiempo disponible para realizar las tareas centrales de desarrollo. Por lo tanto los resultados se verían en menos tiempo o por lo menos con mayor calidad. Además, al tener AK localizable es muy probable que las interrupciones entre los miembros de los equipos se reduzcan, ya que al estar en deuda de documentación (como sucede generalmente), los desarrolladores de AGSD optan por buscar este conocimiento entre sus compañeros (tanto remotos como locales). Además, un ambiente con menos interrupciones es propicio para ser productivos y realizar trabajo de calidad, según estudios con desarrolladores de software (Eccles et al., 2009; Meyer et al., 2014). Finalmente, todos estos factores reducirían el tiempo en concluir tareas de desarrollo, lo cual se traduciría en que los resultados cumplan o excedan los estándares establecidos en la organización, aportando así a la efectividad de equipo.

Tabla 4.1. Relación de elementos de la efectividad de equipo y métricas de desarrollo software.

Métricas	**Efectividad de equipo**		
	Resultados	***Procesos sociales***	***Aprendizaje***
Tiempo para encontrar AK	●	●	●
Cantidad de interrupciones	●	●	
Tiempo en concluir tareas de desarrollo de software	●		

En referencia a los procesos sociales, estos se ven afectados al tener AK localizable, ya que sería más fácil de compartir. De esta manera, los desarrolladores podrían compartir conocimiento sobre el proceso de desarrollo, incluyendo: aspectos técnicos, aclaraciones de requerimientos, cuestiones de despliegue, etc., lo que ayudaría a los equipos a evitar los malentendidos costosos (Dingsøyr and Šmite, 2014). También, al reducirse las interrupciones, podrían reducirse los malestares entre compañeros, ya que como se estableció en el capítulo anterior (y en (Borrego et al., 2016)), preguntar frecuentemente causa sentimiento de vergüenza en la persona que hace la consulta, y en ciertas ocasiones molestias en quien responde, debido a las constantes interrupciones.

Ahora, referente al aspecto de aprendizaje para lograr la efectividad de equipo, el hecho de contar con AK localizable de una manera rápida, fomenta que se comparta el conocimiento, y esto a su vez conduce a la construcción de modelos mentales de equipo, donde los miembros tienden a depender el uno del otro de una manera cognitivamente interdependiente (Ryan and O'Connor, 2013). En términos de rendimiento del equipo, los modelos mentales de equipo mejoran la calidad de las habilidades de trabajo en equipo y la efectividad del equipo (Cannon-Bowers et al., 1993; Orasanu and Salas, 1993). Además, un estudio del rendimiento del desarrollo de software muestra cómo una mejor integración de dominio y conocimiento técnico conduce a una mayor eficiencia y eficacia del desarrollo de software (Tiwana, 2004).

Hasta este punto, la definición y descripción de la condensación del AK sigue siendo abstracta, por lo cual podría haber diferentes maneras de implementar instancias concretas. En la siguiente sección se presenta la manera en la que se decidió diseñar e implementar una solución tecnológica basada en dicho concepto.

4.4. Características para una implementación del concepto de condensación de AK.

Si bien hasta el momento el concepto de condensación de AK es un tanto abstracto, éste debe poder concretarse en una implementación con el fin evaluar su factibilidad para el ambiente de AGSD. Para ello, en esta sección se presentan las características de cada elemento del concepto, que debiera cumplir una implementación del mismo.

4.4.1 Características para la accesibilidad de las bitácoras de UTEM.

Con el fin de lograr esta accesibilidad se logran visualizar dos posibles modalidades: (1) distribuida y (2) centralizada. La modalidad distribuida se refiere a hacer accesibles las interacciones registradas en las bitácoras, aprovechando las interfaces públicas que expongan los UTEM. En la actualidad muchos de ellos (sobre todo basados en Web) ofrecen interfaces para que otros sistemas puedan interactuar con ellos, por ejemplo: Facebook Messenger[19], Jabber[20], Skype[21], Slack[22], etc. Sin embargo, habría que evaluar los mecanismos de recuperación de información que ofrecen los UTEM con los que se quiera trabajar. Hay que recordar que la función principal de los UTEM es la comunicación, y no la recuperación de información. Por lo tanto, los métodos que ofrezcan podrían ser muy básicos (p. ej. sólo basados en texto), lo cual limitaría las opciones para recuperar conocimiento condensado.

Respecto a la modalidad centralizada, consistiría en extraer las interacciones registradas en las bitácoras de los UTEM, para luego conjuntarlas en una base de datos común (repositorio). Con esta modalidad se tendría control total sobre los registros de interacciones, así como en la manera de explotarlos, ya que por lo general una base de datos ofrece opciones robustas para la recuperación de datos. Además, el proceso de explotación de datos sería más sencillo de implementar ya que sólo se consultaría una sola fuente. Sin embargo, centralizar las interacciones de los UTEM implicaría el manejo de grandes volúmenes de datos lo cual influiría en la decisión sobre las tecnologías a usar para alojar el repositorio. Por otro lado, el hecho de que las interacciones entre los distintos involucrados en un ambiente de AGSD se extraigan y estén disponibles en un repositorio, podría representar para algunos una invasión a la privacidad, lo cual tendría que evaluarse con trabajadores de AGSD para definir cómo tratar este riesgo.

4.4.2 Características para el mecanismo de clasificación de bitácoras de UTEM

Este elemento de la condensación del AK se considera de gran importancia, ya que una correcta clasificación de conocimiento facilitaría su recuperación posterior, debido a que la búsqueda de conocimiento no se haría sólo sobre texto plano. Un aspecto a destacar sobre la

[19] https://developers.facebook.com/docs/messenger-platform
[20] https://www.cisco.com/c/en/us/products/unified-communications/jabber-software-development-kit/index.html
[21] https://dev.skype.com/
[22] https://api.slack.com/web

importancia del mecanismo de clasificación, es la recuperación de conocimiento a largo plazo, ya que al pasar el tiempo el conocimiento documentado (que se encuentra en las bitácoras de UTEM) sin clasificar puede perder contexto y significado. Esto se debe principalmente a que el conocimiento no está codificado con base en ningún estándar de representación que ayude a mantener la semántica al pasar el tiempo. Por otro lado, al no estar clasificado, sería más complicado recordar los términos mediante los cuales se pueda recuperar el conocimiento. En cambio, estando clasificado el conocimiento, se pudiera usar como referencia el esquema de clasificación utilizado en vez de tener que recordar los términos exactos de búsqueda.

En cuanto a la implementación del mecanismo de clasificación, ésta debe adecuarse a la dinámica que se vive en el ámbito del AGSD, es decir, no debería representar un gran esfuerzo en término de tareas y tiempo, de tal manera que los directamente involucrados lo pudieran considerar una carga extra y sin sentido. Esto es de gran relevancia, ya que los desarrolladores ágiles generalmente son reacios a las actividades que consideran como secundarias y poco creativas (Clear, 2003).

Resumiendo, una implementación del mecanismo de clasificación de bitácoras de UTEM, debe ser ligero para no representar una carga extra al ambiente de AGSD, pero su vez debe ser perdurable, es decir, que la semántica dada por dicha clasificación no se pierda o altere con el paso del tiempo.

4.4.3 Características para el mecanismo de búsqueda de AK

Una vez definidas las características para hacer accesibles las bitácoras de UTEM, así como las características del mecanismo de clasificación, resta describir como sería un mecanismo para la recuperación el AK. Para ello es importante tomar en cuenta las situaciones que se les presentan a los desarrolladores ágiles en entornos globales al intentar recuperar el AK de los UTEM, las cuales fueron obtenidas mediante el estudio de entendimiento (Borrego et al., 2016) previamente descrito (ver capítulo 4) . Por lo anterior, el mecanismo de recuperación de AK que se implemente debería contar con las siguientes características:

- **Recuperación por período y por autor o destinatario**. Los participantes del estudio de entendimiento reportaron que frecuentemente tratan de recuperar AK de los registros de UTEM basados en un período de tiempo (fechas), ya que generalmente no recuerdan el día exacto en el que dicho conocimiento fue compartido. Esto les consume mucho tiempo

ya que no todos los UTEM cuentan con filtros basados en fechas para recuperar conversaciones. De igual manera, es común que no recuerden exactamente quien compartió el conocimiento, lo cual dificulta su recuperación.

- **Recuperación por texto libre**. Siempre es conveniente contar con esta opción para dar flexibilidad a la manera de buscar conocimiento, y no restringirlo únicamente a las opciones antes expuestas. Por ejemplo, puede usarse cuando se recuerde una palabra en especial con la cual se crea que se pueda recuperar cierto conocimiento.
- **Recuperación basada en el mecanismo de clasificación**. Esta opción de recuperación de conocimiento está enfocada en los casos en los que no se recuerden exactamente los términos por los cuales se pueda recuperar el conocimiento. Se podrían usar los términos del esquema de clasificación que se defina para tomarlo como una base semántica para recuperar conocimiento, sobre todo cuando ha pasado tiempo considerable y no se tiene el mismo contexto mental.
- **Identificación de UTEM**. Del estudio de entendimiento también se obtuvo que se dificulta la recuperación del conocimiento al no recordar cual fue el UTEM por el cual se compartió el conocimiento. Gracias a la accesibilidad de las bitácoras de UTEM que se propone como parte de la condensación del AK, sería posible filtrar las interacciones por el medio en el cual sucedieron. De esta manera sólo sería necesario consultar un solo punto en vez de hacerlo en varios UTEM.

Con un mecanismo de búsqueda con estas características se esperaría que la recuperación de AK en las bitácoras de UTEM se facilite y se haga de manera más rápida. Además, se esperaría una reducción de interrupciones entre los miembros de los equipos, ya que habría una fuente de conocimiento que consultar antes de preguntar a algún compañero. Esto a su vez, tendría impacto en mantener una buena relación entre los miembros del equipo, ya que las interrupciones constantes podrían ser molestas para ciertas personas.

4.5. Resumen del capítulo

En este capítulo se presentó el concepto de condensación del AK, el cual surge a partir de la revisión de literatura presentado en el capítulo 3 y del estudio de entendimiento sobre el manejo del AK en el ambiente de AGSD presentado en capítulo 4. Los principales resultados de este capítulo incluyen:

- Definición de los conceptos de condensación y de vaporización del conocimiento en términos del modelo SECI.
- Extensión del modelo SECI (de manera conceptual y como diagrama de estados) para incluir la vaporización, la condensación y dos tipos de conocimiento explícito adicionales (documentado y formalizado).
- Contextualización del concepto de condensación del AK en el ámbito de AGSD, y la definición de los elementos que se deben contemplar para su implementación.
- Impacto potencial de la condensación de AK en el desarrollo global y ágil de software y su relación la efectividad de equipo.
- Definición preliminar de características de cada elemento de la condensación del AK para el desarrollo de una implementación del concepto.

En el siguiente capítulo se presenta un prototipo no funcional de un condensador de AK para AGSD, así como una evaluación sobre el mismo para determinar preliminarmente la viabilidad del concepto.

Capítulo 5.
Evaluación preliminar de la condensación del conocimiento arquitectónico

En el capítulo anterior, se definió el concepto de condensación de AK, el cual tiene como propósito reducir la vaporización del conocimiento en un ambiente de AGSD. Como primer paso hacia la validación del dicho concepto, en este capítulo se presenta el diseño preliminar de un prototipo de condensador de AK, así como su evaluación con desarrolladores con experiencia en AGSD. Además, con base en los resultados de esta evaluación y lo obtenido anteriormente, tanto en la revisión de literatura, como en el estudio en sitio sobre la compartición de AK en AGSD, se definen las características que debería cumplir una implementación tecnológica del concepto de condensación de conocimiento en AGSD.

5.1. Diseño preliminar de un condensador de conocimiento arquitectónico

En esta sección se presenta la primera aproximación hacia la implementación de un prototipo tecnológico que implemente el concepto de condensación de conocimiento. Dicho concepto consta de tres elementos donde uno de los más relevantes el mecanismo de clasificación de bitácoras de UTEM. Su relevancia reside en que este mecanismo sería la base para recuperar conocimiento sobre todo a largo plazo, ya que se añadiría una estructura semántica que perdure y no se desgaste con el tiempo. Además, se estableció que este mecanismo debería adaptarse al ámbito del AGSD, siendo ligero para no representar una carga extra considerable para los involucrados. En la siguiente parte se establece la propuesta para lograr la implementación del mecanismo de clasificación de bitácoras de UTEM, alrededor de la cual se diseña esta primera aproximación hacia un condensador de AK.

5.1.1 Etiquetado social como mecanismo de clasificación de conocimiento arquitectónico

Con el fin de cumplir con las características expuestas en el capítulo 5 para el mecanismo de clasificación, se decidió explorar la posibilidad de usar el etiquetado social, el cual se define como la práctica para generar etiquetas electrónicas o palabras clave por parte de usuarios,

en lugar de especialistas, como una forma de clasificar y describir el contenido en línea[23] (p. ej. #etiqueta). De esta manera los mismos involucrados clasificarían el AK al momento que lo estén compartiendo en los UTEM. Las ventajas de usar el etiquetado social como mecanismo de clasificación incluyen las siguientes:

1. Actualmente el etiquetado social es usado en las redes sociales para marcar contenido, siendo ampliamente aceptado y popular, por lo que ello podría facilitar su adopción en el ambiente de AGSD.
2. El etiquetado social no representaría un esfuerzo mayúsculo para los usuarios ya que la clasificación de las interacciones se realizaría al mismo tiempo en que éstas ser realizan, en parte debido a la misma popularidad de esta práctica.
3. Los usuarios tendrían la libertad (pero también la responsabilidad) de marcar y clasificar el AK que consideren pertinente, basado en la relevancia o importancia para ser recuperado posteriormente.

En la siguiente parte se presentan algunas soluciones reportadas en la literatura y las herramientas populares del mercado donde se usa el etiquetado social para la ingeniería de software.

5.1.1.1 Etiquetado social en soluciones para la ingeniería de software

El etiquetado social ha sido elegido por investigadores y compañías de software como una manera ligera y discreta de organizar datos dispersos o no estructurados (como registros de interacciones en UTEM) o para agregar significado y metadatos en entornos de desarrollo de software, con el objetivo de recuperar el conocimiento generalmente difícil de encontrar. A nuestro leal saber y entender, en la literatura se reportan siete herramientas que utilizan el etiquetado social en diferentes elementos durante distintas fases del ciclo de desarrollo del software (ver Tabla 5.1).

La mayoría de las herramientas de etiquetado de la Tabla 5.1 están diseñadas para apoyar en la fase de implementación, particularmente enfocadas al etiquetado de código fuente, componentes de software o entradas de control de versiones, es decir, permiten la administración del AK. Sólo se encontró una herramienta diseñada para etiquetar

[23] https://www.igi-global.com/dictionary/social-tagging/27505

Tabla 5.1. Herramientas reportadas en la literatura que utilizan etiquetado social en el desarrollo de software.

Herramienta	Cobertura			Etiquetado		Recuperación de conocimiento	Tipo de herramienta
	Ambiente	***Fases***	***Elementos a etiquetar***	***Mecanismo***	***Tipo***		
IBM® Rational® Jazz®(Treude and Storey, 2009)	Ágil Distribuido	Todas	Artefactos y elementos de trabajo	Auto-completado	Libre	Texto libre y etiquetas	Comercial
TagSEA(Storey et al., 2009)	Distribuido	I	Código fuente	Auto-completado	Libre	Basado en waypoints y etiquetas	Investigación Código abierto
Trac (Calefato et al., 2010)	Distribuido	I, P	Control de versiones y errores	Libre	Libre	Texto libre	Código abierto
eMoose(Dekel and Herbsleb, 2008)	No definido	I	Código fuente	Auto-completado	Libre	"Push" contextual	Investigación
CodeSnippets (Forward et al., 2007)	No definido	I	Fragmentos de código	Libre	Libre	Basado en etiquetas	Código abierto
Paul et al. tool (Paul et al., 2015)	Código abierto	D, I	Componentes de Software	Libre	Libre	Texto libre asociado a etiquetas	Investigación
TAGGER (Richter et al., 2004)	Distribuido	A	Interacciones en UTEM	Libre	Ligado a etiquetas base	No reportado	Investigación

(A = Análisis, D = Diseño, I = Implementación, P = Pruebas, M = Mantenimiento, Todas = Cubre todas las fases)

interacciones personales en UTEM (TAGGER (Richter et al., 2004)), como se propone en este trabajo. Sin embargo, ésta se enfoca en capturar sólo el conocimiento del dominio durante la fase de análisis. Además, la mayoría de estas herramientas están orientadas hacia un entorno de desarrollo distribuido (incluida la herramienta de (Paul et al., 2015), ya que el código abierto a menudo se hace de manera distribuida). Vale la pena destacar que el trabajo donde se utiliza IBM Rational Jazz (una herramienta muy robusta) fue el único en donde la evaluación se llevó a cabo en un entorno ágil.

Con respecto a la implementación del etiquetado, sólo tres herramientas tienen un mecanismo de autocompletado que ayude a recordar las etiquetas disponibles o las más frecuentemente usadas. Además, la mayoría de las herramientas utiliza etiquetas libres, es decir que no hay etiquetas fijas o predefinidas para asignar (en función de una jerarquía u ontología), por lo tanto, los usuarios pueden escribir o crear cualquier etiqueta. Esto podría ocasionar dificultades de etiquetado y problemas de recuperación de información causados por: explosión de etiquetas (un gran número de etiquetas), diferencias en la interpretación del

significado de una etiqueta, un contexto incompleto para entender una etiqueta, localidad de etiquetas (basadas en la jerga de un equipo), etiquetas compuestas (etiquetas que sólo tienen sentido cuando se usan juntas) y similitud oscura (etiquetas con el mismo significado pero escritas de manera diferente) (Bagheri and Ensan, 2016).

Además, la literatura reporta que los desarrolladores prefieren usar etiquetas libres en los sistemas de etiquetado porque agregan una carga cognitiva muy baja al trabajo diario (Storey et al., 2009; Treude and Storey, 2009), a pesar de los problemas mencionados anteriormente. Se han realizado varios esfuerzos para desarrollar mecanismos de auto-etiquetado (Sohan et al., 2010) o sistemas de recomendación basados en etiquetas (Al-kofahi et al., 2010; Bagheri and Ensan, 2016) con el objetivo de reducir la carga cognitiva de los desarrolladores. El mecanismo de etiquetado automático de (Sohan et al., 2010) relaciona los mensajes de correo electrónico con las historias de usuario en proyectos ágiles con una precisión del 70%; sin embargo, el 30% restante de error podría causar problemas en cuanto a la recuperación de conocimiento, al provocar que se obtenga conocimiento incorrecto. En el caso de los sistemas de recomendación de etiquetas, TagRec (Al-kofahi et al., 2010) y LS3 AutoTagger (Bagheri and Ensan, 2016) son enfoques prometedores con los cuales complementar la asignación de etiquetas y mejorar el mecanismo básico de autocompletado, sin embargo estos trabajos aun requieren de una mayor estabilidad.

Con el fin de hacer una exploración más amplia del uso del etiquetado social en la ingeniería de software, se realizó una búsqueda de herramientas actuales que apoyen en la comunicación durante el ciclo de construcción de software y que implementen algún tipo de etiquetado. Se encontraron distintas herramientas de comunicación de las cuales se seleccionaron seis que salieron mejor posicionadas en las búsquedas hechas en Google, ya que se considera que serían las más usadas o las más populares.

Como se puede observar en la Tabla 5.2, se encontraron herramientas mayormente enfocadas a la mensajería instantánea, manejo de tareas y tableros Kanban (Anderson, 2010), donde la mayoría cuentan con canales públicos y privados de comunicación y la capacidad

Tabla 5.2. Herramientas de comunicación enfocadas a la ingeniería de software.

Herramientas de comunicación	**Características particulares**	**Características compartidas**			
		Canales públicos y privados	***Compartir archivos***	***Buscador***	***Etiquetas***
Slack	Mensajero instantáneo	●	●	Texto libre	Canales de comunicación, libres
Basecamp	Tablón de mensajes, mensajero instantáneo y manejador de tareas.	●	●	Texto libre, personas, etiquetas	Libres
Asana	Mensajero instantáneo y manejador de tareas y de tablero Kanban	●	●	Texto libre, personas, etiquetas	Libres
Trello	Manejador de tablero Kanban		●	Texto libre, personas, etiquetas	Libres
Assembla	Manejador de versiones y de tareas, y de tablero Kanban	●	●	Texto libre, personas, etiquetas	Libres
Jira	Manejador de tablero Kanban e integrador de soluciones			Texto libre, personas, etiquetas	Libres

de compartir archivos (Slack[24], Basecamp[25], Asana[26], Trello[27], Assembla[28] y Jira[29]). Todas estas herramientas permiten el etiquetado de los elementos que manejan (tareas, mensajes, tarjetas Kanban, etc.) de forma libre, es decir, el usuario puede asignar cualquier palabra como una etiqueta sin restricción. Además, todos cuentan con un buscador el cual se basa en consultas a partir de texto libre que ingrese el usuario, así como en etiquetas y nombres de usuario (excepto Slack). Todas las herramientas cuentan con un componente de autocompletado en el campo de búsqueda, el cual ayuda a recordar las etiquetas creadas (y canales de comunicación en el caso de Slack) y los nombres de usuario existente, de tal manera que se conviertan en filtros adicionales al texto libre para realizar las búsquedas. Cabe destacar que el tipo de tareas y los mensajes que se comparten podrían contener AK, ya que estas herramientas están diseñadas para soportar la comunicación durante el ciclo de desarrollo de software. Sin embargo, ninguna de ellas está enfocada al manejo de dicho tipo

[24] https://slack.com/
[25] https://basecamp.com/
[26] https://asana.com/
[27] https://trello.com/
[28] https://www.assembla.com/
[29] https://es.atlassian.com/software/jira

de conocimiento, por lo que su recuperación dependería enteramente de la disciplina de organización (para crear etiquetas adecuadas, no repetirlas, escribirlas correctamente, etc.) y memoria de los involucrados (recordar para que es cada etiqueta y que significa). Además, dado que el etiquetado es libre, se puede ocasionar la explosión de etiquetas, similitud obscura, localidad de etiquetas, etc., lo cual dificultaría la recuperación de AK, como se mencionó en párrafos anteriores.

Finalizando esta sección, tanto en la literatura como en herramientas de comunicación para proyectos de software se usa el etiquetado social, como una forma ligera de clasificar diversos elementos. Se presentaron herramientas citadas en trabajos de investigación, donde se usa el etiquetado social para organizar AK generalmente producido en la fase de implementación. En el caso de las herramientas populares para la comunicación durante el ciclo de software, todas usan el etiquetado social, pero no están enfocadas al manejo de dicho tipo de conocimiento. Todas las herramientas presentadas usan el etiquetado libre (excepto TAGGER (Richter et al., 2004)), lo cual no se considera óptimo para que la semántica de las etiquetas perdure más allá del tiempo que duren los proyectos en desarrollo.

En la siguiente sección se presenta un prototipo no funcional de un condensador de AK donde se consideran las características de las herramientas presentadas en esta parte.

5.1.2 Prototipo preliminar de condensador de conocimiento arquitectónico

Tomando en cuenta las características y oportunidades de mejora de las herramientas expuestas en la sección anterior, y con el fin de evaluar preliminarmente el concepto de condensación de conocimiento, se desarrolló un prototipo no funcional de condensador de AK para AGSD usando HTML[30], CSS[31] y Javascript[32]. El eje principal de este prototipo es el uso del etiquetado social como mecanismo ligero para la clasificación de bitácoras de UTEM. Cabe destacar que para este prototipo no se consideró ningún elemento para la accesibilidad las bitácoras de UTEM, ya que se considera un tema que dependería mucho de los UTEM que se seleccionen al momento de realizar una implementación funcional. A continuación, se presenta la funcionalidad proyectada para el mecanismo de clasificación de

[30] https://www.w3.org/TR/html52/
[31] https://www.w3.org/Style/CSS/
[32] https://www.javascript.com/

bitácoras de UTEM y para el de recuperación de AK, así como los bocetos de interfaz de usuario desarrollados para cada uno de ellos.

5.1.2.1 Mecanismo de clasificación de bitácoras de UTEM basado en etiquetas y meta-etiquetas.

Con base en lo que se reporta en la literatura, en las herramientas de comunicación para ingeniería de software presentadas anteriormente y en las condiciones del AGSD, se propone un mecanismo de clasificación basado en etiquetado social semifijo, es decir, que permita a los usuarios agregar etiquetas personalizadas, donde cada una de ellas esté relacionada a una meta-etiqueta predefinida. Esta meta-etiqueta a su vez sería parte de un conjunto de meta-etiquetas las cuales representen el AK que comparten los desarrolladores a través de UTEM en los ambientes de AGSD, lo cual fue obtenido en un estudio anterior (Borrego et al., 2016) (ver Figura 5.1, cada entidad actuaría como una meta-etiqueta).

Las etiquetas de usuario se registran mediante un sistema administrador de etiquetas y ahí mismo se relacionarían con sus respectivas meta-etiquetas (ver Figura 5.2), las cuales son fijas e inalterables. Las etiquetas de usuario se registrarían en alguna de las ceremonias predefinidas por la metodología ágil que se esté utilizando, por ejemplo, si se tratara de Scrum, los usuarios podrían registrar las etiquetas durante la reunión de planeación de Sprint, e incluso ir agregando otras durante el Sprint en las reuniones diarias, en caso de ser necesario.

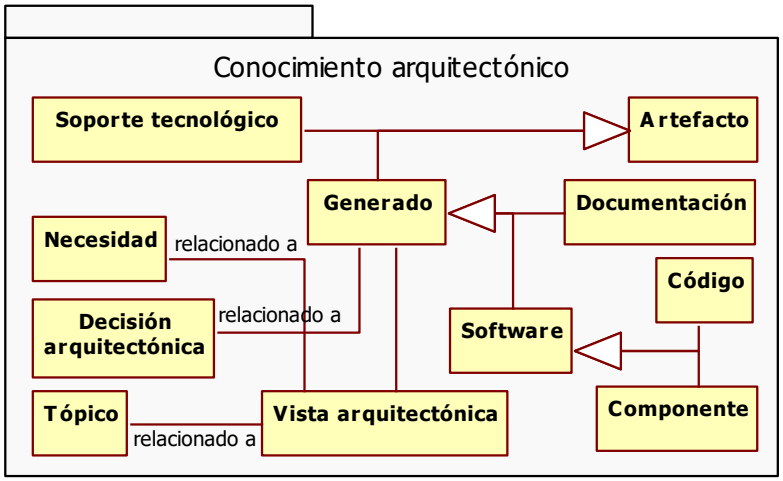


Figura 5.1. Ontología que representa los aspectos involucrados en la articulación de AK a través del uso de UTEM por equipos de AGSD.

Figura 5.2. Interfaces de usuario preliminares del sistema de administración de etiquetas para la condensación de AK.

Las ventajas de esta propuesta de etiquetado social basado en meta-etiquetas con un significado fijo incluyen:

1. Al controlar la creación de etiquetas se reduciría la explosión de etiquetas, ya que no se podría agregar etiquetas a placer durante las interacciones en UTEM.
2. Las diferencias en la interpretación del significado de etiquetas y la similitud oscura se reduciría, ya que cada equipo acordaría las etiquetas a registrar, por lo que todos estarían de acuerdo con el significado de cada una de las etiquetas.
3. Los problemas de localidad de etiquetas y de no contar con el contexto completo para entender alguna etiqueta también se reduciría gracias a la existencia de las meta-etiquetas. Lo anterior debido a que ésas cuentan con una semántica fija, la cual podría ser de ayuda para deducir/comprender el significado de las etiquetas relacionadas.
4. El conjunto de meta-etiquetas proporcionaría un medio más abstracto y estructurado para encontrar AK que se haya almacenado. De esta manera cuando un usuario desee recuperar conocimiento, pero no recuerde el nombre exacto de una etiqueta, pudiera usar una meta-etiqueta para realizar una búsqueda inicial. Por ejemplo, la etiqueta #pythonTricks podría estar relacionada con la meta-etiqueta #Código, de modo que cuando un desarrollador desee recuperar AK, pero no recuerde el nombre exacto de la etiqueta, podría usar la meta-etiqueta para realizar una primera búsqueda.
5. Gracias a la semántica fija de las meta-etiquetas, la semántica de las etiquetas de usuario relacionadas a ellas podría conservarse a través del tiempo. Por otro lado, al utilizar etiquetas totalmente libres, su significado podría erosionarse al pasar el tiempo al no contar con el “ancla semántica” que representan las meta-etiquetas.

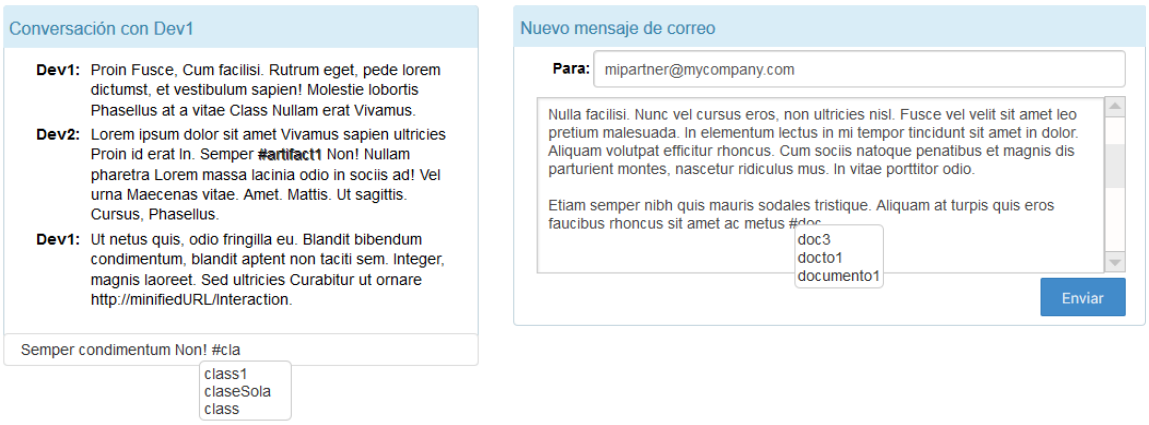


Figura 5.3. Ejemplos de usa del asistente de etiquetado en un mensajero instantáneo y en un mensaje de correo electrónico.

Este mecanismo de etiquetado sería usado durante el transcurso de las interacciones en UTEM, por ejemplo: al estar escribiendo un correo electrónico, al estar usando un mensajero instantáneo, al estar dejando un mensaje en algún foro, etc. Sin embargo, todo sistema de etiquetado manual está propenso al error humano. Pueden producirse errores de captura al etiquetar algún mensaje de la interacción, o pudiera olvidarse la forma exacta en que se registró cada etiqueta, o incluso la existencia de algunas de ellas. Por ello, con el fin de apoyar a los usuarios a etiquetar durante las interacciones en UTEM, se propone agregar un asistente de etiquetado que complete automáticamente las etiquetas (de forma alfabética) mientras los usuarios escriben en una conversación. En la Figura 5.3 se ejemplifica como sería el uso del asistente de etiqueta durante una conversación de mensajero instantáneo, y al estar escribiendo un mensaje de correo electrónico. Con esto se pretende que se reduzca aún más la explosión de etiquetas, ya que se disminuirían los problemas de captura que provocan etiquetas incompletas o mal escritas.

Resumiendo la propuesta de mecanismo de clasificación, ésta se compone de los siguientes elementos: (1) conjunto de meta-etiquetas, basadas en el modelo obtenido en el estudio de entendimiento; (2) sistema de administración de etiquetas, para registrar las etiquetas de usuario y relacionarlas con las meta-etiquetas; y (3) asistente de etiquetado, para apoyar a los usuario mientras etiquetan a través de la técnica de auto-completado.

5.1.2.2 Mecanismo de recuperación de conocimiento arquitectónico

Tomando en cuenta las características definidas en el capítulo 5 para el mecanismo de recuperación de AK, se desarrolló una interfaz de usuario con dos campos para ingresar los

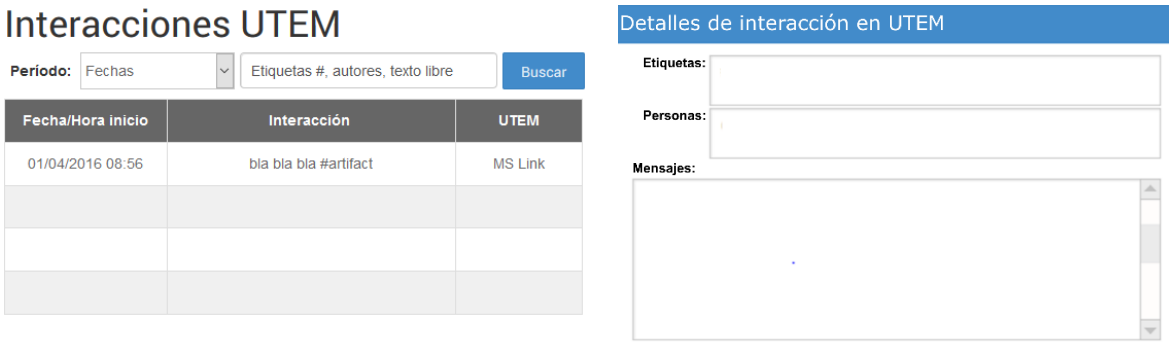


Figura 5.4. Interfaces de usuario de buscador de AK y detalles de interacción

parámetros de búsqueda (ver Figura 5.4). En el campo marcado como período se le permitiría al usuario elegir un rango de fechas deseado para buscar mensajes enviados, y en el campo restante se ingresaría en forma de texto libre: etiquetas, autores, destinatarios, nombres de UTEM o cualquier otro texto por el cual se quiera encontrar algún mensaje. Al presionar el botón de buscar se desplegarían los mensajes que coincidan con los parámetros de búsqueda. Los campos relativos a los mensajes serían la fecha y la hora del mensaje, un fragmento del mensaje y el UTEM fuente del mensaje.

Se sabe que sólo con un mensaje etiquetado es probable que no se tenga el AK deseado, se requeriría de un contexto más amplio para hacerlo. Por ello se plantea que al seleccionar uno de los elementos resultantes de la búsqueda, se despliegue el detalle del entorno del mensaje seleccionado, mostrando las etiquetas relacionadas, las personas involucradas en la interacción y los mensajes de la interacción en donde se encuentra el mensaje etiquetado que fue seleccionado (ver Figura 5.4).

5.1.3 Escenario de uso proyectado

Con el fin de clarificar como se proyecta el uso del prototipo de condensador de AK descrito en la sección anterior, se realizó una gráfica rica que representa el uso proyectado del prototipo en un ambiente de AGSD (ver Figura 5.5). A continuación, se describe el proceso representado en dicho escenario.

En primera instancia, los desarrolladores (remotos y locales) durante la reunión de planeación de un ciclo de desarrollo (un Sprint en el caso de la metodología Scrum) acordarían las etiquetas a usar durante el ciclo a iniciar y las registrarían en el sistema de administración de etiquetas, relacionando cada una a la meta-etiqueta (o incluso a otra

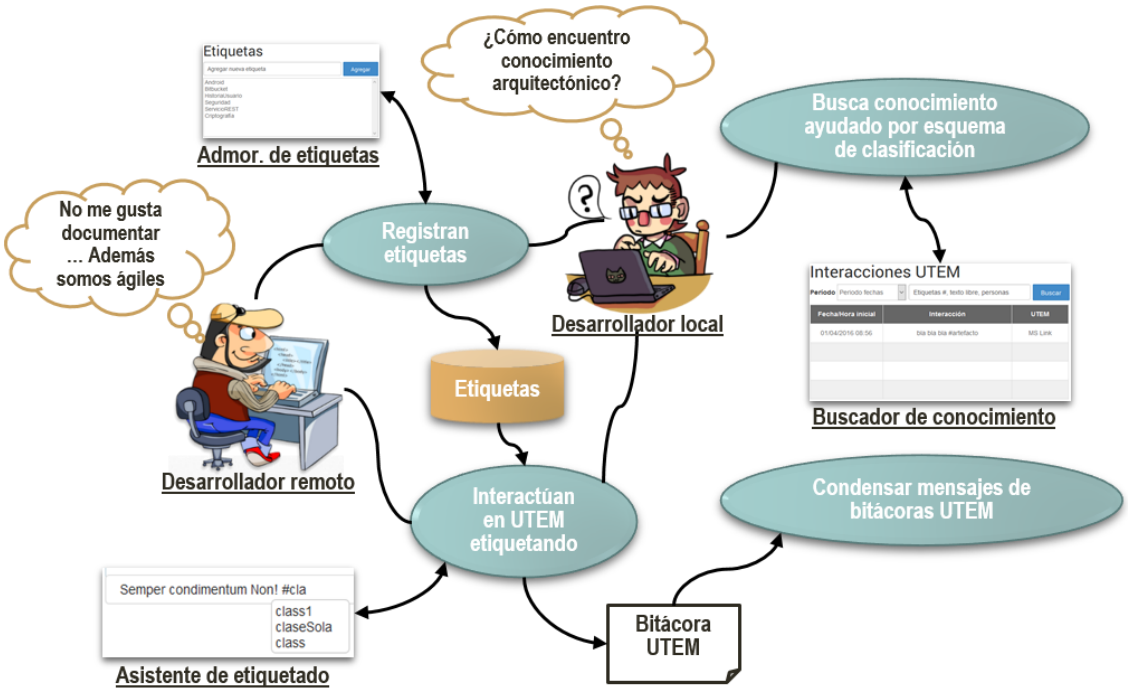


Figura 5.5. Escenario de uso del prototipo preliminar de condensador de AK.

etiqueta de usuario) que consideren más adecuada. Además de registrar etiquetas, podrían eliminar aquellas etiquetas que consideren que no les fueron útiles en los ciclos anteriores.

Posteriormente, durante el transcurso del ciclo de desarrollo, los desarrolladores interactuarían mediante UTEM, y etiquetarían (usando el asistente de etiquetado) aquellos mensajes que crean pertinente recuperar posteriormente, dada la relevancia arquitectónica de conocimiento compartido. De esta manera, en las bitácoras de UTEM se registrarían mensajes marcados con las etiquetas que los desarrolladores registraron. Estos mensajes se condensarían a través de un procedimiento tecnológico hasta ahora no definido.

Finalmente, cuando alguno de los desarrolladores tenga necesidad de recordar sobre algún tema arquitectónico, pero no recuerde exactamente donde, ni cuando y como se compartió, usaría el buscador de AK explotando las opciones de búsqueda que este ofrece, en vez de buscar directamente en los UTEM donde crea el desarrollador que se haya compartido el conocimiento.

5.2. Evaluación de prototipo no funcional de condensador de conocimiento arquitectónico

Como se mencionó anteriormente, la razón principal de desarrollar un prototipo no funcional de un condensador de AK, fue realizar una evaluación preliminar para validar el concepto de condensación de AK. En esta sección se presenta el método, los resultados y la discusión sobre los mismos, de dicha evaluación.

5.2.1 Método

El objetivo de esta evaluación fue obtener retroalimentación de trabajadores del AGSD sobre la propuesta de solución tecnológica para el manejo de AK vertido en conversaciones de UTEM, para pasar de prototipos no funcionales (bocetos), a una implementación real de la propuesta.

Los participantes fueron 10 desarrolladores de AGSD de dos empresas mexicanas: EMCOR[33] (Cd. Obregón, Sonora) con 7 participantes y Softtek[34] (Ensenada, Baja California) con 3 participantes, con una edad promedio de 27 años (desviación estándar = 1.9), con experiencia en desarrollo ágil de 2 años en promedio (desviación estándar = 0.76), experiencia en GSD de 3.3 años en promedio (desviación estándar = 1.26).

El procedimiento de evaluación se llevó a cabo en dos sesiones, una por cada empresa, y se compone por dos grandes partes (ver Figura 5.6): contextualización y co-diseño, las cuales se detallan a continuación.

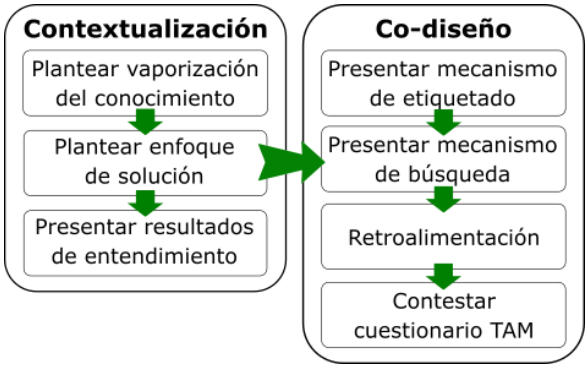


Figura 5.6. Esquema de metodología para evaluación de prototipo no funcional.

[33] http://emcor.mx/
[34] https://www.softtek.com/

- **Contextualización** (15 min. de duración). En esta parte se planteó a los participantes el problema de la vaporización del AK en ambientes de AGSD, para luego plantear el enfoque de solución que se propone, es decir, usar las entradas de UTEM y el etiquetado social para clasificarlas. Además, se presentaron los resultados del estudio de entendimiento (presentado en el capítulo 4), con el fin de darle fuerza a la propuesta de solución.
- **Co-diseño** (45 min. de duración). En esta parte de la evaluación se le presentó a los participantes la funcionalidad y las interfaces de usuario relacionadas con el mecanismo de clasificación de bitácoras de UTEM, basado en el etiquetado social. Posteriormente se presentó la funcionalidad y las interfaces de usuario relacionadas con el mecanismo de recuperación de conocimiento, para luego escuchar la retroalimentación de los participantes respecto al prototipo presentado (esta última parte fue grabada en audio). Finalmente, los participantes contestaron un cuestionario TAM (Davis, 1989) (Likert-7) para obtener su percepción de usabilidad y facilidad de uso.

5.2.2 Resultados

Los archivos de audio obtenidos en las sesiones de evaluación se transcribieron y se obtuvieron los comentarios y sugerencias más relevantes al respecto de cada mecanismo presentado, así como los comentarios generales sobre el prototipo para condensación del AK.

En general los participantes consideraron una muy buena idea el hecho de integrar todas las fuentes de información en donde se pueda compartir AK, ya que es frecuente que no recuerden ni cuando, ni donde les compartieron cierta información, y terminan preguntando a la persona que tiene el conocimiento, lo cual a veces implica tiempo de espera para que se encuentre disponible y/o para recibir la respuesta. Los participantes también consideraron viable la propuesta porque generalmente se determinan UTEM oficiales para la comunicación de los proyectos, por lo que es poco probable que se encuentren demasiados temas personales en las respectivas bitácoras.

En cuanto al uso del etiquetado social para clasificar AK, los participantes lo vieron como algo viable e importante, ya que así se tendría un punto de referencia para encontrar el conocimiento. Algunos participantes refirieron que Jira[35] tiene elementos de configuración

[35] https://es.atlassian.com/software/jira

que actúan como etiquetas, ya que si se busca dicho elemento se puede saber todo lo que ha pasado con él. También comentaron que mantener la lista de etiquetas podría ser un problema, porque los temas van cambiando todos los días. Una duda que les surgió a los participantes al ver los bocetos presentados fue saber cómo se iban a relacionar las etiquetas a los distintos proyectos, ya que cada proyecto tendría temáticas diferentes.

Respecto al mecanismo de recuperación de AK, también fue bien apreciado el hecho de contar con búsquedas por período de fechas, ya que muchos de los UTEM que usan actualmente no cuentan con ello. También, los participantes consideran que con esto un mismo equipo o toda la empresa se enteraría de lo que pasa en cada proyecto, lo cual puede ayudar a que el conocimiento de los distintos proyectos pueda ser útil para todos. Sin embargo, los participantes consideraron que con el buscador se tendría una manera fácil de referir lo que dijo una persona, lo cual podría causar ciertos problemas sociales, que a su vez podrían ayudar a mejorar el comportamiento del equipo, ya que ahora se tendría que tener certeza de lo que se comparte por UTEM. Otro problema que lograron visualizar es que muchas de las interacciones ocurren por teléfono, por lo que no habría manera de condensar ese conocimiento, sin embargo los participantes que refirieron esto tienen clientes que hablan español, por lo que la barrera del lenguaje no existe con ellos; en cambio otros participantes refirieron que las llamadas telefónicas son el último recurso de comunicación, siempre se prefiere comunicarse por UTEM, debido a que su cliente habla inglés.

Siguiendo con el tema de la recuperación de conocimiento, los participantes mostraron cierta preocupación sobre cómo se obtendrían las conversaciones a través del buscador, por ejemplo: si la etiqueta se encuentra en medio de la conversación la pregunta expresa fue si se recuperaría la conversación entera. Al respecto otro participante sugirió que se podría crear una ventana de tiempo alrededor del mensaje etiquetado, de esta manera no se tendría que traer toda la conversación. Este punto se consideró importante, ya que si el buscador no es preciso en mostrar sólo la información que se requiere, sería más fácil ir directamente al UTEM correspondiente y usar los mecanismos de búsqueda que este ofrezca.

Por otro lado, los participantes refirieron situaciones cotidianas que les suceden por la vaporización del AK, la deuda de documentación y dispersión del conocimiento en distintos UTEM. Algunos comentaron que muchas veces tienen los mismos problemas técnicos/arquitectónicos, y lo resuelven al final de cuentas de la misma manera en la que

Tabla 5.3. Estadística descriptiva de los resultados del cuestionario TAM (Likert-7).

	Mediana	Moda	Q.25	Q.50	Q.75
Percepción de utilidad	6	6	5	6	6
Percepción de facilidad de uso	6	7	5	6	7

otros lo hicieron, pero cada quien aprendió por su cuenta. Esto se podría evitar si se tuviera una base de AK fácil de consultar. También refieren que cuando alguien renuncia se lleva el conocimiento consigo y luego ya nadie sabe a ciencia cierta lo que manejaba dicha persona. Algunos participantes comentaron que se han hecho esfuerzos para implementar una base de conocimiento y una red social interna para compartir AK, pero al final nadie los usa porque su manejo requiere de un gran esfuerzo extra, que implica un monto considerable de tiempo que no están dispuestos a invertir. Por ello es que los participantes dicen que: "*entre más fácil e integrada esté la herramienta que se vaya a ofrecer (condensado de conocimiento), va ser mejor su adopción, ya que evitaría meter cosas nuevas al trabajo diario*".

Finalmente, los participantes expresaron su percepción respecto a la facilidad de uso y la utilidad del prototipo contestando el cuestionario TAM. Como se observa en la Tabla 5.3, los participantes tienen una alta percepción de utilidad y de facilidad de uso, habiendo una mayor percepción de facilidad de uso que de utilidad (la moda de facilidad de uso y el cuartil 75 es de 7 en la escala Likert).

5.2.3 Discusión

Los resultados obtenidos reflejan que el problema de la vaporización del AK y la deuda de documentación es un problema palpable para los desarrolladores de AGSD, por lo que la propuesta de rescatar AK de las bitácoras de UTEM les pareció una buena opción para disminuir dichos problemas. La integración de fuentes de conocimiento (bitácoras de UTEM) fue uno de los puntos más apreciados, ya que reportan dedicar mucho tiempo buscando información en los historiales de mensajeros y correos electrónicos principalmente.

En cuanto al mecanismo de clasificación de bitácoras de UTEM basado en etiquetado social, no se percibió signo alguno de incomodidad por etiquetar las interacciones de UTEM, ni por el hecho de usar etiquetas predefinidas (confirmado con los resultados del cuestionario TAM) a diferencia de lo que reporta la literatura (Storey et al., 2009; Treude and Storey, 2009). Sólo se expresaron dudas sobre cómo mantener actualizada la lista de etiquetas

disponibles dado el dinamismo de temas que suele haber en un ambiente de AGSD. Al respecto, simplemente se podría ingresar al sistema de administración de etiquetas durante la reunión diaria de reporte de estado y hacer los ajustes necesarios a la lista de etiquetas. Sin embargo, es conveniente que este mecanismo se evalúe funcionalmente, para que la percepción de los participantes sea a partir de un uso real.

Los participantes también recibieron bien la propuesta del mecanismo de recuperación de conocimiento (también confirmado con los resultados del cuestionario TAM), e hicieron observaciones que conducen a definir mejor la funcionalidad del buscador. A partir de la percepción de los participantes se puede determinar que la búsqueda por períodos de fechas es una funcionalidad que debe mantenerse. Además, la manera de presentar el detalle de una interacción debe hacerse sin saturar de información a los usuarios, lo cual podría ocasionar que no sea práctico el buscador y que se prefiera buscar directamente en el UTEM correspondiente.

En general los participantes refirieron que el prototipo de condensador de conocimiento que se implemente, no debería representar una obstrucción a la dinámica de trabajo actual, es decir, que no represente un trabajo extra que al paso del tiempo se deje de hacer por la dinámica y presiones de tiempo que existen generalmente en un entorno de AGSD, como sucede actualmente con la documentación (Sneed, 2014).

Resumiendo, existe una necesidad de AK durante los ciclos de desarrollo, pero la dinámica de trabajo no permite realizar una documentación adecuada para no perder tanto tiempo buscando conocimiento previamente compartido. Los resultados de esta evaluación permiten determinar que la condensación de AK podría ser viable en un ambiente de AGSD, tomando en cuenta el prototipo no funcional presentado. Con el fin de confirmar esta percepción de viabilidad en los siguientes capítulos se presentan distintas evaluaciones sobre prototipos funcionales que implementan los distintos elementos de la condensación de conocimiento.

5.3. Resumen del capítulo

En este capítulo se presentó un prototipo no funcional de un condensador de AK para AGSD, el cual basa la clasificación de conocimiento en el etiquetado social. Además, se presentó una evaluación de dicho prototipo con desarrolladores de AGSD a fin de obtener su

percepción acerca de la viabilidad del mismo. Los principales resultados de este capítulo incluyen:

- Justificación de la elección del etiquetado social como mecanismo de clasificación de bitácoras de UTEM, dadas las condiciones del AGSD.
- Revisión bibliográfica de las soluciones para ingeniería de software que usan el etiquetado social, donde se evidencia lo siguiente: (1) en la literatura sólo se encontró un trabajo en donde se hace etiquetado de interacciones de UTEM, pero enfocado en capturar conocimiento del dominio durante la fase de análisis (aún sin mecanismo de recuperación); (2) la mayoría las herramientas no utiliza etiquetas fijas o predefinidas, lo cual puede ocasionar problemas como: la explosión de etiquetas, diferencias en la interpretación de etiquetas y similitud oscura, entre otros. Sin embargo, el etiquetado libre es preferido por los desarrolladores; (3) Sólo tres las herramientas tienen un mecanismo de autocompletado que ayude a recordar las etiquetas disponibles.
- Descripción de una propuesta de mecanismo de clasificación compuesta por: (1) conjunto de meta-etiquetas, (2) sistema de administración de etiquetas y (3) asistente de etiquetado.
- Presentación de los bocetos de un prototipo no funcional de un condensador de AK, basado en el etiquetado social y la descripción de un escenario de uso de dicho prototipo en un ambiente de AGSD.
- Resultados de la evaluación del prototipo no funcional con desarrolladores de AGSD, confirmando de manera preliminar la viabilidad del concepto de condensación de AK.

En el siguiente capítulo se presenta una propuesta, y evaluación, para la implementación del mecanismo de clasificación de bitácoras de UTEM, el cual es un elemento crucial para el éxito del concepto de condensación de conocimiento. La evaluación de esta propuesta ayuda a determinar una manera de implementar un prototipo de condensador de AK en AGSD.

Capítulo 6.
Evaluación del etiquetado social como mecanismo de clasificación de conocimiento arquitectónico

En el capítulo anterior se determinó de manera preliminar la viabilidad del concepto de condensación de AK, a través de la evaluación de un prototipo no funcional. El mecanismo de clasificación de bitácoras de UTEM de dicho prototipo está basado en el uso de etiquetado social semi-fijo y asistido, es decir, se tiene una estructura de meta-etiquetas fijas a las cuales los usuarios les asocian etiquetas personalizadas, las cuales son recordadas por un componente de auto-completado al momento de estar escribiendo mensajes en un UTEM determinado.

Aunque la literatura (Storey et al., 2009; Treude and Storey, 2009) reporta que los desarrolladores prefieren el etiquetado libre, al evaluar el prototipo no funcional, los participantes de la evaluación presentada en el capítulo anterior, no mostraron ningún signo de molestia cuando se les indicó que se usarían etiquetas semi-fijas para clasificar conocimiento. Con el fin de determinar fehacientemente la viabilidad del etiquetado social para la clasificación de AK, en este capítulo se presenta la implementación tecnológica del mecanismo de clasificación de bitácoras de UTEM basado en etiquetado social, así como el diseño de su evaluación con desarrolladores de AGSD y los resultados obtenidos.

6.1. Implementación del prototipo de mecanismo de clasificación de bitácoras de UTEM.

Esta implementación del mecanismo de clasificación de bitácoras de UTEM se basó en los bocetos del prototipo no funcional presentado en el capítulo anterior. Por lo tanto, este mecanismo consta de: (1) un conjunto de meta-etiquetas, basadas en el modelo obtenido en el estudio de entendimiento; (2) un sistema de administración de etiquetas, para registrar las etiquetas de usuario y relacionarlas con las meta-etiquetas; y (3) un asistente de etiquetado, para apoyar a los usuarios mientras etiquetan a través de la técnica de auto-completado

Para este prototipo se decidió desarrollar el asistente de etiquetado con un conjunto de etiquetas embebido, es decir, sin que hubiera necesidad de contar con un sistema de administración de las mismas. Esto debido a que el foco de la evaluación estaría centrado en

la interacción con el asistente y no con el administrador de etiquetas, ya que se consideró la acción de etiquetado como la parte crucial para el éxito de este enfoque de clasificación de AK.

En cuanto a la tecnología usada para la implementación de este prototipo, se decidió que todo estuviera basado en Web para facilitar el proceso de evaluación, ya que no sería necesario instalar algún componente para que los potenciales usuarios evaluaran el asistente en cualquier equipo de cómputo. Además, con el propósito de no tener inconvenientes para integrar el asistente de etiquetado a algún UTEM, se decidió desarrollar un mensajero instantáneo expresamente para efectos de la evaluación del mecanismo de clasificación, el cual se denominó “Chatagger”. De esta manera se tendría total control del ambiente para poder desplegar el asistente y hacer las tareas de autocompletado.

Respecto a la funcionalidad, se implementaron cuatro casos de uso (ver Apéndice A para ver el diagrama completo), en donde los más significativos para la evaluación fueron: Enviar mensaje y autocompletar etiqueta, los cuales se describen a continuación. El usuario, al estar escribiendo un mensaje en Chatagger, puede activar el asistente de etiquetado introduciendo la combinación de caracteres: “#” más una letra (e.g. "#a"). Después, se muestra una lista de etiquetas sugeridas basadas en la lista embebida en el asistente, tanto etiquetas de usuario como meta-etiquetas (ver Figura 6.1), las cuales se siguen mostrando mientras (1) no se envíe el mensaje, (b) se elija una etiqueta sugerida, o bien, (c) se teclee un espacio para iniciar otra palabra.

Cuando el usuario selecciona la etiqueta que mejor se ajuste al tema de conversación, esta se inserta en el texto del mensaje actual. La selección y navegación entre las sugerencias mostradas puede hacerse usando el ratón (doble click sobre la opción deseada) o usando el

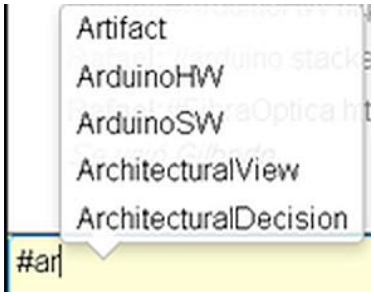


Figura 6.1. Asistente de etiquetado en Web mostrando sugerencias.

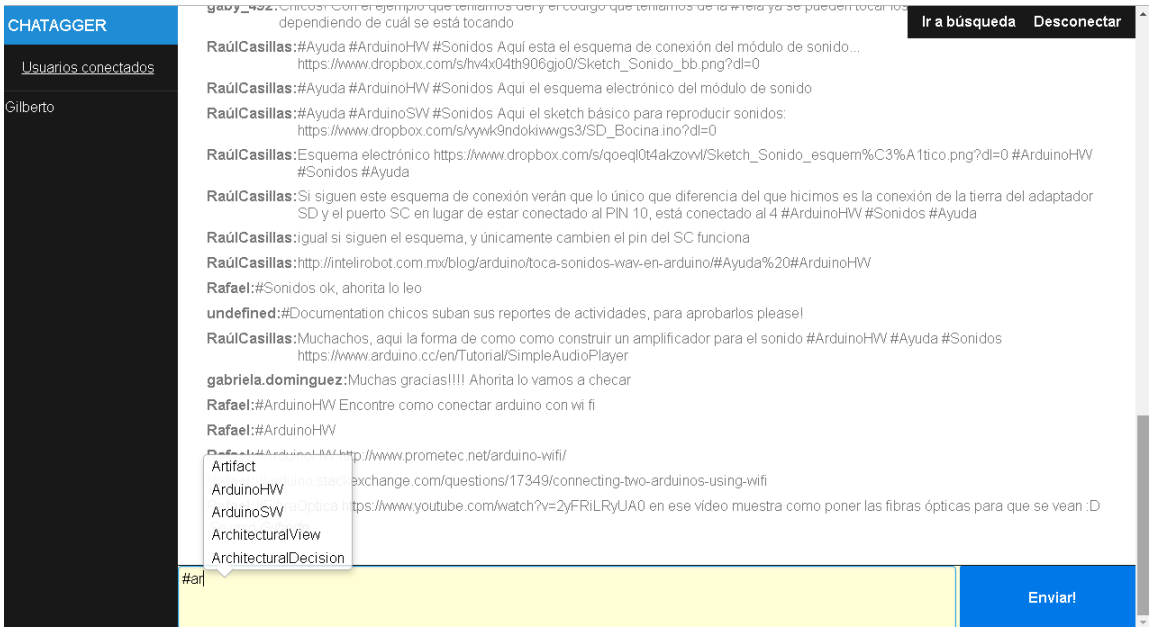


Figura 6.2. Pantalla de conversación de Chatagger.

teclado (usando las flechas arriba/abajo para navegar por la lista de etiquetas y la tecla <enter> para seleccionarla).

Finalmente, el usuario envía el mensaje, y el asistente de etiquetado elimina las etiquetas inexistentes, es decir, las que no sean parte de la lista embebida, antes de que el mensaje sea mostrado a los otros usuarios conectados y antes de que sea guardado en la base de datos. En la Figura 6.2 se muestra una pantalla de conversación del chat, para ilustrar el funcionamiento descrito. En el Apéndice A se puede consultar una especificación más técnica y detallada de la funcionalidad presentada.

6.2. Evaluación del prototipo de mecanismo de clasificación de bitácoras de UTEM

En esta sección se presenta la manera en la que evaluó el mecanismo de clasificación de bitácoras de UTEM, los resultados obtenidos y una breve discusión acerca de los mismos.

6.2.1 Método

Esta evaluación se planeó para obtener la percepción de los usuarios sobre el etiquetado social durante las interacciones de UTEM con apoyo del asistente. Para ello se enfocó en obtener la percepción de usabilidad y la intención de uso del mecanismo de etiquetado, así como a observar la manera en la que se etiquetan las interacciones de los participantes.

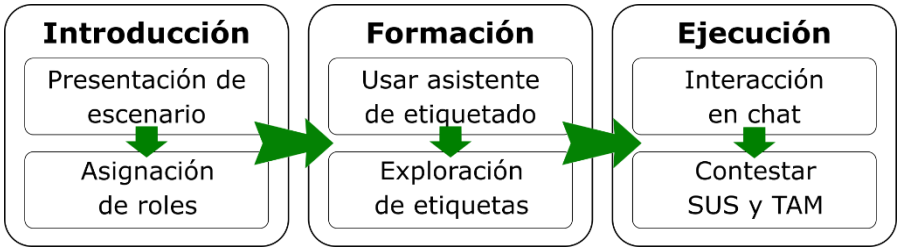


Figura 6.3. Procedimiento de evaluación de mecanismo de clasificación.

Los participantes fueron 30 desarrolladores de cinco empresas mexicanas de desarrollo de software, o bien, con equipos de desarrollo de software interno (EMCOR[36], TUFESA[37], RepoBox[38], Sahuaro Labs[39], Softtek[40]). La evaluación se llevó a cabo en sitio, es decir, en las instalaciones de cada una de las empresas participantes. Los desarrolladores contaban con una edad promedio de 27.2 años (desviación estándar = 3.2), con un promedio de 4.7 años de experiencia en ingeniería de software (desviación estándar = 3.8), 2.3 años de experiencia en desarrollo ágil (desviación estándar = 1.1) y 1.3 años de experiencia en desarrollo de software distribuido y/o global (estándar = 0.8).

El procedimiento de evaluación se dividió en tres partes: introducción, formación y ejecución (ver Figura 6.3). A continuación, se describe cada una de las partes del procedimiento de evaluación.

1. **Introducción**. Se le explicó a cada par de participantes el problema de la vaporización de AK en AGSD, y la solución propuesta basada en el etiquetado de interacciones. Posteriormente, se les pidió a los participantes que se asumieran en un escenario en el que ambos trabajan para diferentes empresas de software (A y B) y que están colaborando en el mismo proyecto. La empresa B depende de un servicio RESTful (Fielding and Taylor, 2002) que está desarrollando la empresa A, y todavía no hay documentación sobre ese servicio. El desarrollador de la empresa B debe obtener AK del servicio del desarrollador de la compañía A (ver detalles del escenario en Apéndice C). Después se asignaron los roles de desarrollador de empresa A y B respectivamente a la pareja de participantes.

[36] http://emcor.mx/
[37] http://www.tufesa.com.mx/
[38] https://www.repobox.com.mx/
[39] http://www.sahuarolabs.com/
[40] https://www.softtek.com/

2. **Formación**. En esta parte se les presentó Chatagger y el asistente de etiquetado. Los participantes usaron libremente ambos elementos durante dos minutos para familiarizarse con el funcionamiento, y al mismo tiempo se hizo una breve exploración de las etiquetas y meta-etiquetas disponibles.
3. **Ejecución**. Se ubicó a los participantes en sitios distintos para emular sitios distantes y que interactuaran mediante Chatagger siguiendo un guion (uno por integrante de cada pareja) en el que se simulaba una situación de dudas técnicas y arquitectónicas (ver detalles de los guiones en Apéndice C). Al estar siguiendo el guion correspondiente, cada participante hace tres preguntas relacionadas con el servicio RESTful (Fielding and Taylor, 2002), y se le presentan tres partes donde se le sugiere al participante que se etiquete el mensaje actual, sin especificar el nombre exacto de la etiqueta. Para ello, se añadieron cuatro etiquetas personalizadas al asistente de etiquetado, las cuales estaban relacionadas con el tema de los guiones. Estas etiquetas fueron: #Seguridad, #ManejarErrores, #HistoriaUsuario y #ReglaNegocio. También, se les indicó a los participantes que podrían etiquetar en cualquier parte donde lo consideraran necesario (teniendo en mente el objetivo del etiquetado, es decir, la recuperación posterior del conocimiento). La interacción mediante Chatagger tuvo una duración promedio de 20 minutos, y al concluir esta actividad los participantes respondieron cuestionarios SUS (Brooke, 1996) y TAM (Davis, 1989), con el fin de obtener sus percepciones de usabilidad e intención de uso. Finalmente, se les pidió a los participantes un comentario o sugerencia sobre el asistente de etiquetado, como mecanismo para clasificar bitácoras de UTEM.

6.2.2 Resultados

En esta sección se presentan los resultados obtenidos divididos en dos aspectos: comportamiento de etiquetado y la percepción de usabilidad e intención de uso de los participantes.

6.2.2.1 Comportamiento de etiquetado

Para obtener este comportamiento se hizo un conteo del número de etiquetas usadas por cada participante, distinguiendo entre etiquetas de usuario (las cuatro que se agregaron para esta evaluación) y meta-etiquetas, e identificando cuándo se usaron de manera correcta, es decir, que la semántica de la etiqueta coincidía con la semántica del mensaje etiquetado. En función de su comportamiento, se identificaron seis perfiles de participante que a continuación se describen:

1. **Esperado (E)**. Se refiere al comportamiento de los participantes que utilizaron sólo las tres instancias de etiquetas que se esperaban en los lugares de la conversación donde se sugirieron.
2. **Esperado + repetido (E+R)**. Se refiere al comportamiento de los participantes con un perfil de "Esperado" que además usaron las instancias de etiquetas esperadas en otro lugar de la conversación.
3. **Esperado + meta-etiquetas (E+M)**. Se refiere al comportamiento de los participantes con un perfil de "Esperado", y que usaron instancias de meta-etiquetas en otros lugares de la conversación.
4. **Menos de lo esperado (ME)**. Se refiere al comportamiento de los participantes que usaron menos de las tres instancias de etiquetas esperadas.
5. **Menos de lo esperado + repetido (ME+R)**. Se refiere al comportamiento de los participantes dentro del perfil "Menos de lo esperado" que además etiquetaron con alguna de las instancias de etiquetas ya utilizadas en otro lugar de la conversación.

Tabla 6.1. Resumen del número de participantes por perfil y el uso correcto o incorrecto de instancias de etiquetas.

Perfil de particpante	Número de particpantes	
	Usaron instancias correctas	***Usaron instancias incorrectas***
Esperado	7	0
Esperado + repetido	10	1
Esperado + meta-etiquetas	7	4
Menos de lo esperado	6	0
Menos de lo esperado + repetido	2	1
Menos de lo esperado + meta- etiquetas	3	1

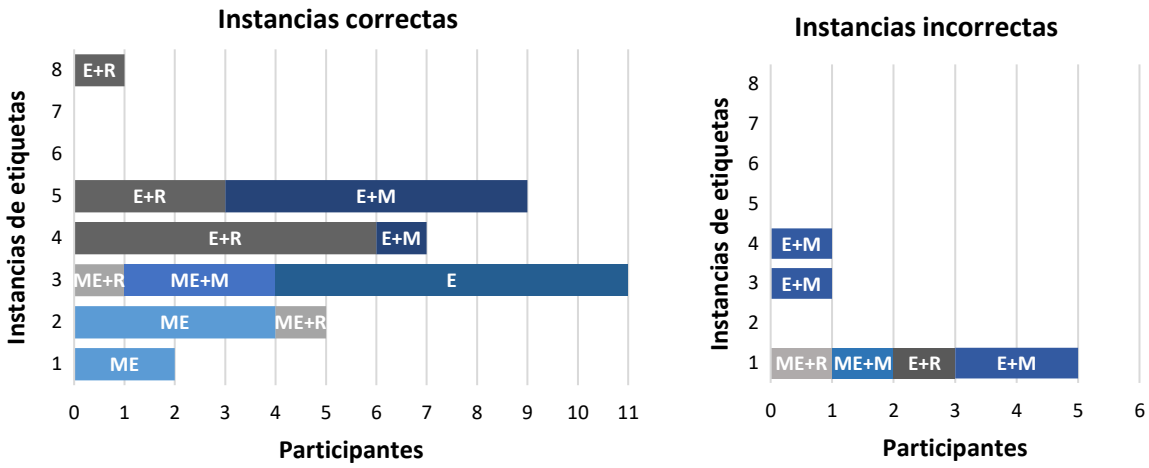


Figura 6.4. Instancias de etiquetas (correctas e incorrectas) usadas por los participantes.

6. **Menos de lo esperado + meta-etiquetas (ME+M)**. Se refiere al comportamiento de los participantes dentro del perfil "Menos de lo esperado" que además usaron instancias de meta-etiquetas en otros lugares de la conversación.

En la Tabla 6.1 se muestra la distribución de los participantes en los perfiles antes descritos, diferenciando entre los que usaron correcta e incorrectamente las instancias de etiquetas. Cabe mencionar que los perfiles no son mutuamente excluyentes, por ejemplo: en el grupo de participantes que usaron las instancias esperadas y además las repitieron (E+R), también hubo participantes que usaron meta-etiquetas (E+M). El mismo caso se presenta en los perfiles ME, ME+R y ME+M.

Como se observa en la Tabla 6.1, sólo 7 de los 30 participantes fueron clasificados con el perfil esperado. También, fue observado que 17 participantes usaron correctamente más de las instancias de etiqueta esperadas ((E+R) + (E+M)), y solo 5 de éstos 17 participantes las usaron incorrectamente. Además, 6 participantes usaron menos de las instancias de etiqueta esperadas, donde 2 de ellos repitieron una instancia esperada y 3 usaron al menos una meta-etiqueta.

En las gráficas de la Figura 6.4 se observa que la mayoría de los participantes usaron hasta 5 instancias de etiquetas de manera correcta. También se aprecia un caso atípico de un participante que usó 8 instancias de etiquetas. En el caso de las instancias de etiquetas utilizadas incorrectamente, la mayoría de los casos son de una sola instancia. Los casos que se muestran de 3 y de 4 instancias usadas de manera incorrecta, corresponden al participante que uso las 8 instancias de etiquetas de manera correcta.

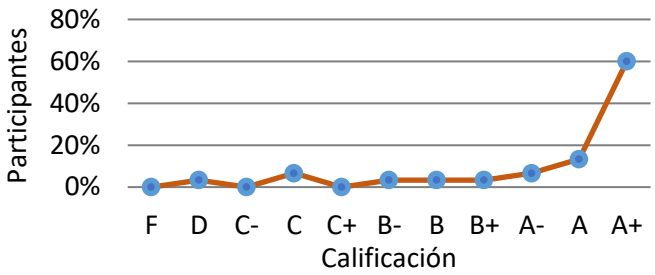


Figura 6.5. Escala de calificación con curva de los puntajes SUS.

6.2.2.2 Usabilidad e intención de uso

En términos de usabilidad, se obtuvo del instrumento SUS una puntuación promedio de 86.66 (desviación estándar = 10.98), lo que sugiere una alta percepción de usabilidad del mecanismo de clasificación basado en etiquetado; de hecho según el estudio de (Bangor et al., 2008) la puntuación SUS obtenida corresponde a una percepción excelente de usabilidad. En la Figura 6.5 se presenta el resultado del instrumento SUS usando una escala de calificaciones al estilo norteamericano (Sauro and Lewis, 2012), donde se observa que alrededor del 10% de los participantes percibió que el mecanismo de clasificación tiene una baja usabilidad (grados C y D en la Figura 6.5).

Con respecto a los resultados del cuestionario TAM, los participantes percibieron una utilidad con mediana de 6 y una moda de 7 utilizando una escala Likert-7, es decir, los participantes consideran que el mecanismo de marcado es muy útil. Sin embargo, se tuvo un valor mínimo de 3 y el valor máximo del cuartil 0.25 fue de 5, es decir, hubo algunos participantes que no consideraron tan útil el mecanismo (ver Tabla 6.2). En cuanto a la facilidad de uso percibida, se obtuvo una mediana y moda de 7 en la escala Likert-7 (ver Tabla 6.2), lo cual indica que se percibió extremadamente fácil de usar en general el mecanismo, pero al igual que con la utilidad un segmento de los participantes tuvo una percepción muy distinta (incluso hasta 3 de facilidad de uso). Este resultado coincide con el resultado de SUS, lo que sugiere que los participantes percibieron que el mecanismo de etiquetado es muy usable.

Tabla 6.2. Resultado del instrumento TAM sobre el mecanismo de clasificación.

	Mediana	**Moda**	**Mínimo**	**Máximo**	**Percentil 0.25**	**Percentil 0.50**	**Percentil 0.75**
Utilidad	6	7	3	7	5	6	7
Facilidad de uso	7	7	3	7	6	7	7

6.2.3 Discusión

Uno de los principales hallazgos de esta evaluación fue que el número de participantes que usaron al menos la cantidad de instancias de etiquetas esperadas fue mayor que la cantidad de participantes que usaron menos de lo esperado ([(E+R) + (E+M) + E] > [ME + (ME+R) + (ME+M)]). Esto sugiere que los desarrolladores participantes no tuvieron ningún problema al etiquetar sus interacciones en UTEM utilizando el mecanismo propuesto. Este comportamiento refleja la popularidad del etiquetado en los sistemas sociales (Yazdani et al., 2012), como lo son los UTEM, sin embargo, se debe explorar mejor esta posibilidad. Además, se observó que los participantes asumieron el rol propuesto en la evaluación, lo que se reflejó en un gran interés en etiquetar correctamente para facilitar la recuperación de conocimiento en el futuro.

Por otro lado, a pesar de la buena percepción de usabilidad, respaldada por los resultados de los cuestionarios SUS y TAM, los participantes sugirieron una lista de posibles mejoras, incluyendo las siguientes:

1. Una forma personalizable de seleccionar y navegar la lista de sugerencias de finalización. Algunos participantes reportaron que tuvieron problemas para usar la tecla de tabulación para seleccionar una etiqueta.
2. Un mecanismo consciente del contexto para sugerir etiquetas, es decir, basadas en el tema de interacción.
3. Incluir una manera de ver todas las etiquetas disponibles, es decir, además del mecanismo de autocompletado, por ejemplo, un buscador de etiquetas, o un árbol de etiquetas disponibles, etc.
4. Un mecanismo para identificar conceptos frecuentes para luego convertirlos en etiquetas.

Estas sugerencias podrían ser la explicación sobre la baja percepción de un 10% de los participantes identificados mediante el cuestionario SUS. Esto a su vez sugiere que es necesario seguir trabajando para mejorar la usabilidad del mecanismo de etiquetado.

Cabe hacer hincapié respecto al uso excesivo de etiquetas durante las interacciones, ya que en esta evaluación uno de los participantes utilizó 8 instancias de etiquetas en lugar de las 3 esperadas (las 3 esperadas y 5 más repetidas). Teniendo en cuenta un período de chat más largo (e.g. un día completo), el etiquetado excesivo podría afectar la efectividad de la recuperación de AK. Hay que recordar que se propuso usar etiquetas para encontrar

conocimiento, por lo que, si hay muchas etiquetas similares en las interacciones de UTEM, la lista de resultados de búsqueda podría crecer, lo que provocaría una sobrecarga de información. De hecho, después de concluir la evaluación, aproximadamente el 50% de los participantes solicitaron las características de la herramienta de búsqueda, y solicitaron considerar la efectividad de los resultados de la búsqueda, de modo que la obtención del conocimiento deseado se hiciera en el menor tiempo posible.

Otro hallazgo fue que los participantes usaron meta-etiquetas de manera correcta, a pesar de que nunca se les explicó su significado. Esto sugiere que los nombres propuestos para las meta-etiquetas son expresivos, y son fácilmente relacionados con situaciones cotidianas en el desarrollo ágil y global de software. Sin embargo, hay algunas meta-etiquetas que se usaron sistemáticamente de forma incorrecta (e.g. #TechnologicalSupport, #ArchitecturalView) y otras que nunca se usaron (e.g. #Topic, #Generated). En el caso de las etiquetas que se usaron incorrectamente, es necesario identificar un nombre más adecuado para ellas, o simplemente aclarar su significado previamente. Teniendo en cuenta las meta-etiquetas que no se usaron, consideramos que el escenario de evaluación no era adecuado para el uso de las mismas. Por lo tanto, es necesario seguir evaluando el conjunto de etiquetas en otros escenarios.

Respecto a los casos donde los participantes usaron menos etiquetas de las esperadas, se observó que los participantes tenían dificultades para seleccionar una etiqueta. Esto era causado ya sea por el mecanismo de selección o porque no encontraron ninguna etiqueta que coincidiera con la temática donde se sugería el etiquetado en el guion. Como resultado, los participantes preferían enviar el mensaje sin una etiqueta. Este comportamiento mostró que una forma personalizable de seleccionar etiquetas es importante y útil para disminuir los casos de "etiquetado mínimo o nulo", así como un mecanismo para agregar nuevas etiquetas durante las interacciones (al estar interactuando en un UTEM). Sin embargo, se debe tener cuidado con un mecanismo para agregar nuevas etiquetas libremente, ya que podría conducir a la explosión de etiquetas.

Otro aspecto de los participantes que usaron menos etiquetas de las esperadas, fue que parecían tener prisa por concluir la evaluación, ya que no querían perder demasiado tiempo tratando de realizar la actividad de manera correcta. En entornos reales, los desarrolladores ágiles se enfrentan a situaciones similares de premura, las cuales provocan la llamada deuda

técnica (Sneed, 2014). Por esta razón, es necesario mejorar la usabilidad del mecanismo para lidiar correctamente con este tipo de situaciones. Además, la proliferación de participantes de este perfil (ME), podría afectar negativamente la recuperación de AK, ya que habría pocos (o inexistentes) puntos a los que referirse al intentar encontrar el conocimiento requerido en las bitácoras de UTEM.

6.3. Resumen del capítulo

En este capítulo se presentó una propuesta de mecanismo de clasificación de AK basada en etiquetado social. Se presentaron las ventajas que tendría este mecanismo al implementarlo en ambientes de AGSD, con la intención de ser parte de una solución completa para la condensación de AK.

Los principales resultados de este capítulo son la implementación del mecanismo de clasificación y su evaluación con desarrolladores, de donde se obtuvo lo siguiente: (1) los participantes percibieron el mecanismo como altamente utilizable y mostraron una alta intención de uso. Sin embargo, se recibieron sugerencias para mejorarlo; (2) se identificaron 6 perfiles de etiquetado de los participantes que se pueden agrupar en (a) participantes que etiquetan como se esperaba, (b) participantes que etiquetan más de lo esperado y (c) participantes que etiquetan menos de lo esperado, donde los que etiquetaron más de lo esperado fueron más que los demás, lo que evidencia que el etiquetado podría ser una buena opción para clasificar las interacciones UTEM.

En el siguiente capítulo se presenta el diseño e implementación de una solución completa de condensación de AK. El mecanismo de clasificación de esta solución se basa en el etiquetado social, debido a los buenos resultados obtenidos mediante la evaluación presentada en este capítulo.

Capítulo 7.
ArchiKCo: Un prototipo de implementación del concepto de condensación de AK

En el capítulo anterior se presentó la implementación y evaluación de una propuesta para un mecanismo de clasificación de bitácoras de UTEM. Los resultados de la evaluación mostraron que el etiquetado social puede ser factible para clasificar AK mientras se está interactuando mediante UTEM con pares remotos. Estos resultados conducen a desarrollar un prototipo de condensador de AK que tenga como base un mecanismo de clasificación basado en el etiquetado social, con el fin de evaluar el concepto de condensación de AK. El diseño de interfaz de usuario y la idea de funcionamiento de todo el prototipo se basó en el prototipo no funcional presentado y evaluado en el Capítulo 6. En este capítulo se presenta el diseño e implementación de dicho prototipo al que se le nombró ArchiKCo, por sus siglas en inglés: **Archi**tectural **K**nowledge **Co**ndenser.

7.1. Características del prototipo ArchiKCo

En esta sección se presentan las características del prototipo ArchiKCo basadas en los tres elementos que componen el concepto de condensación de AK que fueron presentados en el Capítulo 5.

7.1.1 Accesibilidad de bitácoras de UTEM

Para lograr esta accesibilidad se establecieron en el capítulo 5 dos posibles modalidades: (1) distribuida, aprovechando las interfaces públicas que expongan los UTEM; y (2) centralizada, extrayendo las interacciones de las bitácoras de los UTEM, para luego conjuntarlas en un repositorio común. Se analizaron ambas modalidades para la accesibilidad, y se optó por un mecanismo para conjuntar las interacciones de UTEM en un repositorio común. Para lograr esto se requeriría de: (1) un repositorio para almacenar las interacciones registradas en las bitácoras y (2) un componente para recopilar dichas interacciones y almacenarlas en el repositorio. En cuanto al repositorio para las interacciones, y tomando en cuenta la distribución global de los potenciales involucrados en un ambiente de AGSD, éste debería de residir en internet para que los involucrados pudieran acceder a él,

claro está, cuidando la seguridad del repositorio ya que las bitácoras podrían contener información sensible de la empresa. En cuanto al componente recopilador de interacciones, podría implementarse en forma de un servicio que periódicamente esté recopilando las interacciones nuevas y enviándolas el repositorio. Este servicio podría residir en un servidor o en las computadoras de los involucrados, dependiendo de donde queden almacenadas las bitácoras de interacción de los UTEM a incluir.

7.1.2 Mecanismo para la clasificación de bitácoras de UTEM

Este mecanismo se decidió basar en el etiquetado social, tal como se describió en el capítulo anterior. Para este nuevo prototipo se usó un UTEM habitual como base, con el fin de que la evaluación del prototipo fuera más apegada a la realidad del AGSD. Además, en esta ocasión la implementación del mecanismo propuesto sería completa, es decir: (1) un conjunto de meta-etiquetas (embebidas en el prototipo anterior), (2) un sistema administración de etiquetas (no implementado en el prototipo anterior) y (3) un asistente de etiquetado (basado en autocompletado).

7.1.3 Mecanismo para la búsqueda de AK

Además de las características ya establecidas en el capítulo 5 (Recuperación por período, por autor, destinatario, UTEM y texto libre) es conveniente aprovechar los mensajes de UTEM etiquetados para proporcionar un filtro que permita hacer búsquedas a los usuarios por una o varias etiquetas o meta-etiquetas. Cabe mencionar que la recuperación de conocimiento por esta vía podría hacerse mediante meta-etiquetas y/o etiquetas padre, es decir, si no se recuerda exactamente con que etiqueta se marcó cierto conocimiento, podría usarse una etiqueta padre, la cual es más genérica o abstracta. Por ejemplo, podría usarse la meta-etiqueta de “Código”, para encontrar todo lo referente a código, y una vez desplegados los resultados, ir refinando la búsqueda dependiendo de lo que se haya mostrado.

7.2. Diseño del prototipo ArchiKCo

Teniendo como base las características definidas en la sección anterior, se definieron cinco sub-sistemas que se representan en el diagrama de componentes de la Figura 7.1. A

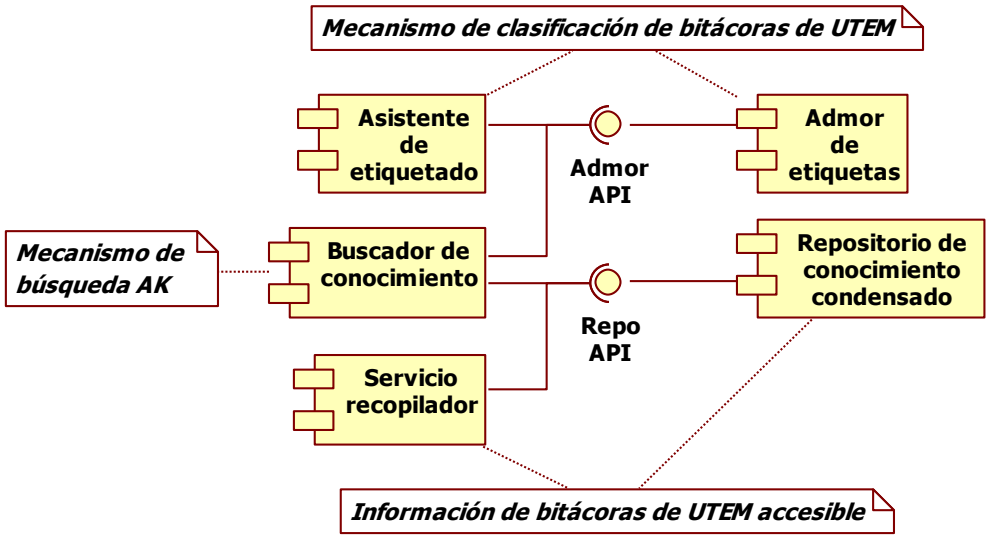


Figura 7.1. Diagrama de componentes de ArchiKCo.

continuación, se describen los componentes que conforman cada uno de los elementos del concepto de condensación de conocimiento.

7.2.1 Diseño para la accesibilidad de bitácoras de UTEM.

Para lograr la accesibilidad de las bitácoras se requiere de la acción conjunta de dos subsistemas: el servicio recopilador y el repositorio de conocimiento condensado (ver Figura 17). El primero tiene como responsabilidad recopilar los mensajes de las bitácoras de los UTEM que se definan. Como su nombre lo indica, se trata de un servicio el cual estaría en ejecución constantemente, de tal manera que cada cierto tiempo se consulten las bitácoras de UTEM en busca de nuevos mensajes para luego enviarlos al repositorio de conocimiento condensado.

Por otro lado, el sub-sistema de repositorio de conocimiento condensado, tiene la responsabilidad de recibir los mensajes de las diferentes bitácoras de UTEM para luego ser almacenados. Además, tiene la responsabilidad de ofrecer vías para consultar dichos mensajes como lo define el mecanismo de búsqueda de conocimiento. Toda la comunicación con este subsistema se realiza a través de su interface “Repo API”.

7.2.2 Diseño del mecanismo de clasificación de bitácoras de UTEM.

Este mecanismo está conformado por dos sub-sistemas: el administrador de etiquetas y el asistente de etiquetado. El subsistema de administración de etiquetas tiene la responsabilidad

de ofrecer los medios para que los usuarios registren, editen, eliminen y busquen las etiquetas a usar en su medio de trabajo. Como se definió en el capítulo anterior, estas etiquetas estarían relacionadas a un conjunto de meta-etiquetas, las cuales residen en este sub-sistema y son inalterables. Además, el subsistema de administración tiene la responsabilidad de ofrecer los medios para que otros sub-sistemas (e.g. el Asistente de etiquetado y el Buscador de conocimiento) consulten las etiquetas y meta-etiquetas existentes a través de su interface "Admor API".

El sub-sistema asistente de etiquetado es el encargado de apoyar a los usuarios al momento de etiquetar algún mensaje, auto-completando alguna etiqueta según se vaya escribiendo (tal como se implementó en el prototipo presentado en el capítulo anterior). Este asistente consultará constantemente al administrador de etiquetas para estar sincronizado con las etiquetas registradas por los usuarios. Con el fin de estar siempre disponible para asistir en el etiquetado, este sub-sistema debe ser implementado como una aplicación en segundo plano, es decir, que esté contantemente en ejecución y censando si la aplicación que se encuentre activa es uno de los UTEM definidos, para determinar si debe activar el autocompletado.

7.2.3 Diseño del mecanismo de búsqueda AK

Este último mecanismo está compuesto por sólo un sub-sistema: el buscador de conocimiento. Este tiene como responsabilidad permitir al usuario hacer búsquedas de AK en los mensajes de UTEM, filtrando por autor, destinatario, fecha de envío del mensaje, etiquetas y texto libre. Para ello, el buscador se comunica con el repositorio de conocimiento y con el administrador de etiquetas para obtener las etiquetas disponibles y ofrecerlas como filtros.

7.3. Implementación del prototipo ArchiKCo

Una vez definidos los sub-sistemas, sus responsabilidades y sus conexiones, fue posible determinar los equipos físicos y sus características, donde los subsistemas se desplegarían, así como la manera de comunicarse entre los equipos. Para ello fue primordial definir quiénes podrían ser los potenciales voluntarios para evaluar ArchiKCo, y así observar las características de sus equipos y de su software. De esta manera se determinó a Skype (en su versión para el sistema operativo Windows) como el UTEM principal a usar. Por ello, el

servicio recopilador debía desplegarse como un servicio de Windows en el equipo donde estuviera instalado Skype, con el fin de aprovechar que su bitácora reside en el mismo equipo. Del mismo modo, se determinó que el asistente de etiquetado se desplegaría como una aplicación en segundo plano en cada uno de los equipos de los usuarios (ver Figura 7.2).

En cuanto al administrador de etiquetas y el buscador de conocimiento, se decidió implementarlo y desplegarlo como una aplicación Web (ver Figura 7.2), con el fin de que estuviera disponible en cualquier lugar donde se encontraran los usuarios. Debido a que el administrador de etiquetas debe ser consultado por el asistente de etiquetado y por el buscador de conocimiento a través de una interface, ésta última se expone como un servicio Web estilo REST (Fielding and Taylor, 2002), para ofrecer un servicio ligero y fácil de consultar por la mayoría de las tecnologías actuales.

Por último, para el repositorio de conocimiento condensado se decidió usar la plataforma en línea llamada Algolia[41], la cual ofrece varios repositorios para almacenar datos masivos, así como una interface para construir herramientas de búsqueda. Dicha interface cumple con las características necesarias para implementar el buscador como fue definido anteriormente. A continuación, se describe la manera en la que se implementó cada uno de los elementos del concepto de condensación de AK.

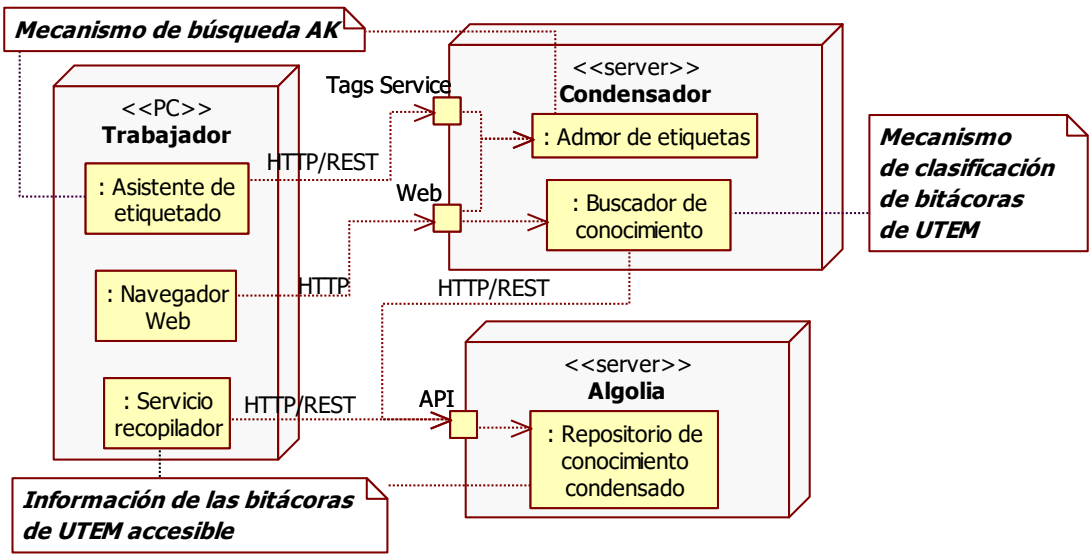


Figura 7.2. Diagrama de despliegue de ArchiKCo.

[41] https://www.algolia.com/

7.3.1 Accesibilidad de bitácoras de UTEM

Este elemento se implementó mediante dos componentes, el servicio recopilador y el repositorio de conocimiento condensado, desplegados en diferentes equipos: la computadora personal del trabajador de AGSD y el servidor de la plataforma Algolia, respectivamente.

El servicio recopilador se implementó como un servicio de Windows en C#.net, el cual revisa cada cierto tiempo si hay mensajes nuevos en las bitácoras de Skype, que no hayan sido enviados al repositorio de interacciones de UTEM (implementado en Algolia). En el Apéndice B se puede consultar la estructura de componentes interna del servicio.

7.3.2 Mecanismo de clasificación de bitácoras de UTEM

Este mecanismo se implementó mediante el sub-sistema asistente de etiquetado y el sub-sistema de administración de etiquetas, los cuales se despliegan en la computadora personal del trabajador de AGSD y en un servidor Web respectivamente (ver Apéndice B). El asistente de etiquetado, fue desarrollado en C# como una aplicación para Windows que se ejecuta en segundo plano para verificar cuando se use el teclado en alguno de los UTEM que haya sido incluido para la condensación del AK, para entonces desplegar la asistencia de auto-completado de etiquetas cuando se teclee el carácter “#”, con el que inicia cualquier etiqueta. En el Apéndice B se muestra la estructura de componentes interna del asistente de etiquetado.

Como resultado de la interacción de los componentes descritos, cuando el usuario se encuentra en la ventana de conversación de Skype, y escribe la combinación de caracteres: “#” más una letra (e.g. "#r"), se despliega sobre esta ventana una lista desplegable con las opciones de autocompletado. En la Figura 7.3 se muestra una captura de pantalla donde el asistente de autocompletado está mostrando las opciones relacionadas a la secuencia de carateres “#r”, en donde se incluyen todas las etiquetas y meta-etiquetas que contienen la letra “r”. A medida que se va formando una palabra completa se van restringiendo las opciones que muestra el asistente. El funcionamiento interno de esta versión de este subsistema, se reutilizó de la versión Web presentada en el capítulo anterior.

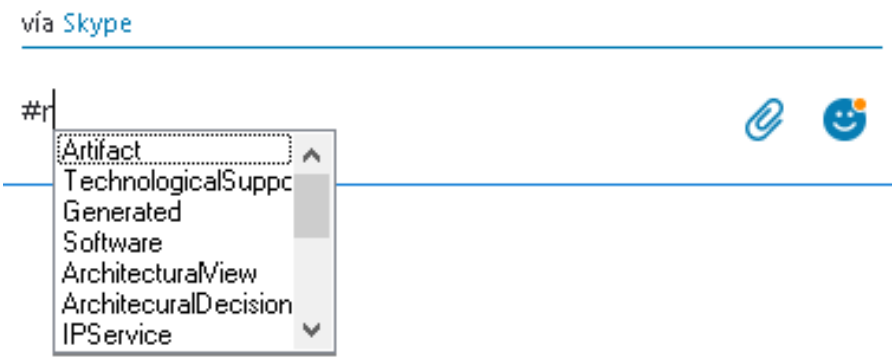


Figura 7.3. Asistente de etiquetado funcionando sobre Skype.

Con el fin de ofrecer una manera fácil de agregar y eliminar etiquetas en este sub-sistema, se implementó una vista de árbol, en la cual se muestran las meta-etiquetas de manera jerárquica (ver Figura 7.4). En esta interfaz se permite agregar y/o eliminar etiquetas de usuario (junto con una breve descripción de su semántica) en cada una de las meta-etiquetas independientemente del nivel del árbol en el que se encuentre. Además, se pueden agregar etiquetas hijas a las etiquetas que el mismo usuario registre, permitiendo crear sus propias jerarquías, siempre y cuando sus raíces estén relacionadas a una de las meta-etiquetas existentes. Cabe mencionar que las meta-etiquetas no pueden ser eliminadas ni modificadas.

Figura 7.4. Interfaz de usuario del administrador de etiquetas.

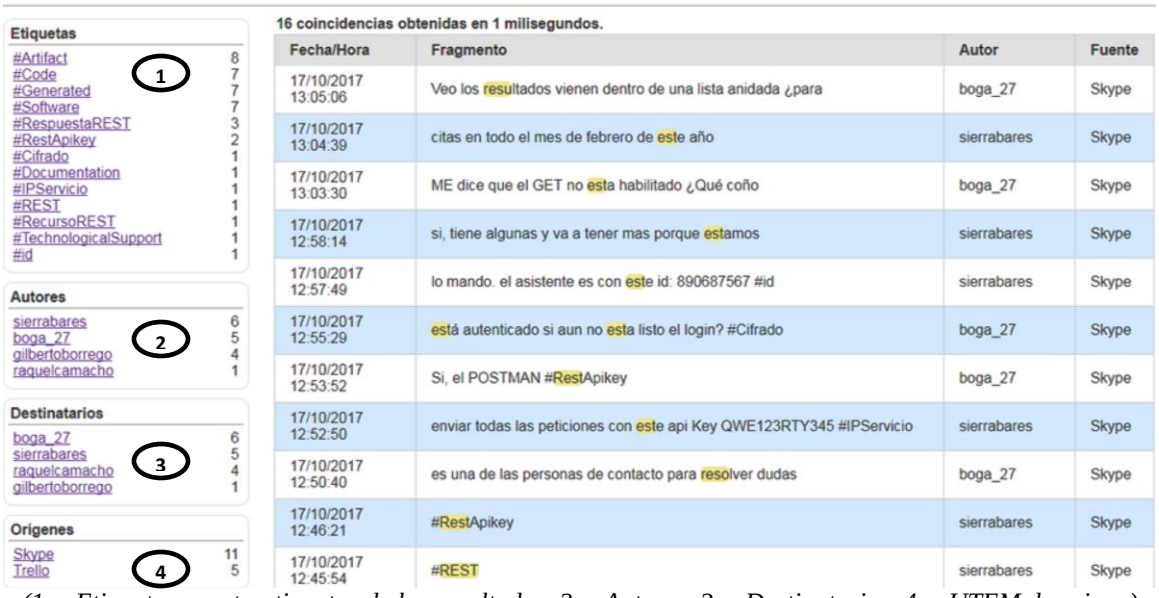
Figura 7.5. Interfaz gráfica inicial del buscador de AK.

7.3.3 Mecanismo de búsqueda de conocimiento arquitectónico

Tal como se mencionó previamente, la interface que expone Algolia cumple con las características necesarias para implementar un buscador que permita la recuperación por distintos campos. En la Figura 7.5 se muestra la interfaz gráfica inicial del buscador de conocimiento, donde se muestran los campos para ingresar los siguientes parámetros:

- Texto libre, en el cual los usuarios pueden ingresar cualquier texto que se desee buscar en el contenido de los mensajes de UTEM; además se pueden hacer búsquedas por autor y/o destinatario de los mensajes.
- Rango de fechas, en el cual los usuarios pueden seleccionar el período en el que se enviaron los mensajes de UTEM.
- Filtro de etiquetas, en el cual los usuarios pueden seleccionar las etiquetas disponibles para conformar un filtro para recuperar mensajes compartidos con ciertas etiquetas. Para ello se muestran meta-etiquetas y etiquetas en una estructura jerárquica de árbol (ver

Figura 7.6. Interfaz gráfica para formar un filtro de etiquetas.



(1 = Etiquetas y meta-etiquetas de los resultados, 2 = Autores, 3 = Destinatarios, 4 = UTEM de origen)

Figura 7.7. Interfaz gráfica del buscador de AK mostrando resultados de una búsqueda.

Figura 7.6), donde los usuarios pueden agregar cualquier cantidad de ellas para formar un filtro.

Una vez que se ejecuta una búsqueda, se muestra la lista de los mensajes coincidentes, junto con cuatro paneles (ver Figura 7.7), donde se muestran: (1) las etiquetas y meta-etiquetas relacionadas a los mensajes resultantes, (2) los autores de los mensajes mostrados, (3) los destinatarios de los mensajes mostrados, y (4) los UTEM de los cuales se originaron los mensajes resultantes. Cabe hacer hincapié que en el panel de etiquetas también se muestran los padres de cada una de ellas; por ejemplo, si hay una etiqueta de usuario llamada #nodeMongoDB, que está relacionada con la meta-etiqueta #Code, el panel de etiquetas muestra ambas y todos los ancestros de #Code. Todos los elementos mostrados en estos cuatro paneles son hipervínculos, que al seleccionarlos van aplicando filtros adicionales para hacer más precisos los resultados. Por ejemplo, si se selecciona cierto autor del panel respectivo, se aplicaría un filtro con el cual sólo se mostrarían los mensajes que hayan sido enviados por el autor seleccionado.

Además, sólo con leer el mensaje etiquetado no sería factible obtener el contexto de la interacción y por consiguiente tampoco sería factible obtener el AK deseado. Por lo tanto, se decidió agregar una manera en la que los usuarios pudieran leer los mensajes que se enviaron cinco minutos antes y cinco minutos después del mensaje etiquetado (e incluso

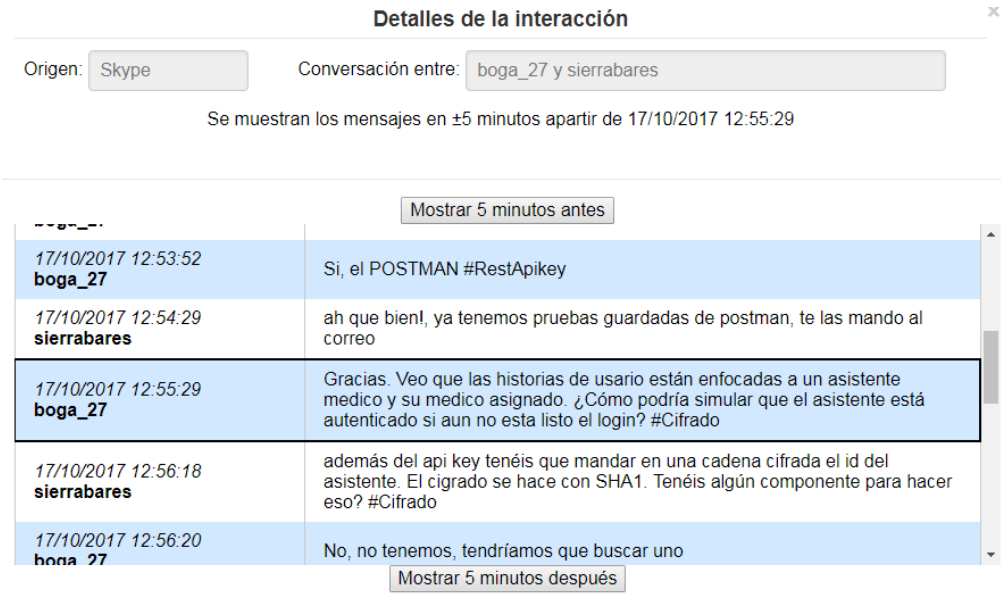


Figura 7.8. Interfaz gráfica mostrando el contexto de un mensaje de UTEM etiquetado.

cargar más mensajes si es necesario), la cual se activa al seleccionar uno de los mensajes mostrados en la búsqueda inicial (ver Figura 7.8).

7.4. Escenario de uso del prototipo ArchiKCo

En esta sección se retoma toda la implementación especificada anteriormente con el fin de describir un escenario de uso de todos los elementos ArchiKCo en un ambiente de AGSD. En la Figura 7.9, se muestra una gráfica rica que ejemplifica un escenario típico en donde podría ser utilizado el prototipo ArchiKCo. A continuación, se describe dicho escenario.

1. En primera instancia los desarrolladores (local y remoto) durante la reunión de inicio de un ciclo de desarrollo (e.g. Sprint meeting de Scrum), acuerdan las etiquetas a usar durante el siguiente ciclo dependiendo de los temas cubiertos por las historias de usuario a desarrollar. Una vez hecho el acuerdo, las etiquetas se registran en la aplicación Web de administración de etiquetas (y consecuentemente en el repositorio de etiquetas), y se continua con el flujo convencional de la metodología que se esté siguiendo.
2. Iniciado el ciclo de desarrollo, es común que los desarrolladores dejen de lado actividades de documentación de AK, debido a la naturaleza del ambiente ágil, donde se prefiere código funcionando e interacción entre individuos sobre documentación y seguimiento de procesos (Beck et al., 2001), además de las presiones de tiempo que generalmente existen en el ambiente de AGSD (Sneed, 2014) y la actitud de los desarrolladores hacia

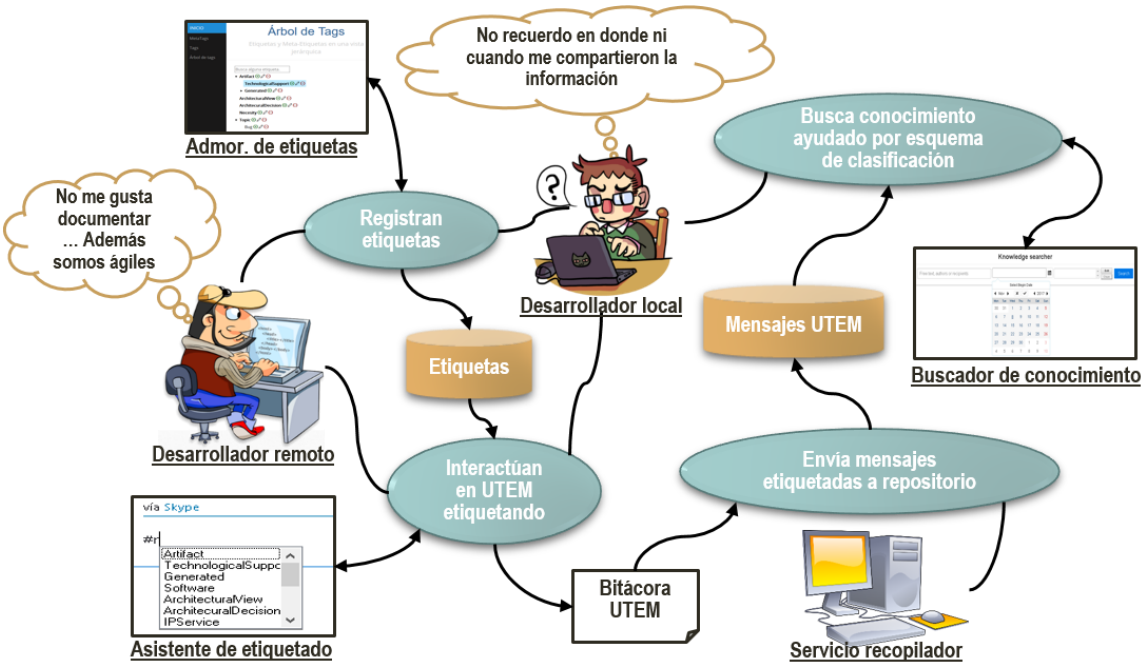


Figura 7.9. Gráfica rica representando un escenario típico de uso de ArchiKCo.

la documentación (Clear, 2003). Esto genera deuda de documentación (Tom et al., 2013), la cual se trata de solventar interactuando constantemente entre los miembros remotos y locales del equipo mediante UTEM para adquirir y/o compartir AK. Durante estas interacciones los desarrolladores tienen la posibilidad de etiquetar el conocimiento (apoyados por el asistente de etiquetado) que se comparten a través de los mensajes, y que crean necesario recuperar posteriormente.

3. Los mensajes etiquetados quedan almacenados en las respectivas bitácoras de UTEM, las cuales son leídas frecuentemente por el servicio recopilador, para luego enviar aquellos mensajes que aún no se encuentren almacenados en el repositorio de mensajes de UTEM
4. Finalmente, debido a la deuda de documentación, es frecuente que los miembros del equipo de desarrollo no recuerden algún detalle arquitectónico (p. ej. especificaciones, diseño detallado, cuestiones de despliegue de aplicaciones, etc.) ya sea durante el mismo ciclo de desarrollo o en alguno posterior. Incluso es frecuente que no se recuerde exactamente la fecha o medio por el cual les fue compartido el conocimiento. Sin embargo, dado que los miembros del equipo etiquetaron sus interacciones y éstas están concentradas en un repositorio común, se podría recuperar el AK usando el buscador respectivo que ofrece ArchiKCo.

Aunque este escenario se describió de manera secuencial, los elementos de ArchiKCo pueden entrar en juego en cualquier momento. Por ejemplo, el registro y/o eliminación de etiquetas podría realizase en cualquier momento durante el ciclo de desarrollo, siempre y cuando exista un acuerdo entre los miembros del equipo. Además, aunque el etiquetado es fundamental para esta implementación de la condensación del conocimiento, la recuperación del mismo se puede llevar acabo incluso antes de que se etiquete algún mensaje, ya sea recuperando por período, autor, destinatario, etc. Sin embargo, el no etiquetar las conversaciones puede conducir a la pérdida de semántica y contexto del conocimiento a través del tiempo, ya que las etiquetas y la estructura de meta-etiquetas a las cuales se relacionan, proporcionan un ancla semántica que ofrecería una manera de recordar o intuir el contexto de la conversación que se esté consultando.

7.5. Beneficios potenciales del prototipo ArchiKCo para el AGSD

Si bien el capítulo 5 se estableció que el concepto de condensación de AK es una manera de reducir la vaporización del mismo, a través del prototipo presentado en este capítulo se esperan también los siguientes efectos benéficos en ambientes de AGSD:

- **Reducción de interrupciones**. Debido a que ArchiKCo ofrece un buscador de AK, aquel miembro del equipo que tenga alguna duda sobre algún tema al respecto, podría usar primero el buscador antes de interrumpir a uno de sus compañeros para hacerle una pregunta.
- **Reducción del tiempo para encontrar AK**. ArchiKCo ofrece un punto común para buscar conocimiento compartido en distintos UTEM, además de ofrecer filtros de búsqueda que la mayoría de los UTEM no ofrece, como: búsqueda por periodo, por autor, por destinatario y por etiqueta.
- **Reducción del tiempo de tareas de desarrollo**. En consecuencia de los dos puntos anteriores, se espera que los tiempos para concluir las tareas de desarrollo se reduzca, ya que el AK se tendría más accesible con ArchiKCo.

Como se expuso en el capítulo 4, la reducción en general de tiempos (para encontrar AK y para concluir tareas) e interrupciones, tiene un impacto positivo en la efectividad de equipo, por lo que se verían beneficios tanto en aspectos cuantitativos, al cumplir o exceder los estándares establecidos en la organización (resultados), como en aspectos cualitativos, al

mejorarse o mantenerse, la capacidad del grupo para trabajar juntos en el futuro (procesos sociales) reduciendo el número de interrupciones que pueden ser molestas, y fomentando el aprendizaje al tener un medio para encontrar conocimiento fácilmente y compartirlo.

Concluyendo esta parte, tomando en cuenta lo expuesto en el párrafo anterior, al implementar la condensación de conocimiento, se estaría dando un paso más hacia una integración que fomente la eficiencia de equipo, entre paradigma del desarrollo global y el paradigma del desarrollo ágil de software, que en principio pareciera que van en sentidos opuestos.

7.6. Resumen del capítulo

En este capítulo se presentaron las características, el diseño e implementación de un prototipo de condensador de AK el cual se nombró ArchiKCo, resaltando cada uno de los elementos del concepto de condensación de conocimiento. Los principales resultados de este capítulo incluyen:

- Definición de las características de cada uno de los elementos de la condensación de conocimiento para implementar el prototipo ArchiKCo.
- Vista de sub-sistemas por los cuales está conformado el prototipo ArchiKCo, indicando cuáles de ellos implementa cada uno de los tres elementos de la condensación de AK.
- Vista donde se especifica cómo se despliegan en equipos físicos los sub-sistemas que conformen al prototipo ArchiKCo, indicando también la ubicación física de los elementos que implementan cada uno de los elementos de la condensación de AK.
- Detalles de implementación de cada uno de los sub-sistemas en términos de componentes y tecnologías usadas, así como ejemplos de interfaz gráfica acompañados de una breve descripción de su funcionamiento a nivel de usuario.
- Explicación de un escenario donde se usa el prototipo ArchiKCo y la especificación de los efectos benéficos que se esperan al implantar este prototipo en AGSD.

En el siguiente capítulo se presenta como se evaluó el prototipo ArchiKCo con el fin de determinar la factibilidad del concepto de condensación de AK.

Capítulo 8.
Evaluación funcional del concepto de condensación del conocimiento arquitectónico

En el capítulo anterior se presentó el diseño e implementación del prototipo de condensador de AK llamado ArchiKCo, así como una perspectiva general de cómo sería su funcionamiento en un ambiente de AGSD. La razón principal de desarrollar este prototipo es evaluar el concepto de condensación de AK con el fin de determinar su viabilidad. En este capítulo se presenta el método usado para realizar dicha evaluación sobre ArchiKCo, así como los resultados obtenidos y la discusión de los mismos, terminando con una argumentación sobre la viabilidad de dicho concepto en AGSD.

8.1. Método

El método empleado para evaluar el prototipo ArchiKCo se describe a continuación. El método está estructurado de la siguiente manera: objetivo y preguntas de investigación, participantes, material, tareas de estudio, variables e hipótesis. Cada una de estas secciones se describe de la siguiente manera.

8.1.1 Objetivo y preguntas de investigación

El objetivo del estudio se estructuró utilizando la plantilla para la definición de objetivos propuesta por (Wohlin et al., 2000). El objetivo de este estudio es el siguiente: *Evaluar la implementación del concepto de condensación de conocimiento de AK, con respecto al comportamiento de marcado y recuperación de AK desde el punto de vista de desarrolladores y estudiantes, en el contexto de AGSD.*

Este objetivo permitió formular las siguientes preguntas de investigación:

- **RQ1**. ¿Es mejor buscar AK usando el buscador de ArchiKCo que sólo usando los medios de búsqueda que ofrecen los UTEM fuente, en términos de conocimiento correcto y tiempo requerido para encontrar el conocimiento correcto?
- **RQ2**. ¿Es preferible para los usuarios buscar AK usando el buscador de ArchiKCo en lugar de buscar directamente en la fuente UTEM?

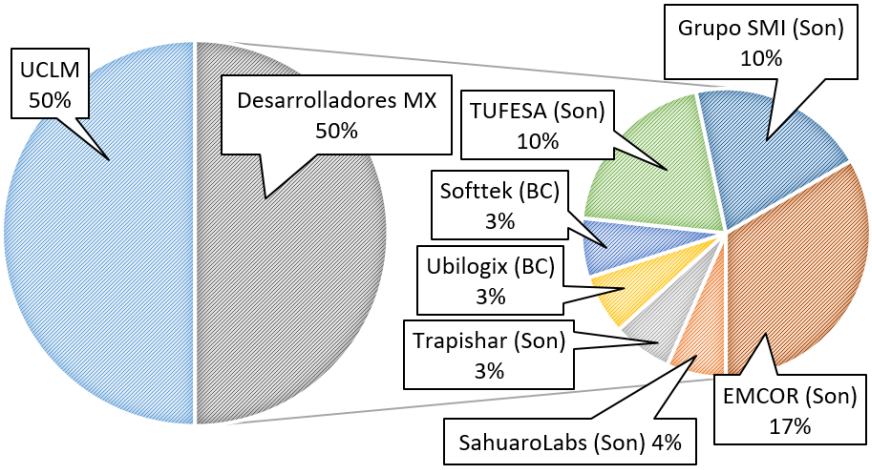


Figura 8.1. Distribución de los participantes por empresa.

8.1.2 Participantes

Para responder a estas preguntas, se llevaron a cabo dos estudios empíricos: uno en la industria y una réplica en un entorno académico. El estudio en la industria se llevó a cabo con 30 desarrolladores en 7 empresas mexicanas, 5 en Ciudad Obregón, Sonora y 2 en Ensenada, Baja California. En la Figura 8.1 se puede observar el porcentaje de participantes de cada empresa.

Todas las compañías participantes están completamente dedicadas al desarrollo de software, con la excepción de TUFESA, cuya principal actividad son los servicios de transporte de pasajeros, pero cuenta con un departamento de tecnología que desarrolla el software de la compañía. Las empresas participantes desarrollan software para diversas áreas (transporte, atención médica, internet de las cosas, administración en general, etc.), trabajan (o han trabajado) en un entorno distribuido o global (al menos algunos de sus proyectos), y todos ellos trabajan de manera ágil. Además, casi todas las empresas tienen menos de 100 empleados y, según el Gartner IT Glossary, pueden considerarse como pequeñas y medianas empresas[42] (sólo una empresa tiene más de 100 empleados).

El estudio réplica en entorno académico se llevó a cabo en Ciudad Real, España, en la Universidad de Castilla-La Mancha (UCLM) con estudiantes de licenciatura y estudiantes de posgrado (maestría y doctorado) de la Escuela Superior de Informática, los cuales contaban con conocimiento de AGSD. En la Tabla 8.1 se muestran las características de los

[42] https://www.gartner.com/it-glossary/smbs-small-and-midsize-businesses

Tabla 8.1. Datos demográficos y experiencia de los participantes del estudio.

Perfil del participante	Número	Género		Edad		Experiencia en IS		Experiencia en DA		Experiencia en DGS	
		M	***F***	***Prom***	***Desv***	***Prom***	***Desv***	***Prom***	***Desv***	***Prom***	***Desv***
Developer	30	84%	16%	28.0	3.9	4.3	2.7	3.1	1.9	2.0	1.4
Student	30	90%	10%	24.1	3.5	2.3	1.8	0.9	0.8	1.1	1.1

(IS = Ingeniería de Software, DA = Desarrollo ágil, DGS = Desarrollo Global de Software). Todos los valores se expresan en años, excepto de la columna Género

dos perfiles de participantes incluyendo: género, edad promedio, promedio de experiencia en ingeniería de software, en desarrollo ágil y en desarrollo global de software.

Ambas muestras (desarrolladores y estudiantes) se seleccionaron de manera no probabilística (por conveniencia), ya que se necesitaban participantes con un perfil particular, es decir, que al menos tuvieran conocimiento sobre las implicaciones de trabajar en proyectos de AGSD.

8.1.3 Material

Para llevar a cabo esta evaluación se utilizaron los siguientes materiales:

- **ArchiKCo**. Se desplegó una instancia del prototipo ArchiKCo para cada par de participantes con el fin de llevar un mejor control de los mensajes y de las etiquetas que se utilicen durante la ejecución de tareas del estudio.
- **Escenario de contexto**. Se desarrolló un escenario en el cual los participantes debían ubicarse mentalmente para ejecutar las tareas de estudio. Este escenario se refiere a dos desarrolladores ágiles de diferentes compañías y ubicaciones que trabajan en el mismo proyecto, uno de ellos requiere información sobre un servicio Web tipo REST, que el otro desarrollador se encuentra desarrollando. Ambos desarrolladores presentan deuda de documentación, por lo que tienen que adquirir conocimientos arquitectónicos sobre el proyecto haciéndose preguntas entre ellos (escenario detallado en el Apéndice D).
- **Etiquetas de usuario**. Se registraron 10 etiquetas de usuario en la aplicación Web de administración de etiquetas vinculadas a diferentes meta-etiquetas, las cuales estaban relacionadas con los temas en el escenario de contexto (ver Tabla 8.2).
- **Guiones de interacción**. Se desarrollaron dos guiones (uno por cada rol del escenario) que cada par de participantes tenía que seguir para simular una conversación técnica que se llevaba a cabo mediante UTEM (en este caso, Skype) dentro del escenario de contexto.

Tabla 8.2. Etiquetas de usuario agregadas para la ejecución de tareas del estudio.

Meta-etiqueta	Etiquetas de usuario
TechnologicalSupport	IPServicio, PruebasREST
Code	RestApikey, SeguridadRest, Cifrado, RespuestaREST, DatosPrueba, RecursoREST
Component	AngularCifrado
Documentation	HistoriaUsuario

Los guiones tenían 7 marcas donde se sugerían etiquetar diferentes mensajes, sin especificar qué etiqueta elegir (ver detalles en Apéndice D).

- **Cuestionario SUS (Brooke, 1996)**. Se preparó un cuestionario SUS (con escala Likert-7) centrado en el asistente de etiquetado. Se agregaron dos preguntas al estilo del mismo cuestionario SUS (una positiva y la otra negativa) para explorar las percepciones de los participantes sobre la obstrucción del asistente en sus tareas cotidianas. Este cuestionario también incluyó una pregunta abierta para recibir comentarios o sugerencias con respecto del asistente de etiquetado (ver Apéndice D).
- **Historias de usuario y comentarios de usuarios en Trello**[43]. Se preparó un tablero de tareas público en Trello (estilo Kanban (Anderson, 2010)), simulando tareas relacionadas con el escenario de contexto. Se agregaron 5 tarjetas de tareas con comentarios de miembros del equipo de desarrollo preguntando y proporcionando aclaraciones de requerimientos (ver detalles de las historias de usuario en Apéndice D).
- **Encuesta electrónica sobre el escenario**. Se preparó una encuesta que contiene 12 preguntas sobre los temas tratados en los comentarios de Trello y en los guiones de interacción (ver el Apéndice D). En las primeras ocho preguntas se indicaba a los participantes en qué medios se deberían buscar respuestas (Skype, Trello o buscador ArchiKCo). En las últimas cuatro preguntas, los participantes fueron libres de seleccionar un medio en el cual buscar, y se les pidió que indicaran el medio donde encontraron la respuesta. Estas últimas preguntas fueron diseñadas de tal manera que las respuestas a las preguntas 9 y 10 sólo se pudieran encontrar usando Skype o el buscador ArchiKCo, y las

[43] https://trello.com/

Tabla 8.3. Distribución de las indicaciones de medio para buscar las respuestas de cada versión de encuesta.

Pregunta	Encuesta	
	Versión 1	**Versión 2**
1	Buscador ArchiKCo	Skype
2	Skype	Buscador ArchiKCo
3	Buscador ArchiKCo	Skype
4	Skype	Buscador ArchiKCo
5	Buscador ArchiKCo	Trello
6	Trello	Buscador ArchiKCo
7	Buscador ArchiKCo	Trello
8	Trello	Buscador ArchiKCo
9 - 12	Libre elección	Libre elección

respuestas a las preguntas 11 y 12 sólo se pudieran encontrar usando Trello o el buscador ArchiKCo. Esta encuesta se cargó en LimeSurvey[44], ya que esta herramienta de encuesta captura el tiempo que se tarda en responder cada pregunta. Para poder comparar el rendimiento de los participantes en cuanto al tiempo necesario para responder las preguntas, se crearon dos versiones de la encuesta (una diferente para cada miembro de la pareja), en la que se variaron los medios indicados por pregunta (ver Tabla 8.3).

- **Cuestionario TAM (Davis, 1989) extendido**. Se preparó un cuestionario TAM (Likert-7) enfocado al buscador de ArchiKCo, y se extendió agregando preguntas sobre la reducción de interrupciones (una pregunta), sobre la facilidad y oportunidad (cuando se necesita) de encontrar AK relevante (3 preguntas) y la impresión general de todo el prototipo ArchiKCo (2 preguntas). Ver las preguntas de extensión en Apéndice D.

8.1.4 Tareas

Se les pidió a los participantes ser parte de tres sesiones de evaluación en la que se ejecutaron las tareas que se describen a continuación (Figura 8.2).

8.1.4.1 Sesión de instalación.

En esta sesión, se dio una explicación general a los participantes sobre las sesiones del estudio, así como el objetivo de cada una de ellas. También se organizó a los participantes en parejas, y se apoyó a configurar el asistente de etiquetado (para trabajar con Skype) y el

[44] https://www.limesurvey.org/

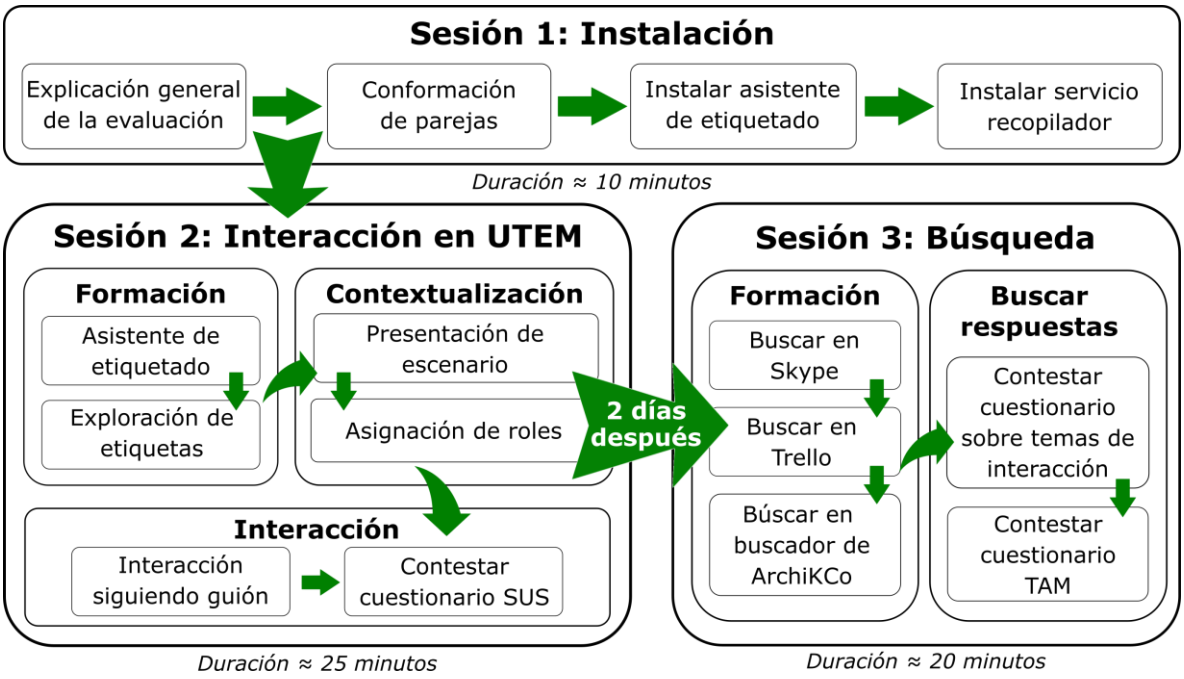


Figura 8.2. Flujo de las sesiones de evaluación de ArchiKCo.

servicio recopilador, con el fin de que las conversaciones entre los miembros de la pareja se enviaran al servidor asignado para cada pareja.

8.1.4.2 Sesión de interacción en UTEM.

En esta parte, se les proporcionó a los participantes una breve sesión de capacitación sobre el uso del asistente de etiquetado (3 minutos aprox.), Además, los participantes exploraron rápidamente las etiquetas disponibles (2 minutos, aprox.), para luego describirles el escenario en el que se ubicarían para llevar a cabo las tareas. Después se les asignó un rol a cada miembro de las parejas: el desarrollador que trabaja desarrollando el servicio Web tipo REST o el desarrollador que necesita conectarse a dicho servicio. Se le solicitó a cada miembro de cada pareja que se sentara en partes distantes de la sala en la que se desarrolló la sesión y que no se comunicaran vía oral entre ellos, de tal manera de que se simulara una distribución geográfica de los miembros. Posteriormente las parejas participantes usaron Skype para interactuar entre ellas, siguiendo el guion correspondiente a cada rol, en los cuales había 7 marcas que sugerían etiquetar un cierto mensaje. Los participantes debían seleccionar las etiquetas de la lista de opciones que ofrecía el asistente de etiquetado. También se les comentó que podrían etiquetar cualquier otro mensaje si lo consideraban necesario y que tenían permitido escribir una nueva etiqueta (que no figuraba en la lista de opciones) si no se

encontraba alguna que se ajustara al mensaje a etiquetar. Después de que las parejas terminaron de seguir el guion de interacción, respondieron el cuestionario SUS, para finalmente ejecutar el servicio de recopilación para enviar los mensajes de chat al repositorio asignado a cada pareja.

8.1.4.3 Sesión de búsqueda.

El objetivo de esta sesión fue medir el rendimiento de búsqueda de los participantes al emplear diferentes fuentes de AK. Por ello, esta sesión debía realizarse al menos dos días después de la sesión anterior, para evitar una situación en la que los participantes recordaran fácilmente los temas tratados durante la interacción en Skype, mitigando así el efecto de aprendizaje. Concretamente, en esta sesión los participantes debían responder 12 preguntas sobre los temas tratados en la sesión anterior, y buscar las respuestas en alguno de los tres diferentes medios disponibles: Skype, Trello y el buscador de ArchiKCo, según como lo indicaba la versión de la encuesta que se les hubiera asignado (de forma aleatoria). Se menciona Trello, ya que antes de comenzar esta sesión, se agregaron al repositorio de cada pareja los comentarios de las tarjetas que se prepararon en esta plataforma, por lo que estos comentarios podrían ser encontrados también mediante el buscador de ArchiKCo. Los participantes fueron entrenados para hacer búsquedas en los 3 medios disponibles, y se les indicó que podrían responder "No sé" si no encontraban alguna respuesta. Finalmente, después de responder las 12 preguntas de la encuesta, los participantes respondieron el cuestionario TAM extendido.

8.1.5 Variables e hipótesis

En esta sección se definen las variables independientes y dependientes (y la manera de medirlas), así como las hipótesis del estudio donde estas variables son utilizadas.

Respecto a las variables independientes, sólo se tuvo una que consistió en los diferentes medios utilizados para buscar las respuestas en la encuesta electrónica: Skype, Trello y el buscador de ArchiKCo. Las variables dependientes fueron las siguientes:

- **Preferencia de medios**. Esta variable se midió obteniendo el porcentaje de preguntas (de la 9 a la 12) en las que los participantes indicaron los medios (es decir, Skype, Trello o buscador de ArchiKCo) que habían utilizado para encontrar la respuesta.

- **Exactitud de respuestas**. Esta variable se midió obteniendo el porcentaje de respuestas correctas recibidas en la encuesta electrónica.
- **Tiempo para encontrar el conocimiento correcto**. Esta variable se midió mediante el uso de la encuesta electrónica en LimeSurvey, la cual registra el tiempo entre la presentación de una pregunta y cuando el participante hace clic en el botón "siguiente" para pasar a la siguiente pregunta. Esta variable fue posible medirla en el entorno académico, ya que sólo ahí se dieron las condiciones para controlar la ejecución de las tareas sin interrupciones.

Con base en estas variables dependientes se establecieron las siguientes hipótesis a probar con esta evaluación:

- **Preferencia de los medios**.
    - Hipótesis nula, H_{0a}: No hay diferencia en cuanto a la preferencia de buscar conocimiento utilizando cualquiera de los medios provistos.
    - Hipótesis alternativa H_{1a}: Existe una diferencia significativa en cuanto a la preferencia de buscar conocimiento usando un medio en específico.
- **Exactitud de las respuestas**.
    - Hipótesis nula, H_{0b}: No hay diferencia en el porcentaje de respuestas correctas encontradas usando cualquiera de los medios provistos.
    - Hipótesis alternativa H_{1b}: Existe una diferencia significativa en el porcentaje de respuestas correctas encontradas usando un medio en específico.
- **Tiempo para encontrar el conocimiento correcto**.
    - Hipótesis nula, H_{0c}: No hay diferencia en el tiempo requerido para encontrar las respuestas correctas usando cualquiera de los medios proporcionados.
    - Hipótesis alternativa H_{1c}: Existe una diferencia significativa en el tiempo para encontrar respuestas correctas usando un medio en específico.

Además de las variables dependientes que están asociadas a las hipótesis, se definieron tres variables más para tener una manera cuantitativa de determinar el comportamiento de etiquetado que se observe durante la ejecución de las tareas. Estas variables son las siguientes:

- **Validez de etiquetas**. La validez de una etiqueta se determina observando si ésta es parte del conjunto de etiquetas de usuario o parte del conjunto de meta-etiquetas, de tal forma que una etiqueta inválida es aquella que no es parte de ninguno de estos dos conjuntos.
- **Exactitud semántica de etiqueta**. Esto se determina observando si la semántica de la etiqueta usada en un mensaje, corresponde a la semántica de este último. Se considera que las únicas etiquetas que pueden ser catalogadas como correctas semánticamente hablando también deben de ser válidas.
- **Puntaje SUS del asistente de etiquetado**. Esta variable se midió mediante el procesamiento del cuestionario SUS (Sauro and Lewis, 2012) aplicado a los participantes.

8.2. Resultados

En esta parte se presentan los resultados de la evaluación descrita anteriormente. Estos resultados se presentan en tres partes: (1) los resultados relativos a la recuperación de AK en términos de preferencia de los medios, la exactitud de las respuestas, el tiempo de r espuesta y las percepciones de los participantes del buscador de ArchiKCo y de toda la solución de condensación de conocimiento; (2) los resultados relativos al comportamiento de etiquetado en términos de las instancias de etiqueta utilizadas correctamente, los errores de etiquetado y la usabilidad del ayudante de etiquetado; y (3) las percepciones generales de los participantes sobre el prototipo ArchiKCo.

8.2.1 Recuperación de AK.

En esta parte, presentamos las pruebas de las tres hipótesis planteadas para esta evaluación y los resultados obtenidos en el cuestionario TAM extendido, con el cual el buscador de ArchiKCo fue evaluado cualitativamente.

8.2.1.1 Preferencia de medios

El buscador de ArchiKCo fue preferido por los participantes sobre Skype y Trello en todos los casos (ver Figura 8.3). Vale la pena recordar que los participantes solo tuvieron la

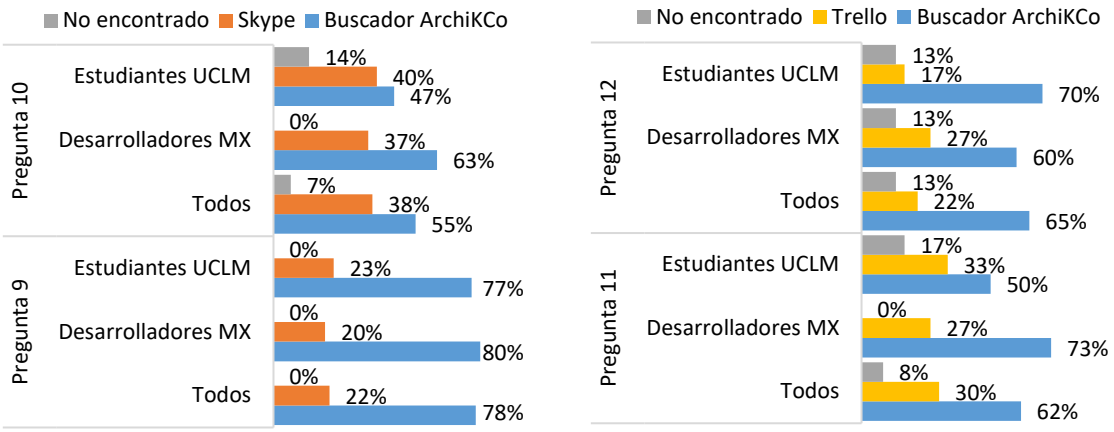


Figura 8.3. Preferencia de los medios con respecto a las preguntas de libre elección.

posibilidad de elegir el medio de búsqueda en las preguntas 9 a 12, que las respuestas a las preguntas 9 y 10 solo podían ser encontradas usando el buscador de ArchiKCo o Skype, y las respuestas a las preguntas 11 y 12 solo con el buscador de ArchiKCo o Trello. También cabe resaltar que hubo preguntas en las que los participantes no encontraron las respuestas y, por lo tanto, no se registró ningún medio preferido. Al respecto, en la Figura 8.3 se aprecia que algunos desarrolladores mexicanos no encontraron la respuesta sólo en la pregunta 12. Esto puede ser debido a la experiencia laboral de ellos, ya que el caso de estudio se apegó lo más posible a situaciones reales, y se usaron términos usados diariamente en proyectos tipo RESTful (Fielding and Taylor, 2002), para los cuales los estudiantes de la UCLM no tenían tanta experiencia, y por ello se les dificultó responder algunas preguntas, de hecho sólo en el caso de la pregunta 9 todos los participantes de la UCLM encontraron la respuesta.

Tomando en cuenta solamente las preguntas que pudieron responder los participantes de ambos perfiles (desarrolladores y estudiantes), se obtuvo que el buscador de ArchiKCo fue preferido el 69% de las veces (preferido 80 veces) y Skype sólo el 31% (preferido 36 veces), considerando las preguntas 9 y 10. En el caso de las preguntas 11 y 12, el buscador de ArchiKCo obtuvo el 71% de preferencia (75 veces preferido) y Trello obtuvo el 29 (preferido 31 veces). Para comprobar la significancia estadística de estos datos se aplicó la prueba de ajuste de bondad de Chi-cuadrada ($\alpha = 0.05$), donde se supuso una distribución uniforme en la preferencia de los medios como hipótesis nula (ver Tabla 8.4). Con esto se obtuvo evidencia suficiente para afirmar que las preferencias de los participantes no están distribuidas uniformemente y, por lo tanto, existe una tendencia a preferir el buscador de ArchiKCo en todos los casos. En consecuencia, es posible rechazar la hipótesis nula H_{0a} (no

Tabla 8.4. Prueba de ajuste de bondad de Chi-cuadrada (α = 0.05) sobre la preferencia de medios

	Preguntas 9 y 10		**Preguntas 11 y 12**	
	Buscador ArchiKCo	***Skype***	***Buscador ArchiKCo***	***Trello***
Veces preferido	80	36	76	31
Esperado	50%	50%	50%	50%
Valor X^2	16.69		18.925	
p-Value	< 0.001		< 0.001	

hay diferencia en cuanto a la preferencia de buscar conocimiento utilizando cualquiera de los medios provistos), ya que la prueba del ajuste de bondad indica una diferencia significativa en la preferencia de los medios.

8.2.1.2 Exactitud de las respuestas

Ambos grupos de participantes (estudiantes de la UCLM y desarrolladores mexicanos) tuvieron variaciones en cuanto al porcentaje de respuestas correctas cuando usaron el buscador de ArchiKCo o Skype (ver Figura 8.4). Por ejemplo, hubo casos en los que el Buscador ArchiKCo obtuvo una mayor corrección de respuestas que Skype, mientras que hubo otros en los que obtuvo una corrección de respuestas más baja que Skype, y otros en los que la corrección de las respuestas fue la misma para ambos medios.

También es posible observar que ambos conjuntos de participantes se comportaron de manera similar cuando usaron el buscador de ArchiKCo o Trello (ver Figura 8.5). Sin embargo, en este caso, los resultados sugieren que el buscador de ArchiKCo pudo haber obtenido un mayor porcentaje de respuestas correctas.

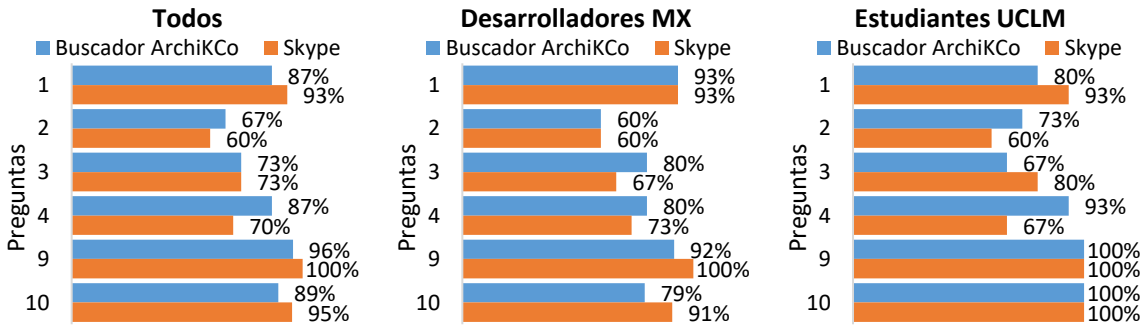


Figura 8.4. Comparación entre la exactitud de las respuestas recibidas usando el buscador de ArchiKCo y Skype.

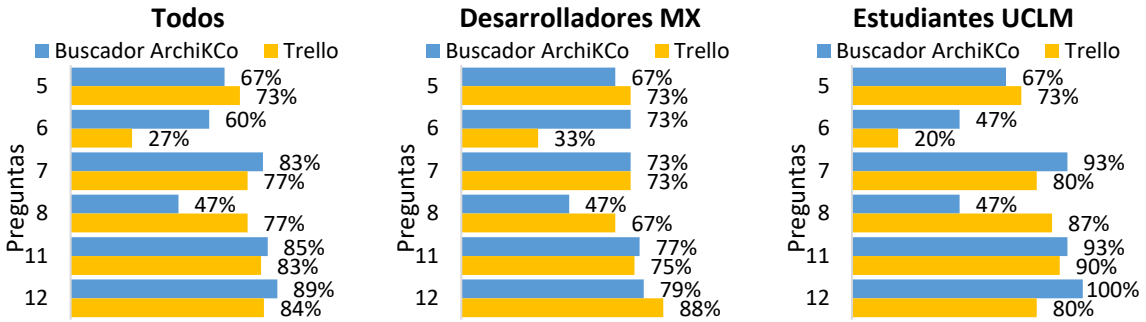


Figura 8.5. Comparación entre la exactitud de las respuestas recibidas usando el buscador de ArchiKCo y Trello.

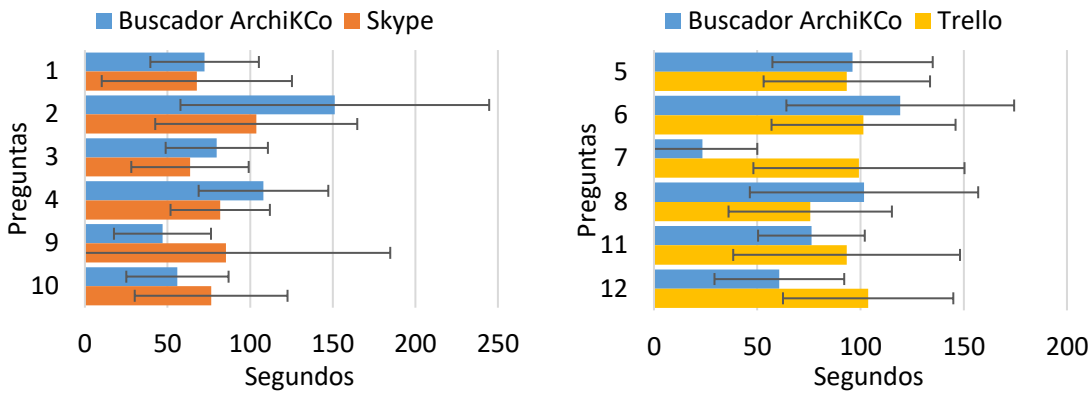
Con el fin de determinar si hay una diferencia significativa en la exactitud de las respuestas, se agruparon los datos de las preguntas 1 a 4, 9 y 10 (Buscador ArchiKCo vs Skype) y de las preguntas 5 a 8, 11 y 12 (Buscador ArchiKCo vs Trello) de ambos conjuntos de participantes (desarrolladores y estudiantes), de modo que se obtuvieron dos conjuntos de datos emparejados (Buscador ArchiKCo vs Skype y Buscador ArchiKCo vs Trello), donde cada uno de ellos contenía 12 elementos (los resultados de las 6 preguntas de los estudiantes y 6 preguntas de los desarrolladores). Posteriormente se aplicó la prueba de los rangos con signo de Wilcoxon (los datos no se distribuían normalmente), donde $\alpha = 0.05$, para ambos conjuntos. Se obtuvo que no hay diferencia entre la exactitud de las respuestas al usar el Buscador ArchiKCo o Trello, o usando el Buscador ArchiKCo o Skype, ya que W-value > $W_{\alpha = 0.05}$ en ambos casos (ver Tabla 8.5). Por lo tanto, no es posible rechazar la hipótesis nula H_{0b}: No hay diferencia en el porcentaje de respuestas correctas encontradas usando cualquiera de los medios provistos.

8.2.1.3 Tiempo para encontrar el conocimiento correcto

En la Figura 8.6 se observa que el tiempo requerido para obtener el conocimiento correcto con el buscador ArchiKCo fue mayor que el requerido usando Skype y Trello para la mayoría

Tabla 8.5. Detalles de la prueba de los rangos con signo de Wilcoxon sobre la exactitud de las respuestas ($\alpha = 0.05$).

	Buscador ArchiKCo vs Skype	**Buscador ArchiKCo vs Trello**
W - value	15	30
Diferencia de medias	0.35	0.42
Suma de rangos positivos	21	36
Suma de rangos negativos	15	30
$W_{\alpha = 0.05}$	3	10



(Las líneas horizontales representan la desviación estándar)

Figura 8.6. Tiempos de respuesta de las preguntas realizadas a los participantes.

de las preguntas en las que se indicó el medio a usar (preguntas 1 a 8). Sin embargo, en el caso de las preguntas donde los participantes fueron libres de elegir el medio de búsqueda (preguntas 9 a 12), se observó que usando el buscador ArchiKCo se requirió menos tiempo para obtener respuestas que usando Skype y Trello. Vale la pena recordar que las respuestas a las preguntas 9 y 10 sólo podrían encontrarse usando buscador ArchiKCo y Skype, y las respuestas a las preguntas 11 y 12 sólo usando buscador ArchiKCo y Trello.

Para comprobar estadísticamente la significancia de estos resultados, se agruparon los datos por medios y tipo de preguntas (elección libre de medios o medio indicado) y se prom ediaron los tiempos de respuesta registrados (ver Tabla 8.6). En el caso de las preguntas con un medio indicado, los participantes registraron menos tiempo usando el buscador ArchiKCo que usando Trello, pero usando Skype registraron un tiempo de respuesta menor que usando el buscador ArchiKCo, aunque en ambos casos se obtuvieron desviaciones estándar considerables. En el caso de las preguntas de libre elección, usando el buscador ArchiKCo los participantes registraron un menor tiempo de respuesta que usando los otros dos medios.

Tabla 8.6. Estadísticas descriptivas de los tiempos de respuesta registrados por los participantes (todos los valores están en segundos).

	Medios indicados				**Libre elección de medios**			
	Preguntas 1 - 4		**Preguntas 5 - 8**		**Preguntas 9 - 10**		**Preguntas 11 - 12**	
	Buscador ArchiKCo	***Skype***	***Buscador ArchiKCo***	***Trello***	***Buscador ArchiKCo***	***Skype***	***Buscador ArchiKCo***	***Trello***
Prom.	102.91	79.24	51.47	80.879	68.47	97.25	85.16	92.44
DE	60.86	48.86	29.88	68.01	29.92	50.55	56.85	43.51
Min.	39.37	14.07	7.51	15.04	23.76	21.71	7.32	15.15
Max.	341.54	250.34	131.52	239.99	140.83	179.26	215.4	174.33

**Prom. = Promedio, DE = Desviación estándar, Min. = Mínimo, Max. = Máximo*

Tabla 8.7. Detalles de la prueba de la U de Mann-Whitney sobre los tiempos de respuesta de los participantes ($\alpha = 0.05$).

	Medios indicados		**Libre elección de medios**	
	Buscador ArchiKCo vs Skype	**Buscador ArchiKCo vs Trello**	**Buscador ArchiKCo vs Skype**	**Buscador ArchiKCo vs Trello**
Suma de rangos	4278	3003	1596	1176
Media de rangos	46.5	39	28.5	24.5
Desviación estándar	128.03	98.15	57.79	43.10
U-value	724	600	285	141
Z-Score	2.60	1.43	-1.14	-1.99
p-value	0.009	0.153	0.254	0.045

Para comprobar la significancia en los tiempos obtenidos se aplicó la prueba U de Mann-Whitney (los datos no se distribuyeron normalmente) donde $\alpha = 0.05$, comparando el buscador ArchiKCo vs Skype, el buscador ArchiKCo vs Trello (ver detalles en Tabla 8.7). Se utilizó esta prueba estadística porque las muestras no estaban emparejadas, ya que sólo se consideró el tiempo necesario para encontrar las respuestas correctas.

Como se puede observar en la Tabla 8.7, en el caso de las preguntas con un medio indicado, hay una diferencia bastante grande en el tiempo de respuesta (p-valor $<\alpha$) entre el buscador ArchiKCo y Skype, lo que indica que los participantes encontraron las respuestas correctas más rápido al usar este último medio. En el mismo caso, no se obtuvo una diferencia considerable entre el buscador ArchiKCo y Trello, a pesar de que los participantes promediaron menos tiempo al usar el buscador ArchiKCo. En el caso de las preguntas de libre elección, hay una diferencia bastante grande en el tiempo de respuesta (p-valor $<\alpha$) entre el buscador ArchiKCo y Trello, lo que indica que los participantes encontraron las respuestas correctas más rápido cuando utilizan el buscador ArchiKCo. Sin embargo, no hay una diferencia considerable entre el buscador ArchiKCo y Skype, a pesar de que en promedio los participantes fueron más rápidos usando el buscador ArchiKCo. Esta evidencia permite rechazar la hipótesis nula H_{0c}: No hay diferencia en el tiempo requerido para encontrar las respuestas correctas usando cualquiera de los medios proporcionados, ya que hubo casos en que los participantes fueron más rápidos al usar el buscador ArchiKCo y otros en los que fueron más rápidos al usar Skype.

8.2.1.4 TAM extendido

Con respecto a las percepciones de los participantes sobre al buscador ArchiKCo (ver Tabla 8.8), los resultados del cuestionario TAM extendido permitieron concluir que el buscador ArchiKCo es percibido como muy útil y fácil de usar. Además, los participantes consideraron que este buscador podría ayudarlos a encontrar AK importante de manera oportuna. También percibieron que las interrupciones podrían reducirse usando el buscador ArchiKCo, ya que tendrían una fuente de AK, aparte de sus compañeros de equipo. El buscador también se percibió como una herramienta con la que se podría encontrar AK fácilmente durante los ciclos de desarrollo.

Para resumir los resultados presentados en esta sección, se puede afirmar que el buscador ArchiKCo fue ampliamente preferido por los participantes para buscar AK en comparación con Skype y Trello, cuando se les dio libre elección de los medios de búsqueda. También es posible afirmar que nuestros resultados evidencian que los desarrolladores de AGSD podrían tener un mejor rendimiento encontrando AK si utilizaran el buscador ArchiKCo cuando no conocen la fuente (aunque esto fue estadísticamente significativo sólo comparado contra Trello). Finalmente, el buscador ArchiKCo podría ser confiable al buscar conocimiento, ya que los participantes obtuvieron un alto porcentaje de respuestas correctas y no hubo una diferencia significativa con la exactitud obtenida con Skype o Trello.

Tabla 8.8. Resultados de TAM extendido sobre el buscador ArchiKCo

	Desarrolladores mexicanos					**Estudiantes UCLM**				
	Med.	***Mod.***	***Q.25***	***Q.50***	***Q.75***	***Med.***	***Mod.***	***Q.25***	***Q.50***	***Q.75***
Utilidad	6	6	5	6	6	6	6	5	6	6
Facilidad de uso	6	6	6	6	6	6	6	6	6	7
Reducción de interrupciones	6	6	5	6	7	6	5	5	6	6.75
Encontrar conocimiento importante	6	6	6	6	6	6	6	5.25	6	7
Encontrar conocimiento fácilmente	6	6	6	6	7	6	6	5	6	7
Encontrar conocimiento oportunamente	6	6	5	6	6	6	6	5	6	6.75

(Med = Mediana, Mod = Modo, Q = Cuartil). Todos los valores en Likert-7.

8.2.2 Comportamiento de etiquetado

En esta sección se analiza la manera en la que los participantes etiquetaron los mensajes durante la sesión de interacción, así como las percepciones de usabilidad al usar el asistente de etiquetado de ArchiKCo. Vale la pena recordar que la inclusion de este asistente en el prototipo ArchiKCo fue para ayudar a los usuarios a que utilicen la escritura exacta de etiquetas y, para que en consecuencia, se pueda reducir la explosión de etiquetas y sus problemas relacionados.

Para analizar el comportamiento de etiquetado se obtuvo el número de instancias de etiquetas utilizadas por cada participante y se identificaron las instancias de etiquetas válidas (etiquetas de usuario o meta-etiquetas) y no válidas (nuevas etiquetas y errores de tecleo). También se identificaron las instancias de etiquetas que se usaron correctamente en términos de su semántica. A continuación, se presentan los análisis respecto al uso de instancias de etiquetas válidas, inválidas y correctamente utilizadas.

8.2.2.1 Validez de las etiquetas

En la Figura 8.7 se observa que la mayoría de los desarrolladores mexicanos (80% aproximadamente) usaron entre 6 y 9 instancias de etiquetas durante la sesión de interacción en Skype, mientras que los estudiantes de la UCLM (80% aproximadamente) usaron entre 3 y 8 instancias de etiquetas. Esto significa que un porcentaje considerable de todos los participantes (50% aproximadamente) usaron etiquetas válidas como se esperaba o más (al menos 7 mensajes etiquetados). Además, algunos participantes usaron 12 o más instancias de etiquetas válidas (13% aproximadamente), es decir, al menos usaron 5 instancias más de

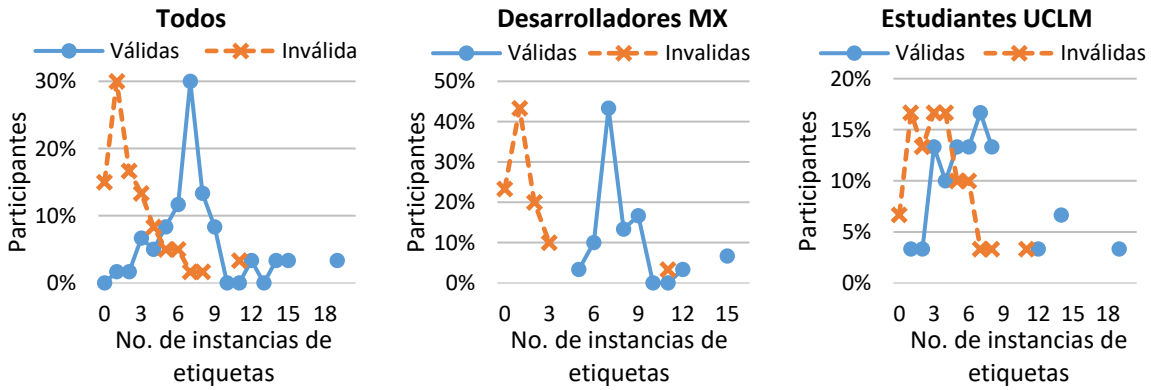


Figura 8.7. Número de instancias de etiquetas válidas e inválidas de los participantes.

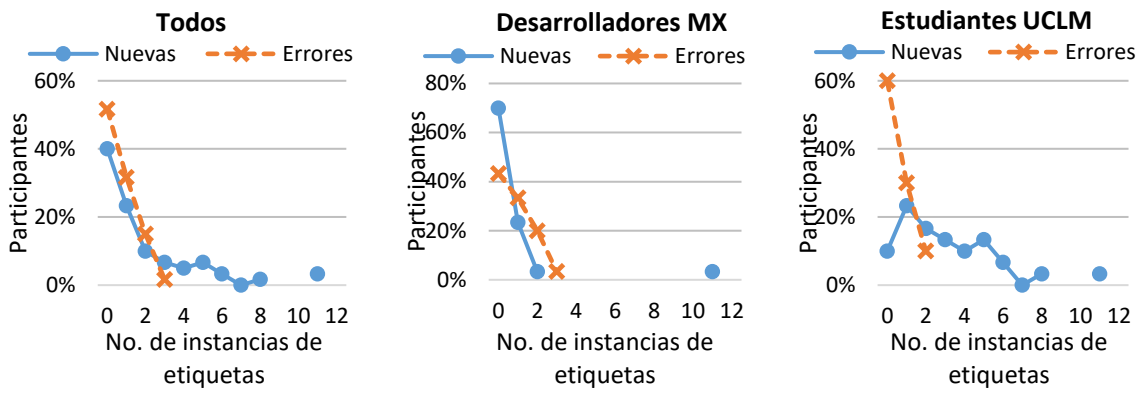


Figura 8.8. Número de instancias de etiquetas nuevas y de errores de tecleo de los participantes.

lo esperado. Esto evidencia que los participantes tuvieron la iniciativa de etiquetar mensajes aun en partes del guion donde no estaba sugerido.

Considerando instancias de etiquetas inválidas, el 23% de los desarrolladores mexicanos no usaron etiquetas inválidas, mientras que sólo el 6% de los estudiantes de la UCLM tampoco usaron este tipo de etiquetas (ver la líneas punteadas en la Figura 8.7). Por lo tanto, es posible afirmar que los desarrolladores mexicanos fueron más disciplinados al etiquetar mensajes. De hecho, la mayoría de los desarrolladores mexicanos (73% aproximadamente) sólo usaron de 1 a 3 instancias de etiquetas inválidas, mientras que el 86% de los estudiantes de la UCLM usaron instancias de etiquetas inválidas (de 1 a 12). Las etiquetas inválidas pueden representar errores de tecleo o nuevas etiquetas, ya que los participantes tuvieron la opción de crear nuevas etiquetas en caso de no encontrar una adecuada. En ese sentido, el 53% de los estudiantes de la UCLM usaron entre una y tres instancias de etiquetas nuevas, mientras que sólo el 23% de los desarrolladores mexicanos usaron una instancia de etiqueta nueva (ver líneas continuas en la Figura 8.8). Además, el 33% de los estudiantes usaron entre 4 y 8 instancias de etiquetas nuevas, lo que indica una gran disposición a etiquetar mensajes, pero también refleja su inexperiencia en los temas del guion, lo que se tradujo en la creación de nuevas etiquetas que se ajustaran a su conocimiento. La Figura 8.8 también muestra que alrededor del 50% de todos los participantes no cometieron errores de tecleo en las instancias de etiquetas usadas, y que el 46% de ellos solo tuvieron uno o dos errores. Esto podría significar que el asistente de etiquetado realmente fue útil para tener una baja tasa de errores.

8.2.2.2 Exactitud semántica de las etiquetas

Mientras que un alto porcentaje de participantes usaron instancias de etiquetas válidas, el 15% de todos los participantes no tuvieron errores semánticos al usar esas instancias, y el 46% usaron erróneamente sólo una o dos instancias de etiquetas (ver la línea punteada la Figura 8.9). Esto es significativo, ya que los participantes no registraron sus propias etiquetas y, por lo tanto, no conocían la semántica exacta de cada una de ellas. Sin embargo, el 50% de los desarrolladores mexicanos tuvieron entre 3 y 5 instancias usadas incorrectamente, mientras que sólo el 23% de los estudiantes de la UCLM etiquetaron incorrectamente en el mismo rango. Lo anterior pudiera deberse a que los estudiantes no tenían ninguna presión por reincorporarse a sus actividades una vez terminada la evaluación, por lo tanto podían dedicar más tiempo para elegir una etiqueta adecuada. Respecto a las instancias de etiquetas correctamente utilizadas, el 78% de todos los participantes tuvieron entre 2 y 6 instancias correctas; los desarrolladores mexicanos registraron más variabilidad que los estudiantes de la UCLM (ver línea continua en la Figura 8.9). Un pequeño grupo de participantes (13%, aproximadamente) registraron entre 8 y 10 instancias utilizadas correctamente, destacando nuevamente que los participantes no conocían las etiquetas por adelantado.

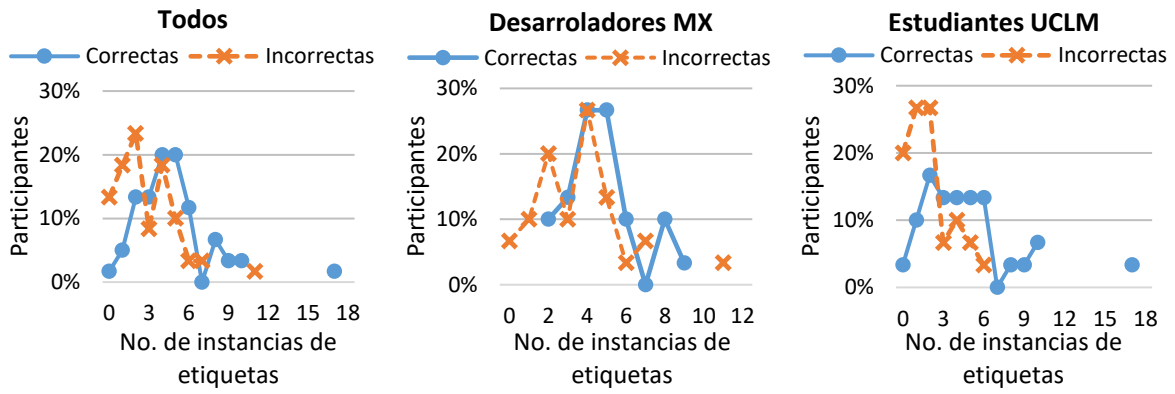


Figura 8.9. Número de instancias de etiquetas correctas e incorrectas de los participantes.

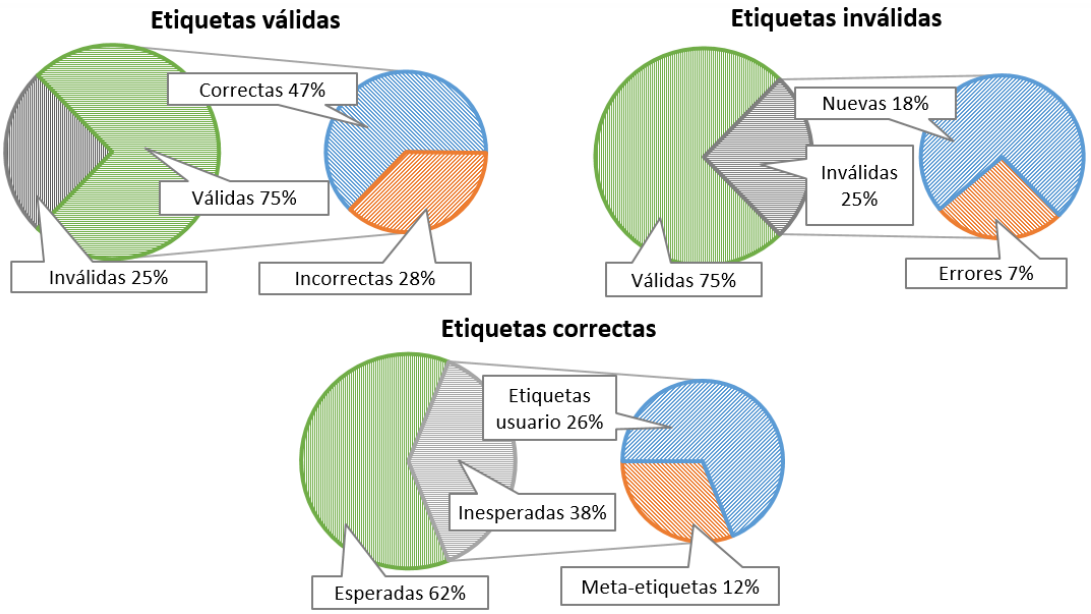


Figura 8.10. Distribución general de las etiquetas válidas, inválidas y correctas.

8.2.2.3 Comportamiento en general del etiquetado

Resumiendo los resultados sobre comportamiento de etiquetado (ver Figura 8.10), se puede determinar que los participantes utilizaron más instancias de etiquetas válidas (75%) que inválidas (25%), y más instancias de etiqueta correctamente usadas (47%) que las usadas incorrectamente (28%). Ambas diferencias se confirmaron estadísticamente mediante la aplicación de la prueba de rangos con signo de Wilcoxon (ver Tabla 8.9). Con respecto a las etiquetas inválidas, hubo un 18% de instancias de etiquetas nuevas y sólo un 7% de errores de tecleo (diferencias que también fueron estadísticamente significativas, ver Tabla 8.9), es

Tabla 8.9. Prueba de rangos con signo de Wilcoxon: detalles de instancias de etiquetas válidas y no válidas, instancias de etiquetas correctas e incorrectas, e instancias de etiquetas nuevas y errores de escritura ($\alpha = 0.05$).

	Válidas vs Inválidas	**Correctas vs. Incorrectas**	**Etiquetas nuevas vs Errores**
W - value	109.5	283	317.5
Diferencia de medias	4.64	4.07	-0.17
Suma de rangos postivos	1601.5	1202	858.5
Suma de rangos negativos	109.5	283	317.5
Z-value	-5.7758	-3.9564	-2.7744
Media (W)	855.5	742.5	588
Desviación estándar (W)	129.16	116.14	97.5
p-value	0	0.000008	0.0056

decir, las instancias inválidas sucedieron más por la necesidad de etiquetar de los participantes que por los errores causados por el mecanismo de etiquetado.

También se observó que el 38% de las instancias de etiquetas usadas correctamente eran inesperadas (ver Figura 8.10), es decir, instancias que encajaban en la semántica del mensaje, pero no era la instancia que se esperaba, o bien, instancias que usaron correctamente en mensajes donde no se sugirió etiquetar. Las etiquetas inesperadas están compuestas por el 26% de etiquetas de usuario y el 12% de meta-etiquetas. Las meta-etiquetas inesperadas fueron las siguientes: #TechnologicalSuport, #Component, #Software, #Code y #Necesity, lo que significa que estas entidades de la ontología, que son la base de las meta-etiquetas, parecen ser intuitivas, ya que no se explicó su significado.

8.2.2.4 Percepción de usabilidad y de obstrucción del asistente de etiquetado

Después de que los participantes terminaron la sesión de interacción por Skype, respondieron a un cuestionario SUS centrado en el asistente de etiquetado. Este cuestionario incluía dos preguntas al estilo SUS para explorar la percepción de los participantes sobre la obstrucción del asistente en sus tareas cotidianas. Se obtuvo el puntaje SUS de todos los cuestionarios, y se transformaron éstos puntajes de acuerdo con la escala de calificación con curva (Sauro and Lewis, 2012) (ver Figura 8.11), incluyendo las dos preguntas respecto obstrucción del asistente. El asistente de etiquetado obtuvo una media del puntaje SUS de 77 (desviación estándar = 13, mediana = 78 = B+), que de acuerdo con la escala de calificación con curva, corresponde a una calificación de B. Según el estudio de (Bangor et al., 2008), la puntuación SUS obtenida corresponde a una percepción buena de usabilidad. Ambos conjuntos de participantes se comportaron de manera similar con respecto a la percepción de usabilidad

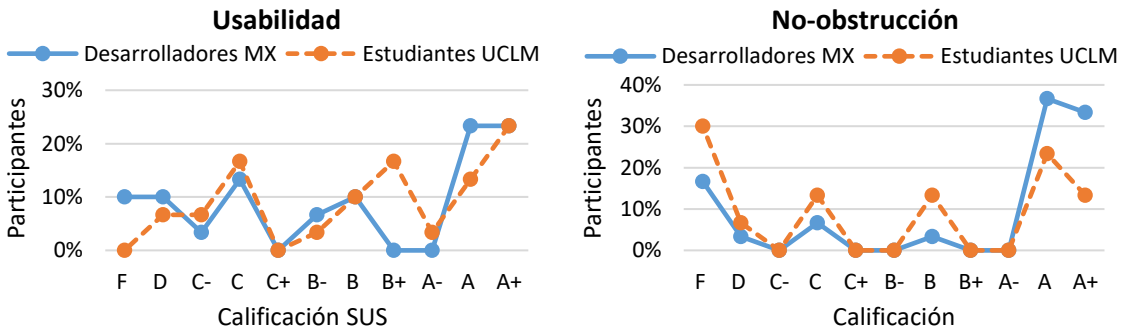


Figura 8.11. Percepción de los usuarios acerca de la usabilidad y la no-obstrucción del asistente de etiquetado.

(ver Figura 8.11). Sin embargo, alrededor del 20% de los desarrolladores mexicanos tuvieron una baja percepción de usabilidad (grados D y F), principalmente causada por el mecanismo utilizado para navegar en la lista de etiquetas sugeridas y para seleccionar una etiqueta; los participantes tenían que presionar la tecla <Alt> más las flechas de navegación para moverse en la lista de sugerencias, y la tecla <Alt> + <Enter> para seleccionar una etiqueta.

Por otro lado, ambos grupos de participantes se comportaron de manera similar en lo que respecta a su percepción de obstrucción del asistente (ver Figura 8.11). En general, los participantes obtuvieron un puntaje promedio de 72 (desviación estándar = 20, mediana = 83 = A), que corresponde a un grado C+. En este caso, un porcentaje importante de los participantes consideró que el asistente de etiquetado sería molesto en su trabajo diario (30% de los estudiantes de la UCLM y 17% de los desarrolladores mexicanos). Al analizar los comentarios de los participantes, se concluyó que esta percepción fue causada por el mecanismo de navegación y selección de etiquetas, pero también por las etiquetas disponibles, ya que los participantes no fueron quienes las registraron en la aplicación web de administración de etiquetas.

Para resumir los resultados sobre el comportamiento de etiquetado, se observó una tasa significativa de instancias de etiquetas válidas que fueron utilizadas correctamente, con una baja tasa de errores de tecleo. Este comportamiento podría conducir a una reducción en la explosión de etiquetas (y sus problemas relacionados) al utilizar el asistente de etiquetado para clasificar AK en UTEM. Por lo anterior, el asistente tiene una buena posibilidad de ser adoptado en el futuro, ya que se percibe como usable y con poca obstrucción en el trabajo diario por la mayoría de los participantes.

8.2.3 Percepción general del prototipo ArchiKCo

Se incluyeron dos preguntas en el cuestionario TAM para obtener la percepción general sobre el prototipo ArchiKCo, tomando en cuenta que ellos sólo utilizaron el asistente de etiquetado y el buscador, que se puede considerar que son los componentes críticos que fueron usados directamente por los participantes. Una de las preguntas pidió a los participantes que calificaran el prototipo en una escala del 1 al 10, resultando que tanto los desarrolladores mexicanos como los estudiantes de la UCLM dieron una calificación con una mediana de 8 (moda = 8), lo cual se puede considerar como satisfactorio.

Además de calificar a ArchiKCo, los participantes también indicaron la razón de la calificación otorgada. La mayoría de ellos comentó que ArchiKCo podría acelerar la recuperación de conocimiento, y que podría reducir las interrupciones y la repetición de información entre sus compañeros de equipo. Los participantes también apreciaron que el buscador de ArchiKCo podría funcionar como un único punto de referencia, en lugar de tener que buscar en múltiples fuentes varias veces, ya que con frecuencia no recuerdan dónde se encuentra el conocimiento que se necesita. A este respecto, uno de los participantes comentó: *"... las notas y los requisitos no se discuten en el mismo lugar... hay conflictos porque no todas las partes involucradas tienen acceso a esta información todo el tiempo...*".

Aunque ArchiKCo fue bien recibido y percibido por los participantes, señalaron que el etiquetado es un aspecto crucial y que las etiquetas deberían ser previamente acordadas por las partes interesadas. Además, los participantes comentaron que deben acostumbrarse a etiquetar sus conversaciones, refiriéndose a Skype, ya que no es común etiquetar en este tipo de medios síncronos (el etiquetado social es más común en medios asíncronos, como Facebook y Twitter). Con respecto al uso de etiquetas para recuperar conocimiento, algunos participantes comentaron que éstas no fueron relevantes al hacer las búsquedas; uno de ellos declaró que: "*Terminaría no usando etiquetas, ya que buscaría palabras, no etiquetas*". De hecho, mientras los participantes usaban el buscador de ArchiKCo, se observó que la mayoría de los participantes prefería buscar usando texto libre, pero algunos de ellos usaban las etiquetas para refinar los resultados obtenidos mediante una búsqueda de texto libre (usando los filtros ofrecidos a través de los paneles a lado de la lista de resultados). También, al respecto del buscador un participante comentó: "*podría causar mucha distracción porque las conversaciones realmente importantes se mezclan con conversaciones personales, chistes, etc.*"; esta situación podría ser realmente problemática para la recuperación de AK y, por lo tanto, sería conveniente agregar un mecanismo para excluir mensajes que se consideren irrelevantes o sólo buscar mensajes etiquetados.

Como último punto, en el cuestionario TAM se les pidió a los participantes sugerencias de mejora a todo el prototipo. Se recibieron varias sugerencias de mejora al asistente de etiquetado, siendo las siguientes las más relevantes: un etiquetado automático o sugerencia inteligente de etiquetas dependiendo de los temas de conversación, agregar

etiquetas nuevas durante la conversación y, seleccionar etiquetas de una lista de tendencia de temas.

8.3. Amenazas a la validez

Para entender en qué medida son válidos los resultados anteriormente presentados y cómo se podrían usar, a continuación se presenta una discusión sobre las amenazas de la validez del estudio expuesto, según la taxonomía de (Wohlin et al., 2000).

8.3.1 Amenazas a la validez del constructo

El tiempo requerido para responder cada una de las preguntas relacionadas con los temas de la interacción en Skype se midió utilizando una función Limesurvey (herramienta en la que se subió dicha encuesta) que registra el tiempo que transcurre entre que se muestra una pregunta y que el participante hace clic en el botón para mostrar la siguiente. Este tiempo podría verse afectado por la velocidad de lectura de cada participante y por el tiempo necesario para comprender la pregunta en curso. Se consideró que ambos perfiles de participantes (los desarrolladores mexicanos y los estudiantes de la UCLM) tenían habilidades similares, por lo que esta amenaza podría haberse reducido con la disposición de los grupos.

Con el fin de saber si los participantes perciben al asistente de etiquetado como una obstrucción a su trabajo diario, se extendió el cuestionario estándar SUS agregando dos preguntas respetando el estilo, es decir, una pregunta en positivo y la otra negativo en referencia a este tema de la obstrucción. Las respuestas a estas dos preguntas se procesaron siguiendo los pasos convencionales para obtener la puntuación SUS. Se es consciente de que la extensión del cuestionario SUS no es común, pero de esta manera se contó con un medio estructurado para obtener las percepciones de los participantes sobre temas que el cuestionario SUS no incluye.

8.3.2 Amenazas a la validez interna

Durante la evaluación se lidió con los siguientes temas: diferencias entre los sujetos, efecto de aprendizaje, efectos de la fatiga, efectos de persistencia, motivación de los sujetos y el paquete de estudio. A continuación, se proporciona una explicación sobre cómo se manejaron estas amenazas.

- **Diferencias entre sujetos**. Cuando se utiliza un diseño intra-sujetos, como el que fue utilizado en este estudio, se reduce la variación de error debido a las diferencias entre los mismos sujetos. En este estudio, todos los participantes (desarrolladores y estudiantes) estaban familiarizados con las herramientas Skype y Trello. Sin embargo, para asegurar que todos tuvieran el mismo nivel de conocimiento de estas herramientas, los participantes recibieron una breve sesión de capacitación sobre cómo buscar información en estos medios.
- **Efectos de aprendizaje**. Todos los participantes respondieron las mismas 12 preguntas durante la sesión de búsqueda usando el prototipo. Para mitigar el efecto de aprendizaje, se utilizó una técnica de contrapeso, es decir, se colocó a los participantes en grupos y se presentaron las condiciones para contestar cada pregunta (los medios indicados para buscar) a cada grupo en un orden diferente (especificado en la Tabla 6).
- **Efectos de fatiga**. Se estructuró el estudio en tres sesiones, las cuales duraron un tiempo suficientemente corto como para evitar que los participantes se fatigaran durante el estudio, lo que hizo que los efectos de fatiga fueran irrelevantes.
- **Efectos de persistencia**. Para evitar los efectos de persistencia, el estudio se realizó con sujetos que nunca habían tomado parte en un estudio similar. Además, el escenario de contexto no se trataba de un proyecto real, por lo tanto, no era posible que los participantes tuvieran conocimiento previo del mismo. Además, se realizó la sesión de búsqueda de conocimiento al menos dos días después de la sesión de interacción en Skype, para evitar que los participantes recordaran detalles sobre los temas del guion.
- **Motivación de los sujetos**. Los participantes fueron voluntarios, lo que garantiza que los sujetos tuvieran motivación a ser parte del estudio científico. De hecho, algunos de los participantes solicitaron que se les notificara sobre las siguientes versiones del prototipo, debido al interés que les generó el tema que se está atendiendo.
- **Paquete de estudio**. A fin de observar claramente los resultados de los tratamientos sobre el desempeño de los sujetos, ellos recibieron el mismo conjunto de preguntas a ser buscadas en los tres medios disponibles. Además, todas las ellas podían responderse en los medios indicados, reduciendo el riesgo de que los participantes no pudieran encontrar la respuesta correcta. Si esta condición no se hubiera respetado, las diferencias entre las tareas podrían haber sido factores de confusión y haber perjudicado el análisis. La prueba

de rangos con signo de Wilcoxon confirmó que no hubo una diferencia significativa en las respuestas correctas entre los medios utilizados (ver Tabla 8), por lo tanto, los resultados se pueden considerar independientes del paquete de estudio.

8.3.3 Amenazas a la validez externa

A continuación, se presenta como se manejaron las amenazas a la validez externa de este estudio, seccionada en: validez de los materiales y tareas, y validez de los sujetos.

- **Materiales y tareas**. Los guiones que se prepararon para llevar a cabo la interacción mediante Skype, se basaron en un escenario de proyecto ficticio, pero con características del mundo real de tal manera que los participantes sintieran que participaban en una conversación típica de su trabajo. De hecho, muchos de los participantes se adentraron tanto en su papel que incluso agregaban elementos de broma a los mensajes del guion, tal como sucede en su cotidianidad. En cuanto al etiquetado de los mensajes, se recibieron etiquetas en partes donde el guion lo sugería, lo cual refleja la adopción del papel de los participantes, así como la cercanía a la realidad que tuvo tanto el escenario como los guiones diseñados. Sin embargo, la necesidad de buscar conocimiento estuvo motivada por un cuestionario, y no por una necesidad genuina del proyecto en cuestión. Por lo tanto, para futuras evaluaciones se tendría que considerar el uso de escenarios reales, ya que estos son más complejos y articulados, y podrían motivar naturalmente la necesidad de obtener AK.
- **Sujetos**. Los desarrolladores con experiencia en AGSD que participaron en el estudio, ayudan a dar validez externa al mismo. Sin embargo, también se incluyeron estudiantes como participantes del estudio para tener condiciones más controladas en un entorno académico, particularmente para medir los tiempos de respuesta. Desafortunadamente, estos estudiantes no tenían experiencia real en proyectos globales y ágiles, pero reportaron haber participado en cursos sobre metodologías ágiles y desarrollo de software global durante su educación universitaria.

8.4. Discusión

En esta sección se discuten los resultados presentados anteriormente, divididos en tres partes: (1) una discusión sobre el mecanismo de clasificación de AK (implementado con etiquetado

social), (2) una discusión sobre el mecanismo de búsqueda de AK, y (3) una discusión sobre la viabilidad del concepto de condensación de AK.

8.4.1 Mecanismo de clasificación de AK

La literatura reporta que los desarrolladores prefieren usar etiquetado libre (creado por ellos mismos al vuelo) dada su baja carga cognitiva en el trabajo diario (Storey et al., 2009; Treude and Storey, 2009). En el caso del prototipo ArchiKCo, su mecanismo de clasificación se basó en el etiquetado asistido (a través del asistente de etiquetado), al igual que IBM® Rational® Jazz® (Treude and Storey, 2009), TagSEA (Storey et al., 2009) y eMoose (Dekel and Herbsleb, 2008). El asistenten de etiquetado de ArchiKCo, a diferencia de éstas últimas tres herramientas, utiliza etiquetas predefinidas por el usuario, que a su vez están asociadas a un conjunto fijo de meta-etiquetas, en lugar de permitir el etiquetado libre. Los resultados no reflejan disgusto por parte de los participantes acerca de usar etiquetas predefinidas, simplemente comentaron que hubiera sido mejor si ellos definieran las etiquetas que usaron durante la evaluación. Además, los participantes indicaron que podría ser apropiado incluir un mecanismo para agregar etiquetas sobre la marcha o una manera de sugerir etiquetas en función del contexto de los temas de interacción, pero ninguno de ellos mencionó el etiquetado libre. Sin embargo, los resultados sobre etiquetado reportados en la literatura se obtuvieron de estudios empíricos, por lo tanto, como siguiente paso se debería explorar las percepciones de los participantes en estudios a más largo plazo usando el asistente de etiquetado

Por otro lado, los participantes no mostraron ningún signo de disgusto con respecto a etiquetar mensajes de conversación, de tal manera que alrededor del 50% de ellos usaron al menos siete instancias de etiquetas, cuando las que se esperaban eran máximo siete, ya que sólo había siete marcas de etiquetado en los guiones. En el estudio reportado en el capítulo 6 (así como en (G Borrego et al., 2017)), donde se evaluó sólo el mecanismo de etiquetado, alrededor del 90% de los participantes utilizaron al menos tres instancias de etiquetas, usando guiones con 3 sugerencias de etiquetado. Ambos resultados evidencian que los desarrolladores pueden presentar la necesidad y voluntad de etiquetar mensajes durante las conversaciones, lo que indica que un mecanismo de clasificación de AK basado en el etiquetado social podría ser exitoso.

Vale la pena recordar que se decidió usar un mecanismo de etiquetado asistido para evitar problemas como la explosión de etiquetas, lo que podría arruinar la clasificación de AK. En ese sentido, los resultados no muestran signos de explosión de etiquetas, a pesar de que se permitió a los participantes usar nuevas etiquetas si lo consideraban necesario (sólo hubo 18% de las instancias de etiquetas no registradas).

El mecanismo de etiquetado también podría ayudar a reducir el problema de la similitud oscura (Bagheri and Ensan, 2016), ya que los participantes seleccionaron etiquetas de una lista de sugerencias, por lo tanto los mensajes se etiquetaron con instancias de etiquetas escritas correctamente. Esto es respaldado por el hecho de que sólo hubo un 7% de errores de tecleo de etiquetas durante las sesiones de interacción en Skype. Esta baja tasa de error podría contribuir a mantener la funcionalidad el mecanismo de clasificación de AK.

Aunque los participantes no conocían las etiquetas registradas, el 47% de las instancias de etiquetas se usaron correctamente (una diferencia significativa en comparación con los usos incorrectos de etiquetas), es decir, la semántica de las etiquetas correspondió a la semántica del mensaje etiquetado. Sin embargo, esta exactitud fue menor que la obtenida por (Sohan et al., 2010) usando un mecanismo inteligente de auto-etiquetado (70% de precisión). Al respecto, si los participantes definen en grupo sus propias etiquetas en futuras evaluaciones, podría incrementarse considerablemente la precisión de etiquetado. Pensando en una evaluación a largo plazo, podría darse el caso de que un grupo de desarrolladores haya definido sus etiquetas y posteriormente se integre un nuevo elemento al equipo. Este elemento tendría que contextualizarse con etiquetado durante un tiempo, en el cual probablemente no tenga la misma precisión semántica que sus compañeros usando las etiquetas.

Una parte clave del esquema de clasificación propuesto en esta implementación del concepto de condensación de AK, es el uso de meta-etiquetas fijas. Al respecto, el 43% de los participantes usaron meta-etiquetas de manera correcta, donde un 12% fueron en partes del guion donde no se indicaba el etiquetado. En el estudio anterior, dedicado únicamente al etiquetado (presentado en el capítulo 6 y en [51]), el 30% de los participantes usaron meta-etiquetas correctamente. Por lo tanto, esto podría considerarse como evidencia de que las entidades de la ontología en la que se basan las meta-etiquetas son expresivas, y que están

relacionadas con situaciones que suceden en AGSD en términos de AK. Sin embargo, se deben realizar estudios centrados sólo en la validación de la ontología.

Todos los resultados obtenidos conducen a que el asistente de etiquetado de ArchiKCo podría ser parte de un buen mecanismo de clasificación de AK para ser usado durante las conversaciones de UTEM. Aunque los participantes percibieron que el asistente es usable (puntuación SUS 77 = B), esta percepción fue inferior a la obtenida en el estudio anterior [51] (puntuación SUS 87 = A+), en la que se evaluó la implementación Web del asistente de etiquetado, que estaba integrado a un mensajero instantáneo Web desarrollado expresamente para dicha evaluación. Por lo anterior, se tenía control absoluto sobre las características de autocompletado y, por lo tanto, los participantes reportaron menos problemas con la selección y navegación de etiquetas. En cambio, el asistente empleado en el estudio reportado en este capítulo, se integró a Skype, para lo cual se tuvieron que adaptar las funciones de selección y navegación para no interferir con las funciones de Skype. A pesar de esta diferencia en usabilidad, el asistente de etiquetado basado en Skype no fue percibido como una obstrucción al trabajo diario (puntaje 72 = grado C+), pero esta percepción podría mejorarse si la navegación y la selección de etiquetas también se mejoran, aunque algunos participantes comentaron que sólo era cuestión de acostumbrarse a las características del asistente de etiquetado.

8.4.2 Mecanismo de búsqueda de AK

Los resultados presentados anteriormente indicaron que el buscador ArchiKCo es una aplicación confiable y preferida por los desarrolladores cuando buscan AK. Además, indican que usando el buscador ArchiKCo se tiende a encontrar conocimiento más rápido que en otros medios particularmente cuando no se conoce la fuente de antemano. Sin embargo, cuando se indicó el medio para hacer la búsqueda, los participantes fueron significativamente más rápidos usando Skype que usando el buscador ArchiKCo. Se presume que este comportamiento es causado por el diseño de la interfaz de usuario. Si bien la búsqueda en Skype consta de al menos tres pasos (1º - Ctrl + F para mostrar el cuadro de texto de búsqueda, 2º - escribir texto para buscar, 3º - presionar <Enter>), con el buscador ArchiKCo se necesitan más pasos para mostrar un resultado específico (1º - escribir texto buscar, 2º - presionar <Enter>, 3º - buscar un mensaje interesante en la lista de resultados, 4º - abrir la conversación correspondiente, 5º - buscar el conocimiento en la conversación). Por lo tanto,

en una situación real, si un desarrollador recuerda la fuente de un determinado elemento de conocimiento, podría ser mejor buscar directamente en esa fuente. Sin embargo, si un desarrollador necesita algo más que búsqueda de texto libre para encontrar un elemento de conocimiento específico, el buscador ArchiKCo podría ser la mejor opción, ya que ofrece más parámetros (fecha de rango y filtro de etiquetas) con los cuales realizar una búsqueda, además de elementos para refinar el conjunto de resultados (por etiqueta, por autor del mensaje, por destinatario del mensaje y por fuente de conocimiento).

Durante la sesión de búsqueda, se observó que la mayoría de los participantes preferían buscar usando texto libre, y que algunos de ellos usaban el panel de etiquetas para refinar los resultados obtenidos. Esto significa que los participantes utilizaron muy poco el mecanismo de clasificación propuesto para encontrar conocimiento como primera opción, aunque comentaron que etiquetar las conversaciones es una manera interesante para buscar conocimiento más adelante. Esto conduce a pensar que las etiquetas podrían ser más útiles para buscar AK si un desarrollador desea obtener una visión general para luego llegar a una vista detallada. Por ejemplo, cuando un nuevo miembro del equipo necesita adquirir conocimiento del proyecto en curso, un buen punto de partida podría ser explorar las etiquetas existentes y luego explorar los comentarios de una etiqueta en particular con mayor profundidad.

Los comentarios de los participantes y las observaciones durante la ejecución de las tareas mostraron la necesidad de realizar mejoras en el buscador ArchiKCo. Una de estas mejoras es presentar los resultados de diferentes maneras dependiendo de la naturaleza de la fuente de AK, es decir, dependiendo de si es un medio síncrono o asíncrono. En el estudio presentado en este capítulo, Trello podría considerarse asincrónico, ya que los comentarios de las tarjetas de tareas no se reciben con la misma frecuencia que los mensajes de Skype en una conversación. Entonces, para el caso de los medios asíncronos la función del buscador ArchiKCo con la que se muestra un rango de mensajes 5 minutos después y antes de un mensaje seleccionado podría ser inútil ya que los comentarios de este tipo de medios podrían tener una mayor diferencia de tiempo, y por lo que de entrada podrían no mostrarse mensajes usando esa función. La alternativa en este caso sería mostrar un número fijo de mensajes previos y posteriores al mensaje seleccionado.

Otra posibilidad de mejora del buscador ArchiKCo sería incluir una función para mostrar sólo los mensajes etiquetados en el primer conjunto de resultados. Con esto, el problema de mostrar mensajes irrelevantes, como interacciones personales, bromas, etc., podría reducirse, ya que estos tipos de mensajes no se etiquetarían. Otra forma de reducir este problema sería agregar etiquetas de exclusión que le indiquen al Servicio Recopilador que no envíe ciertos mensajes al repositorio.

Con el fin de intentar comparar el buscador ArchiKCo con los buscadores de las herramientas de etiquetado presentadas en el capítulo 6, se presenta una lista de diferencias importantes entre éstas herramientas y el buscador ArchiKCo:

1. Las herramientas presentadas anteriormente no están enfocados en la búsqueda de conocimiento proveniente de interacciones electrónicas (con la excepción de TAGGER (Richter et al., 2004), que maneja este tipo de conversaciones pero no incluye un buscador);
2. Casi todas se basan únicamente en la búsqueda de texto libre, y solo TagSEA (Storey et al., 2009) e IBM Rational Jazz (Treude and Storey, 2009) incluyen parámetros adicionales como etiquetas o puntos de referencia;
3. No tienen un medio para refinar un conjunto de resultados;
4. No integran AK de diferentes fuentes (IBM Rational Jazz puede configurarse para hacerlo, pero en el artículo (Treude and Storey, 2009) al que se hace referencia no presenta ninguna integración);
5. No presentan resultados empíricos con respecto a la búsqueda en sus respectivos artículos.

Todas estas diferencias impiden hacer una comparación directa con los resultados obtenidos en la evaluación reportada en este capítulo.

A pesar de las oportunidades de mejora que se detectaron en la evaluación, el buscador ArchiKCo se percibió como un medio usable y útil que facilita el descubrimiento de AK durante un ciclo de desarrollo ágil, lo que podría reducir las interrupciones entre los compañeros de equipo cuando tengan preguntas sobre la arquitectura de algún proyecto. Además, los participantes percibieron que al utilizar el buscador podían encontrar AK de manera oportuna, a pesar de que éste sólo fue más rápido que Trello.

8.5. Viabilidad de la condensación de AK

El comportamiento de etiquetado observado y los resultados sobre la recuperación de conocimiento proporcionan evidencia suficiente para establecer que es factible implementar el concepto de condensación de AK en AGSD. Las razones para aseverar esto se explican a continuación.

- Los desarrolladores de AGSD son conscientes de que los registros de UTEM contienen AK importante, y con frecuencia ellos necesitan buscar temas de arquitectura en esos registros. Sin embargo, a menudo pasan demasiado tiempo buscando este conocimiento porque está disperso en diferentes UTEM y no recuerdan el lugar particular en el que se encuentra el conocimiento compartido.
- Los registros de UTEM carecen de una estructura que facilite la búsqueda de AK. Por ello se propuso un mecanismo de clasificación basado en el etiquetado social asistido. Este mecanismo fue bien utilizado por los desarrolladores de AGSD y se percibió como usable, útil y con poca obstrucción al trabajo diario.
- El mecanismo de recuperación de AK que se implementó en ArchiKCo fue bien recibido por los desarrolladores de AGSD, ya que integra diferentes fuentes de conocimiento, integrando diferentes interacciones en UTEM, por lo que en la evaluación prefirieron buscar el conocimiento solicitado en el buscador que hacerlo directamente en los UTEM. Además, al usar el mecanismo de recuperación, los desarrolladores de AGSD tendieron a descubrir conocimiento más rápido que cuando lo hicieron directamente en los UTEM. Todas estas razones llevaron a los desarrolladores de AGSD a percibir que el mecanismo de recuperación de ArchiKCo es usable y útil.

Cabe recordar que en esta tesis se está presentando una sola manera de implementar el concepto de condensación de conocimiento. Se podrían crear diferentes implementaciones únicamente considerando los elementos básicos de este concepto: (1) información de registro de UTEM accesible, (2) el mecanismo de clasificación de bitácoras de UTEM y (3) el mecanismo de búsqueda de AK. Por lo que podría ser interesante evaluar otras maneras de implementación para confirmar la viabilidad del concepto de condensación de AK.

8.6. Resumen del capítulo

En este capítulo se presentó una evaluación del prototipo ArchiKCo, como prueba de la viabilidad del concepto de condensación del AK. En dicha evaluación, desarrolladores mexicanos de AGSD y estudiantes españoles de licenciatura y de posgrado de la UCLM usaron el asistente de etiquetado y el buscador ArchiKCo para ejecutar diferentes tareas, que estuvieron basadas en situaciones con necesidad de compartir y buscar conocimiento en ambientes AGSD. Los principales hallazgos de esta evaluación se a continuación.

- El buscador ArchiKCo fue preferido por los participantes para buscar conocimiento por encima del resto de medios ofrecidos para ello (Skype y Trello), principalmente debido a integración de fuentes de conocimiento y a las herramientas de búsqueda ofrecidas.
- El buscador ArchiKCo registró menor tiempo en promedio para encontrar conocimiento cuando no se conoció de antemano el medio para hacer la búsqueda, pero sólo se tuvo una diferencia significativa cuando se comparó con Trello. Sin embargo, se proyecta que con tareas más complejas, o en situaciones reales de trabajo, el buscador ArchiKCo podría registrar menor tiempo que cualquier otro medio similar a Skype (e.g. Jabber).
- El buscador ArchiKCo fue percibido como usable y útil, al grado que se tuvo la sensación de encontrar AK más rápido con el buscador, cuando cuantitativamente no fue así.
- A pesar de que la literatura reporta que el etiquetado libre es preferido por los desarrolladores, el etiquetado asistido y basado en etiquetas ligadas a una estructura de meta-etiquetas, tiene potencial para ser una manera de clasificar AK que se comparta en UTEM en un ambiente de AGSD. Esto debido a que en más del 50% de los casos se recibieron al menos la cantidad de instancias de etiquetas esperadas, lo que significa que no se tuvo mucho inconveniente en etiquetar interacciones en UTEM.
- El asistente de etiquetado de ArchiKCo fue percibido también como útil, usable y con poca obstrucción al trabajo, aunque en menor medida que la versión anterior del mismo, presumiblemente porque esta versión de asistente se integró a un UTEM comercial (Skype) y no a un UTEM ad-hoc para una evaluación controlada.
- El uso de un asistente de etiquetado podría ayudar a elegir etiquetas semánticamente correctas aunque no se conozca de antemano el significado de ellas. Esto se determinó porque se tuvo un 47% de las instancias de etiquetas y meta-etiquetas usadas correctamente.

Los resultados de la evaluación presentada evidencian que el concepto de condensación de AK puede ser viable para ser implementado en entornos AGSD. El uso de implementaciones de este concepto podría reducir la cantidad de tiempo que se emplea para encontrar soluciones a problemas pasados, a reducir el número de interrupciones entre compañeros de equipo, ya que se dispondría de una fuente adicional de AK además de los mismos compañeros, es decir, se contaría con un condensador de conocimiento. En consecuencia, la implementación del concepto de condensación de AK podría reducir problemas como la deuda de documentación y la deuda técnica arquitectónica en entornos AGSD.

En el siguiente capítulo se retoman todos los hallazgos presentados en esta tesis para conformar las conclusiones de la misma, así como para puntualizar las aportaciones derivadas de esta investigación, y finalmente trazar los pasos hacia el trabajo futuro que se deriva de los resultados obtenidos.

Capítulo 9.
Conclusiones, aportaciones y trabajo futuro

En este trabajo de tesis se atiende la problemática del manejo del AK en el AGSD, la cual es provocada por la tendencia a la vaporización del AK, que a su vez es causada por la deuda de documentación existente en este ambiente. Esta deuda se produce por la preferencia de manejar AK en su estado tácito en lugar de manejarlo en su estado explícito, propiciado por las condiciones del AGSD. La deuda (o falta) de documentación y consecuentemente la vaporización de AK provoca en general retrasos en el término de los productos (parciales o completos), y que el producto de software no evolucione como se esperaba, dado que algunos lineamientos arquitectónicos pudieran haberse vaporizado.

Por medio de una revisión de literatura (presentada en el Capítulo 2) se obtuvo que los desarrolladores de AGSD prefieren compartir AK a través de UTEM, los cuales dejan rastros de las comunicaciones establecidas entre los involucrados, por lo que estas trazas pueden contener AK articulado, es decir, que fue convertido de tácito a explícito al interactuar en el UTEM. Con el fin de aprovechar este AK articulado, se realizó un estudio en empresas practicantes del AGSD para lograr un entendimiento sobre la manera de compartir AK en UTEM (presentado en el Capítulo 3), de donde se obtuvo principalmente una descripción del AK compartido en UTEM, así como que los desarrolladores consideran importante poder acceder de manera eficiente al AK que se comparte en estos medios.

A partir de este resultado y del obtenido en la revisión de literatura surgió la idea del concepto de condensación de AK (y los elementos que lo conforman), en analogía al concepto de vaporización (presentado en el Capítulo 4). Además, se determinó cómo afectaría este nuevo concepto al modelo SECI, los beneficios potenciales que se esperarían en ambientes de AGSD, y la relación de estos beneficios con métricas de desarrollo de software y con la efectividad de equipo.

Una vez definido el concepto de condensación de AK y sus elementos, se inició un proceso iterativo de diseño, implementación y evaluación de prototipos de software que instancian este concepto. Se inició con un prototipo no funcional (presentado en el Capítulo 5), hasta llegar a una versión funcional del mismo (presentada en el Capítulo 7), haciendo hincapié en el mecanismo de clasificación de AK (presentado en el Capítulo 6), el cual es

particularmente importante para el éxito de cualquier instancia de la condensación de AK. Todos los prototipos fueron evaluados por desarrolladores practicantes de AGSD, incluyendo estudiantes en la evaluación de la versión completa del prototipo (presentada en el Capítulo 8). Como resultado final de las evaluaciones de los diferentes prototipos se obtuvo evidencia de que permite determinar que la condensación de AK es viable para ambientes de AGSD, tanto por aspectos cuantitativos como por aspectos cualitativos.

En las siguientes secciones se presentan de manera más detallada las conclusiones de este trabajo y su relación con los objetivos planteados inicialmente. Además, se presentan las contribuciones obtenidas, así como las publicaciones que se derivaron de esta investigación. Finalmente, se presentan diversas oportunidades de trabajo futuro respecto a los resultados obtenidos en las diferentes fases de este trabajo.

9.5. Conclusiones

Las conclusiones de este trabajo son presentadas en tres partes: (1) conclusiones respecto al manejo de AK en AGSD, (2) conclusiones referentes a la condensación de AK, y (3) conclusiones respecto a las métricas afectadas por la implementación de la condensación de AK. Además, en la Figura 9.1 se esquematiza que objetivos fueron abordados en cada parte de la metodología, así como los capítulos en donde se desarrolló cada parte de la misma.

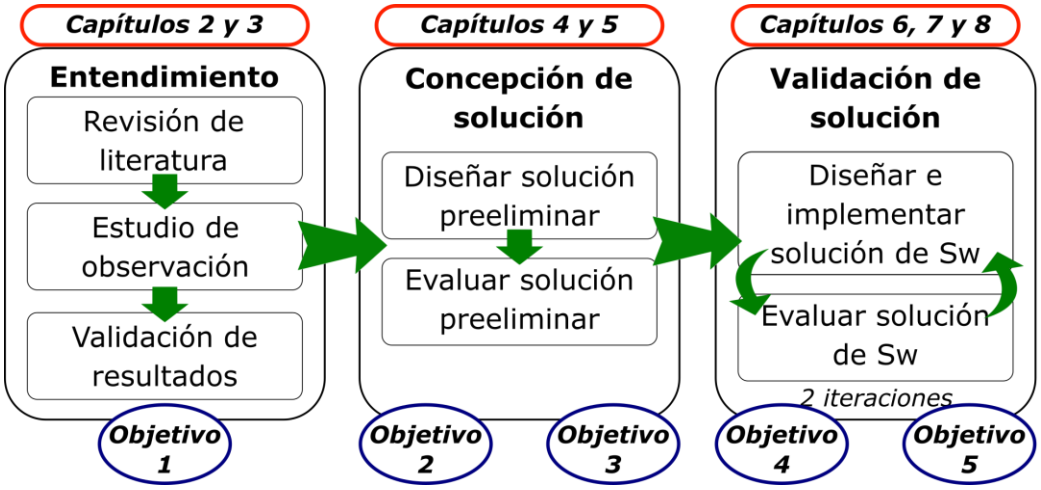


Figura 9.1. Relación de objetivos y capítulos con la metodología de investigación del trabajo de tesis.

9.5.1 Manejo del conocimiento arquitectónico en AGSD

Mediante el mapeo sistemático de literatura presentado en el Capítulo 2 (ver Figura 9.1) se identificaron tres formas en las que se maneja el conocimiento arquitectónico en AGSD: mediante artefactos de documentación, mediante métodos de comunicación, y mediante modificaciones metodológicas. Esta última manera consiste en incluir roles o fases arquitectónicas, y fue la más recurrente presumiblemente porque es la más fácil de implementar, pero sólo en empresas medianas o grandes (>100 empleados según el Glosario de TI de Gartner[45]), donde la infraestructura y presupuesto no es un gran impedimento para adoptar esta manera de manejar AK.

En cuanto a los artefactos de documentación sobresalen los artefactos con notaciones ligeras y los artefactos autogenerados, donde los primeros tienen el problema de la adopción ya que son notaciones poco comunes o poco populares en el ambiente ágil como UML. En el caso de los segundos (artefactos autogenerados) se tienen que realizar actividades adicionales al flujo común de desarrollo para que los artefactos se puedan generar automáticamente, o bien, se requiere refinar las técnicas de inteligencia artificial utilizadas para obtener artefactos con mejor precisión.

El manejo de AK mediante métodos de comunicación es predominantemente implementado con videoconferencias, sin embargo, cuando se tienen brechas grandes en el lenguaje se prefiere el uso de UTEM, sobre todo en el caso de los desarrolladores, que podrían considerarse como el perfil que más comparte AK en el AGSD.

Debido al potencial de encontrar AK articulado en las bitácoras de UTEM, se realizó un estudio para categorizar el AK compartido en estos medios en empresas de AGSD (presentado en el Capítulo 3, ver Figura 9.1). Se obtuvo que en las interacciones de UTEM se refieren y comparten artefactos como: documentos, componentes de software y código fuente, se toman y refieren decisiones arquitectónicas y se tocan temas principalmente relacionados con requerimientos y con codificación. Además, se obtuvo que el único AK documentado en las empresas participantes eran los requerimientos de software, por ello los desarrolladores consideraron importante el AK que se comparte mediante UTEM, ya que se tocan tópicos más allá de los requerimientos.

[45] https://www.gartner.com/it-glossary/smbs-small-and-midsize-businesses

Para cerrar esta parte, con lo anteriormente presentado se puede considerar cubierto el primer objetivo específico de esta tesis: *Caracterizar la forma de usar y de diseminar el conocimiento arquitectónico en el AGSD.*

9.5.2 Condensación de conocimiento arquitectónico

Al determinar la existencia e importancia del AK en las bitácoras de UTEM, además de la necesidad de los desarrolladores de AGSD de acceder de manera eficiente a dicho conocimiento, surgió la idea de ofrecer una manera de aprovechar el AK articulado en las bitácoras de UTEM, sin añadir demasiado esfuerzo para los involucrados manteniendo la agilidad del proceso. De esa manera surge el concepto de condensación de AK (presentado en el Capítulo 4, ver Figura 9.1), el cual se compone de tres elementos: (1) accesibilidad a las bitácoras de UTEM, (2) un mecanismo de clasificación de AK de las bitácoras de UTEM, y (3) un mecanismo de recuperación de AK que aproveche dicho mecanismo de clasificación. Con lo anterior se podría dar por cumplido el tercer objetivo específico de esta tesis: *Establecer los elementos que debe contemplar una solución para reducir la vaporización el conocimiento arquitectónico en el AGSD, sin perder las características ágiles y del GSD.*

Con los resultados de esta investigación se puede concluir que el elemento crucial de la condensación del AK es el mecanismo de clasificación, ya que es la base para el mecanismo de recuperación y es el elemento que podría introducir un esfuerzo considerable que disminuya la agilidad de un equipo de AGSD. Por esta razón se propuso el etiquetado social como un mecanismo que permita clasificar AK al momento de estar interactuando en UTEM, donde se usen etiquetas definidas por el equipo de trabajo y ligadas a un ancla semántica para que el significado de estas etiquetas no se vaporice, es decir, sería un etiquetado social semifijo. Esta ancla semántica está representada por alguna de las entidades del modelo de AK compartido en UTEM (meta-etiquetas) obtenido del estudio de observación presentado en el Capítulo 3. De esta manera, mediante el etiquetado social semifijo se tiene una manera viable y aceptada por los desarrolladores para clasificar AK compartido en UTEM en ambientes de AGSD (como lo mostraron los resultados de los capítulos 6 y 8).

Por otro lado, con la implementación del etiquetado social como mecanismo de clasificación de AK, se atiende uno de los problemas detectados en la revisión de literatura presentada en el Capítulo 2, es decir, el predominio de AK en sus estados tácito y explícito

documentado (sin estructura formal), lo cual lo hace propenso a vaporizarse. Se dice que este problema es atendido ya que se estructura AK articulado en UTEM (conocimiento explícito documentado) a través del etiquetado basado en el modelo de AK compartido en UTEM. Debido a que este modelo no es parte de un estándar, no se podría considerar que se está convirtiendo de conocimiento explícito documentado a explícito formalizado, sin embargo tampoco se queda totalmente sin estructura. Por este motivo, al analizar el impacto del concepto de condensación de AK en el modelo SECI, se identificaron dos sub-estados del estado explícito documentado (ver Capítulo 4 y Figura 9.1): no organizado y organizado, donde este último es donde se situaría el AK etiquetado.

A través de varias iteraciones el concepto de condensación de AK se logró implementar como un prototipo de software llamado ArchiKCo, tomando en cuenta los tres elementos que lo componen. La disponibilidad de las bitácoras de UTEM se implementó a través de un servicio que envía las conversaciones de Skype (el UTEM elegido para este prototipo) a un repositorio común. Como se mencionó anteriormente, el mecanismo de clasificación de AK se implementó usando el etiquetado social, pero además se implementó un componente de autocompletado para ayudar a los desarrolladores a recordar y encontrar las etiquetas adecuadas durante la interacción en Skype, de tal manera que las interacciones se guarden ya etiquetadas en el repositorio. Además, se implementó un sistema de administración de etiquetas en Web, para registrar las etiquetas de usuario y relacionarlas con sus respectivas meta-etiquetas. En cuanto al mecanismo de recuperación de AK, éste fue implementado mediante un buscador Web que consulta el repositorio de interacciones y ofrece distintos parámetros de búsqueda, como: período de fechas, autor, destinatario, texto libre y las mismas etiquetas y meta-etiquetas, que generalmente no ofrecen los UTEM. Con esto se puede considerar cumplido el cuarto objetivo específico de esta tesis (ver Figura 9.1): *Diseñar e implementar una solución de software que permita la reducción de la vaporización del conocimiento arquitectónico en el AGSD, sin perder las características ágiles y del GSD*.

El prototipo ArchiKCo fue evaluado por estudiantes y desarrolladores practicantes de AGSD, obteniendo una preferencia por buscar AK usando ArchiKCo en lugar de usar los medios ofrecidos por UTEM como Trello y Skype, sin perder precisión en los resultados de la búsqueda, ya que no hubo diferencia significativa en ello usando ArchiKCo o usando alguno de los UTEM. Cabe mencionar que la preferencia del buscador de ArchiKCo estuvo

influenciada por los parámetros de búsqueda de AK, así como por la integración de fuentes, es decir, bitácoras de UTEM, ya que es común que los desarrolladores de AGSD no recuerden en que UTEM fue compartido el AK que necesitan en un momento determinado. El buscador de ArchiKCo también fue evaluado respecto a la rapidez en encontrar AK, obteniendo que ArchiKCo obtuvo menor tiempo cuando no se sabía de antemano en que medio buscar, lo cual también influyó en la preferencia de los participantes. Los participantes percibieron al buscador como una herramienta usable y útil para encontrar AK durante un ciclo de desarrollo, con la cual se podrían reducir las interrupciones para preguntar sobre AK, y consecuentemente se podrían terminar más rápido las tareas de desarrollo.

Además de centrarse en el desempeño al buscar AK, durante la evaluación se observó el comportamiento de etiquetado, obteniendo que los participantes usaron mayormente etiquetas válidas (existentes dentro de una lista predefinida) de manera semánticamente correcta y con baja tasa de error (al teclear la etiqueta) durante la interacción en UTEM que sostuvieron como parte de la evaluación. Esto significa que el asistente de etiquetado fue útil para encontrar etiquetas y evitar errores que pudieran afectar la clasificación de AK durante las interacciones en UTEM. Además, significa que las etiquetas pueden llegar a ser muy expresivas, dado que los participantes no las conocían de antemano y aun así las usaron correctamente la mayoría de las veces. También, el mecanismo de etiquetado tuvo una alta percepción de usabilidad y una baja percepción de interferencia en el trabajo diario.

Por los resultados de la evaluación se concluyó que el concepto de condensación de AK es viable para el ambiente de AGSD, porque se puede acceder al AK articulado en UTEM de una manera fácil, rápida y con baja obstrucción al trabajo. Hay que recordar que los desarrolladores de AGSD tienen la necesidad de acceder a dicho conocimiento, como una manera de solventar la deuda de documentación propia de un ambiente ágil. Sin embargo, con base en los resultados del mapeo sistemático del Capítulo 2, las empresas practicantes de AGSD con más probabilidad de presentar deuda de documentación serían aquellas que no hayan integrado un rol de arquitecto o una fase arquitectónica, es decir, empresas pequeñas o medianas que por cuestiones de estructura y presupuesto no puedan hacer esa modificación metodológica. Por ello también se concluye que este tipo de empresas serían las que podrían beneficiarse más al implementar la condensación de AK.

Otro aspecto a resaltar sobre la condensación de AK es que la cantidad y tipo de AK que se condense dependería mucho de la distribución del trabajo entre equipos locales y remotos que tenga la empresa o proyecto donde se implemente. Como se evidenció en el estudio presentado en el Capítulo 3 (Figura 9.1), el AK sobre aspectos de codificación y diseño predomina cuando existe un equipo virtual de desarrolladores que comparten AK entre las distintas locaciones; o bien, cuando de un lado están los dueños del producto y de otro está sólo el equipo de desarrollo, el tipo de AK que predomina es sobre requerimientos o cuestiones de arquitectónicas de alto nivel. Esto quiere decir que la distribución del trabajo sería un factor para evaluar el impacto de implementar la condensación de AK en alguna empresa de AGSD.

9.5.3 Efectos de la condensación de conocimiento arquitectónico

Los efectos que se esperan al implementar la condensación de AK en AGSD se pueden clasificar en efectos directos e indirectos. Los efectos directos fueron determinados a partir de la evidencia obtenida con el mapeo sistemático (presentado en el Capítulo 2) y con el estudio de observación en empresas de AGSD (presentado en el Capítulo 3), mediante la cual se logró hacer una lista de métricas que potencialmente se verían afectadas. A continuación, se presentan dichas métricas y sus afectaciones una vez evaluado el prototipo ArchiKCo.

9.5.3.1 Tiempo para encontrar conocimiento arquitectónico

En la evaluación del prototipo ArchiKCo los participantes fueron más rápidos encontrando AK mediante su buscador, particularmente cuando no sabían de antemano el UTEM donde se encontraba el conocimiento, pero sólo significativamente contra uno de los UTEM utilizados (Trello). Se proyecta que en un ambiente real donde es muy frecuente que no se recuerde en qué UTEM está el AK que se requiere para cierta situación, siempre se encontrará más rápido el conocimiento deseado usando el buscador de ArchiKCo, en comparación de hacerlo con los medios que ofrece un UTEM. Esto debido a la integración de bitácoras de UTEM, a los parámetros de búsqueda disponibles y la buena percepción de usabilidad resultante de la evaluación. Por lo tanto, el tiempo para encontrar AK se disminuiría en general implementando la condensación de AK.

9.5.3.2 Cantidad de interrupciones

Al contar con un medio eficiente para encontrar AK, es coherente pensar que el número de interrupciones entre los compañeros de trabajo se reduzca, ya que primero se preferiría buscar en ArchiKCo antes de consultar a un compañero de trabajo. Esto se concluye debido a dos factores, el primero es que en el estudio realizado en empresas de AGSD, se observó que los desarrolladores prefieren buscar por su cuenta el AK necesario para completar sus tareas antes que consultar a un compañero remoto, con el fin de no interrumpirlo y causar alguna molestia. El segundo factor es que en la evaluación del ArchiKCo los participantes consideraron casi de manera unánime que usando el buscador del prototipo se podrían disminuir las interrupciones entre compañeros, además se percibió que se encontraría AK de manera fácil y oportuna, lo cual fue confirmado por los resultados cuantitativos de la misma evaluación.

9.5.3.3 Tiempo en concluir tareas de desarrollo de software

Consecuentemente, al disminuir el tiempo en encontrar AK y el número de interrupciones es natural pensar que el tiempo para concluir tareas de desarrollo de software también se reduzca. De hecho, en el estudio de observación en empresas de AGSD, algunos participantes comentaron que han tenido retrasos causados por no contar con el AK requerido para cierta tarea, ya sea porque pierden mucho tiempo buscando el AK o buscando a la persona con el conocimiento, o bien, porque esta persona con conocimiento no se encuentra o no tiene tiempo para atenderlos. Es por ello también que otros esfuerzos para mejorar el manejo de AK en AGSD consisten en formar un directorio de expertos en ciertas áreas, para no perder tiempo buscando a la persona adecuada (Gilberto Borrego et al., 2017).

Referente a los aspectos indirectos que se ven afectados por la implementación de la condensación del AK está la efectividad de equipo y la calidad del producto. A continuación, se detalla sobre ello.

9.5.3.4 Efectividad de equipo

En el Capítulo 4 se estableció una relación entre las métricas presentadas anteriormente y la efectividad de equipo, que según Hackman (Hackman, 1987) se define con tres elementos: resultados, procesos sociales y aprendizaje. Es evidente que los resultados se obtendrían más rápido implementando la condensación de conocimiento, dado la reducción del tiempo en

terminar tareas. Los procesos sociales se verían beneficiados también, ya que al reducirse las interrupciones se reduciría una de las posibles fuentes de disgustos en la persona que tienen el AK, o bien, podría reducirse el sentimiento de vergüenza ocasionado por estar solicitando constantemente AK. Referente al aprendizaje, gracias a la integración de bitácoras de UTEM de todos los involucrados en los proyectos de software, se tendrían disponibles soluciones, técnicas, sugerencias, etc. referentes a situaciones arquitectónicas que beneficiarían al aprendizaje conjunto de todos los integrantes de un proyecto o empresa.

9.5.3.5 Calidad del producto

Uno de los mayores problemas de la vaporización del AK es que el producto de software no evolucione como inicialmente se proyectó al diseñar las reglas y lineamientos arquitectónicos. Al vaporizarse dichas reglas y lineamientos es probable que se incurra en errores grandes que afecten el funcionamiento del producto y por tanto su calidad. Por ejemplo, supongamos que para una aplicación se tenía proyectado usar cierto patrón arquitectónico que le facilitaría el manejo de peticiones concurrentes; al pasar del tiempo este lineamiento se vaporiza y tal vez en una etapa de mantenimiento se decide cambiar de patrón para facilitar otra funcionalidad; al desplegar la nueva versión de la aplicación se vería rápidamente una saturación de peticiones lo cual afectaría negativamente la calidad del servicio y por consiguiente la percepción del usuario. Por lo tanto, la implementación de la condensación de AK en AGSD, podría evitar este tipo de problemas de calidad en un futuro, debido a que el conocimiento no estaría totalmente vaporizado, y habría una manera de darle seguimiento a las razones de decisiones arquitectónicas.

Con base en lo anterior se pueden considerar como cumplidos los objetivos específicos 2 y 5: (2) *Determinar las métricas de desarrollo de software y los aspectos relacionados con la efectividad de equipo que son afectados al implementar una solución que reduzca la vaporización del conocimiento arquitectónico en AGSD*; y (5) *Determinar el impacto en las métricas y la efectividad del equipo al implementar la solución de software para reducir la vaporización del conocimiento arquitectónico en el AGSD.*

9.6. Aportaciones

La principal aportación de esta tesis es el concepto de condensación de AK, así como la determinación de viabilidad del mismo a través de la evaluación de un prototipo (ArchiKCo)

que implementa dicho concepto. La condensación de AK representa una manera viable para solventar la deuda de documentación (sin ser una manera que sustituya a la documentación), con la que a su vez se puede reducir la vaporización del AK en ambientes de AGSD, lo cual es uno de los retos establecidos en la literatura al practicar este paradigma de desarrollo de software (Dorairaj et al., 2012; Jiménez et al., 2009; Yanzer Cabral et al., 2014a). Una de las principales razones por la que se considera que la condensación de AK es una aportación, es que hasta el momento de iniciar este trabajo de tesis no se había planteado en la literatura el aprovechamiento del AK articulado en UTEM en ambientes de AGSD para solventar la deuda de documentación y consecuentemente disminuir la vaporización de AK.

Por otro lado, al introducir el concepto de condensación de AK y dada su estrecha relación con el concepto de vaporización del conocimiento, se hizo un trabajo de definición de este último basado en el modelo SECI (que establece la transformación de conocimiento tácito a explícito y viceversa), ya que no existe un consenso en la literatura al respecto (ver Capítulo 4). De tal manera, la vaporización del conocimiento se definió como: *la transición hacia un estado intermedio (vaporizado) entre el conocimiento tácito o explícito y la pérdida total del conocimiento, teniendo aun la posibilidad de regresar al estado tácito del conocimiento a través de la recuperación y al explícito a través de la condensación*. Asimismo, el concepto de condensación en términos del modelo SECI se definió como: *una manera de preservar la espiral de creación del conocimiento, reduciendo la posibilidad de pérdida de conocimiento explícito documentado en su forma no organizada, ya sea organizando el conocimiento antes de que se vaporice o incluso recuperando el conocimiento vaporizado antes de que se pierda de manera definitiva*. Además, una aportación consecuente de este trabajo de definición fue la introducción de dos nuevos sub-estados al estado explícito documentado del conocimiento (organizado y no organizado), así como un nuevo estado llamado vaporizado, como parte del modelo SECI, es decir, se realizó una extensión del mismo.

Otra aportación asociada a la condensación del AK es el diseño del prototipo que implementa este concepto, de donde se podría partir para implementar instancias que puedan adoptarse en la industria. Por otro lado, como parte de la definición de la condensación del AK, se dejaron las bases para realizar otros diseños que implementen este concepto.

Finalmente, una aportación previa a la definición de la condensación de AK son las tres grandes áreas en las que se agrupan los enfoques para el manejo de AK en AGSD que reporta la literatura: (1) modificaciones metodológicas, (2) artefactos de documentación y (3) métodos de comunicación. De este último, surge la idea de aprovechar los rastros de UTEM para recuperar AK articulado, y consecuentemente surge el modelo que representa los aspectos de comunicación y propios del AK que intervienen al compartir este conocimiento en UTEM en ambientes de AGSD, lo cual se puede considerar como otra aportación.

9.6.1 Publicaciones

Un resultado importante de este trabajo de investigación, lo constituyen las publicaciones realizadas. A continuación, se enlistan las referencias a los trabajos publicados directamente relacionados con este trabajo de tesis.

Artículo de revista en JCR:

- **G. Borrego**, A. L. Morán, R. R. Palacio, O. M. Rodríguez-Elias, and E. García-Canseco, "Review of approaches to manage architectural knowledge in Agile Global Software Development," *IET Softw.*, vol. 11, no. 3, pp. 77–88, 2017.

Conferencias arbitradas:

- **G. Borrego**, A. L. Moran, R. Palacio, and O. M. Rodriguez, "Understanding architectural knowledge sharing in AGSD teams: An empirical study," in *Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016*, 2016, pp. 109–118.
- **G. Borrego**, "Condensing architectural knowledge from unstructured textual media in agile GSD teams," in *2016 IEEE 11th International Conference on Global Software Engineering Workshops*, 2016, pp. 69–72.
- **G. Borrego**, A. L. Moran, R. Palacio, and O. M. Rodriguez, "Findings on AGSD architectural knowledge sharing," in *Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016*, 2016, pp. 193–194.
- **G. Borrego**, A. L. Morán, and R. Palacio, "Preliminary Evaluation of a Tag-Based Knowledge Condensation Tool in Agile and Distributed Teams," in *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 2017, pp. 51–55.

Además, hubo oportunidad de participar en publicaciones también relacionadas con el trabajo de tesis, pero que se derivan de seguir distintos opciones de investigación producto de este trabajo.

Artículo de revista de divulgación:

- S. González-López, **G. Borrego**, A. López-López, A. L. Morán, "Clasificando conocimiento arquitectónico a través de técnicas de minería de texto", *Komputer Sapiens*, vol. 1, no. 10, 2018 (**aceptado en 2017, aun sin publicarse**)

Conferencias arbitradas:

- L. T. Portela and **G. Borrego**, "Scrumconix: Agile and documented method to AGSD," in Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016, 2016, pp. 195–196.

9.7. Trabajo futuro

Como parte del trabajo de tesis surgieron algunas ideas que podrían haber mejorado algunos aspectos, o que dirigían el trabajo hacia otros enfoques interesantes referentes a la problemática abordada. Sin embargo, se decidió no contemplar estas ideas debido al alcance propuesto para esta tesis. Por consiguiente, estas ideas delinean el trabajo futuro de esta tesis y se presentan a continuación.

- **Privacidad de las interacciones de UTEM**. Generalmente las empresas de software determinan ciertos medios de comunicación como los oficiales para tratar asuntos del trabajo, donde generalmente están los UTEM. Por ello se espera que en las bitácoras de estos medios sólo exista información referente al trabajo, sin embargo, la misma socialización de un equipo propicia ciertas conversaciones personales en los UTEM. Por lo tanto se requiere explorar opciones para atender la privacidad al condensar el AK en estos medios. Preliminarmente se piensa en un cierto tipo de etiquetas que marquen aspectos que no se deben condensar, pero haría falta más análisis y evaluaciones para determinar la mejor opción.
- **Condensación de AK articulado en audio y video**. Este trabajo de tesis se enfoca en el AK articulado en UTEM, sin embargo, como parte de los hallazgos de este trabajo, se obtuvo que mientras no haya una brecha considerable de lenguaje entre los equipos locales y remotos, las videoconferencias y las llamadas telefónicas son preferidas para

aclarar temas arquitectónicos, por lo tanto también se articula AK en estos medios. Aprovechando el marco conceptual desarrollado para definir la condensación de AK, podría explorarse la utilización de audios y videos como fuente de conocimiento e integrarlo todo como una solución completa.

- **Evaluación en sitio y largo plazo de la condensación de AK**. La evaluación realizada para determinar la viabilidad de la condensación de AK fue en un ambiente controlado utilizando escenarios predefinidos. El siguiente paso sería conducir un estudio en una o varias empresas de AGSD, en proyectos reales, durante algunos meses, de tal manera que se abarquen varios ciclos ágiles de desarrollo, para confirmar la viabilidad del concepto, refinarlo, confirmar los efectos de su implementación y determinar los aspectos primordiales para su adopción.
- **Integración de inteligencia artificial al mecanismo de clasificación de AK**. Una de las sugerencias más recurrentes sobre el mecanismo de clasificación, fue contar con una manera más inteligente de sugerir etiquetas durante las interacciones de UTEM. Para ello se propone la inclusión de técnicas de inteligencia artificial que puedan sugerir etiquetas dependiendo del contexto de conversación y/o del proyecto, o incluso que pueda hacer un etiquetado automático. Como se observó en las publicaciones asociadas a esta tesis, ya se inició a trabajar en ello, pero aún falta la integración de estas técnicas al componente de autocompletado y conducir la evaluación correspondiente.
- **Diferentes implementaciones de la condensación de AK**. Como se mencionó anteriormente, en esta tesis se presenta sólo una manera de llegar a la implementación del concepto de condensación de AK. En el Capítulo 4 se esbozan diferentes maneras de implementar cada uno de los elementos de este concepto, las cuales valdría la pena explorar, para luego compararlas con las implementaciones realizadas en esta tesis, y así determinar cuáles serían las mejores opciones dependiendo de las condiciones de cada ambiente en el que se vaya a implementar.
- **Integración del modelo de manejo de conocimiento arquitectónico en AGSD con otros modelos**. En la literatura existen otros modelos que describen distintos aspectos del manejo del AK en GSD o en ágil, como el modelo de (Nour Ali et al., 2010), con los cuales se podría hacer una integración con la intención de formar un modelo unificado, o bien, crear extensiones entre los distintos modelos.

- **Integración de la condensación de AK con otros enfoques para el manejo de AK en AGSD**. En el Capítulo 2 se delineó como sería una solución completa, abarcando las tres áreas identificadas para el manejo del AK en AGSD y contemplando las tres fases del ciclo integrado del manejo de conocimiento, lo cual sería de la siguiente manera: (1) Agregar un rol o una fase para abordar los problemas arquitectónicos; (2) Usar artefactos de documentación ligeros para codificar AK; (3) Aprovechar el AK articulado en UTEM; y (4) Diseminar artefactos autogenerados y/o ligeros a través de wikis, repositorios o herramientas de groupware, con opciones de búsqueda adaptadas a las necesidades del AK. De esta proyección de solución, sólo se abordó en esta tesis el tercer punto, por lo que sería de gran interés explorar el resto para conformar una solución más completa para atender la problemática del manejo de AK en AGSD.
- **Condensación de AK en ambientes ágiles no distribuidos**. En el estudio realizado en empresas de AGSD se observó que se comparte AK en UTEM entre miembros locales de equipo distribuido, y aunado a la poca o nula documentación de AK que se manejan en estos ambientes, es muy probable que el AK articulado en estos medios sea la única fuente de conocimiento disponible. Por lo tanto, valdría la pena evaluar la viabilidad del concepto de condensación de AK en equipos ágiles co-localizados.
- **Condensación de conocimiento en otros ámbitos**. Este trabajo de tesis está totalmente enfocado al área de ingeniería de software, por lo que el concepto de condensación se refiere particularmente al AK. Sin embargo, pudieran existir otros ámbitos en donde también se articule conocimiento valioso en UTEM y donde exista la necesidad de medios eficientes para su recuperación. Por ello, una de las líneas futuras de investigación podría explorar la aplicabilidad y viabilidad del concepto de condensación de conocimiento en otros ámbitos laborales o simplemente en la vida cotidiana.

Apéndice A.
Especificación técnica del prototipo Chatagger

A.1. Arquitectura del prototipo Chatagger.

Se definió una arquitectura sencilla de cliente/servidor (ver Figura A.1), donde el asistente de etiquetado y la funcionalidad de usuario del mensajero instantáneo se desarrollaron en Javascript[46]. La parte de servidor de este último se desarrolló en Node.js[47], con una base de datos no relacional implementada en MongoDB[48], para guardar las conversaciones para su posterior análisis.

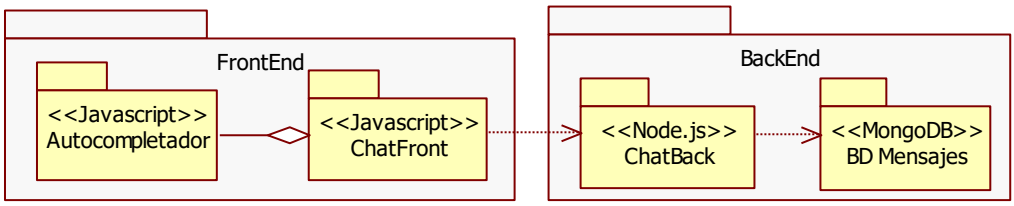


Figura A.1. Arquitectura de Chatagger.

A.2. Diagrama de casos de uso del prototipo Chatagger

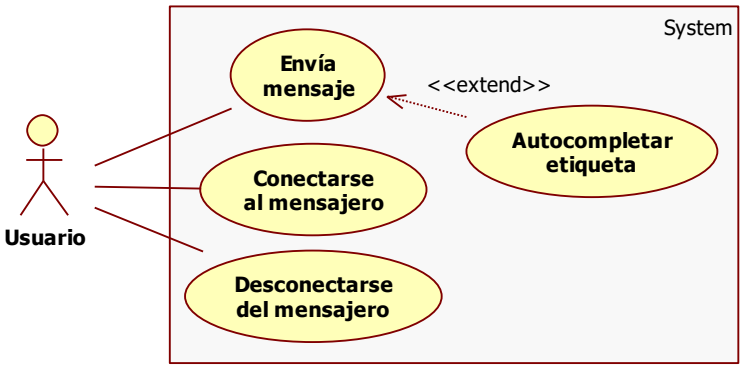


Figura A.2. Diagrama de casos de uso de Chatagger.

[46] https://www.javascript.com/
[47] https://nodejs.org/
[48] https://www.mongodb.com/

A.3. Diagrama de secuencia del funcionamiento general de Chatagger

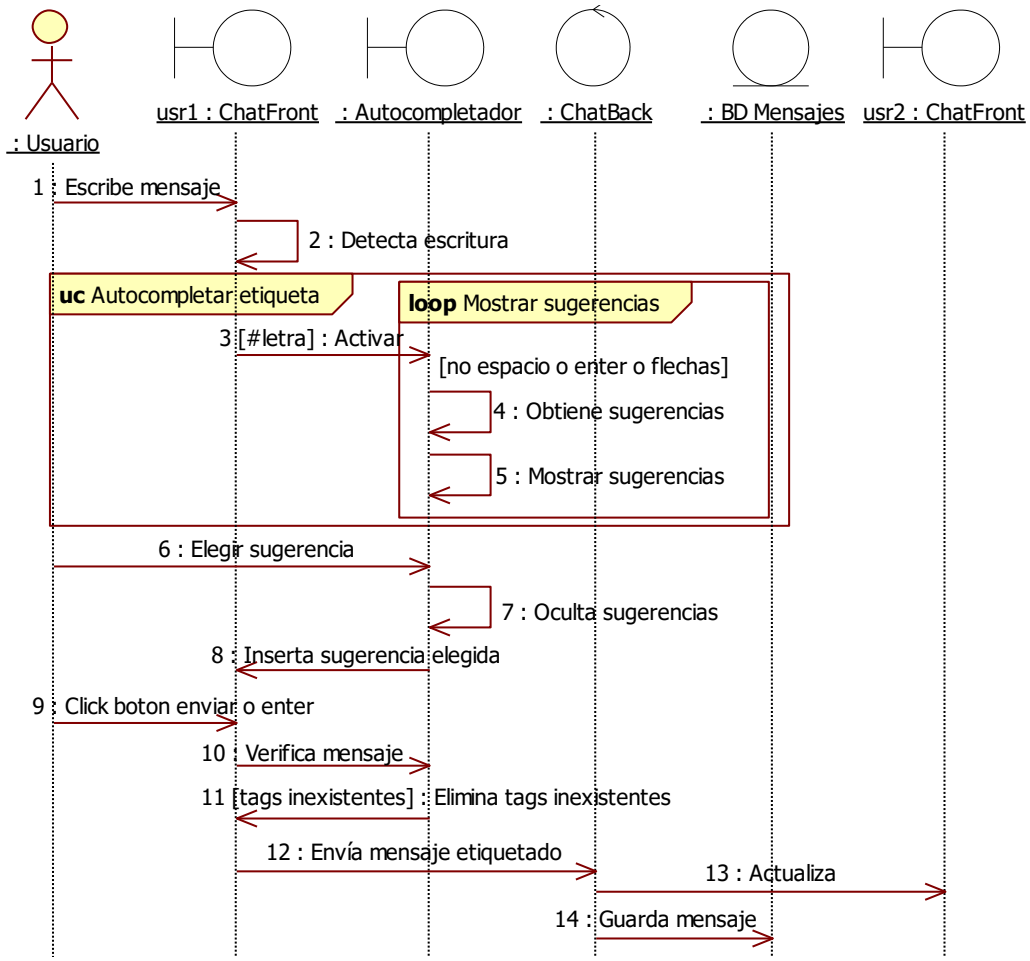


Figura A.3. Diagrama de secuencia del funcionamiento general Chattager.

Apéndice B.
Implementación de los sub-sistemas del prototipo ArchiKCo

B.1. Servicio recopilador de interacciones

A continuación, se describe la funcionalidad de cada uno de los componentes que forman el sub-sistema de recopilación de interacciones en UTEM, el cual se representa en la Figura B.1.

- **Componente servicio recopilador**. Representa componente principal del servicio, le cual verifica la existencia de mensajes nuevos por enviar al repositorio, cada cierto tiempo según lo que indique el archivo servicio.config.
- **Artefacto servicio.config**. Se trata de un archivo de configuración en formato XML, donde se establece la frecuencia con la que se verifica la existencia de nuevos mensajes, se establece los contactos de quien se extraerán los mensajes, se define la ubicación de la base de datos de Skype (artefacto main.db), la URL de la interface de Algolia, los datos de autenticación a este último y el nombre del repositorio donde se enviarán los mensajes.
- **Componente Lógica Skype**. Representa el componente que ejecuta las operaciones de extracción de Skype y ejecuta el envío de los mensajes al repositorio.
- **Componente Entidades Skype**. Define las entidades necesarias para capturar los mensajes de Skype.
- **Componente Extractor Skype**. Tiene la responsabilidad de ejecutar las consultas hacia la base de datos Skype para extraer los mensajes que se requieran.
- **Componente Interface Algolia**. Tiene la responsabilidad de hacer la conexión con la plataforma de Algolia y de ejecutar las operaciones de envío de mensajes de UTEM hacia dicha plataforma.

B.2. Asistente de etiquetado

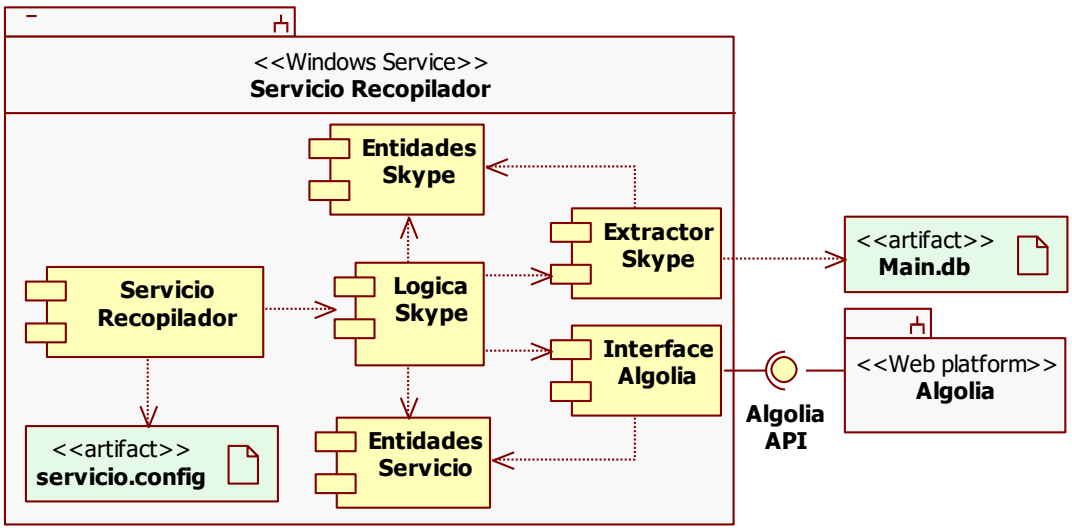


Figura B.1. Diagrama de componentes del servicio recopilador de interacciones.

A continuación, se describe la funcionalidad de cada uno de los componentes que forman el sub-sistema de asistente de etiquetado para interacciones en Skype, el cual se representa en la Figura B.2.

- **Componente ayudante etiquetado**. Representa el componente que se ejecuta para iniciar todo el asistente de etiquetado. Tiene la responsabilidad de iniciar el censado de las acciones del teclado y a su vez de determinar si se debe iniciar el proceso de autocompletado si la ventana activa es de uno de los UTEM definidos para ser asistidos. Esta definición de UTEM se toma del archivo ayudante.config.
- **Artefacto ayudante.config**. Se trata de un archivo de configuración en formato XML, donde se establece los nombres de los procesos de Windows y las ventanas de los UTEM que serán asistidos con el autocompletado. Además, se define la URL a consultar para obtener las etiquetas registradas en el administrador de etiquetas.
- **Componente Utilidades WinAPI**. Tiene la responsabilidad de hacer la interface entre el asistente de etiquetado y los métodos del sistema operativo Windows, que permiten censar la actividad del teclado, determinar ventanas activas y manejar el texto de dichas ventanas para concretar el autocompletado.
- **Componente Controladores de Autocompletado**. Se trata de un conjunto de controladores encargados de administrar las acciones de autocompletado y de censado de acciones del teclado.
- **Componente de Dominio**. Contiene las entidades con las que se manejaran los datos de que se obtienen del administrador de etiquetas.
- **Componente Controladores de etiquetas**. Se trata de un conjunto de controladores encargados de administrar las acciones referentes a la obtención de etiquetas del sub-sistema de administración de etiquetas.
- **Componente Conector REST**. Administra la conexión y consultas hacia el sub-sistema de administración de etiquetas, lo cual se lleva a cabo a través de un servicio tipo REST que éste último expone.

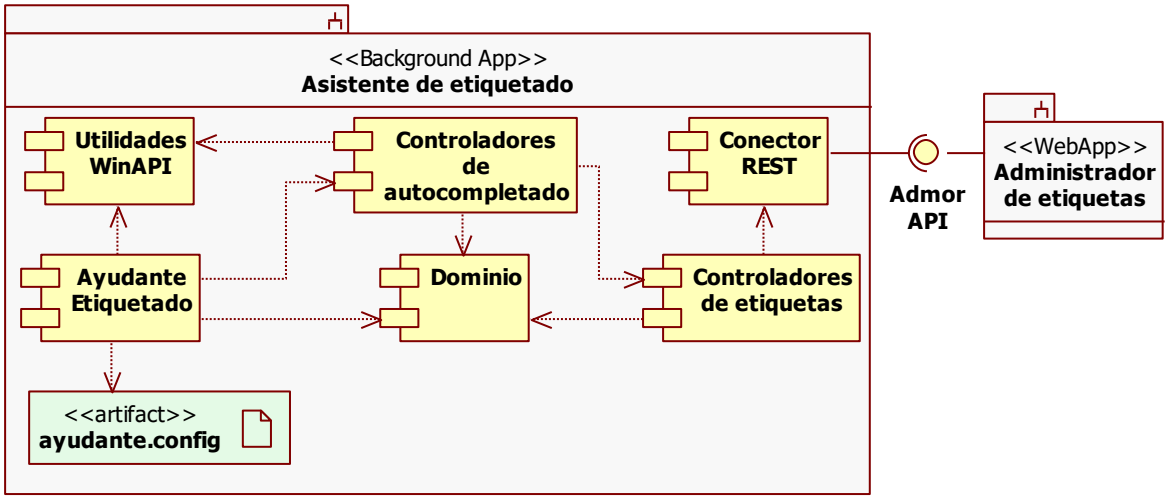


Figura B.2. Diagrama de componentes del sub-sistema asistente de etiquetado

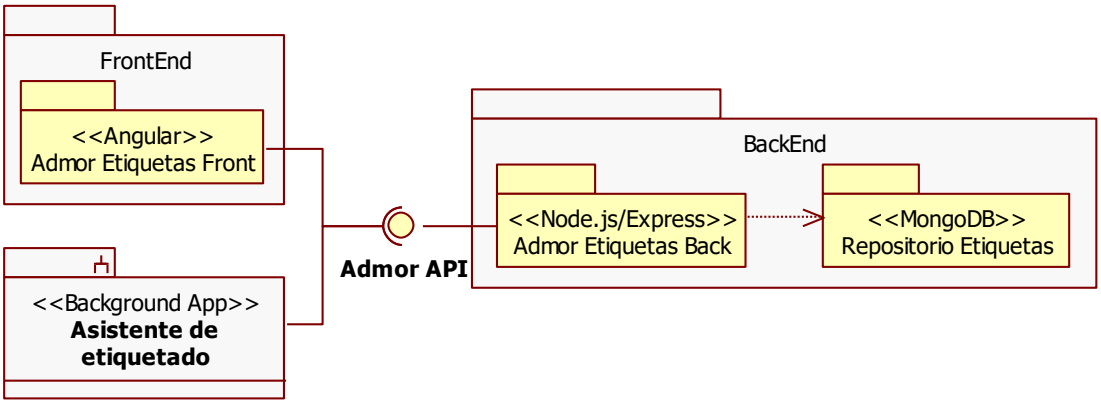


Figura B.3. Arquitectura del sub-sistema de administración de etiquetas.

B.3. Administración Web de etiquetas

Respecto al sub-sistema de administración de etiquetas, fue desarrollado como una aplicación Web usando la pila de herramientas de desarrollo llamada MEAN.JS[49] (MongoDB, Express, Angular, Node.js). Se utilizó una arquitectura cliente/servidor (ver Figura C.3), que es común para las aplicaciones basadas en MEAN, donde la parte de cliente está desarrollada con Angular y es responsable de manejar la interfaz de usuario; y la parte de servidor está desarrollada con Node.js, la biblioteca Express para manejar la lógica de negocio y la interacción con el repositorio de datos implementado con MongoDB. Cabe destacar en la Figura B. la presencia de la interface Admor API, la cual representa un servicio Web tipo REST mediante el cual se comunica la parte de cliente con la parte servidor, y además es la vía por la cual el sub-sistema de asistente de etiqueta consulta las etiquetas y meta-etiquetas disponibles.

B.4. Buscador de conocimiento

Este sub-sistema fue desarrollado como una aplicación Web en Angular y Node.js, usando el estilo arquitectónico de cliente/servidor. Como se puede apreciar en la Figura C., el buscador de conocimiento consulta al repositorio de mensajes implementado en Algolia, y al repositorio de etiquetas (parte del componente de administración de etiquetas), este último con fin de crear filtros basados en etiquetas. Las consultas a ambos sub-sistemas (repositorio de mensajes y repositorio de etiquetas) se realizan a través de servicios Web tipo REST, que estos dos sub-sistemas exponen.

[49] http://meanjs.org/

Apéndice C.
Material para evaluación de mecanismo de etiquetado

C.1. Planteamiento inicial de escenario

Supongan que ambos se encuentran laborando en dos distintas empresas de desarrollo de software, donde una le brinda servicios a la otra. Ambas empresas se encuentran en países distintos y trabajan bajo el paradigma ágil, es decir, aplican metodologías como Scrum o programación extrema. La empresa que llamaremos A, acaba de iniciar un proyecto con la empresa B, en donde ésta última tiene que consumir un servicio web tipo REST, el cual está apenas está siendo desarrollado. Por lo tanto no se tiene documentación del servicio.

C.1.1. Planteamiento de escenario cliente

A usted se le asignó desarrollar 3 historias de usuario, todas relacionadas con accesos al servicio REST. Entonces, usted debe empaparse técnicamente de las cuestiones relacionadas con el servicio, así como de las reglas de negocio que se encuentran embebidas en él. Particularmente usted debe de resolver las siguientes interrogantes:

1. ¿Cómo se debe autenticar un cliente al servicio REST?
2. ¿Cómo responde el servicio cuando hay un error?
3. ¿Cuáles son las URL que debo usar, así como los parámetros y comandos (PUT, POST, GET, DELETE) para las 3 historias de usuario?

C.1.2. Planteamiento de escenario servidor

Usted es el punto principal de contacto para saber detalles técnicos del servicio REST, ya que usted lo desarrolló. Recibirá preguntas del programador que desarrollará la parte de cliente, pero también usted hará las siguientes preguntas para que el servicio éste en las mejores condiciones posibles.

- ¿El cliente que se va a desarrollar va a residir en el mismo dominio del servicio?
- ¿Los campos para la historia de usuario 2 están completos?

C.2. Guion para cliente

1. Saludar
2. Presentarse. Ejemplo: Soy el programador que va interactuar con el servicio REST, tengo entendido que tú eres el encargado de ese servicio.
3. Recibirás confirmación de que él es el encargado del servicio.
4. Hacer pregunta 1: ¿Cómo me debo autenticar al servicio REST? (***usar etiqueta***)
5. Recibirás respuesta.
6. Preguntar: Me imagino que el token de autenticación va ir en el encabezado, ¿con que nombre de campo iría?
7. Recibirás respuesta
8. Otra pregunta: ¿Cómo responde el servicio cuando hay un error? (***usar etiqueta***)
9. Recibirás respuesta
10. Por último, necesito saber a qué URLs debo consultar para hacer las historias de usuario 1, 2 y 3. (***usar etiqueta***)
11. Recibirás respuesta
12. Gracias, por último, en la URL de la historia 2, ya vez que es de actualizar, si se usar el comando PUT?

13. Recibirás respuesta.
14. Gracias, cualquier otra duda de digo.
15. Recibirás una pregunta tu contraparte, a lo que contestas que el cliente no estará en el mismo dominio.
16. Recibirás otra pregunta de tu contraparte, a lo que contestarás que al parecer si son suficientes los campos, pero que en el transcurso del desarrollo le comentarás si se necesitará algo más.

C.3. Guion para servidor

1. El cliente te saludará y se presentará contigo.
2. Te preguntará si eres el encargado del servicio, a lo que contestarás que sí.
3. Te preguntará sobre el método de autenticación. Contestarás lo siguiente: primero tienes que enviar el usuario y contraseña que te pasamos a la siguiente URL 109.212.5.2/servicio/autentica, y te responderá un token.
4. Te hará otra pregunta a lo que contestas que el token va en el encabezado de cada petición en un campo que se llame authToken.
5. Recibirás una pregunta al respecto del manejo de errores del servicio. Responderás que el servicio usa los errores estándar de HTTP y que el mensaje de error viene en el cuerpo de la respuesta.
6. Recibirás otra pregunta sobre las URLs para las historias de usuario. Responderás: Para la historia 1, usarás 109.212.5.2/servicio/usuario?busqueda=[texto], para la 2: 109.212.5.2/servicio/usuario/[id] y para la 3 109.212.5.2/servicio/roles
7. Recibirás una pregunta acerca de la URL de la historia 2. Respondes: Si, si estamos respetando tal cual el estándar. Oye y a provechando que preguntas sobre las historias de usuario, para la búsqueda debes asegurarte que no te manden vacío el campo, porque si no el servicio te responde con un error. (***usar etiqueta***)
8. Tu contraparte te agradecerá tus respuestas, y a continuación preguntas lo siguiente: Oye, el cliente que están desarrollando va a estar corriendo en el mismo dominio que el servicio?
9. Te responderá que no. A lo que tu dices: Ah, entonces activo el CORS en el servidor. (***usar etiqueta***)
10. Una última pregunta: los campos que regresa la URL de la historia 1 son suficientes para desarrollarla? es que en eso me quedó duda cuando lo hice. (***usar etiqueta***)
11. Recibirás respuesta.
12. Te despides: Muy bien, entonces seguimos en contacto.

Apéndice D.
Material para la evaluación de prototipo funcional del concepto de condensación de AK.

D.1. Planteamiento inicial de escenario

Supongan que ambos se encuentran laborando en dos distintas empresas de desarrollo de software, donde una le brinda servicios a la otra. Ambas empresas se encuentran en países distintos y trabajan bajo el paradigma ágil, es decir, aplican metodologías como Scrum o programación extrema. La empresa que llamaremos Acaba de iniciar un proyecto con la empresa B, el cual se trata de una aplicación Web basada en la que la parte del cliente (front-end) será desarrollada en Angular 2, y la parte del servidor (back-end) será desarrollada en Node.js y MongoDB. La manera en la que back-end y front-end se comunican es a través de un servicio tipo REST, por lo que este no presenta una definición de interface como lo hacen los servicios tipo SOAP. Este servicio está siendo desarrollado por la empresa A casi al paralelo que el front-end, el cual está a cargo de la empresa B, por lo tanto no se tiene documentación sobre los recursos que el servicio expone, ni de los métodos disponibles para cada uno de ellos. Además, la empresa A es quien tiene contacto directo con el cliente, por lo que las historias de usuario las proporciona esta misma empresa, y a quien la empresa B acude para resolver dudas sobre la funcionalidad. Por otro lado, las historias de usuario se están definiendo y refinando a medida que avanza el proyecto, por lo que es común que haya modificaciones en elementos desarrollados con anterioridad.

D.2. Historias de usuario

- **Historia de usuario 1**. Yo como asistente médico quisiera buscar un paciente por nombre o número de seguridad social al momento de otorgarle una cita para el médico al cual estoy asignado.
  - Criterios de aceptación.
    1. Al presionar el botón de buscar debe iniciarse la búsqueda.
    2. Se tiene un solo campo para búsqueda, por lo que se debe identificar automáticamente si se trata de un número de seguridad social o el nombre del paciente.
    3. Sólo se hace búsquedas de los pacientes adscritos a la clínica actual.
    4. En caso de hacer la búsqueda por nombre se deben ingresar por lo menos 2 palabras.
    5. Si la búsqueda es de más de 30 resultados se debe pedir al usuario que refine los criterios de búsqueda.
    6. Al dar doble click a uno de los elementos de los resultados se considera como elegido el paciente.
- **Historia de usuario 2**. Yo como asistente médico deseo registrarle una cita a un paciente para el médico al cual estoy asignado.
  - Criterios de aceptación.
    1. Se debe seleccionar una fecha mayor o igual a la fecha actual. En caso se ser la fecha actual la hora debe ser mayor a la actual.
    2. La fecha debe ser seleccionada de un calendario con la capacidad de poder navegar por días y meses.

3. En el calendario para escoger fecha se deben marcar de rojo los días en los que ya no hay disponibilidad porque se llenó la agenda y de azul los días en los que no se deben asignar citas, ya sea por inhábil (festivo o fin de semana) o por vacaciones del médico. Las fechas en rojo y las fechas en azul no deben poder seleccionar.
4. Al presionar el botón de asignar cita debe verificar que no exista otra cita en la misma fecha y hora. Además, debe asegurarse que no exista una cita asignada en una fecha posterior a la fecha actual, para el mismo paciente. En caso de ello debe de sugerirse cambiar la fecha de la cita ya existente. Una vez asegurado lo anterior se debe guardar el registro de la cita.
5. Todos los campos son requeridos.

- **Historia de usuario 3**. Yo como asistente médico me gustaría poder buscar citas por paciente o por rango de fechas sólo del médico al cual estoy asignado.
  - Criterios de aceptación.
    1. Debe existir una manera de seleccionar la modalidad de búsqueda: por fechas o por paciente.
    2. Cuando se busca por fechas, podrá seleccionarse buscar por una fecha exacta o por rango de fechas. Al tener un rango de fechas, la fecha inicial debe ser menor a la fecha final y ambas fechas deben estar llenas.
    3. La selección de fechas se debe hacer sobre un calendario en donde deben de estar marcados de azul los días inhábiles (festivo o fin de semana) o y las vacaciones del médico. Las fechas en azul no debe ser seleccionadas.
    4. Para buscar por paciente, selecciona el paciente tal como lo indica la historia de usuario 1.
    5. Al presionar el botón de buscar se debe mostrar en una tabla los resultados obtenidos. Los campos a mostrar son: fecha, hora y nombre del paciente. En caso de no obtener resultados debe mostrarse una leyenda indicando que no hubo coincidencias. Si son más de 30 resultados deben paginarse los resultados.
- **Historia de usuario 4**. Yo como asistente médico deseo poder cancelar una cita de un paciente adscrito al médico al cual estoy asignado.
  - Criterios de aceptación.
    1. Una vez mostrados los resultados de la búsqueda (historia 4), se selecciona la opción de cancelar. Se debe preguntar en una ventana emergente si se está seguro de cancelar la cita del paciente actual (mostrar nombre del paciente). De ser así, se marca como cancelada la cita y se elimina de los resultados de búsqueda. En caso contrario sólo se cierra la ventana emergente.
    2. No se mostrará la opción de cancelar de citas con fecha menor a la actual.
- **Historia de usuario 5**. Yo como asistente médico deseo cambiar de fecha citas de pacientes adscritos al médico al cual estoy asignado.
  - Criterios de aceptación.
    1. Una vez mostrados los resultados de la búsqueda (historia 4), selecciono la opción de cambiar fecha. Se debe mostrar un calendario para seleccionar la nueva fecha, el cual se debe mostrar como se indica en la historia 2. También se aplicarán los mismos criterios para dar de alta una nueva cita, tal como se indican en la historia 2.
    2. No se mostrará la opción de cancelar de citas con fecha menor a la actual.

D.3. Guión para cliente.

1. Saludar
2. Esperar respuesta
3. Presentarse. Ejemplo: Soy uno de los que va a desarrollar el front-end de las historias de usuario, según me comentaron usted es uno de los puntos de contacto para resolver dudas respecto al back-end y sobre funcionalidad.
4. Esperar respuesta
5. Para empezar en tengo dudas de como conectarme, no me han dado ni la IP, me puede ayudar con eso?
6. Esperar respuesta
7. Ah muy bien, me imagino que el api key se envía en los encabezados del HTTP, como se debe llamar el campo en el encabezado?
8. Esperar respuesta
9. Si, el postman (***etiquetar***)
10. Esperar respuesta
11. Gracias. Veo que las historias de usuario están enfocadas a un asistente médico y su médico asignado. Como podría simular que el asistente ésta autenticado si aun no está listo el login? (***etiquetar***)
12. Esperar respuesta
13. No, no tenemos, tendríamos que buscar uno.
14. Esperar respuesta
15. Ah muy bien, gracias nuevamente. Acerca de los asistentes, me podrías pasar una lista de ellos para probar.
16. Esperar respuesta
17. Gracias, con ese es suficiente por ahora. Oye, y ese asistente ya tiene citas dadas de alta?
18. Esperar respuesta
19. OK, ahora tengo unas dudas sobre la historia de usuario 1. Que recurso del servicio debo consultar?
20. Esperar respuesta
21. También en la historia 1, dice que se debe identificar automáticamente cuando es un nombre y cuando es un numero de seguridad social, como es ese número (***etiquetar***)
22. Esperar respuesta
23. Muy bien. Ahora de la historia 2, que recurso se usa? (***etiquetar***)
24. Esperar respuesta
25. Y para saber la fechas en las que ya no hay citas disponibles?
26. Esperar respuesta
27. Voy a probar ahorita mismo con los datos que me diste.
28. Esperar respuesta
29. Si funciona, y veo que las fechas llevan un formato especial. Ese mismo forma debo usar para enviarlas? (***etiquetar***)
30. Esperar respuesta
31. Y en la misma historia cuando se manda guardar, las validaciones del criterio 4 las aplican en el backend?
32. Esperar respuesta
33. Ya casi termino, ahora de la historia 3, el recurso que consulta es el mismo de la historia 2? (***etiquetar***)

34. Esperar respuesta
35. Voy a hacer una prueba
36. Esperar respuesta
37. Oye me dice que el método GET no está habilitado, que pasa?
38. Esperar respuesta
39. Me imagino que los datos te los mando en json, como sería el formato?
40. Esperar respuesta
41. Me puedes decir en que fechas puedo encontrar citas, para hacer una prueba?
42. Esperar respuesta
43. Veo los resultados vienen dentro de una lista anidada en el objeto de respuesta, los campos que vienen extra para que son?
44. Esperar respuesta
45. Muy bien, bueno creo que eso es todo por el momento, si tengo mas dudas te digo. Gracias!
46. Esperar respuesta
47. Perfecto.

D.4. Guion para servidor

1. Responder al saludo
2. Esperar respuesta
3. Si, a mí me puedes preguntar todo lo referente al servicio y las historias de usuario.
4. Esperar respuesta
5. Mira la IP es 45.65.234.9, tienes que usar el puerto 8080 para desarrollo. Y también tienes que enviar todas las peticiones con este api key QWE123RTY345 (***etiquetar***).
6. Esperar respuesta
7. Si es por el encabezado, el campo se debe llamar key. Vas a usar algún programa para probar las peticiones?
8. Esperar respuesta
9. A que bien, aquí ya tenemos unas pruebas guardadas de postman, te las mando al correo.
10. Esperar respuesta
11. Ah OK, además del api key tienes que mandar en una cadena encriptada el id del asistente. La encripción se hace con SHA1. Ya tienes algún componente para hacer esa encripción? (***etiquetar***)
12. Esperar respuesta
13. Acá tenemos uno que usamos en proyectos anteriores de Angular 2, te mando una liga para que lo bajes (***etiquetar***)
14. Esperar respuesta
15. Ahorita nomas tengo uno, pero si ocupas otro, al rato te lo mando. El asistente es con esta matrícula: 890687567
16. Esperar respuesta
17. Si, tiene algunas y va a tener mas porque estamos haciendo unas pruebas acá en el backend
18. Esperar respuesta
19. El de pacientes, tal cual pacientes. (***etiquetar***)
20. Esperar respuesta
21. Es una secuencia de 9 números únicamente.
22. Esperar respuesta

23. Se llama citas tal cual. (***etiquetar***)
24. Esperar respuesta
25. Ahí es citas/diasnodisponibles/ y se le manda por query string la matrícula del asistente (***etiquetar***)
26. Esperar respuesta
27. OK
28. Esperar respuesta
29. Si, se me había pasado especificar eso, formato yyyy-mm-ddThh:00
30. Esperar respuesta
31. Así, eso es para nosotros.
32. Esperar respuesta
33. Así es
34. Esperar respuesta
35. OK
36. Esperar respuesta
37. Ah es que ahí decidimos usar POST porque se nos facilita el manejo acá en le backend. (***etiquetar***)
38. Esperar respuesta
39. FechaIni, FechaFin, nombre, nss, todo en string
40. Esperar respuesta
41. Seguro, vas a encontrar citas en todo el mes de febrero de este año.
42. Esperar respuesta
43. Esos son los que manejan el paginado, si en la consulta agregas el campo página, es la que se te regresará.
44. Esperar respuesta
45. OK, cuando gustes, también puede dejarnos comentarios en trello si quieres.

D.5. Extensión para cuestionario SUS

1. Me pareció que la herramienta de etiquetado se integra con la manera en la que me comunico con mis compañeros de trabajo que están de manera remota.

Totalmente en desacuerdo	1	2	3	4	5	6	7	Totalmente de acuerdo

2. La herramienta de etiquetado interfiere con la manera en la que interactúo con mis compañeros de trabajo que están de manera remota.

Totalmente en desacuerdo	1	2	3	4	5	6	7	Totalmente de acuerdo

D.6. Extensión para cuestionario TAM

1. Mediante el uso del buscador de AK se reducirían las interrupciones entre los compañeros de trabajo.

Totalmente en desacuerdo	1	2	3	4	5	6	7	Totalmente de acuerdo

2. Usando el buscador de AK encontraría conocimiento técnico importante relacionad con los proyectos de software en los que trabajo.

Totalmente en desacuerdo	1	2	3	4	5	6	7	Totalmente de acuerdo

3. El buscador de AK facilitaría la localización de conocimiento técnico durante las tareas de desarrollo.
   Totalmente en desacuerdo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Totalmente de acuerdo

4. Con el buscador de AK encontraría conocimiento técnico justo cuando lo necesito.
   Totalmente en desacuerdo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Totalmente de acuerdo

5. ¿Cómo considera la solución completa de condensación del AK (ayudante de etiquetado + buscador)? (explique su respuesta)
   Pésima herramienta | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Excelente herramienta

6. ¿Qué sugeriría para mejorarla?

D.7. Cuestionario basado en el escenario

1. ¿Cuál es la dirección IP y el puerto del servidor de desarrollo del proyecto?
2. ¿Qué recurso REST está indicado para la historia usuario 3?
3. ¿Qué recurso se necesita para descubrir las fechas no disponibles para una cita?
4. ¿Cómo es posible simular que se está autenticado con un determinado usuario de asistente médico?
5. ¿La búsqueda por número de seguro social debe ser precisa en términos de recibir un número completo o sólo un fragmento?
6. ¿Cómo se identifica que debe sugerirse un cambio en la fecha de una cita?
7. ¿Qué significan los códigos HTTP no convencionales en el servicio?
8. Para cambiar la fecha de la cita, ¿se envía su identificador junto con la URL o en el cuerpo de la solicitud?
9. ¿En qué período se pueden encontrar citas para hacer las pruebas?
10. ¿Cuál es el formato del número de seguro social?
11. Al usar el comando PUT para cualquier actualización, ¿el identificador del recurso se envía como una cadena de consulta o como parte del cuerpo del mensaje?
12. ¿Los mensajes de error del servicio son adecuados para mostrarse al usuario final?

Referencias

Al-kofahi, J.M., Tamrawi, A., Nguyen, T.T., Nguyen, H.A., Nguyen, T.N., 2010. Fuzzy Set Approach for Automatic Tagging in Evolving Software. In: Software Maintenance (ICSM), 2010 IEEE International Conference On. IEEE, Timisoara, Romania.

Ali, N., Beecham, S., Mistrik, I., 2010. Architectural Knowledge Management in Global Software Development: A Review. In: Global Software Engineering (ICGSE), 2010 5th IEEE International Conference On. pp. 347–352.

Ali, N., Beecham, S., Mistrik, I., 2010. Architectural Knowledge Management in Global Software Development: A Review. In: 2010 5th IEEE International Conference on Global Software Engineering. Ieee, pp. 347–352.

Allen, T.J., 1984. Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information Within the R&D Organization. MIT Press.

Anderson, D.J., 2010. Kanban: successful evolutionary change for your technology business. Blue Hole Press.

Antonino, P.O., Keuler, T., Antonino, P., Keuler, T., Germann, N., Cronauer, B., 2014. A Non-Invasive Approach to Trace Architecture Design , Requirements Specification , and Agile Artifacts. In: Software Engineering Conference (ASWEC), 2014 23rd Australian. IEEE, Milsons Point, NSW, pp. 220–229.

Augustine, S., 2005. Managing Agile Projects. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Avritzer, A., Bronsard, F., Matos, G., 2010. Improving Global Development Using Agile - How Agile Processes Can Improve Productivity in Large Distributed Projects. In: Šmite, D., Moe, N.B., Ågerfalk, P.J. (Eds.), Agility Across Time and Space: Implementing Agile Methods in Global Software Projects. Springer-Verlag, pp. 133–148.

Awar, K.B., Sameem, M.S.I., Hafeez, Y., 2017. A model for applying Agile practices in Distributed environment: A case of local software industry. In: Proceedings of 2017 International Conference on Communication, Computing and Digital Systems, C-CODE 2017. pp. 228–232.

Babar, M. a., 2009. An exploratory study of architectural practices and challenges in using agile software development approaches. In: 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture. pp. 81–90.

Babar, M.A., 2009. Supporting the software architecture process with knowledge management. In: Software Architecture Knowledge Management: Theory and Practice. pp. 69–86.

Bagheri, E., Ensan, F., 2016. Semantic tagging and linking of software engineering social content. Autom. Softw. Eng. 23, 147–190.

Bangor, A., Kortum, P.T., Miller, J.T., 2008. An empirical evaluation of the system usability

scale. Int. J. Hum. Comput. Interact. 24, 574–594.

Bass, J.M., 2016. Artefacts and agile method tailoring in large-scale offshore software development programmes. Inf. Softw. Technol. 75, 1–16.

Bass, L., Clements, P., Kazman, R., 2003. Software Architecture in Practice, 2nd ed. Addison-Wesley Professional.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., 2001. The Agile Manifesto.

Beecham, S., Noll, J., Richardson, I., Ali, N., 2010. Crafting a global teaming model for architectural knowledge. Proc. - 5th Int. Conf. Glob. Softw. Eng. ICGSE 2010 55–63.

Bider, I., Söderberg, O., 2016. Becoming Agile in a Non-disruptive Way - Is It Possible? In: Proceedings of the 18th International Conference on Enterprise Information Systems. pp. 294–305.

Bjørnson, F.O., Dingsøyr, T., 2008. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. Inf. Softw. Technol. 50, 1055–1068.

Boehm, B., Lane, J.A., Koolmanojwong, S., Turner, R., 2010. Architected Agile Solutions for Software Reliant Systems. In: Dingsøyr, T., Dybå, T., Moe, N.B. (Eds.), Agile Software Development,. Springer Berlin Heidelberg, pp. 165–184.

Borrego, G., Morán, A.L., Palacio, R., 2017. Preliminary Evaluation of a Tag-Based Knowledge Condensation Tool in Agile and Distributed Teams. In: 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE). pp. 51–55.

Borrego, G., Morán, A.L., Palacio, R., Rodríguez-Elias, O.M., García-Canseco, E., 2017. Review of approaches to manage architectural knowledge in Agile Global Software Development. IET Softw. 11, 77–88.

Borrego, G., Morán, A.L., Palacio, R., Rodríguez, O.M., 2016. Understanding Architectural Knowledge Sharing in AGSD Teams: An Empirical Study. In: 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE). IEEE Computer Society, Los Alamitos, CA, USA, pp. 109–118.

Bosch, J., 2004. Software Architecture: The Next Step. In: Oquendo, F., Warboys, B., Morrison, R. (Eds.), EWSA, Lecture Notes in Computer Science. Springer, pp. 194–199.

Brooke, J., 1996. SUS-A quick and dirty usability scale. Usability Eval. Ind. 189, 194.

Bruun, L., Hansen, M.B., 2014. Handling Design-Level Requirements across Distributed Teams : Developing a New Feature for 12 Danish Mobile Banking Apps. In: Requirements Engineering Conference (RE), 2014 IEEE 22nd International. IEEE, Karlskrona, pp. 335–343.

Calefato, F., Gendarmi, D., Lanubile, F., 2010. Investigating the Use of Tags in Collaborative Development Environments: A Replicated Study. Proc. 2010 ACM-IEEE Int. Symp. Empir. Softw. Eng. Meas. 24:1--24:9.

Cannon-Bowers, J.A., Salas, E., Converse, S., 1993. Shared mental models in expert team decision making. In: Individual and Group Decision Making: Current Issues. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, US, pp. 221–246.

Capilla, R., Jansen, A., Tang, A., Avgeriou, P., Babar, M.A., 2016. 10 years of software architecture knowledge management: Practice and future. J. Syst. Softw. 116, 191–205.

Carmel, E., 1999. Global Software Teams: Collaborating Across Borders and Time Zones. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Cho, J., 2009. A Hybrid Software Development Method For Large-Scale Projects: Rational Unified Process With Scrum. J. Issues Inf. Syst. 10, 340–348.

Clear, T., 2003. Documentation and Agile Methods: Striking a Balance. SIGCSE Bull. 35, 12–13.

Clerc, V., Lago, P., 2011. Architectural Knowledge Management Practices in Agile Global Software Development.

Clerc, V., Lago, P., Vliet, H. Van, 2011. Architectural Knowledge Management Practices in Agile Global Software Development. In: 2011 IEEE Sixth International Conference on Global Software Engineering Workshop. Ieee, pp. 1–8.

Cockburn, A., 2002. Agile Software Development Joins the " Would-Be " Crowd what ' s your center ? 6–12.

Cockburn, A., 2006. Agile Software Development: The Cooperative Game, 2nd ed, Agile Software Development Series. Pearson Education.

Conchúir, E.Ó., Ågerfalk, P.J., Olsson, H.H., 2009. Global Software Development : Where are the Benefits ? Commun. ACM 52, 127–131.

Costa, N., Santos, N., Ferreira, N., Machado, R., 2014. Delivering User Stories for Implementing Logical Software Architectures by Multiple Scrum Teams. In: Murgante, B., Misra, S., Rocha, A.C., Torre, C., Rocha, J., Falcão, M., Taniar, D., Apduhan, B., Gervasi, O. (Eds.), Computational Science and Its Applications – ICCSA 2014 SE - 55, Lecture Notes in Computer Science. Springer International Publishing, pp. 747–762.

Dalkir, K., 2011. Knowledge Management in Theory and Practice, Second. ed. The MIT Press.

Davenport, T.H., Prusak, L., 2000. Working Knowledge: How Organizations Manage What They Know. Harvard Business Review Press.

Davis, F.D., 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Q. 13, 319–340.

de Graaf, K.A., Liang, P., Tang, A., van Hage, W.R., van Vliet, H., 2014. An exploratory study on ontology engineering for software architecture documentation. Comput. Ind. 65, 1053–1064.

Dekel, U., Herbsleb, J.D., 2008. Pushing relevant artifact annotations in collaborative software development. In: Proceedings of the ACM 2008 Conference on Computer

Supported Cooperative Work - CSCW ’08. p. 1.

Del Nuevo, E., Piattini, M., Pino, F.J., 2011. Scrum-based methodology for distributed software development. In: Proceedings - 2011 6th IEEE International Conference on Global Software Engineering, ICGSE 2011. pp. 66–74.

Díaz, J., Pérez, J., Garbajosa, J., 2014. Agile product-line architecting in practice: A case study in smart grids. Inf. Softw. Technol. 56, 727–748.

Dingsøyr, T., 2009. Strategies and Approaches for Managing Architectural Knowledge. In: Babar, M.A., Dingsøyr, T., Lago, P., van Vliet, H. (Eds.), Software Architecture Knowledge Management. Springer Berlin Heidelberg, pp. 59–68.

Dingsøyr, T., Conradi, R., 2002. A Survey of Case Studies of the Use of Knowledge Management in Software Engineering. Int. J. Softw. Eng. Knowl. Eng. 12, 391–414.

Dingsøyr, T., Šmite, D., 2014. Managing Knowledge in Global Software Development Projects. IT Prof. 16, 22–29.

Dorairaj, S., Noble, J., Malik, P., 2012. Knowledge Management in Distributed Agile Software Development. In: 2012 Agile Conference. Ieee, Dallas, USA, pp. 64–73.

Duhon, B., 1998. It’s All in Our Heads.

Dumitriu, F., Oprea, D., Mesnita, G., 2011. Issues and strategy for agile global software development adoption. In: Gavriluta, N., Raducanu, R., Iliescu, M., Costin, H., Mastorakis, N., Olej, V., Strouhal, J. (Eds.), Proceeding of the 3rd World Multiconference on Applied Economics, Business and Development, AEBD. WSEAS Press, Iasi, Romania, pp. 37–42.

Eccles, M., Smith, J., Tanner, M., Van Belle, J.-P., van der Watt, S., 2009. The Influence of Collocation on Team Effectiveness. In: Knowledge Management and Innovation in Advancing Economies: Analyses and Solutions - Proceedings of the 13th International Business Information Management Association Conference, IBIMA 2009. International Business Information Management Association, IBIMA, pp. 687–696.

Eeles, P., 2013. Building a Platform for Innovation: Architecture and Agile as Key Enablers. In: Babar, M.A., Brown, A.W., Koskimies, K., Mistrik, I. (Eds.), Agile Software Architecture: Aligning Agile Processes and Software Architectures. Elsevier Inc., pp. 315–333.

Ellis, C.A., Gibbs, S.J., Rein, G., 1991. Groupware: Some issues and experiences. Commun. ACM 34, 39–58.

Eloranta, V., Koskimies, K., 2012. Aligning architecture knowledge management with Scrum. In: Proceedings of the WICSA/ECSA 2012 …. pp. 112–115.

Eloranta, V., Koskimies, K., 2013. Lightweight Architecture Knowledge Management for Agile Software Development. In: Agile Software Architecture: Aligning Agile Processes and Software Architectures. Elsevier Science, pp. 189–211.

Esbensen, M., Tell, P., Cholewa, J.B., Pedersen, M.K., Bardram, J., 2015. The dBoard: a Digital Scrum Board for Distributed Software Development. In: Proceedings of the

2015 International Conference on Interactive Tabletops & Surfaces - ITS ’15. pp. 161–170.

Estler, H.-C., Nordio, M., Furia, C.A., Meyer, B., Schneider, J., 2012. Agile vs. Structured Distributed Software Development: A Case Study. In: Proceedings of the IEEE International Conference on Global Software Engineering.

Estler, H.-C., Nordio, M., Furia, C.A., Meyer, B., Schneider, J., 2012. Agile vs. structured distributed software development: A case study. Proc. - 2012 IEEE 7th Int. Conf. Glob. Softw. Eng. ICGSE 2012 11–20.

Fielding, R.T., Taylor, R.N., 2002. Principled design of the modern Web architecture. ACM Trans. Internet Technol. 2, 115–150.

Fitriani, W.R., Rahayu, P., Sensuse, D.I., 2016. Challenges in Agile Software Development : A Systematic Literature Review. Icacsis 155–164.

Forward, A., Lethbridge, T., Deugo, D., 2007. CodeSnippets Plug-in to Eclipse: Introducing Web 2.0 Tagging to Improve Software Developer Recall. In: Proceedings - SERA 2007: Fifth ACIS International Conference on Software Engineering Research, Management, and Applications. pp. 498–502.

Fowler, M., 2002. Patterns of Enterprise Application Architecture. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Galster, M., Babar, M.A., 2014. Empirical Study of Architectural Knowledge Management Practices. 2014 IEEE/IFIP Conf. Softw. Archit. 239–242.

Gayer, S., Herrmann, A., Keuler, T., Riebisch, M., Antonino, P.O., 2016. Lightweight Traceability for the Agile Architect. IEE Comput. 49, 64–71.

Gervigny, M.L.I., Nagowah, S.D., 2017. Knowledge Sharing for Agile Distributed Teams : A Case Study of Mauritius. In: 2017 International Conference on Infocom Technologies and Unmanned Systems (ICTUS’2017). IEEE Computer Society, Dubay, UAE, pp. 413–419.

Ghani, I., Bello, M., 2015. Agile adoption in IT organizations. KSII Trans. Internet Inf. Syst. 9, 3231–3248.

Gotterbarn, D., 2004. Irresponsible Development. inroads – SIGCSE Bull. 36.

Grant, M.J., Booth, A., 2009. A typology of reviews: An analysis of 14 review types and associated methodologies. Health Info. Libr. J. 26, 91–108.

Gutwin, C., Greenberg, S., Roseman, M., 1996. Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation. In: Sasse, M.A., Cunningham, R.J., Winder, R.L. (Eds.), People and Computers XI. Springer London, London, pp. 281–298.

Hackman, J.R., 1987. The design of work teams, Handbook of organizational behavior. Prentice Hall, Englewood Cliffs, NJ.

Hadar, I., Sherman, S., Hadar, E., Harrison, J.J., 2013. Less is more: Architecture

documentation for agile development. 2013 6th Int. Work. Coop. Hum. Asp. Softw. Eng. CHASE 2013 - Proc. 121–124.

Hamed, A.M.M., Abushama, H., 2013. Popular agile approaches in software development: Review and analysis. In: 2013 International Conference on Computing, Electrical and Electronic Engineering (Icceee). pp. 160–166.

Harrison, R., Veerappa, V., 2014a. Social Media Collaboration in Software Projects. In: Ruhe, G., Wohlin, C. (Eds.), Software Project Management in a Changing World. Springer Berlin Heidelberg, pp. 401–4014.

Harrison, R., Veerappa, V., 2014b. Social Media Collaboration in Software Projects. In: Ruhe, G., Wohlin, C. (Eds.), Software Project Management in a Changing World SE - 16. Springer Berlin Heidelberg, pp. 401–424.

Herbsleb, J.D., Moitra, D., 2001. Global Software Development. IEEE Softw. 18, 16–20.

Herbsleb, J.D., Paulish, D.J., Bass, M., 2005. Global software development at Siemens: experience from nine projects. Proceedings. 27th Int. Conf. Softw. Eng. 2005. ICSE 2005. 524–533.

Hochmüller, E., Mittermeir, R.T., 2008. Agile Process Myths. Proc. 2008 Int. Work. Scrutinizing Agil. Pract. or Shoot. Agil. corral - APOS '08 5–8.

Hoda, R., Noble, J., Marshall, S., 2010. How much is just enough? In: Proceedings of the 15th European Conference on Pattern Languages of Programs - EuroPLoP '10. ACM Press, New York, New York, USA, p. 13.

Holmstrom, H., Conchuir, E.O., Agerfalk, P.J., Fitzgerald, B., 2006. Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance. Glob. Softw. Eng. 2006. ICGSE '06. Int. Conf. 3–11.

Holmström, H., Fitzgerald, B., Ågerfalk, P.J., Conchúir, E.Ó., 2006. Agile Practices Reduce Distance in Global Software Development. Inf. Syst. Manag. 23, 7–18.

Holz, H., Maurer, F., 2003. Knowledge management support for distributed agile software processes. Adv. Learn. Softw. Organ. 2640, 60–80.

IEEE, A.W.G., 2000. IEEE Std 1471-2000, Recommended practice for architectural description of software-intensive systems. IEEE.

Isaacs, E., Whittaker, S., Frohlich, D., O'Conaill, B., 1997. Informal communication re-examined: New functions for video in supporting opportunistic encounters. … - Mediated Commun. 1–30.

Jain, R., Suman, U., 2016. Effectiveness of Agile Practices in Global Software Development. Int. J. Grid Distrib. Comput. 9, 231–248.

Jalali, S., Wohlin, C., 2010. Agile practices in global software engineering-A systematic map. In: … , 2010 5th IEEE International Conference On. Princeton, NJ, pp. 45–54.

Jiménez, M., Piattini, M., Vizcaíno, A., 2009. Challenges and Improvements in Distributed Software Development : A Systematic Review. Adv. Softw. Eng. 2009, 14.

Kamaruddin, N., Arshad, N., Mohamed, A., 2012. Comparison of drivers between global software development and agile global software development: A SURVEY. In: Computer and Mathematical Sciences Graduates National Colloqium 2013, At Faculty of Computer and Mathematical Sciences, UiTM Shah Ala. Kuala Lumpur, Malasia.

Kim, W., Chung, S., Endicott-popovsky, B., 2014. Software architecture model driven reverse engineering approach to open source software development. In: Proceedings of the 3rd Annual Conference on Research in Information Technology. New York, NY, USA, pp. 9–14.

Klobas, J.E., Beesley, A., 2006. Wikis: tools for information work and collaboration, Chandos information professional series. Chandos.

Korkala, M., Maurer, F., 2014. Waste identification as the means for improving communication in globally distributed agile software development. J. Syst. Softw. 95, 122–140.

Kozlowski, S.W.J., Ilgen, D.R., 2006. Enhancing the Effectiveness of Work Groups and Teams. Psychol. Sci. Public Interes. 7, 77–124.

Kruchten, P., 2009. Documentation of Software Architecture from a Knowledge Management Perspective - Design Representation. In: Ali Babar, M., Dingsøyr, T., Lago, P., van Vliet, H. (Eds.), Software Architecture Knowledge Management: Theory and Practice. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 39–57.

Kruchten, P., Obbink, H., Stafford, J., 2006. The Past, Present, and Future for Software Architecture. IEEE Softw. 23, 22–30.

Kruchten, P.B., 1995. The 4+ 1 view model of architecture. Software, IEEE November 1, 9.

Levy, M., Hazzan, O., 2009. Knowledge management in practice: The case of agile software development. 2009 ICSE Work. Coop. Hum. Asp. Softw. Eng. 60–65.

Li, Y., Maedche, A., 2012. Formulating Effective Coordination Strategies in Agile Global Software Development Teams. In: Proceedings of the International Conference on Information Systems. Orlando, Florida, USA.

Lipnack, J., Stamps, J., 1997. Virtual Teams: Reaching Across Space, Time and Organizations with Technology. John Wiley & Sons, Inc., New York, NY, USA.

Lloyd, D., Moawad, R., Kadry, M., 2017. A supporting tool for requirements change management in distributed agile development. Futur. Comput. Informatics J. 2, 1–9.

Martin, P.Y., Turner, B.A., 1986. Grounded Theory and Organizational Research. J. Appl. Behav. Sci. 22, 141–157.

Martini, A., Bosch, J., 2014. Role of Architects in Agile Organizations. In: Bosch, J. (Ed.), Continuous Software Engineering. Springer International Publishing, pp. 39–50.

Martini, A., Pareto, L., Bosch, J., 2013. Communication factors for speed and reuse in large-scale agile software development. Proc. 17th Int. Softw. Prod. Line Conf. - SPLC '13 42.

Meyer, A.N., Fritz, T., Murphy, G.C., Zimmermann, T., 2014. Software Developers' Perceptions of Productivity. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014. ACM Press, Hong Kong, China, pp. 19–29.

Moe, N.B., Börjesson, A., Andréasson, P., 2014. Networking in a Large-Scale Distributed Agile Project. In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. New York, NY, USA, p. 8.

Moraes, A., Silva, E., da Trindade, C., Barbosa, Y., Meira, S., 2010. Recommending experts using communication history. In: 2nd International Workshop on Recommendation Systems for Software Engineering - RSSE '10. pp. 41–45.

Nonaka, I., Takeuchi, H., 1995. The knowledge-creating company: How japanese companies create the dynamics of innovation, The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. Oxford University Press, New York.

Nonaka, I., Toyama, R., Konno, N., 2000. SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation. Long Range Plann. 33, 5–34.

Noordeloos, R., Manteli, C., 2012. From RUP to Scrum in Global Software Development : A Case Study.

Nowak, M., Pautasso, C., 2010. Architectural Decision Modeling with Reuse : Challenges and Opportunities. In: SHARK '10 Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge. pp. 13–20.

O'Reilly, C., Bustard, D., Morrow, P., 2005. The war room command console: shared visualizations for inclusive team coordination. In: Proceedings of the 2005 ACM Symposium on Software Visualization. pp. 57–65.

Orasanu, J., Salas, E., 1993. Team decision making in complex environments. In: Decision Making in Action: Models and Methods. Ablex Publishing, Westport, CT, US, pp. 327–345.

Paasivaara, M., 2017. Adopting SAFe to scale agile in a globally distributed organization. In: Proceedings - 2017 IEEE 12th International Conference on Global Software Engineering, ICGSE 2017. pp. 36–40.

Paasivaara, M., Behm, B., Lassenius, C., Hallikainen, M., 2018. Large-scale agile transformation at Ericsson: a case study. Empir. Softw. Eng. 1–47.

Paasivaara, M., Lassenius, C., 2014. Communities of practice in a large distributed agile software development organization - Case Ericsson. Inf. Softw. Technol. 56, 1556–1577.

Papatheocharous, E., Andreou, A.S., 2013. Evidence of Agile Adoption in Software Organizations: An Empirical Survey. In: Communications in Computer and Information Science. pp. 237–246.

Paper, C., Dings, T., Brede, N., Sintef, M., 2014. Towards Principles of Large-Scale Agile Development : A Summary of the workshop at XP2014 and a revised research agenda.

In: Workshop at XP2014 and a Revised Research Agenda.

Paredes, J., Anslow, C., Maurer, F., 2014. Information Visualization for Agile Software Development Teams. In: 2014 Second IEEE Working Conference on Software Visualization. pp. 157–166.

Paul, S., Makkar, T., Chandrasekaran, K., 2015. Software Development Using Context Aware Searching Of Components In Large Repositories. In: International Conference on Computing, Communication and Automation (ICCCA2015). IEEE Computer Society, pp. 765–772.

Pérez, J., Díaz, J., Garbajosa, J., Alarcón, P.P., 2010. Flexible working architectures: Agile architecting using PPCs. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 6285 LNCS, 102–117.

Pfleeger, S.L., 2001. Software Engineering: Theory and Practice, 2nd ed. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Phalnikar, R., Deshpande, V.S., Joshi, S.D., 2009. Applying agile principles for distributed software development. In: Proceedings - International Conference on Advanced Computer Control, ICACC 2009. pp. 535–539.

Philippe, K., Patricia, L., Hans, van V., 2006. Building up and reasoning about architectgural knowledge. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 4214, 95–110.

Ramesh, B., Cao, L., Mohan, K., Xu, P., 2006. Can distributed software development be agile? Commun. ACM 49, 41.

Razzak, M.A., Smite, D., 2015. Knowledge Management in Globally Distributed Agile Projects – Lesson Learned. In: Global Software Engineering (ICGSE), 2015 IEEE 10th International Conference On. Ciudad del Real, Spain, pp. 81–89.

Read, K., Maurer, F., 2003. Issues in scaling agile using an architecture-centric approach: A tool-based solution. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 2753, 142–150.

Richter, H., Abowd, G., Miller, C., Funk, H., 2004. Tagging Knowledge Acquisition Sessions To Facilitate Knowledge Traceability. Int. J. Softw. Eng. Knowl. Eng. 14, 3–19.

Richter, I., Raith, F., Weber, M., 2016. Problems in Agile Global Software Engineering Projects especially within Traditionally Organised Corporations. Proc. Ninth Int. C* Conf. Comput. Sci. Softw. Eng. - C3S2E ’16 33–43.

Rost, D., Naab, M., Lima, C., von Flach Garcia Chavez, C., 2013. Architecture Documentation for Developers : A Survey. In: Drira, K. (Ed.), Software Architecture. Springer Berlin Heidelberg, Montpellier, France, pp. 72–88.

Ryan, S., O’Connor, R. V., 2013. Acquiring and sharing tacit knowledge in software development teams: An empirical study. Inf. Softw. Technol. 55, 1614–1624.

Sangwan, R., Bass, M., Mullick, N., Paulish, D., Kazmeier, J., 2006. Global Software

Development Handbook (Auerbach Series on Applied Software Engineering Series). Auerbach Publications, Boston, MA, USA.

Sauro, J., Lewis, J.R., 2012. Quantifying the User Experience: Practical Statistics for User Research, 1st ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Schmidt, N., Meures, C., 2016. "Mind the Gap": An Analysis of Communication in Agile Global Outsourced Software Development Projects. In: 49th Hawaii International Conference on System Sciences. IEEE Computer Society, pp. 501–510.

Schwaber, K., Sutherland, J., 2011. The scrum guide. Scrum. org, Oct. 2, 17.

Selic, B., 2009. Agile Documentation, Anyone? IEEE Softw. 26, 11–12.

Sharp, H., Giuffrida, R., Melnik, G., 2012. Information Flow within a Dispersed Agile Team: A Distributed Cognition Perspective. In: Wohlin, C. (Ed.), Agile Processes in Software Engineering and Extreme Programming SE - 5, Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 62–76.

Shen, X., Li, Y., Sun, Y., 2016. An Agile Enterprise Architecture-Driven Model for Geographically Distributed Agile Development. Transform. Healthc. Through Inf. Syst. 17, 185–197.

Shore, J., Warden, S., 2007. The Art of Agile Development, First. ed. O'Reilly.

Shrivastava, S.V., Date, H., 2010. Distributed Agile Software Development: A Review. J. Comput. Sci. Eng. 1, 10–17.

Šmite, D., Ågerfalk, P.J., Moe, N.B., 2010a. Fundamentals of Agile Distributed Software Development. In: Agility Across Time and Space: Implementing Agile Methods in Global Software Projects. pp. 3–7.

Šmite, D., Ågerfalk, P.J., Moe, N.B., 2010b. Coordination between global agile teams: from process to architecture. In: Agility Across Time and Space: Implementing Agile Methods in Global Software Projects. pp. 1–341.

Šmite, D., Brede, N., Aivars, Š., Wohlin, C., 2017. Software teams and their knowledge networks in large-scale software development. Inf. Softw. Technol. 86, 71–86.

Smith, G., Sidky, A., 2009. Becoming Agile: ...in an imperfect world. Manning.

Sneed, H.M., 2014. Dealing with Technical Debt in agile development projects. Lect. Notes Bus. Inf. Process. 166 LNBIP, 48–62.

Sohan, S.M., Richter, M.M., Maurer, F., 2010. Auto-tagging emails with user stories using project context. Lect. Notes Bus. Inf. Process. 48 LNBIP, 103–116.

Solinski, A., Petersen, K., 2016. Prioritizing agile benefits and limitations in relation to practice usage, Software Quality Journal. Springer US.

Sommerville, I., 2010. Software Engineering, 9th ed. Addison-Wesley, Harlow, England.

Sriram, R., Mathew, S.K., 2012. Global software development using agile methodologies: A review of literature. 2012 IEEE Int. Conf. Manag. Innov. Technol. 389–393.

Stettina, C.J., Heijstek, W., 2011. Necessary and neglected? An empirical study of internal documentation in agile software development teams. In: Proceedings of the 29th ACM International Conference on Design of Communciation (SIGDOC 2011). pp. 159–166.

Stol, K.J., Babar, M.A., Avgeriou, P., Fitzgerald, B., 2010. A comparative study of challenges in integrating Open Source Software and Inner Source Software. Inf. Softw. Technol. 53, 1319–1336.

Storey, M.A., Ryall, J., Singer, J., Myers, D., Cheng, L.T., Muller, M., 2009. How software developers use tagging to support reminding and refinding. IEEE Trans. Softw. Eng. 35, 470–483.

Tanveer, M., 2015. Agile For Large Scale Projects – A Hybrid Approach. In: 2015 National Software Engineering Conference (NSEC 2015). pp. 14–18.

Tiwana, A., 2004. An Empirical Study of the Effect of Knowledge Integration on Software Development Projects. Inf. Softw. Technol. 46, 899–906.

Tom, E., Aurum, A., Vidgen, R., 2013. An exploration of technical debt. J. Syst. Softw. 86, 1498–1516.

Treude, C., Storey, M.A., 2009. How tagging helps bridge the gap between social and technical aspects in software development. In: Proceedings - International Conference on Software Engineering. pp. 12–22.

Turk, D., France, R., Rumpe, B., 2002. Limitations of agile software processes. In: Third International Conference on Extreme Programming and Flexible Processes in Software Engineering. Springer-Verlag, pp. 43–46.

Uikey, N., Suman, U., Ramani, A., 2011. A Documented Approach in Agile Software Development. Int. J. Softw. … 2, 13–22.

Urrego, J., Munoz, R., Mercado, M., Correal, D., 2014. Archinotes: A global agile architecture design approach. Lect. Notes Bus. Inf. Process. 179 LNBIP, 302–311.

Van Hillegersberg, J., Ligtenberg, G., Aydin, M.N., 2011. Getting agile methods to work for Cordys global software product development. In: Kotlarsky, J., Willcocks, L.P., Oshri, I. (Eds.), New Studies in Global IT and Business Service Outsourcing. Springer Berlin Heidelberg, pp. 133–152.

Vizcaíno, A., García, F., Piattini, M., Garrido, P., 2014. El desarrollo global del software. In: Vizcaíno, A., García, F., Piattini, M. (Eds.), Desarrollo Global de Software. Ra-ma, pp. 37–58.

Wenger, E., McDermott, R., Snyder, W., 2002. Cultivating Communities of Practice: A Guide to Managing Knowledge. Harvard Business School Press, Boston, MA, USA.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2000. Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Norwell, MA, USA.

Xu, P., Cao, L.A.N., Mohan, K., 2006. Can distributed software development be agile? Commun. ACM 49, 41–46.

Yagüe, A., Garbajosa, J., Díaz, J., González, E., 2016. An exploratory study in communication in Agile Global Software Development. Comput. Stand. Interfaces 48, 184–197.

Yang, C., Liang, P., Avgeriou, P., 2016. A systematic mapping study on the combination of software architecture and agile development. J. Syst. Softw. 111, 157–184.

Yanzer Cabral, A.R., Ribeiro, M.B., Noll, R.P., 2014a. Knowledge Management in Agile Software Projects: A Systematic Review. J. Inf. Knowl. Manag. 13, 1450010.

Yanzer Cabral, A.R., Ribeiro, M.B., Noll, R.P., 2014b. Knowledge Management in Agile Software Projects: A Systematic Review. J. Inf. & Knowl. Manag. 13, 1450010.

Yazdani, S., Ivanov, I., AnaLoui, M., Berangi, R., Ebrahimi, T., 2012. Spam Fighting in Social Tagging Systems. In: Aberer, K., Flache, A., Jager, W., Liu, L., Tang, J., Guéret, C. (Eds.), Social Informatics: 4th International Conference, SocInfo 2012, Lausanne, Switzerland, December 5-7, 2012. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 448–461.

Young, C., Terashima, H., 2008. How did we adapt Agile processes to our distributed development? In: Proceedings - Agile 2008 Conference. pp. 304–309.

Zernadji, T., Tibermacine, C., Cherif, F., 2014. Processing the evolution of quality requirements of web service orchestrations: A pattern-based approach. Proc. - Work. IEEE/IFIP Conf. Softw. Archit. 2014, WICSA 2014 139–142.

Zimmermann, O., Wegmann, L., Koziolek, H., Goldschmidt, T., 2015. Architectural Decision Guidance Across Projects-Problem Space Modeling, Decision Backlog Management and Cloud Computing Knowledge. In: Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015. pp. 85–94.