CIMAT Centro de Investigación en Matemáticas, A.C.

GUÍA DE ACTIVIDADES, TÉCNICAS, HERRAMIENTAS Y ARTEFACTOS DE INGENIERÍA DE REQUISITOS ÚTILES PARA EL DESARROLLO DE ARQUITECTURA DERIVADA DE TRES ENFOQUES ÁGILES

T E S I S

Que para obtener el grado de

Maestro en Ingeniería de Software

Presenta

Gabriela Mayanin Sobrevilla Dominguez

Codirectores de Tesis:

Dra. Perla Inés Velasco Elizondo

MID. José Guadalupe Hernández Reveles

Autorización de la versión final

Zacatecas, Zac., 14 de Noviembre de 2017

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS, A.C.

BIBLIOTECA

AUTORIZACION
PUBLICACION EN FORMATO ELECTRONICO DE TESIS

El que suscribe

Autor(s) de la tesis: ----------

Institución y Lugar: ----------

Grado Académico: Licenciatura ( ) Maestría ( ) Doctorado ( ) Otro ( ) ----------

Año de presentación: ----------

Área de Especialidad: ----------

Director(es) de Tesis: ----------

Correo electrónico: ----------

Domicilio: ----------

Palabra(s) Clave(s): ----------

ORCID: ----------

Por medio del presente documento autorizo en forma gratuita a que la Tesis arriba citada sea divulgada y reproducida para publicarla mediante almacenamiento electrónico que permita acceso al público a leerla y conocerla visualmente, así como a comunicarla públicamente en la Página WEB del Repositorio Institucional y Repositorio Nacional.

La vigencia de la presente autorización es por tiempo indefinido a partir de la firma de presente instrumento, quedando en el entendido de que si por alguna razón el alumno desea revocar la autorización tendrá que hacerlo por escrito con acuse de recibo de parte de alguna autoridad del CIMAT

Atentamente

_____

Gabriela Mayanin Sobrevilla Dominguez

CALLE JALISCO S/N MINERAL DE VALENCIANA APDO. POSTAL 402
C.P. 36240 GUANAJUATO, GTO., MÉXICO
TELÉFONOS (473) 732 7155, (473) 735 0800 EXT. 49609 FAX. (473) 732 5749
E-mail: biblioteca@cimat.mx

Resumen

El presente proyecto de tesis se basó en generar una guía que integra un conjunto de actividades, técnicas, herramientas y artefactos de ingeniería de requisitos enfocados en el desarrollo ágil de arquitectura de software. Los métodos de desarrollo ágil actualmente son poco prescriptivos en el proceso de desarrollo de la arquitectura. La falta de actividades explícitas sobre manejo de requisitos en los métodos ágiles genera dificultades a los equipos de desarrollo para administrar sus requisitos considerando la información útil para la construcción de la arquitectura de manera ágil.

La guía propuesta busca orientar a desarrolladores de software en el manejo de los requisitos poniendo especial atención en actividades que generarán artefactos útiles para el desarrollo de la arquitectura. Esto a través de la recopilación, clasificación y síntesis de actividades, técnicas, herramientas y artefactos los cuales han sido colocados en tarjetas auto contenidas para promover la facilidad de uso.

La recopilación de información de la guía se realizó tomando como base el framework propuesto por Hofmeister et al. (2007), para la comparación de métodos de diseño de arquitectura. En este caso, se extendió a la ingeniería de requisitos analizando los mismos elementos: actividades, técnicas, herramientas y artefactos. Luego, para la identificación y clasificación de actividades por prioridad, se utilizaron encuestas en línea que permitieron a dos grupos de ingenieros de software realizar una priorización de las actividades.

Finalmente se compararon sus resultados, se identificaron las actividades de mayor prioridad y se aplicó la prueba de los rangos con signo de Wilcoxon para analizar su similitud. Para validar los resultados obtenidos de la prueba de Wilcoxon se utilizó una herramienta de terceros con la cual se comprobó la correctitud de los resultados obtenidos.

Palabras clave:

Requisitos de software, arquitectura de software, métodos ágiles

Agradecimientos

A Pablo, por haber decidido acompañarme en esta etapa tan importante en nuestras vidas. Ha sido un largo e inolvidable viaje.

A mis padres, que aunque han tomado diferentes caminos, lo que aprendí de ustedes me ha ayudado a llegar al lugar donde estoy y superar los obstáculos que he encontrado a través de mi camino.

A mi familia y amigos por su apoyo a través de la distancia.

A mis asesores de tesis, Dra. Perla Velasco-Elizondo y MID. José Guadalupe Hernández Reveles por su valioso apoyo, tiempo y comprensión.

A CONACyT por hacer posible a través de su programa de becas, la realización de este proyecto de tesis.

A CIMAT por impulsar a sus alumnos en el ámbito científico, especialmente gracias por el apoyo económico recibido que permitió mi asistencia a diversas conferencias que me dejaron gran valor en lo profesional.

Finalmente, a todos los ingenieros de software que ayudaron en los experimentos de este proyecto por su valioso tiempo.

Índice

Índice de Figuras

Índice de Tablas

1 INTRODUCCIÓN

Este capítulo tiene como finalidad proporcionar un panorama general del proyecto de tesis incluyendo el contexto en el cual se desarrolla, problemática, objetivos y finalmente una descripción de la organización del documento.

1.1 Contexto

En la actualidad existen diferentes métodos para el desarrollo de software, sin embargo todos tienen como característica considerar en mayor o menor grado actividades técnicas relacionadas con la planificación del proyecto, análisis de requisitos, diseño, construcción, pruebas y despliegue de sistemas. Murch (2012) describe estas actividades técnicas de la siguiente forma:

- **Planificación del proyecto**. El propósito de la planificación es identificar las necesidades de tecnologías de la información derivadas de los objetivos empresariales. En esta fase, el enfoque de desarrollo, beneficios así como una estimación razonable del costo de desarrollo del diseño conceptual son determinados.
- **Análisis de requisitos**. El propósito del análisis es formular y formalizar los requisitos del sistema. Esto es logrado por medio del establecimiento de lo que el sistema debe hacer de acuerdo a los requisitos y expectativas de los usuarios finales.
- **Diseño.** El propósito del diseño es planificar un sistema que cumpla los requisitos definidos en el análisis. El objetivo de la fase de diseño es haber definido un medio para implementar la solución del proyecto sin haberla implementado.
- **Construcción**. El propósito de la fase de construcción es completar el diseño detallado de las arquitecturas requeridas (desarrollo, ejecución y operaciones) y construir la aplicación o sistema utilizando el ambiente y herramientas de desarrollo acordados. Los desarrolladores crean la aplicación deseada utilizando los documentos de diseño como su guía.
- **Pruebas**. El propósito de esta fase es determinar las pruebas requeridas para producir un sistema de calidad y asegurar que existe el plan e información adecuada para realizar las pruebas. Las pruebas a menudo se convierten en una actividad crítica debido a que raramente es una tarea planeada hasta que los programas se escriben en gran medida e incluso cuando se planea es subestimada y se carece de personal que realice esta actividad.

- **Despliegue**. El propósito de esta fase es iniciar todas las actividades de preparación necesarias para desplegar el nuevo sistema a los usuarios finales y en ambiente de producción. Esta fase es realizada para asegurar que el despliegue del nuevo sistema será suave y en tiempo.

Estas actividades técnicas pueden llevarse a cabo utilizando un orden particular según el método que se adopte, como por ejemplo ágil o tradicional. Por otro lado, es claro que las actividades técnicas del ciclo de vida del software constan de un grado de complejidad considerable que se verá influenciado por el método de desarrollo seleccionado.

Dentro de la serie de actividades técnicas descritas, el trabajo realizado en este proyecto de tesis se ha enfocado principalmente en el análisis de requisitos, etapa que ha sido marcada con color verde en la Figura 1. Esto debido a que se busca identificar actividades de análisis de requisitos, derivadas en enfoques ágiles especialmente aquellas que sean de utilidad para el diseño de arquitectura.

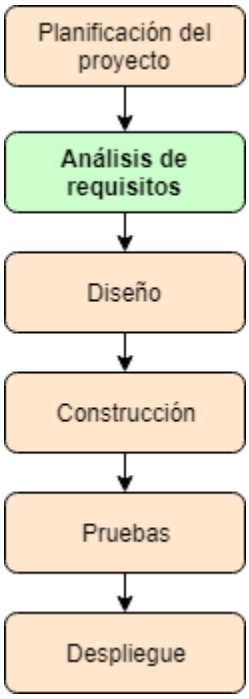

Figura 1. Actividades técnicas del desarrollo de software

1.2 Problemática

Desde la formulación del Manifiesto Ágil han surgido diferentes métodos de desarrollo basados en sus valores, principios, roles, prácticas y artefactos (Meyer, 2014). En este contexto, los métodos ágiles han ganado popularidad debido principalmente a la mejor capacidad de respuesta que proveen frente a las demandas de mercado. En relación a ello The Standish Group International (2016) indica en su reporte anual llamado *Chaos Report,* que el utilizar un método ágil de desarrollo de software reduce las posibilidades de terminar en un proyecto fallido del 27% al 9% respecto a los métodos tradicionales.

Por otro lado, Fairbanks (2010) menciona que una limitación de algunos métodos de desarrollo ágil, es que son poco prescriptivos en el proceso de desarrollo de la arquitectura de software. La arquitectura de software es un artefacto fundamental porque describe la estructura, funcionamiento e interacción entre los componentes de un sistema y, por lo tanto, guía la construcción del mismo. El proceso de desarrollo de la arquitectura de software depende fuertemente de los requisitos. Brown (2016) señala que los requisitos son fundamentales debido a que determinan el diseño de la arquitectura, pues lo que se debe lograr finalmente es una arquitectura que permita satisfacerlos. Fairbanks (2010) también señala que los diseñadores de arquitectura tienden a enfocar su atención en atributos de calidad y restricciones debido a que la naturaleza de la arquitectura tiene un gran impacto en atenderlos.

La falta de actividades explícitas sobre manejo de requisitos y desarrollo de la arquitectura de software en los métodos ágiles genera dificultades a los equipos de desarrollo para administrar sus requisitos de software considerando la información útil sin perder agilidad. Algunas de estas dificultades incluyen:

- La revisión de múltiples fuentes para obtener una visión general del manejo de requisitos para el diseño de arquitectura de software implica una cantidad considerable de tiempo y esfuerzo.
- La falta de experiencia en el desarrollo de requisitos y arquitectura de software por parte de los desarrolladores puede llevar a que requisitos con impacto en las decisiones de arquitectura no sean considerados en la construcción del producto de software.

- La asunción de los autores sobre el conocimiento del lector acerca de cuáles técnicas, herramientas y artefactos se ven involucrados en la ejecución de una actividad específica es un problema común al encontrar actividades técnicas relacionadas al manejo de requisitos.
- El encontrar actividades técnicas relacionadas al manejo de requisitos es relativamente sencillo de realizar, ya que existe gran variedad de información al respecto en la literatura. Sin embargo, a menudo resulta difícil identificar estas actividades dentro de una sección en documentos o libros de este tema, pues por lo general se encuentran dispersas en la documentación.
- La identificación de técnicas, herramientas y artefactos que complementen la ejecución de las actividades técnicas a realizar a menudo requiere que el lector investigue estos elementos por cuenta propia.

1.3 Objetivos del proyecto de tesis

1.3.1 Objetivo general

Proponer una guía de actividades, técnicas, herramientas y artefactos de ingeniería de requisitos que facilite a los equipos de desarrollo, la administración de los requisitos de software considerando la información útil para el desarrollo de arquitectura en entornos de desarrollo ágil.

1.3.2 Objetivos específicos

1. Identificar y analizar actividades relacionadas a ingeniería de requisitos sugeridas por los autores *J. Coplien y G. Bjørnvig, S. Brown y G. Fairbanks* en sus libros enfocados al desarrollo de arquitectura de software en entornos de desarrollo ágil:
    - Lean Architecture for Agile Software Development (2010)
    - Software Architecture for Developers (2016)
    - Just Enough Software Architecture (2010)
2. Clasificar y agrupar las actividades encontradas a partir del análisis de los tres enfoques revisados.
3. Proponer una guía que integre además de actividades, sugerencias de técnicas, herramientas y artefactos para complementar su aplicación.
4. Validar la guía realizada por medio de un caso de estudio.
5. Publicar la guía en un sitio web disponible en internet para facilitar su diseminación de manera digital.

1.4 Organización del documento de tesis

Las secciones restantes del documento se han estructurado de la siguiente manera:

En el capítulo 2, titulado *antecedentes*, se ha incluido además del marco teórico en el cual se definen términos importantes del contexto del proyecto, una visión general de los enfoques manejados para el desarrollo de arquitectura de los autores mencionados así como el método utilizado para compararlos.

En el capítulo 3, titulado *desarrollo de la guía de actividades, técnicas, herramientas y artefactos de ingeniería de requisitos,* se describen de manera detallada las etapas y acciones realizadas para construir la guía. Estas son: 1) Análisis, identificación y síntesis de la información; 2) Refinamiento de la información; 3) Priorización de la información y finalmente; 4) Diseño de la presentación de la información.

En el capítulo 4, titulado *Caso de estudio*, se describe paso a paso el desarrollo del caso de estudio mediante el cual se validó la priorización del contenido de la guía realizada en el capítulo 3. Esto, en colaboración con equipos de expertos y practicantes del desarrollo de software.

En el capítulo 5, destinado a integrar las *conclusiones* del proyecto, se resume como los objetivos del proyecto han sido resueltos, se han incluido reflexiones sobre el proceso de construcción de la guía, el caso de estudio y el producto obtenido. Finalmente se proponen algunas líneas de trabajo futuro que sería de interés explorar.

Por último, se han agregado las referencias bibliográficas y anexos del material utilizado para el desarrollo de este proyecto.

2. ANTECEDENTES

El propósito de este capítulo es proporcionar al lector la base teórica sobre la cual se desarrolla el proyecto. Para lograrlo, se ha dividido en dos secciones: la primera sección se compone del marco teórico, donde se describen los términos más relevantes que serán utilizados de manera recurrente en este documento. En la segunda sección se explican los enfoques de desarrollo de arquitectura de los autores que fueron analizados para la realización de la guía de actividades, técnicas, herramientas y artefactos de ingeniería de requisitos para el desarrollo ágil de arquitectura de software.

Los enfoques a analizar son: J. Coplien y G. Bjørnvig en *Lean Architecture for Agile Software Development* (2010), S. Brown en *Software Architecture for Developers* (2016) y G. Fairbanks en *Just Enough Software Architecture* (2010), mismos que fueron seleccionados debido a que intentan mostrar con diferentes perspectivas cómo el desarrollo de arquitectura puede ser combinado con métodos de desarrollo ágil. Para este proyecto en particular, el alcance dentro del ciclo de vida del software se encuentra principalmente en la etapa de requisitos de software, por lo cual dentro de los enfoques se buscó por actividades relacionadas al área de ingeniería de requisitos. Finalmente, se ha incluido una descripción del método tomado como referencia para comparar los tres enfoques mencionados.

2.1 Marco Teórico

2.1.1 Desarrollo ágil

En la actualidad muchas organizaciones dentro de la industria del software se han unido a la corriente del desarrollo ágil con la finalidad de adaptarse a las necesidades cambiantes de la industria en general. Debido, entre otros aspectos, a sus principios sobre entrega de valor y obtención de retroalimentación constantes durante el desarrollo. Moya (2016: 202) define al desarrollo ágil de la siguiente manera:

- El desarrollo ágil de software es una filosofía que trata de reducir al mínimo la burocracia típica de las metodologías de desarrollo tradicionales. *Se basa en la idea de que el software que funciona es la principal medida del progreso.* El desarrollo ágil recoge la herencia de varias corrientes plasmadas a finales de los años 90 en el Manifiesto Ágil.

2.1.2 Manifiesto ágil

El Manifiesto Ágil (2001) es una colección de valores y principios a partir de los cuales se han diseñado diferentes métodos de desarrollo ágil que proponen marcos de trabajo bien definidos. Estos valores y principios se describen a continuación:

Valores

- Individuos e interacciones sobre procesos y herramientas.
- Software funcional sobre documentación comprensiva.
- Colaboración con el cliente sobre negociación de contratos.
- Responder al cambio sobre seguir un plan.

Principios

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

2.1.3 Ingeniería de requisitos

La ingeniería de requisitos es un proceso medular del desarrollo de sistemas en el que la interacción con los involucrados y la manera en que sus necesidades son interpretadas como requisitos de software proporciona un punto de partida que sirve como referencia sobre el producto final deseado y las características que debe cumplir. Sommerville (2005) la define de la siguiente manera:

- La Ingeniería de Requisitos es el proceso de comprensión y definición de qué servicios se requieren del sistema y de identificación de las restricciones del funcionamiento y desarrollo del mismo. La ingeniería de requisitos es una etapa particularmente crítica en el proceso del software ya que los errores en esta etapa originan inevitablemente problemas posteriores en el diseño e implementación del sistema.

Este proceso tiene una serie de etapas que son obtención y análisis, especificación y validación de requisitos, mismas que pueden realizarse de manera incremental como se muestra en la Figura 2:

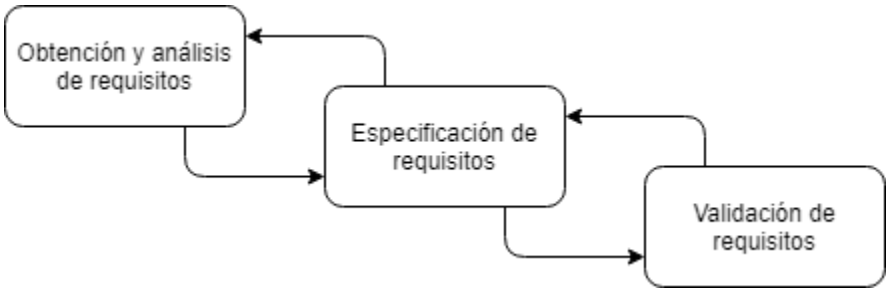

Figura 2. El proceso de la ingeniería de requisitos. Adaptado de "Ingeniería del Software" por Sommerville, 2005, p. 70. Derechos reservados Pearson Education

En las siguientes secciones se explican cada una de estas etapas.

2.1.3.1 Obtención y análisis de requisitos

Según Sommerville (2005) la obtención y análisis de requisitos es un proceso que se compone por cuatro fases, que pueden ser iterativas e incrementales, en las cuales los requisitos:

- Se descubren
- Se clasifican y organizan
- Se ordenan por prioridades y se negocian
- Finalmente, se documentan de manera formal o informal

Al finalizar esta etapa se obtendrá un modelo del sistema, el cual es una representación simplificada del sistema que ayudará a comprender su comportamiento y validar las

asunciones realizadas sobre la funcionalidad que debe proporcionar con los usuarios finales e involucrados.

2.1.3.2 Especificación de requisitos

Somerville (2005) describe la especificación de requisitos como la actividad de traducir la información recopilada durante la actividad de análisis en un documento que define un conjunto de requisitos. Este documento debe ser formal y servirá como guía durante el desarrollo para la evaluación de la implementación de los requisitos del cliente en el producto de software construido.

El ejemplo más común de documento de especificación, es el llamado Especificación de Requisitos de Software o SRS *(Software Requirements Specification)* que provee lineamientos bien definidos sobre el contenido que debe incluir. Su similar en entornos ágiles son generalmente historias de usuario, sin embargo es recomendable en cualquier entorno contar con un documento que sirva como punto de partida para la definición del producto esperado y que además sea actualizado durante el proyecto.

2.1.3.3 Validación de requisitos

La validación de los requisitos consiste en verificar si estos realmente reflejan lo que el cliente necesita. Somerville (2005: 70) menciona que en la validación inevitablemente se descubren errores en el documento de requisitos. Por lo tanto, la modificación de este documento para corregir los problemas encontrados forma parte de esta etapa.

Existen diversas técnicas de validación que ayudarán a mitigar re-trabajo futuro revisando la validez, consistencia, completitud y realismo de los requisitos a través de las técnicas sugeridas por Soria y Condor (2016) como:

- Revisiones de requisitos con los involucrados
- Construcción de prototipos
- Generación de casos de prueba

2.1.4 Arquitectura de software

El término arquitectura de software puede ser utilizado de manera indistinta para referirse al proceso seguido para su construcción y al artefacto de software resultante. A continuación se define en ambos contextos:

El artefacto. La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema. Comprende elementos de software, relaciones entre ellos y propiedades de ambos. (Bass, Clements y Kazman, 2012).

El proceso. Es entender lo que se necesita construir, crear una visión para construirlo y tomar las decisiones de diseño apropiadas. Todo esto debe basarse en los requisitos, ya que impulsan la arquitectura. También es crucial comunicar esa visión e introducir el liderazgo técnico para que todos los involucrados en la construcción del producto entiendan la visión y puedan contribuir de manera positiva a su éxito (Brown, 2016).

2.1.5 Actividad

Una **actividad** según el International Standard ISO/IEC 12207 (2008) es un conjunto de tareas cohesivas de un proceso. Las **tareas** son requisitos, recomendaciones o acciones permisibles que pretenden contribuir al logro de uno o más resultados de un proceso. En el contexto de este proyecto, las actividades estudiadas están relacionadas principalmente a las etapas de análisis de requisitos y diseño del desarrollo de sistemas de software.

- La obtención de atributos de calidad durante la etapa de requisitos es un ejemplo de una actividad que proporcionará ciertos artefactos que serán de necesarios durante el diseño.

2.1.6 Técnica

La **técnica** es un conjunto de procedimientos reglamentados y pautas que se utiliza como medio para llegar a un cierto fin. La técnica supone que, en situaciones similares, repetir conductas o llevar a cabo un mismo procedimiento producirán el mismo efecto (Pérez y Merino, 2008). En el contexto de ingeniería de software han surgido técnicas ligadas a lenguajes de programación, profundización en los estudios en ingeniería de software; arquitectura de software y herramientas CASE (*Computer-aided software engineering*) (Noriega, 2017).

- Retomando el ejemplo anterior de la obtención de atributos de calidad, los talleres de atributos de calidad o *Quality Attribute Workshops* (Software Engineering Institute, 2017), son un ejemplo de una técnica que se puede utilizar para llevar a cabo la actividad.

2.1.7 Herramienta

La **herramienta** es un instrumento que permite realizar ciertos trabajos (...) más allá del objeto físico. El concepto de herramienta también se utiliza para nombrar a cualquier procedimiento que mejora la capacidad de realizar ciertas tareas (Pérez y Merino, 2010). En el contexto de ingeniería de software las herramientas *CASE* tienen como objetivo auxiliar en el proceso de creación de sistemas de software. Existen diversas herramientas de este tipo para la planificación, edición, prototipado, prueba, documentación, etc. (Árias, 2015).

- En el contexto de la actividad de obtención de atributos de calidad, los formatos predefinidos para llenar los escenarios serían un ejemplo de una herramienta con la cual se podría complementar la ejecución del taller de atributos de calidad.

2.1.8 Artefacto

El **Artefacto** es cualquier pieza de información usada o producida al desarrollar sistemas. Podría ser un diagrama, texto descriptivo, instrucciones de usuario, métodos del código, programas o cualquier otro componente del sistema (Kendall y Kendall, 2005).

- Siguiendo el ejemplo anterior, los artefactos obtenidos como producto final de la actividad de obtención de atributos de calidad serían los escenarios de atributos de calidad.

2.2 Análisis de tres enfoques de desarrollo ágil de arquitectura

En esta sección se presenta cada uno de los enfoques de desarrollo de arquitectura de software propuestos por los autores: J. Fairbanks con *Just Enough Software Architecture*, S. Brown con *Software Architecture for Developers* y finalmente, J. Coplien y G. Bjørnvig con *Software Architecture for Agile Software Development* los cuales serán analizados en el capítulo 3.

2.2.1 Enfoque 1: Modelo general de desarrollo de arquitectura dirigido por riesgos

Fairbanks presenta un enfoque de desarrollo de arquitectura dirigido por riesgos, en el cual propone tres etapas bien definidas que han de llevarse a cabo de manera iterativa para su ejecución efectiva (véase Figura 3). Este enfoque ha sido llamado *M**odelo de Desarrollo de Arquitectura Dirigido por Riesgos*** y con él se pretende que los desarrolladores de software conozcan un conjunto mínimo de técnicas para ayudar a reducir los riesgos dentro de un proyecto.

Las actividades del modelo propuesto son: 1) ***identificar y priorizar los riesgos***, que preferentemente se debería realizar de manera grupal. Una vez realizado lo anterior, el equipo debe 2) ***seleccionar y aplicar un conjunto de técnicas*** que permitan mitigar estos riesgos, para finalmente 3) ***evaluar la reducción del riesgo****.* Con estas actividades se intenta responder a las interrogantes ¿cuáles son los riesgos del proyecto?, ¿cuáles son

las mejores técnicas para reducirlos? y si ¿ha sido mitigado el riesgo y se puede iniciar el trabajo de desarrollo?.

Un factor importante para el éxito de un proyecto donde la arquitectura es decisiva, es la alineación de las actividades relacionadas a su construcción con las técnicas a aplicar para mitigar los riesgos. Por ejemplo, la aplicación de técnicas para selección de las tecnologías adecuadas que soporten ciertos atributos de calidad. Debido al ambiente técnico en el que se desenvuelven los desarrolladores, estos pueden identificar más rápidamente riesgos relacionados al desarrollo. Sin embargo, se debe tener presente que existen dos tipos de riesgos principales en los proyectos de software: riesgos de ingeniería y riesgos relacionados a la administración de proyectos.

Actualmente el enfoque de desarrollo de arquitectura a la medida es el más común, sin embargo es subjetivo debido a que la selección de técnicas utilizadas depende puramente del equipo de desarrollo. No obstante, el enfoque basado en riesgos soporta la construcción rápida de software con alta calidad a un costo razonable. Finalmente, la idea principal del modelo es que los desarrolladores de software puedan contestar a la pregunta: *¿cuáles son los principales riesgos del proyecto y cuáles son las técnicas de ingeniería para correspondientes para mitigarlos?*.

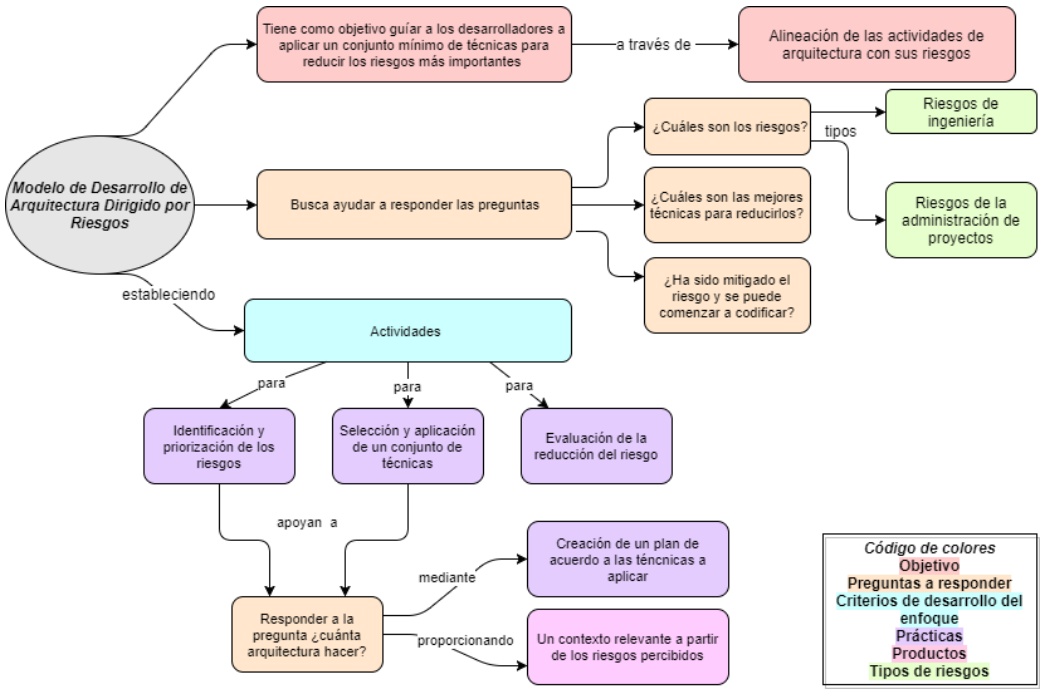

Figura 3. Esquema general del modelo de desarrollo de arquitectura dirigido por riesgos

2.2.2 Enfoque 2: Fundamentos para el desarrollo de arquitectura de software

Brown propone un enfoque para el desarrollo de arquitectura basado en contar con firmes fundamentos de la ***estructura*, *visión* y *riesgos* del proyecto**. El enfoque que presenta busca que los desarrolladores de software se centren en los riesgos para construir un fundamento sólido de arquitectura de software con el cual trabajar. Para lograrlo, el autor responde dos preguntas sobre estos tres factores: **el *qué hacer* y *cómo hacerlo*** (véase Figura 4).

- Respecto a la *estructura*, el qué es respondido al lograr el entendimiento de los elementos estructurales significativos y cómo estos encajan juntos basados en los drivers arquitecturales[1], y el cómo, implica la realización del diseño y descomposición hasta llegar a componentes de software.
- Por otro lado, la visión responde a su qué, cuando se crea y comunica una visión que permite al equipo tener una base con la cual trabajar y su cómo, es la visualización de la arquitectura utilizando una colección de diagramas a diferentes niveles de abstracción.
- Finalmente, el qué de los riesgos, es la identificación y mitigación de aquellos que se hayan identificado con mayor prioridad y el cómo, aunque depende del contexto, puede ser llevado a cabo utilizando técnicas como la lluvia de ideas sobre riesgos por mencionar un ejemplo.

Fairbanks (2010) menciona que si se cumple el conjunto de actividades anterior que responden al cómo de la estructura, visión y riesgos, se contará con fundamentos firmes para soportar el resto de entregas de software ya que se tendrá una visión del producto en construcción. Además, Fairbanks también señala que es importante reconocer que algunas arquitecturas necesitan ser realizadas por adelantado, otras en cambio pueden evolucionar naturalmente, sin embargo depende del contexto. Resume finalmente que el decidir dónde

1 Los drivers arquitecturales son aquellos requisitos que tienen mayor influencia respecto de la forma que tomarán los elementos que componen las estructuras arquitectónicas (Cervantes et. al., 2015: 15).

se encuentra la línea entre lo que debe tenerse por adelantado y el diseño evolutivo es la clave.

La identificación de elementos arquitecturalmente significativos y los riesgos correspondientes deberían ser realizados en todos los proyectos de software sin importar la metodología aplicada. Sin embargo, no existe una receta exacta sobre qué técnicas aplicar ya que en los proyectos de software se ven involucrados muchos factores. Por esta razón se debe realizar lo que sea que funcione basado en el propio contexto en que se desarrolla.

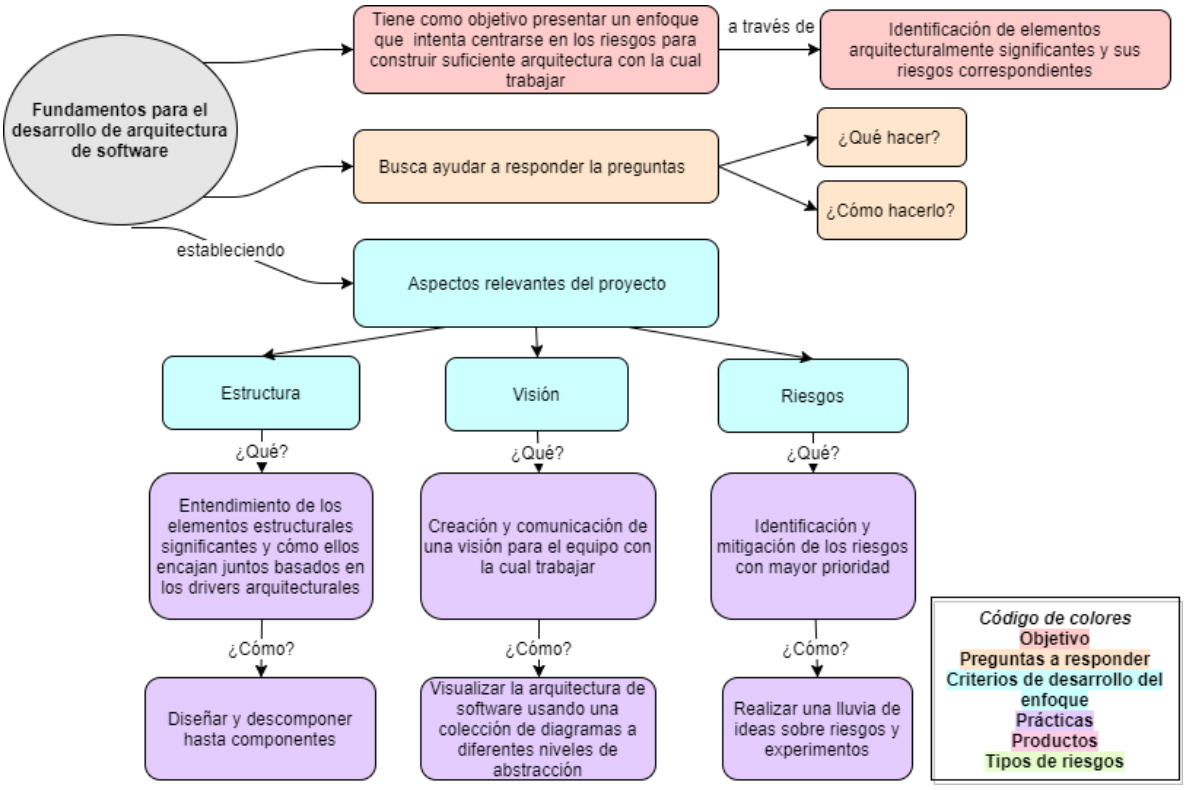

Figura 4. Fundamentos para el desarrollo de arquitectura de software

2.2.3 Enfoque 3: Producción ágil de arquitectura

Coplien y Bjørnvig sugieren una serie de actividades relacionadas al desarrollo de arquitectura de software que deben realizarse durante un proyecto, menciona porqué es importante desarrollar arquitectura de software y cuáles deberían ser los artefactos obtenidos al final del proceso de su desarrollo. ***Las razones principales por las que se realiza el trabajo de arquitectura es para capturar las perspectivas de los involucrados que afectan el diseño***, para responder al cambio de manera efectiva, crear una visión compartida a través del equipo e involucrados así como apoyar al proceso de toma de decisiones.

Las actividades sugeridas necesitan el compromiso de los involucrados, definición del problema, *lo que el sistema es*, es decir la consideración de actividades relacionadas a la construcción de las estructuras que lo conforman y así como la construcción de *lo que el sistema hace* o los servicios proveídos por este. Las actividades sugeridas pueden darse en cualquier momento durante el desarrollo dado que no han sido indicados como pasos en secuencia (véase Figura 5).

El compromiso con los involucrados se refiere a la identificación de la gente y sistemas a los que les importa que el sistema construido exista, lo que hace o cómo lo hace, además de mantener el contacto y cooperación de los usuarios finales durante el desarrollo. Los principales roles que pueden ser encontrados en todas las fases del desarrollo son: *usuarios finales, gente del negocio, clientes, expertos de dominio y desarrolladores*. Es importante notar que todos los involucrados obtienen valor del producto desarrollado.

La definición del problema fomenta que el equipo se reúna y escriba una corta y concisa descripción de manera temprana en el proyecto, la cual debe ser actualizada periódicamente y funciona como catalizador para la auto organización de equipos. La importancia de la definición del problema radica en que sin una, es difícil saber cuándo el trabajo de desarrollo ha terminado. Para crear una definición de manera sencilla basta con pensar en la diferencia entre el estado actual y el estado deseado al que se quiere llegar con el software a construir.

La construcción de la arquitectura o *lo que el sistema es*, trata lo que comúnmente es llamado plataforma de arquitectura, que refleja las estructuras estables del negocio a través del tiempo. Entre las actividades recomendadas se encuentran el particionar el diseño estático del dinámico, sin embargo estas actividades se realizan de forma paralela pues no existe una división explícita entre ambas. Luego, se debe seguir particionando hasta obtener elementos que pueden ser manejados de manera tan autónoma como sea posible. Los involucrados claves en esta parte del sistema son *expertos de dominio, arquitectos de software y la gente del negocio.*

La construcción de lo que el sistema hace, trata la vista de usuario sobre los servicios proveídos por el sistema, las tareas que el sistema puede realizar y la forma en que estas tareas son estructuradas. Las actividades principales son: convertir los casos de uso a algoritmos, es decir, código que se pueda inspeccionar, analizar y probar, escribir pruebas de sistema, codificar algoritmos y soportar la lógica del dominio refactorizando en el camino y finalmente, correr las pruebas nuevas al código. Los involucrados claves en esta parte son: *usuarios finales, diseñadores de interfaz y encargados de requisitos.* Así, al final del desarrollo se obtendrá software corriendo, probado, código documentado y pruebas.

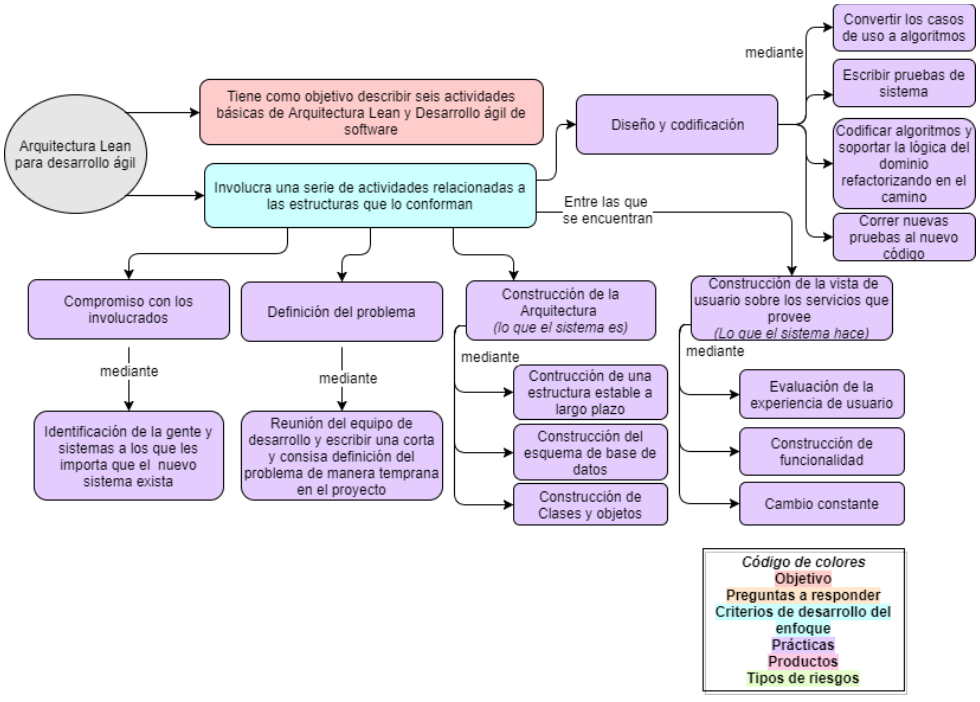

Figura 5. Elementos relevantes en el desarrollo de arquitectura

2.3 Modelo general de diseño de arquitectura derivado de cinco enfoques industriales

En esta sección es descrito de manera general el método utilizado en este proyecto de tesis para comparar los tres enfoques mencionados en la sección anterior, y de los cuales se extrajeron actividades, técnicas, herramientas y artefactos para formar la guía. Este método se basa en el trabajo de los autores Hofmeister et al. (2007). Los autores evaluaron cinco métodos de diseño de arquitectura de software de los que obtuvieron un modelo general de diseño de arquitectura y de la comparación se derivó una tabla que se propone, pueda ser utilizada para otras comparaciones. La Tabla 1 se tomó como punto de referencia para realizar el análisis de este proyecto y se muestra a continuación:

Tabla 1. Tabla para analizar métodos de diseño de arquitectura de software

Artefactos en x	Actividades X	Técnicas y herramientas en X
¿Cómo son llamados y representados los artefactos?	¿Cómo son las actividades nombradas y documentadas?	¿Cuáles son las técnicas y herramientas específicas asociadas al método x?
Artefactos clave en el método X no coincidente con los otros	Otras actividades clave no encajando en otros apartados	---

La tabla anterior puede ser utilizada para analizar cada método por separado. Una vez realizado el análisis individual, se puede realizar el análisis entre métodos utilizando como referencia la Tabla 2. Esta tabla muestra una comparación entre las actividades de cada método.

Tabla 2. Comparando métodos: Actividades

Actividad genérica	Método A	Método B	Método C	Método D	Método E
Nombre de la actividad genérica	Actividad del método A	Actividad del método B	Sin Actividad	Actividad del método D	Actividad del método E
---	---	---	---	---	---

De esta forma los autores analizaron los métodos de diseño individualmente y entre métodos por actividades, artefactos, técnicas y herramientas, lo cual permitió identificar de manera clara las similitudes y diferencias entre cada uno. La Figura 6 intenta resumir el trabajo realizado por los autores:

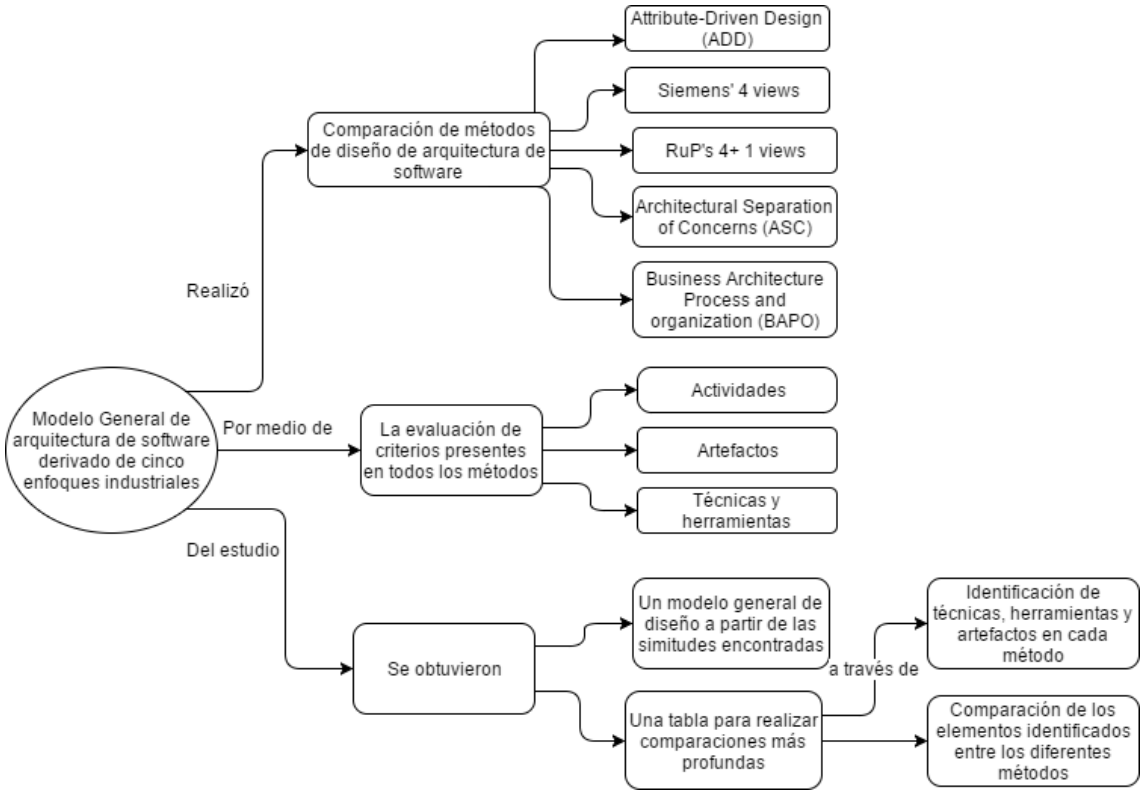

Figura 6. Características del modelo general de arquitectura de software derivado de cinco enfoques industriales

3. DESARROLLO DE LA GUÍA DE ACTIVIDADES, TÉCNICAS, HERRAMIENTAS Y ARTEFACTOS DE INGENIERÍA DE REQUISITOS

En este capítulo se aborda el proceso seguido para la generación de la guía de actividades, técnicas, herramientas y artefactos de ingeniería de requisitos útiles para el desarrollo de arquitectura derivada de tres enfoques ágiles. La Figura 7 describe el proceso que consistió en 4 etapas. En las siguientes secciones son descritas a detalle.

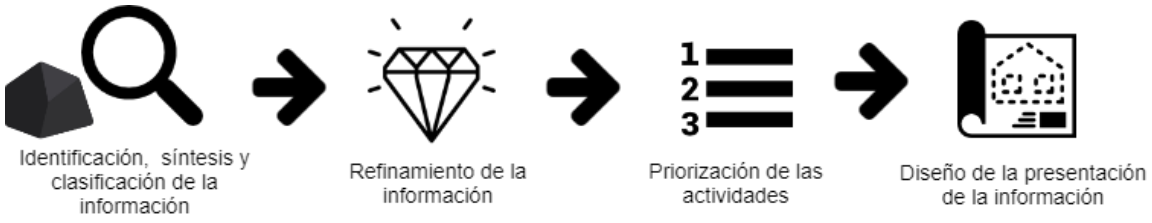

Figura 7. Etapas del proceso realizado para la generación de la guía

3.1 Identificación, síntesis y clasificación de la información

El propósito de esta etapa es la identificación, síntesis y clasificación de actividades, técnicas, herramientas y artefactos para el contenido de la guía a desarrollar; este material fue obtenido de las fuentes:

- J. Coplien y G. Bjørnvig en Lean Architecture for Agile Software Development (2010)
- S. Brown en Software Architecture for Developers (2016)
- G. Fairbanks en Just Enough Software Architecture (2010)

En la sección 2.2 del capítulo 2, titulada *Análisis de tres enfoques de desarrollo ágil de arquitectura de software,* se describe de manera general el enfoque de desarrollo de arquitectura de software de cada autor.

El primer paso fue la *identificación* de los elementos que conforman el contenido de la guía. Posteriormente se sintetizó y clasificó la información tomando como referencia el método de comparación utilizado en el trabajo *Modelo general de diseño de arquitectura de software derivado de cinco enfoques industriales,* descrito en la sección 2.3 del capítulo 2. Específicamente, se adaptó la matriz de comparación propuesta para incluir los siguientes datos:

- Número para identificar la actividad
- Descripción de la actividad
- Técnicas y herramientas mencionadas por el autor para complementar la actividad
- Artefactos resultantes o productos esperados de la actividad
- Finalmente se agregó el número de página o páginas de libro en las que se encontró la información

Una vez completada la matriz se identificaron actividades similares y se unieron en una sola actividad o renglón de la matriz, con la finalidad de evitar incluir actividades repetidas. Debido a que en primera instancia se intentó realizar un análisis de las fuentes de manera individual se requirió generar una matriz por cada una.

La Tabla 3 muestra un fragmento de la matriz resultante del análisis del libro *Software Architecture for Developers,* que contiene actividades, técnicas, herramientas y artefactos.

El contenido completo de las matrices resultantes del análisis de las tres fuentes se puede consultar en los Anexos A, B y C.

Tabla 3. Ejemplos de actividades, técnicas, herramientas y artefactos obtenidos del análisis del libro Software Architecture for Developers

No.	Actividad	Técnicas y herramientas	Artefactos	Página de libro
2	Obtener la lista de deseos de los involucrados acerca de lo que necesitan que el sistema haga	Historias de usuario, casos de uso, especificaciones de requisitos tradicionales, criterios de aceptación, etc.	Lista de deseos	55
4	Priorizar requisitos funcionales, historias de usuario, etc.	Uso de la técnica *MosCow* para priorizar	Historias de usuario	50
8	Refinar la información obtenida sobre los atributos de calidad asociando por ejemplo cantidades a los mismos, evitando vaguedad en su definición	Preguntas como las siguientes pueden ayudar: ¿Cuántos usuarios concurrentes debería el sistema soportar en promedio?, ¿Cómo intervienen las horas pico?, ¿Qué tiempo de respuesta es considerado aceptable?, ¿El tiempo de respuesta debe ser el mismo en todas las partes del software o sólo para características específicas? además de preguntar sobre el número de usuarios concurrentes, volúmenes de datos y número máximo de respuestas	* Conjunto específico de requisitos no funcionales, idealmente que se pueden cuantificar explícitamente. * Criterios de aceptación para realizar pruebas posteriormente	59,60
10	Revisar si existen otros sistemas que se necesiten integrar con el sistema que se va a construir	Revisión de restricciones organizacionales dictando protocolos y tecnologías a utilizar para los puntos de integración	No especificado	62

Una vez que se obtuvo la matriz por cada autor, se creó una nueva matriz para comparar las actividades, técnicas, herramientas y artefactos mencionados por los tres autores. Esta nueva matriz se adaptó para incluir los siguientes datos, que facilitan la descripción de actividades complementarias y equivalentes entre autores:

- **Actividad.** Describe en una oración un conjunto de tareas cohesivas.
- **Actividad genérica**. Engloba las perspectivas de los autores en una misma oración. En ella se reformuló la definición de una actividad si fue mencionada por más de un autor, en caso contrario permaneció sin cambio.

- **Actividad genérica.** Describe el porqué de la acción (su objetivo). Esto es la razón de llevar a cabo una actividad.
- **Tipo.** Indica si la actividad es **complementaria** o **equivalente**. Una actividad es señalada como complementaria cuando sólo ha sido mencionada por un autor y equivalente al conjunto cuando ha sido mencionada por dos o más.

La Tabla 4 muestra un fragmento de la matriz descrita anteriormente.

Tabla 4. Ejemplos de identificación de actividades complementarias y equivalentes entre autores

Actividad			**Actividad genérica (explicando el qué)**	**Actividad genérica (explicando el qué y el porqué)**	**Tipo**
Software Architecture for Developers - Brown	**Lean Architecture for Agile Software Development - Coplien**	**Just Enough Software Architecture - Fairbanks**			
Obtener la lista de deseos de los involucrados acerca de lo que necesitan que el sistema haga	No encontrada	No encontrada	Obtener la lista de deseos de los involucradas	Obtener la lista de deseos de los involucrados para conocer lo que necesitan que el sistema haga	Complementaria
Elaborar una lista de características o historias de usuario	Iniciar la obtención de historias de usuario y proceder a los casos de uso dirigidos por metas, incluyendo escenarios y motivaciones del usuario para la funcionalidad Utilizar narrativas de los que sucede antes y después de una acción para entender la motivación del usuario o del negocio	Utilizar escenarios de funcionalidad y casos de uso para expresar comportamiento del sistema	Incluir lo que sucede antes y después de cada acción en las historias de usuario	Incluir lo que sucede antes y después de cada acción en las historias de usuario para entender la motivación del usuario o del negocio	Equivalente
Priorizar requisitos funcionales, historias de usuario, etc.	Organizar las historias de usuario con una lista ordenada basada en las relaciones entre sí, su costo, el valor del negocio y el “time to market”	No encontrada	Priorizar los requisitos	Priorizar los requisitos para identificar aquello que aporten mayor valor al negocio	Equivalente

Utilizando la matriz anterior se ordenaron las actividades y se clasificaron en un conjunto de 36 actividades, las cuales han sido materia prima para el refinamiento de la información. La matriz completa puede consultarse en el Anexo D.

3.2 Refinamiento de la información

El propósito de la etapa de refinamiento es la reformulación de la redacción de las actividades para mejorar su lectura. Igualmente, se buscó proporcionar a las oraciones que describen las actividades un sentido de lista de verificación y definir los elementos complementarios de cada actividad. Esta lista será el insumo del caso de estudio, que se explicará en el próximo capítulo.

A continuación se describen los elementos que complementan cada actividad ilustrados con un ejemplo, mismos que forman parte de una matriz construida para concentrar los elementos encontrados al analizar los diferentes autores. Un segmento de esta matriz es mostrada en la Tabla 5:

- **Reformulación de la definición de la actividad**. Esto, para que su lectura sea más orientada a una lista de verificación. Ejemplo:
    - Descripción original de la actividad: Elaborar una definición de problema corta y nítida.
    - Descripción de la actividad reformulada: La definición del problema elaborada es corta y nítida.
- **Especificación del objetivo de la actividad.** Ejemplo:
    - La definición del problema ayudará a saber cuándo el trabajo de desarrollo ha terminado.
- **Identificación del objeto de la actividad.** Es decir, el elemento al cual se refiere. Ejemplo:
    - Objeto de la actividad "elaborar una definición del problema corta y nítida" sería el "problema".

Esta acción se realizó para cada una de las 36 actividades y permitió agruparlas por su relación a: *atributos de calidad*, *riesgos*, *restricciones*, *requisitos emergentes* y *usuarios*.

A partir de la identificación del objeto de la actividad, la distribución de las mismas dentro de los grupos mencionadas fue la siguiente:

- 6 actividades relacionadas con atributos de calidad
- 5 actividades relacionadas a riesgos en el proyecto
- 4 actividades relacionadas a restricciones del proyecto
- 2 actividades relacionadas a requisitos emergentes
- 2 actividades relacionadas a los usuarios del sistema
- 17 actividades sin grupo. Este conjunto quedó de esta forma debido a que todas presentaron diferentes objetos de la actividad sin relación directa en común.

Según la definición de grupos anterior, la actividad abordada en este ejemplo quedaría marcada "*sin grupo"*.

- **Identificación de Categoría.** Identificación de la etapa del proceso de ingeniería de requisitos a la cual pertenece la actividad. Estas etapas fueron definidas en el capítulo 2 y son: *obtención y análisis*, *especificación* y *validación de requisitos*.
  Ejemplo: La categoría identificada de la actividad "la definición del problema elaborada es corta y nítida" es *obtención y análisis de requisitos.*
- **Identificación de rol o roles involucrados.** Identificación del rol o roles que se ven involucrados en la ejecución de la actividad. Tomando como referencia el marco ágil Scrum (Schwaber y Sutherland, 2016) estos roles pueden ser: Scrum master, Dueño del producto y el Equipo de desarrollo.
  Ejemplo: Los roles identificados en la actividad "la definición del problema elaborada es corta y nítida" son; Scrum master, Dueño del producto y Equipo de desarrollo.
- **Identificación de la etapa**. Indica la etapa del ciclo de vida en que podría ejecutarse la actividad. En el ámbito de este proyecto las etapas que se han identificado son: *Análisis de requisitos*, *diseño*, *construcción* y *pruebas*.
  Con la definición de la etapa se hace notorio que actividades de análisis y obtención, especificación y validación de requisitos pueden llevarse a cabo en diferentes momentos del desarrollo de software.
  Ejemplo: La etapa asociada a la actividad "la definición del problema elaborada es corta y nítida" es análisis de requisitos.

- **Identificación de técnicas y herramientas**. Se separaron técnicas de herramientas; y dentro de ellas, elementos que fueron identificados como consejos propuestos por los autores fueron separados a otra columna llamada "**Consejos**". No todas las actividades presentaron esta última característica.

  Ejemplo: La técnica asociada a la actividad "la definición del problema elaborada es corta y nítida" es la definición del problema de Jerry Weinberg: consiste en escribir la diferencia entre el estado actual y el estado deseado. Ver *"Are Your Lights On?: A Treatise on the Definition of Diverse Problems"* (1977).

  La herramienta asociada a la actividad "la definición del problema elaborada es corta y nítida" es el uso de procesadores de texto.

  No se identificaron consejos propuestos por los autores para la ejecución de esta actividad.
- **Artefactos**. Por último se incluyeron los artefactos esperados de la ejecución de cada actividad.

  Todas estas acciones fueron realizadas para el conjunto de las **36 actividades.** Como se mencionó anteriormente, la Tabla 5 mostrada a continuación, ilustra un fragmento de la matriz resultante. El contenido completo de la matriz se encuentra en el Anexo E.

Tabla 5. Fragmento de la matriz realizada para el refinamiento de la información

Elementos de la actividad	Ejemplo de actividad 1	Ejemplo de actividad 2
Actividad genérica (explicando sólo el qué)	Elaborar narrativas para explorar la funcionalidad del sistema, incluyendo lo que sucede antes y después de cada acción	Priorizar las historias de usuario
Reformulación de la actividad para la lista de verificación	Narrativas para explorar la funcionalidad del sistema han sido elaboradas, incluyendo lo que sucede antes y después de cada acción	Las historias de usuario han sido priorizadas
Objetivo	Entendimiento de la motivación del usuario o del negocio para una funcionalidad dada	Identificación de aquellas historias de usuario que aporten mayor valor al negocio
Objeto de la actividad	Narrativas	Historias de usuario
Categoría (etapa del proceso de ingeniería de requisitos)	Especificación	Validación
Rol	Equipo de desarrollo	Dueño del producto + equipo de desarrollo
Etapa del ciclo de vida	Requisitos	Requisitos
Técnicas	Especificaciones de casos de uso	Técnicas de priorización: * Mapa de historias de usuario o *Story Mapping* * Juego del Árbol del Producto o Prune the Product Tree…
Consejos	----	----
Herramientas	Plantillas para la especificación de casos de uso	Mapa de historias de usuario, Plantilla de árbol, Herramientas de software como: * idea engine (comercial), decision engine (comercial) , TransparentChoice's AHP online tool (comercial), easyAHP (libre)
Artefactos	Documentación de las especificaciones de casos de uso	Conjunto de requisitos o historias de usuario priorizadas

Una vez realizado el refinamiento de la información, esta se encontraba preparada para la siguiente etapa que se centró en la priorización de las actividades.

3.3 Priorización de las actividades

Esta etapa se centró en realizar la priorización de las 36 actividades que conformarán el contenido de la guía. Para lograrlo, se utilizó la herramienta de Software Libre *Wiki Surveys*. A continuación se describen aspectos relacionados con la aplicación de la encuesta así como los resultados obtenidos.

3.3.1 Wiki Surveys: Herramienta para la recopilación de datos sociales abiertos y cuantificables

Esta herramienta permite realizar encuestas personalizadas en línea con la particularidad de mostrar al usuario pares de elementos obtenidos de un banco de ideas para priorizarlas. Estas ideas son mostradas de manera aleatoria, y finalmente a partir de los votos de todos los usuarios se obtiene una priorización. En los siguientes párrafos se describe cómo realiza el análisis de datos de manera interna la herramienta según los autores Salganik y Levy (2015):

Análisis de los datos

Los datos son analizados en dos principales pasos. Primero, se utilizan las respuestas obtenidas para estimar la matriz de opinión Θ que incluye una estimación de cuánto valora la persona que responde cada elemento. Posteriormente, se resume la matriz de opinión para producir una calificación para cada elemento que estima la probabilidad de que el elemento sea aleatoriamente escogido para un encuestado también aleatoriamente seleccionado. Porque este análisis es modular, cualquiera, tanto la estimación y la forma de resumir pueden ser mejorados de manera independiente.

Estimando la matriz de opinión

El análisis empieza con un conjunto de respuestas de comparación por pares que están anidadas dentro de los encuestados. Por ejemplo la Figura 8 muestra cinco respuestas hipotéticas de dos encuestados. Estas respuestas son utilizadas para estimar la matriz de opinión. En este caso, el elemento $\Theta_{1,1}$ hace referencia a la primera respuesta, el elemento $\Theta_{1,2}$ a la segunda, así sucesivamente hasta llegar a $\Theta_{1,k}$ como respuesta a la última pregunta.

$$\Theta = \begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \dots & \theta_{1,K} \\ \theta_{2,1} & \theta_{2,2} & \dots & \theta_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{J,1} & \theta_{J,2} & \dots & \theta_{J,K} \end{bmatrix}$$

Figura 8. Matriz de opinión

La cual tiene una línea por cada encuestado y una columna por cada elemento, donde $\Theta_{j,k}$ es la cantidad que cada encuestado $_j$ valora el elemento $_k$ (o la cantidad que cada encuestado $_j$ cree que el elemento $_k$ responde la pregunta siendo realizada).

Tres características de los datos de respuesta complican el proceso de estimar la matriz de opinión Θ. Primero debido a que *wiki survey* es codicioso, se tiene un desigual número de respuestas por cada encuestado. Segundo, porque *wiki survey* es colaborativo, hay algunos elementos que pueden no ser presentados para algunos encuestados. Por ejemplo, si el usuario $_j$ contribuye con un elemento, ninguno de los encuestados previos podría haber visto ese elemento (En el caso de este proyecto de tesis la funcionalidad asociada a este problema no será utilizada). El tercer problema es que las respuestas son presentadas en forma de pares de comparación, lo cual significa que se puede observar solamente una preferencia relativa sobre dos elementos, no el sentimiento absoluto del usuario acerca de cada elemento.

Para manejar estos tres retos, se ha propuesto un modelo estadístico que asume que las respuestas de los encuestados reflejan sus preferencias relativas entre los elementos y que la distribución de preferencias a través de los encuestados por cada elemento sigue una distribución normal. Dadas estas asunciones, se puede realizar una inferencia Bayesiana para estimar la $\Theta_{j,k}$ que son más consistentes con las respuestas que se observaron y las asunciones que han sido realizadas. Una característica importante de esta estrategia de modelado es que para aquellos que contribuyen con muchas respuestas, se puede estimar mejor su renglón en la matriz de opinión, y para aquellos quienes contribuyen con menos respuestas, se tiene que depender más de la puesta en común de información de otros encuestados.

Las formas específicas funcionales que se asumen, resultan en la distribución mostrada en la ecuación representada en la Figura 9, que se asemeja a un modelo jerárquico:

$$p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid \boldsymbol{Y}, \boldsymbol{X}, \sigma, \boldsymbol{\mu_0}, \boldsymbol{\tau_0^2}) \quad \propto \quad \prod_{i=1}^{V} \Phi(\boldsymbol{x}_i^T \boldsymbol{\theta})^{y_i} (1 - \Phi(\boldsymbol{x}_i^T \boldsymbol{\theta}))^{1-y_i} \times \prod_{j=1}^{J} \prod_{k=1}^{K} N(\theta_{j,k} \mid \mu_k, \sigma)$$
$$\times \prod_{k=1}^{K} N(\mu_k \mid \mu_{0[k]}, \tau^2_{0[k]})$$

Figura 9. Fórmula de la distribución de las respuestas semejante a un modelo jerárquico

Donde *X* es una matriz de diseño construida apropiadamente, *Y* es un vector de salida apropiadamente construido, $\mu = \mu_1 \ldots \mu_k$ que representa una petición de cada elemento, y $\boldsymbol{\mu_0} = \mu_{0\,[1]} \ldots \mu_{0\,[k]}$ y $\tau^2_o = \tau^2_{o\,[1]} \ldots \tau^2_{o[k]}$ son parámetros anteriores para cada petición de cada elemento (μ). Para más información véase Salganik, M., & Levy, K. (2015).

Resumiendo la matriz de opinión

Una vez estimada, la matriz de opinión Θ puede incluir cientos o miles de parámetros - ya que en ocasiones hay miles de encuestados y cientos de elementos - que son medidos en una escala no intuitiva. Por lo tanto, el segundo paso para el análisis es resumir la matriz de opinión Θ para hacerla más interpretable. El resumen ideal de la matriz de opinión probablemente variará de ajuste en ajuste, pero el resumen estadístico preferido es el que se llama la calificación de cada elemento $\hat{s}_i$, el cual es la oportunidad estimada de que será aleatoriamente seleccionado para un encuestado también seleccionado aleatoriamente. Esto se define en la ecuación representada en la Figura 10:

$$\widehat{s_i} = \frac{\sum_{j=1}^{J} \sum_{k \neq i} \Phi(\hat{\theta}_{j,i} - \hat{\theta}_{j,k})}{J \times (K-1)} \times 100$$

Figura 10. Oportunidad de que un elemento sea seleccionado aleatoriamente para un encuestado seleccionado también de manera aleatoria

La calificación mínima es 0 para un elemento que siempre se espera que pierda (que no reciba el voto) y la máxima calificación es 100 para un elemento que se espera que gane (o que reciba el voto). Por ejemplo, una calificación de 50 para una idea dada significa que es igualmente probable a ganar o perder cuando es comparada a una idea seleccionada aleatoriamente para un usuario seleccionado también aleatoriamente.

Para construir el 95% de los intervalos posteriores alrededor de las calificaciones estimadas, se utilizan *t* extracciones posteriores de la matriz de opinión ($\Theta^{(1)}$, $\Theta^{(2)}$...$\Theta^{(t)}$) para calcular las extracciones posteriores de s ($s\hat{}^{(1)}$,$s\hat{}^{(2)}$,…,$s\hat{}^{(t)}$). De estas extracciones, se calcula que el 95% de los intervalos posteriores alrededor de $s\hat{}_i$ por valores de hallazgo *a* y *b* como la probabilidad (*Pr*), *Pr* ($s\hat{}_i > a$) = 0.025 y *Pr* ($s\hat{}_i > a$) = 0.025.

Se ha seleccionado llevar un proceso de análisis de dos pasos: estimando y resumiendo la matriz de opinión Θ, en lugar de estimar las calificaciones directamente por tres razones. Primero, se cree que en la matriz de opinión Θ, un objetivo explícito de inferencia subraya la posible heterogeneidad de las preferencias entre los encuestados. Segundo, por la estimación de la matriz de opinión como un paso intermedio, el enfoque tomado puede ser extendido a casos en los cuales co-variables son agregadas al nivel de cada encuestado (por ejemplo género, edad, etc.) o al nivel de cada elemento (por ejemplo acerca de la economía, ambiente, etc.). Finalmente, a pesar de que se tiene el interés en la calificación como un resumen estadístico, hay muchos posibles resúmenes de la matriz de opinión que pueden ser importantes, y por la estimación de Θ se habilita a futuros investigadores a seleccionar otras maneras de resumir que puedan ser importantes.

3.3.2 Características de las personas encuestadas

Para realizar la encuesta en línea se solicitó la colaboración de expertos de dominio que cumplieran las siguientes características:

- Experiencia en desarrollo de software, especialmente en el manejo de requisitos de software y desarrollo de arquitectura.
- Practicantes de metodologías tanto tradicionales como ágiles.

3.3.3 Resultados obtenidos

La encuesta obtuvo la respuesta de **17** expertos, que realizaron un total de **594** votos para priorizar las **36** actividades. A partir de cada comparación la herramienta asigna un voto a cada actividad seleccionada.

Además, se buscó dividir las actividades en prioridades **"alta"** y **"baja"**. Para lograrlo se tomó como referencia el principio de Pareto (Javier e Izquierdo, 2009) según el cual, en un conjunto constituido por un gran número de unidades, sólo una pequeña parte de las mismas son realmente significativas. Se le ha llamado también la ley del 20/80 porque generalmente, el 20% de los elementos del grupo constituyen el 80% del total ponderado. De esta forma se determina que el **20%** de las actividades que conforman la guía aportarán el **80%** del valor. Aplicado el principio de Pareto a la lista de actividades priorizadas obtenida de la encuesta, se tiene que las **7** actividades con mayor votación son aquellas que dan este **20%** de mayor valor y por lo tanto han sido catalogadas con prioridad **alta.** Las **29** actividades restantes que obedecen al **80%** han sido calificadas con prioridad **baja**.

En la Tabla 6 se muestran las actividades con la calificación obtenida por las votaciones. Las actividades marcadas en color verde son aquellas que han sido catalogadas con prioridad alta.

Tabla 6. Lista de actividades ordenada con la calificación obtenida de las votaciones de los usuarios

Ideas (Actividades)	Calificación (0 - 100)
Las metas del proyecto y del negocio son entendidas	92
Condiciones de prueba han sido incluidas en la información de los atributos de calidad	80
Riesgos en los requisitos de atributos de calidad han sido identificados y se ha obtenido una lista priorizada de escenarios de fallo	67
Prototipos rápidos han sido utilizados para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso	66
Casos de prueba han sido incluidos en las historias de usuario o especificaciones de casos de uso	66
Narrativas para explorar la funcionalidad del sistema han sido elaboradas, incluyendo lo que sucede antes y después de cada acción	62
Retroalimentación del producto es obtenida durante el desarrollo	61
Las restricciones técnicas y de negocio han sido identificadas	59
Los atributos de calidad relevantes han sido priorizados	57
Los usuarios finales e incluso clientes han sido estudiados en su hábitat natural	56
El sistema o desarrollo ha sido clasificado de acuerdo a su importancia táctica o estratégica	55
Los riesgos identificados han sido incluidos en la lista del producto o backlog	54

Tabla 6. Lista de actividades ordenada con la calificación obtenida de las votaciones de los usuarios (continuación)

Ideas (Actividades)	Calificación (0 - 100)
Las dependencias con sistemas externos han sido revisadas	53
Los principios de desarrollo y/o arquitectura han sido identificados y seleccionadas	51
La obtención de atributos de calidad ha sido incluida como parte del rol de arquitecto de software	51
Las restricciones han sido utilizadas en la prueba de escenarios de atributos de calidad	51
Las restricciones y reglas de negocio han sido incluidas en historias de usuario	50
Las historias de usuario han sido priorizadas	49
Las reglas de negocio han sido consideradas como fuente de requisitos emergentes	48
Los riesgos del proyecto han sido priorizados	47
La aparición de requisitos emergentes es soportada durante el desarrollo	47
Información sobre los usuarios ha sido recolectada y se han identificado los diferentes perfiles que juegan los involucrados en el proyecto	46
Los drivers arquitecturales han sido identificados en los atributos de calidad y escenarios de funcionalidad	46
Los escenarios de atributos de calidad han sido priorizados	46
Riesgos en los requisitos han sido identificados	44
La plataforma de despliegue destino ha sido revisada	44
La definición del problema elaborada es corta y nítida	41
La lista de deseos de los involucrados sobre lo que necesitan que el sistema haga ha sido obtenida	41
Los atributos de calidad relevantes han sido identificados	41
Los riesgos del proyecto han sido identificados	40
El esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad ha sido estimado	40
Todos los involucrados acuden a las mismas reuniones	40
Las características del equipo son consideradas cuando se está arquitectando un sistema de software	34
Las restricciones organizacionales han sido priorizadas	33
La lista de tecnologías aprobadas por la compañía ha sido revisada	24

Se ha aprendido sobre los atributos de calidad comunes del dominio	24

Para el final de esta etapa se contaba con la lista de actividades refinadas y priorizadas con sus elementos complementarios (técnicas, herramientas, artefactos, etc.) listas para el diseño de la presentación de la guía.

3.4 Diseño de la presentación de la información

Durante esta etapa se generaron prototipos a diferentes niveles de abstracción, los cuales fueron evolucionando hasta obtener una versión aceptada del formato en que se presentará la información de la guía.

Los prototipos generados fueron los siguientes:

- Una primera versión generada en papel
- Una segunda versión generada en una herramienta de diseño
- Una tercera versión generada por un diseñador profesional
- Versión final del diseño

En primera instancia se definió que la guía sería presentada a manera de tarjetas, en las cuales se colocarían los 11 elementos (definidos en la sección 3.2) que conforman la información de cada actividad.

Para diseñar la primera versión se identificaron los datos que podrían servir para realizar diferentes clasificaciones de las tarjetas que son:

- La prioridad de la actividad (alta o baja)
- La etapa del ciclo de vida en que se lleva a cabo
- La etapa del proceso de ingeniería de requisitos a la que se relaciona
- El rol o roles que se ven involucrados en su ejecución
- Por su relación a atributos de calidad, riesgos, restricciones, requisitos emergentes y usuarios

3.4.1 Prototipo 1

Esta versión buscó principalmente mostrar una propuesta de cómo los elementos que conformarán cada tarjeta podrían distribuirse en un área de trabajo rectangular. En este momento del diseño las medidas no han sido especificadas. En la Figura 11 se muestra el diseño propuesto:

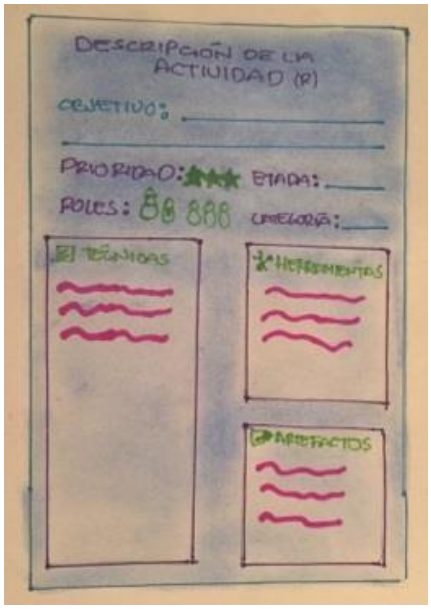

Figura 11. Diseño del prototipo de papel

En esta etapa del diseño se estableció la distribución de los elementos, quedando en la parte superior aquellos datos que sirven para dar un contexto a la actividad y en la inferior los que dan soporte a su ejecución, como las técnicas y herramientas para ejecutarlas así como los artefactos esperados.

3.4.2 Prototipo 2

En la segunda versión se estableció que sería utilizado el tamaño estándar que utilizan las cartas de póker (5.9 x 9.1 cm) para generar la versión digital, en la cual los elementos de la tarjeta pasaron de ocupar sólo una cara de la tarjeta a ambas.

Las Figuras 12 y 13 muestran un ejemplo del contenido real, y cómo se distribuyó en las dos nuevas áreas de trabajo:

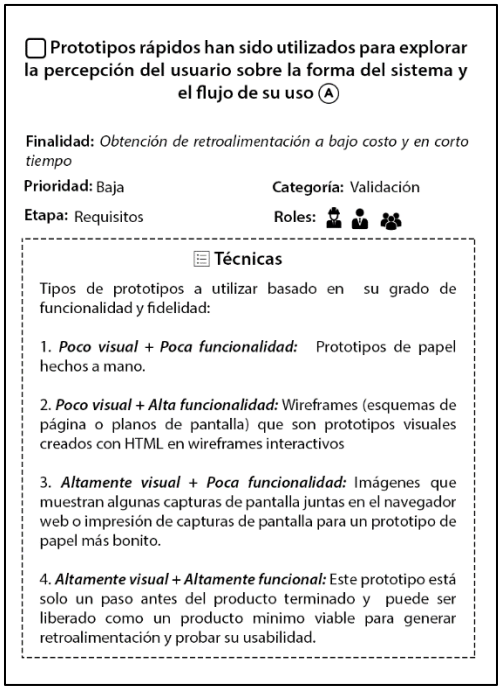

Figura 12. Frente de la tarjeta

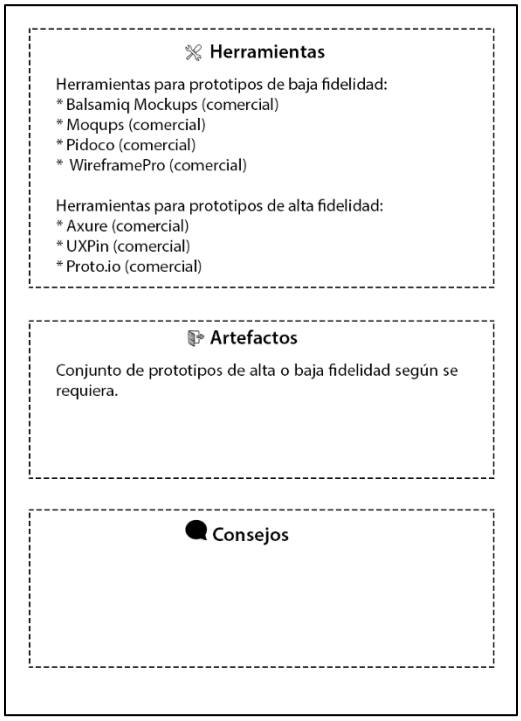

Figura 13. Reverso de la tarjeta

A esta versión se agregó posteriormente el elemento consejos como parte de la tarjeta y se procedió a armar cada una para facilitar el trabajo de diseño. La Figura 14 muestra un ejemplo de las tarjetas elaboradas:

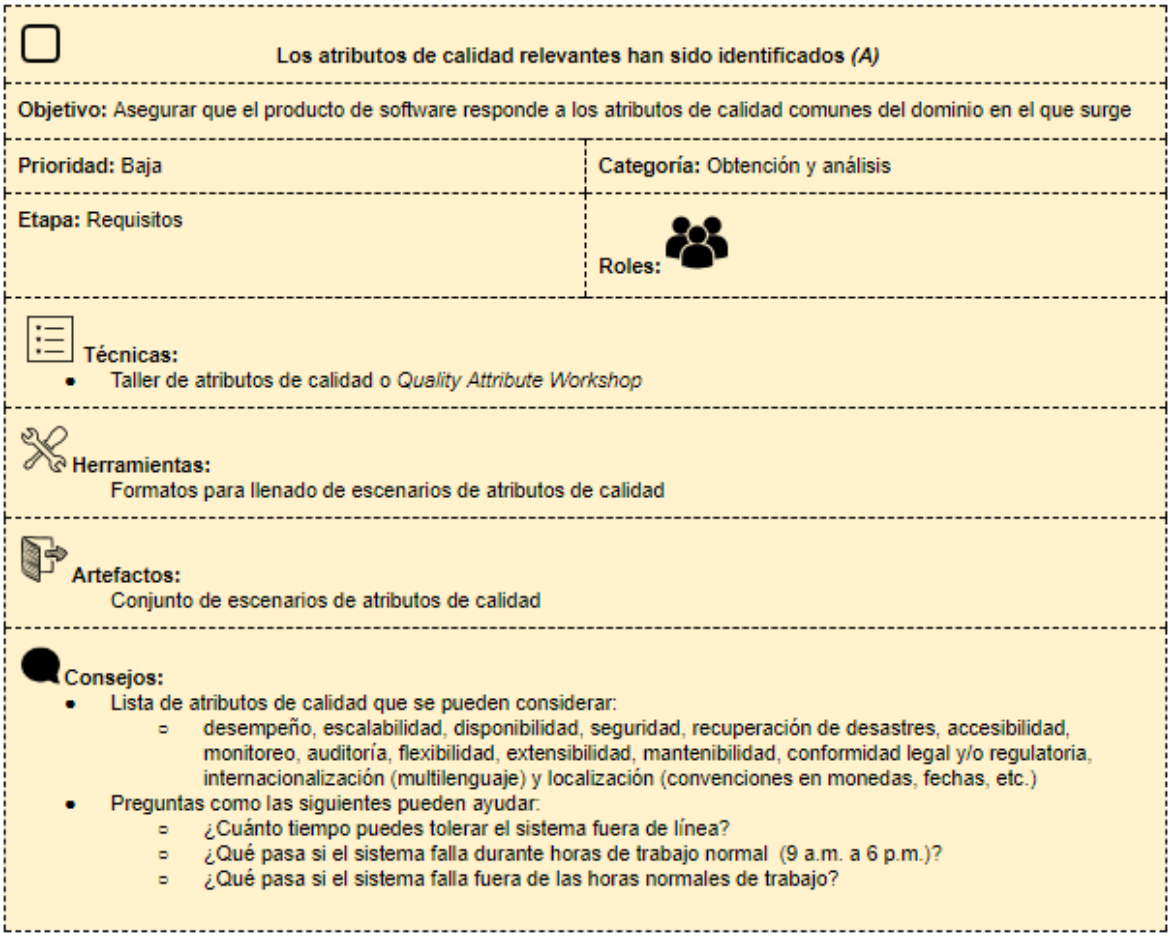

Figura 14. Bosquejo de tarjeta para entrega a diseño

De esta forma se generaron las 36 fichas que posteriormente serían utilizadas para construir el juego de 36 tarjetas (para acceder al contenido completo de todas las tarjetas véase anexo F).

Posteriormente se agregó a manera de instrucciones, una descripción de los elementos que conforman la estructura de las tarjetas, así como una explicación de su propósito y la forma en que pueden ser utilizadas. Lo cual es mostrado en las Figuras 15 y 16 a continuación:

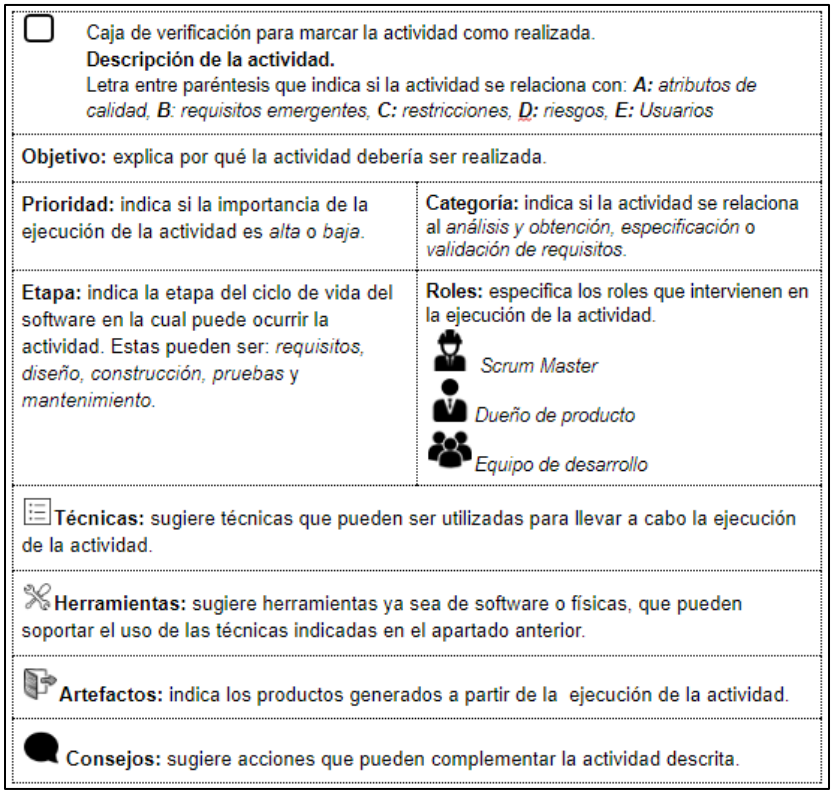

Figura 15. Explicación del contenido de las tarjetas

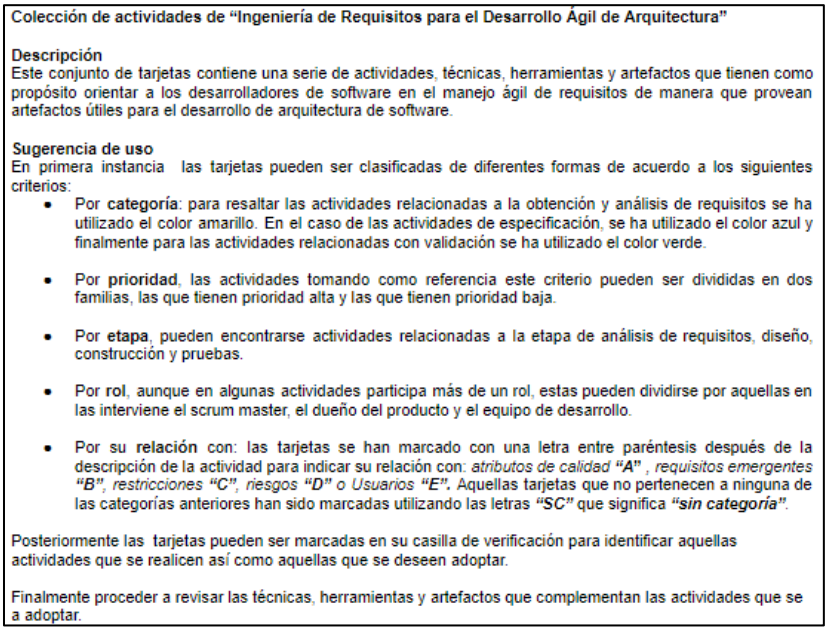

Figura 16. Explicación de la guía y las opciones para clasificar las tarjetas

3.4.3 Prototipo 3

En esta sección se muestran como al pasar por el trabajo de diseño profesional, pasó de nuevo de ocupar el contenido las dos caras de la tarjeta a ocupar sólo una aumentando su tamaño. Las siguientes figuras (17, 18, 19) muestran cómo quedaron el frente y el reverso en esta versión para cada una de las tres categorías principales en las que se han dividido las tarjetas:

- Obtención y análisis de requisitos
- Especificación de requisitos
- Validación de requisitos

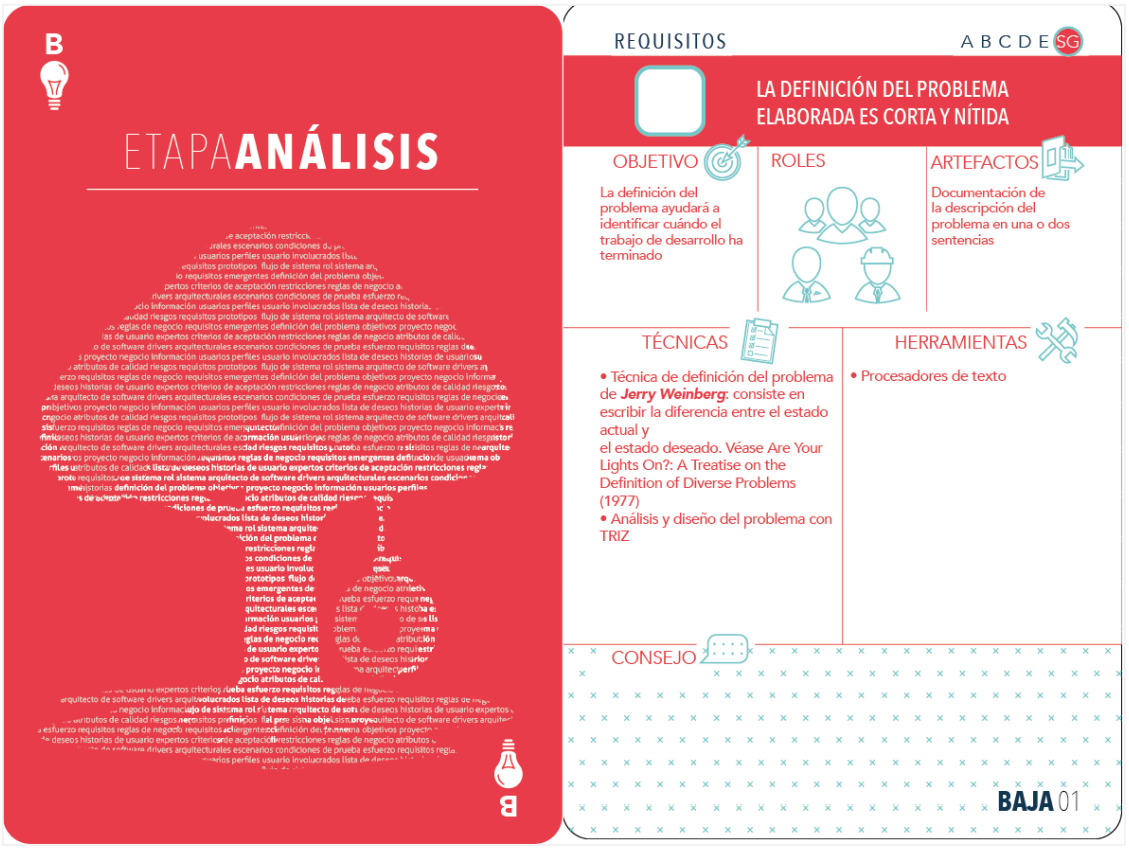

Figura 17. Ejemplo de tarjeta de la categoría Obtención y análisis de requisitos

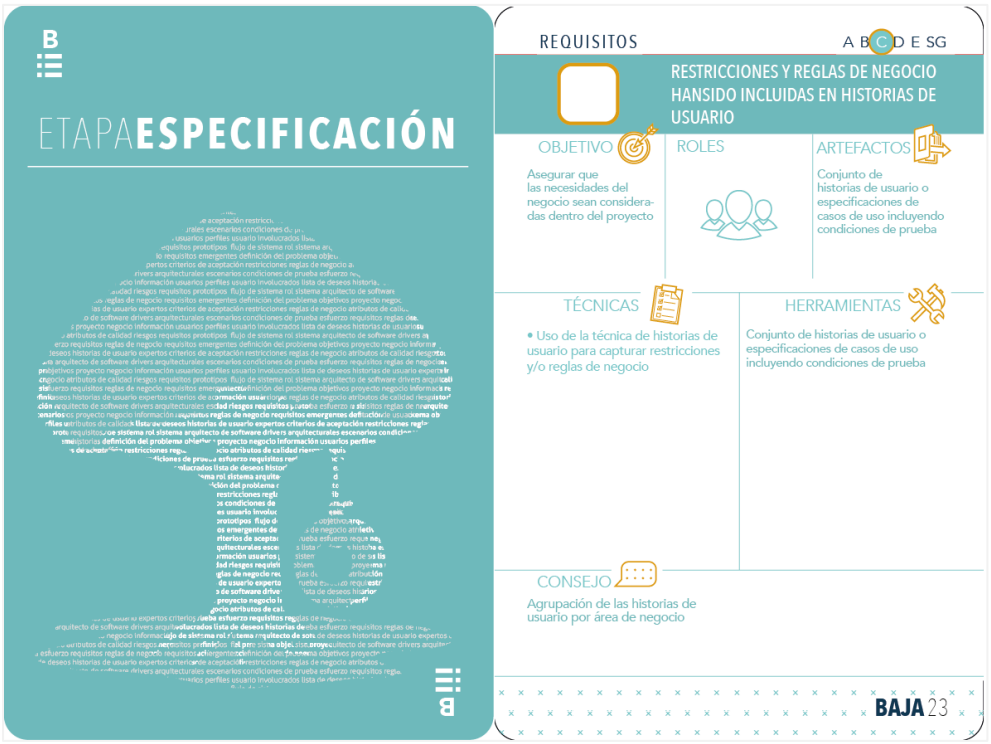

Figura 18. Ejemplo de tarjeta de la categoría Especificación de requisitos

Figura 19. Ejemplo de tarjeta de la categoría Validación de requisitos

3.4.4 Versión final del diseño

En esta versión se tienen algunos cambios realizados a la versión entregada por el diseñador en el prototipo anterior. Versión a la cual se hicieron algunas correcciones menores como:

- Utilizar la palabras completas para referirse a la prioridad (Alta y Baja) en lugar de las letras A y B para evitar confusiones con la categoría que agrupa las actividades en los subconjuntos A, B, C, D, E y SC por su relación con atributos de calidad, requisitos emergentes, etc.
- Modificaciones menores a la distribución de algunos elementos.
- Modificaciones menores en algunos colores.

Las Figuras 20, 21 y 22 muestran como quedó al final el diseño de las tarjetas de cada una de las tres categorías principales.

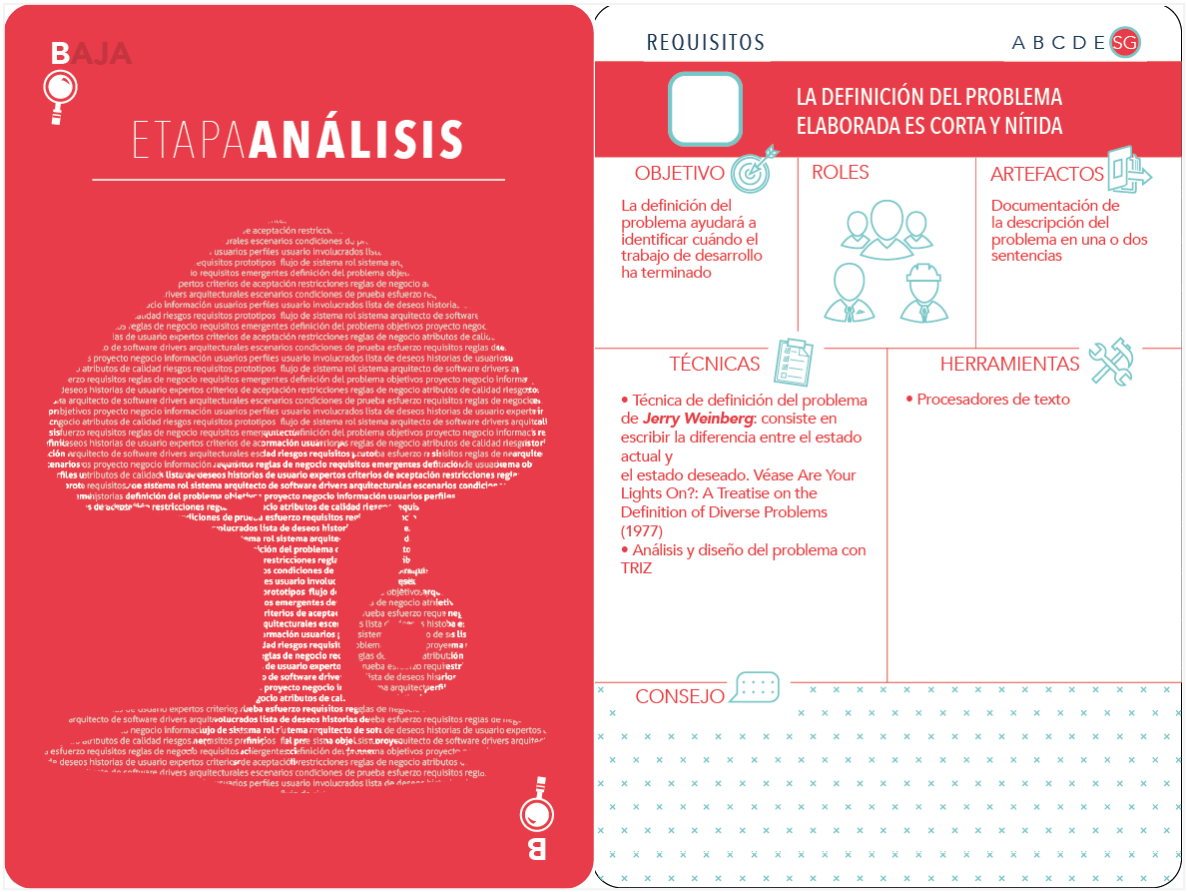

Figura 20. Versión final de la tarjeta de Obtención y análisis de requisitos

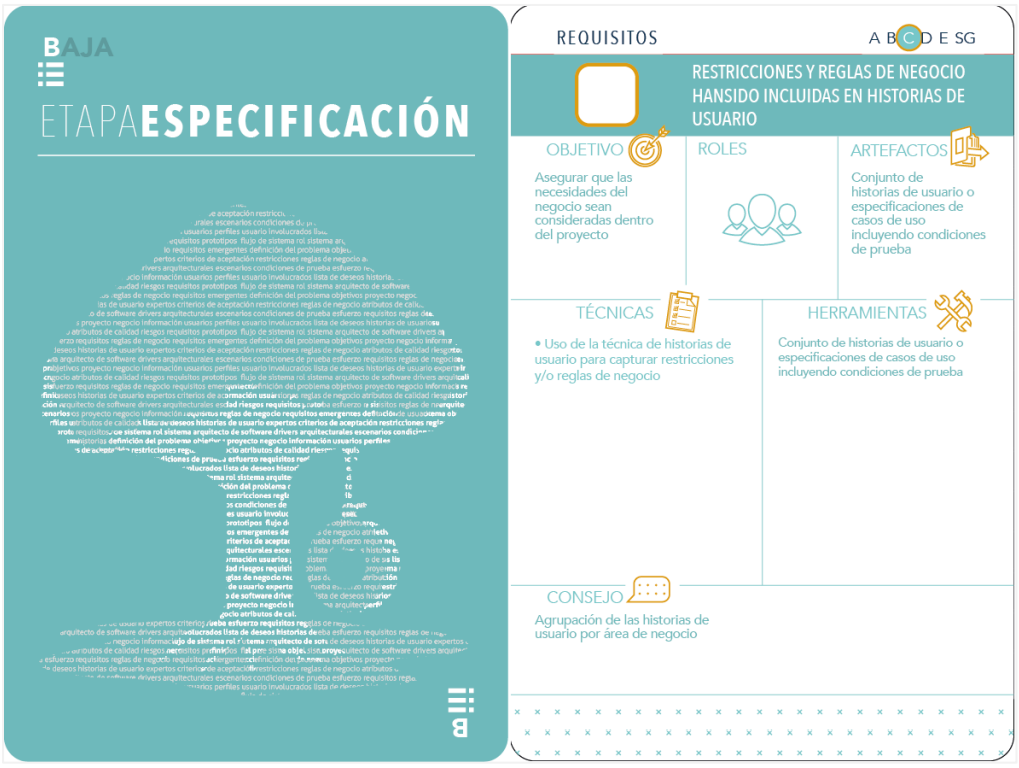

Figura 21. Versión final de la tarjeta de Especificación de requisitos

Figura 22. Versión final de la tarjeta de Validación de requisitos.

4. CASO DE ESTUDIO

Con el propósito de confirmar la prioridad asignada a las actividades que contendrá la guía, en este capítulo se describe el ejercicio de validación de dicha priorización. Para el desarrollo de este ejercicio se siguió la estructura de diseño de casos de estudio descrita en Runeson, Host, Reiner y Regnell (2012), la cual ha sido incluida en el Anexo G.

4.1 Fundamento del estudio

En este proyecto de tesis se han recopilado un conjunto de actividades, técnicas, herramientas y artefactos que serán presentados en forma de tarjetas auto contenidas, con el cual se pretende orientar a desarrolladores novatos y estudiantes sobre qué actividades relacionadas al análisis y obtención, especificación y validación de requisitos son necesarias realizar en los proyectos de software.

Como se mencionó anteriormente, con la realización de este caso de estudio se validó el conjunto de actividades que han sido valoradas con prioridad alta y baja por expertos en el tema, comparándolo con las valoraciones de prioridad asignadas por un segundo grupo conformado por practicantes del desarrollo de software.

La finalidad de esta validación es asegurar la prioridad de las actividades catalogadas con prioridad alta, proporcionarle al usuario un panorama general sobre a cuáles actividades debería prestarle más atención así como facilitar la identificación de actividades a adoptar en su trabajo habitual.

4.2 Objetivos del caso de estudio

Los objetivos de este caso de estudio son:

1. Identificar el **20%** de las actividades del conjunto que serán catalogadas con prioridad alta y que representan el **80%** del valor de actividades relacionadas con requisitos. En consecuencia identificar también el **80%** de las actividades restantes que presentarán prioridad baja, y que representan el **20%** del valor.
2. Comparar la priorización de actividades realizada por un grupo de expertos en desarrollo de arquitectura y manejo de requisitos de software contra la priorización del mismo conjunto realizada por un grupo de practicantes del desarrollo de software con menor experiencia.

4.3 Caso y unidades de análisis

En este caso de estudio se analiza la priorización de un conjunto de actividades relacionadas al análisis de requisitos por dos diferentes grupos de individuos con experiencia en desarrollo de software. Las características de los grupos se describen en la Tabla 7.

Tabla 7. Características de los grupos estudiados

Características	Grupo 1: Expertos en desarrollo de software	Grupo 2: Practicantes de desarrollo de software
Años de experiencia en desarrollo de software	Más de 5 años	Menos de 5 años
Tipos de productos de software desarrollados (por entorno)	Aplicaciones web, aplicaciones para dispositivos móviles y aplicaciones de escritorio	Aplicaciones web, aplicaciones para dispositivos móviles y aplicaciones de escritorio
Tipos de productos de software desarrollados (por área de aplicación)	Proyectos de investigación, aeroespaciales, ciber-físicos, empotrados, de aplicación y transformación de negocio	Sistemas administrativos y/o contables, proyectos de celebración de eventos
Metodologías de desarrollo utilizadas	Ágiles y tradicionales: Scrum, Kanban, TSP, RUP	Sin aplicación de metodologías de desarrollo de software
Especializaciones y/o certificaciones relacionadas al desarrollo de software	Si	No
Perfiles	Ingenieros de software sr con maestría y/o doctorado en el área de ingeniería del software	Ingenieros de software junior
Número de participantes	17	17

4.4 Teoría

El marco teórico sobre el cual se desenvuelve la investigación ha sido abordado en el capítulo 2. Los conceptos principales son:

- Desarrollo ágil
- Ingeniería de requisitos
- Obtención y análisis de requisitos
- Especificación de requisitos
- Validación de requisitos
- Actividad
- Técnica
- Herramienta
- Artefacto

4.5 Preguntas de investigación

- PI1: ¿Las priorizaciones realizadas por los dos grupos son similares en su distribución?
- PI2: ¿Cuáles actividades conforman el 20% que aporta el 80% del valor según el principio de Pareto, y que serán catalogadas con prioridad alta dentro del conjunto de actividades?
- PI3: ¿Cuáles actividades forman el 80% que serán catalogadas con prioridad baja y aportan el 20% del valor según el principio de Pareto, dentro del conjunto de actividades?

4.6 Proposiciones e hipótesis

Las proposiciones e hipótesis asociadas a las preguntas de investigación definidas en el apartado anterior se describen en la Tabla 8.

Tabla 8. Proposiciones e hipótesis de las preguntas de investigación

Pregunta de investigación	Proposición con hipótesis
PI1: ¿Las priorizaciones realizadas por los dos grupos son similares en su distribución?	**Proposición 1:** Si dos grupos de individuos priorizan las actividades, entonces se encontrará una relación entre ambas priorizaciones. **Hipótesis 1:** Grupos de individuos con diferentes niveles de experiencia en desarrollo de software pueden identificar las mismas actividades con prioridad alta y baja pertenecientes a un mismo conjunto.
PI2: ¿Cuáles son las actividades que forman el 20% del conjunto que aporta el 80% del valor según el principio de Pareto, y que serán catalogadas con prioridad alta?	**Proposición 2**: El 20% de actividades catalogadas con prioridad alta en ambos grupos será el mismo.
PI3: ¿Cuáles son las actividades que forman el 80% del conjunto que aportan el 20% del valor según el principio de Pareto, y serán catalogadas con prioridad baja?	**Proposición 3**: El 80% de actividades catalogadas con prioridad baja de ambos grupos será el mismo.

4.7 Definición de conceptos y medidas

Los conceptos más relevantes a los que se hace referencia en las preguntas de investigación se describen en la Tabla 9 mostrada a continuación:

Tabla 9. Definición de conceptos relevantes utilizados en las preguntas de investigación

Concepto o medida	Definición
Actividad	La actividad es un conjunto de tareas cohesivas de un proceso. Ha sido abordada a detalle en la sección 2.1.5 del capítulo 2 de este documento.
Priorización	La priorización, en el contexto de este proyecto de tesis, consistió en elegir entre un par de actividades, cuál de ellas aporta mayor valor en un proyecto de software. El resultado de la priorización es un conjunto de actividades organizadas con la calificación obtenida de esta priorización.
Prioridad Alta	Prioridad alta, en el contexto de este proyecto de tesis, se refiere a aquellas actividades que deberían ser realizadas en todos los proyectos de software sin importar el contexto en el que se encuentre.
Prioridad baja	Prioridad baja, en el contexto de este proyecto de tesis, se refiere a aquellas actividades cuya realización es deseable en todos los proyectos de software sin importar el contexto en el que se encuentre.
Principio del 80%-20% de Pareto	Se refiere a que generalmente, el 20% de los elementos de un grupo constituyen el 80% del total ponderado. En el contexto de este proyecto de tesis, se aplicó el principio de Pareto al conjunto de actividades para dividirlas en dos categorías: prioridad alta y baja que corresponden al 20% y 80% de los elementos del conjunto respectivamente. Esto ha sido abordado a profundidad en la sección 3.3.3 del capítulo 3 de este documento.

4.8 Método de recolección de datos

En este caso de estudio, la actividad de recolección de datos ha sido realizada utilizando la herramienta *Wiky Surverys*[2] de Salganik y Levy (2015) descrita en la sección 3.3.1 del capítulo 2. La herramienta fue utilizada para priorizar las actividades y se siguieron los siguientes criterios:

- Se aplicó la encuesta a dos grupos compuestos por 17 individuos, cada uno de los cuales realizó la priorización de manera individual.
- Cada persona realizó 40 votos por lo que al final se obtuvieron dos listas priorizadas de actividades en base a 680 votos recibidos.
- Cada voto consistió en la selección de una actividad entre un par teniendo como criterio elegir aquella actividad que aporte mayor valor en un proyecto de software.

Las Tablas 10 y 11 muestran las listas priorizadas obtenidas de las priorizaciones realizadas por los grupos de expertos y practicantes respectivamente.

Tabla 10. Lista de actividades priorizadas por el grupo de expertos

Actividad	Calificación (0 a 100)
Las metas del proyecto y del negocio son entendidas	92
Condiciones de prueba han sido incluidas en la información de los atributos de calidad	80
Riesgos en los requisitos de atributos de calidad han sido identificados y se ha obtenido una lista priorizada de escenarios de fallo	67
Prototipos rápidos han sido utilizados para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso	66
Casos de prueba han sido incluidos en las historias de usuario o especificaciones de casos de uso	66

[2] Wiki Surveys de Salganik y Levi (2015) es una herramienta en línea que permite realizar encuestas en las cuales los participantes seleccionan entre pares de opciones provenientes de un banco de ideas. Como criterio para seleccionar una opción se muestra una pregunta la cual es mantenida a través de todos los pares de opciones que se van mostrando a los participantes, y estos pueden votar por tantos pares de opciones como deseen debido a que la herramienta no limita el número de votos recibidos.

Tabla 10. Lista de actividades priorizadas por el grupo de expertos (continuación)

Actividad	Calificación (0 a 100)
Narrativas para explorar la funcionalidad del sistema han sido elaboradas, incluyendo lo que sucede antes y después de cada acción	62
Retroalimentación del producto es obtenida durante el desarrollo	61
Las restricciones técnicas y de negocio han sido identificadas	59
Los atributos de calidad relevantes han sido priorizados	57
Los usuarios finales e incluso clientes han sido estudiados en su hábitat natural	56
El sistema o desarrollo ha sido clasificado de acuerdo a su importancia táctica o estratégica	55
Los riesgos identificados han sido incluidos en la lista del producto o backlog	54
Las dependencias con sistemas externos han sido revisadas	53
Los principios de desarrollo y/o arquitectura han sido identificados y seleccionadas	51
La obtención de atributos de calidad ha sido incluida como parte del rol de arquitecto de software	51
Las restricciones han sido utilizadas en la prueba de escenarios de atributos de calidad	51
Las restricciones y reglas de negocio han sido incluidas en historias de usuario	50
Las historias de usuario han sido priorizadas	49
Las reglas de negocio han sido consideradas como fuente de requisitos emergentes	48
Los riesgos del proyecto han sido priorizados	47
La aparición de requisitos emergentes es soportada durante el desarrollo	47
Información sobre los usuarios ha sido recolectada y se han identificado los diferentes perfiles que juegan los involucrados en el proyecto	46
Los drivers arquitecturales han sido identificados en los atributos de calidad y escenarios de funcionalidad	46
Los escenarios de atributos de calidad han sido priorizados	46
Riesgos en los requisitos han sido identificados	44
La plataforma de despliegue destino ha sido revisada	44
La definición del problema elaborada es corta y nítida	41
La lista de deseos de los involucrados sobre lo que necesitan que el sistema haga ha sido obtenida	41
Los atributos de calidad relevantes han sido identificados	41

Tabla 10. Lista de actividades priorizadas por el grupo de expertos (continuación II)

Actividad	Calificación (0 a 100)
Los riesgos del proyecto han sido identificados	40
El esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad ha sido estimado	40
Las características del equipo son consideradas cuando se está arquitectando un sistema de software	34
Las restricciones organizacionales han sido priorizadas	33
Se ha aprendido sobre los atributos de calidad comunes del dominio	24
La lista de tecnologías aprobadas por la compañía ha sido revisada	24

Tabla 11. Lista de actividades priorizadas por el grupo de practicantes

Actividad	Calificación (0 a 100)
Las metas del proyecto y del negocio son entendidas	77
Las restricciones técnicas y de negocio han sido identificadas	75
Riesgos en los requisitos de atributos de calidad han sido identificados y se ha obtenido una lista priorizada de escenarios de fallo	70
El sistema o desarrollo ha sido clasificado de acuerdo a su importancia táctica o estratégica	68
Las reglas de negocio han sido consideradas como fuente de requisitos emergentes	68
Retroalimentación del producto es obtenida durante el desarrollo	67
Información sobre los usuarios ha sido recolectada y se han identificado los diferentes perfiles que juegan los involucrados en el proyecto	65
Narrativas para explorar la funcionalidad del sistema han sido elaboradas, incluyendo lo que sucede antes y después de cada acción	63
Los atributos de calidad relevantes han sido priorizados	61
Prototipos rápidos han sido utilizados para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso	58
Los riesgos del proyecto han sido priorizados	58
Las restricciones organizacionales han sido priorizadas	56
Los usuarios finales e incluso clientes han sido estudiados en su hábitat natural	54
Condiciones de prueba han sido incluidas en la información de los atributos de calidad	52
Los riesgos identificados han sido incluidos en la lista del producto o backlog	52

Tabla 11. Lista de actividades priorizadas por el grupo de practicantes (continuación)

Actividad	Calificación (0 a 100)
Casos de prueba han sido incluidos en las historias de usuario o especificaciones de casos de uso	51
La aparición de requisitos emergentes es soportada durante el desarrollo	51
Las restricciones y reglas de negocio han sido incluidas en historias de usuario	50
Las restricciones han sido utilizadas en la prueba de escenarios de atributos de calidad	49
Riesgos en los requisitos han sido identificados	49
Los riesgos del proyecto han sido identificados	48
Los escenarios de atributos de calidad han sido priorizados	45
La definición del problema elaborada es corta y nítida	45
Las características del equipo son consideradas cuando se está arquitectando un sistema de software	45
Los principios de desarrollo y/o arquitectura han sido identificados y seleccionadas	44
Las dependencias con sistemas externos han sido revisadas	43
Las historias de usuario han sido priorizadas	43
La lista de deseos de los involucrados sobre lo que necesitan que el sistema haga ha sido obtenida	40
La obtención de atributos de calidad ha sido incluida como parte del rol de arquitecto de software	38
Los drivers arquitecturales han sido identificados en los atributos de calidad y escenarios de funcionalidad	38
El esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad ha sido estimado	38
Se ha aprendido sobre los atributos de calidad comunes del dominio	35
Los atributos de calidad relevantes han sido identificados	34
Todos los involucrados acuden a las mismas reuniones	33
La plataforma de despliegue destino ha sido revisada	30
La lista de tecnologías aprobadas por la compañía ha sido revisada	27

4.9 Análisis de los datos

En esta sección se describe cómo fue realizado el análisis de los datos para responder las tres preguntas de investigación presentadas en la sección 4.5 de este capítulo.

4.9.1 Asignación de identificador a las actividades

Una vez obtenidos los dos conjuntos de actividades priorizadas se les asignó un identificador. Quedando la Tabla 12 con las actividades enumeradas de la siguiente manera:

Tabla 12. Tabla de actividades con identificador asignado

Actividad	ID asignado
La definición del problema elaborada es corta y nítida	1
Las metas del proyecto y del negocio son entendidas	2
Información sobre los usuarios ha sido recolectada y se han identificado los diferentes perfiles que juegan los involucrados en el proyecto	3
Los usuarios finales e incluso clientes han sido estudiados en su hábitat natural	4
La obtención de atributos de calidad ha sido incluida como parte del rol de arquitecto de software	5
Los atributos de calidad relevantes han sido identificados	6
Los drivers arquitecturales han sido identificados en los atributos de calidad y escenarios de funcionalidad	7
Se ha aprendido sobre los atributos de calidad comunes del dominio	8
El esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad ha sido estimado	9
Las reglas de negocio han sido consideradas como fuente de requisitos emergentes	10
Las restricciones técnicas y de negocio han sido identificadas	11
La lista de tecnologías aprobadas por la compañía ha sido revisada	12
Las dependencias con sistemas externos han sido revisadas	13
La plataforma de despliegue destino ha sido revisada	14

Tabla 12. Tabla de actividades con identificador asignado (continuación)

Actividad	ID asignado
Las características del equipo son consideradas cuando se está arquitectando un sistema de software	15
El sistema o desarrollo ha sido clasificado de acuerdo a su importancia táctica o estratégica	16
Los principios de desarrollo y/o arquitectura han sido identificados y seleccionadas	17
Los riesgos del proyecto han sido identificados	18
Los riesgos del proyecto han sido priorizados	19
Riesgos en los requisitos han sido identificados	20
La lista de deseos de los involucrados sobre lo que necesitan que el sistema haga ha sido obtenida	21
Narrativas para explorar la funcionalidad del sistema han sido elaboradas, incluyendo lo que sucede antes y después de cada acción	22
Las restricciones y reglas de negocio han sido incluidas en historias de usuario	23
Los riesgos identificados han sido incluidos en la lista del producto o *backlog*	24
Casos de prueba han sido incluidos en las historias de usuario o especificaciones de casos de uso	25
Las historias de usuario han sido priorizadas	26
Prototipos rápidos han sido utilizados para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso	27
Los atributos de calidad relevantes han sido priorizados	28
Condiciones de prueba han sido incluidas en la información de los atributos de calidad	29
Los escenarios de atributos de calidad han sido priorizados	30
La aparición de requisitos emergentes es soportada durante el desarrollo	31
Las restricciones han sido utilizadas en la prueba de escenarios de atributos de calidad	32
Las restricciones organizacionales han sido priorizadas	33
Retroalimentación del producto es obtenida durante el desarrollo	34
Todos los involucrados acuden a las mismas reuniones	35
Riesgos en los requisitos de atributos de calidad han sido identificados y se ha obtenido una lista priorizada de escenarios de fallo	36

Posteriormente el estudio se dividió en dos etapas. En la primera se compararon los dos conjuntos de actividades priorizadas para analizar su similitud utilizando técnicas estadísticas. En la segunda se identificó el 20% de las actividades con prioridad alta del conjunto de 36 actividades.

4.9.2 Comparación de priorizaciones entre grupos

Para la comparación de ambos grupos se seleccionó la prueba de rangos con signos de Wilcoxon que puede ser utilizada cuando los conjuntos comparados derivan de algún tipo de escala de calificación (Lowry, 1998). Para aplicar esta técnica se trabajará con las posiciones de las actividades en las listas priorizadas en lugar de las calificaciones recibidas en la priorización, debido a que la a herramienta utilizada *Wiki Surveys (*Salganik y Levy, 2015*)*, cuenta con su propio algoritmo para generar una calificación (de 0 a 100) de cada elemento priorizado (en este caso actividades). Por lo tanto, no es posible acceder a los valores que originaron dichas calificaciones y estudios más profundos no pueden ser realizados.

La aplicación de la prueba de Wilcoxon a los dos conjuntos ayudará a definir qué tan similares son, para lograrlo la prueba asume los siguientes factores (Lowry, 1998):

- Que los valores emparejados de X_A y X_B son aleatorios e independientes (es decir, cada par es independiente de todos los otros pares).
- Que la variable dependiente (por ejemplo, la estimación de la probabilidad de un sujeto) es intrínsecamente continua.
- Que las medidas de X_A y X_B tienen las propiedades de al menos una escala ordinal de medida, como "mayor que", "menor que" y "igual a".

4.9.2.1 Asignación de número de posición a la priorización de actividades de los conjuntos

Para aplicar la técnica de Wilcoxon primeramente se asignó un número de posición a cada actividad de los dos conjuntos según la lista priorizada obtenida. Esto es mostrado en las Tablas 13 y 14.

Tabla 13. Asignación de número de posición a cada actividad en la priorización del grupo de expertos

ID de la actividad	Calificación (0 a 100)	Posición
2	92	36
29	80	35
36	67	34
27	66	33
25	66	32
22	62	31
34	61	30
11	59	29
28	57	28
4	56	27
16	55	26
24	54	25
13	53	24
17	51	23
5	51	22
32	51	21
23	50	20
26	49	19
10	48	18
19	47	17
31	47	16

Tabla 13. Asignación de número de posición a cada actividad en la priorización del grupo de expertos (continuación)

ID de la actividad	Calificación (0 a 100)	Posición
3	46	15
7	46	14
30	46	13
20	44	12
14	44	11
1	41	10
21	41	9
6	41	8
18	40	7
9	40	6
35	40	5
15	34	4
33	33	3
8	24	2
12	24	1

Tabla 14. Asignación de número de posición a cada actividad en la priorización del grupo de practicantes

ID de la actividad	Calificación (0 a 100)	Posición
2	77	36
11	75	35
36	70	34
16	68	33
10	68	32
34	67	31
3	65	30
22	63	29
28	61	28

Tabla 14. Asignación de número de posición a cada actividad en la priorización del grupo de practicantes (continuación)

ID de la actividad	Calificación (0 a 100)	Posición
27	58	27
19	58	26
33	56	25
4	54	24
29	52	23
24	52	22
25	51	21
31	51	20
23	50	19
32	49	18
20	49	17
18	48	16
30	45	15
1	45	14
15	45	13
17	44	12
13	43	11
26	43	10
21	40	9
5	38	8
7	38	7
9	38	6
8	35	5
6	34	4
35	33	3
14	30	2

Tabla 14. Asignación de número de posición a cada actividad en la priorización del grupo de practicantes (continuación II)

ID de la actividad	Calificación (0 a 100)	Posición
12	27	1

4.9.2.2 Emparejamiento de las actividades con su posición en cada conjunto priorizado

Enseguida se emparejaron las actividades con el número de su posición en cada conjunto tomando su identificador como criterio para ordenarlas de manera ascendente. Esto es mostrado en la Tabla 15:

Tabla 15. Asignación de posiciones a la actividad en las priorizaciones realizadas

ID de la actividad	Posición en el grupo 1 - Expertos	Posición en el grupo 2 - Practicantes
1	10	14
2	36	36
3	15	30
4	27	24
5	22	8
6	8	4
7	14	7
8	2	5
9	6	6
10	18	32
11	29	35
12	1	1
13	24	11
14	11	2
15	4	13
16	26	33
17	23	12

Tabla 15. Asignación de posiciones a la actividad en las priorizaciones realizadas (continuación)

ID de la actividad	Posición en el grupo 1 - Expertos	Posición en el grupo 2 - Practicantes
18	7	16
19	17	26
20	12	17
21	9	9
22	31	29
23	20	19
24	25	22
25	32	21
26	19	10
27	33	27
28	28	28
29	35	23
30	13	15
31	16	20
32	21	18
33	3	25
34	30	31
35	5	3
36	34	34

4.9.2.3 Desarrollo del método de Wilcoxon

Una vez realizado lo anterior, se desarrolló el método de Wilcoxon con los siguientes pasos que se muestran gráficamente en la Tabla 16.

1) Se establecen las hipótesis:
    - H_0: Las priorizaciones de las actividades realizadas por los grupos de expertos y practicantes tienen la misma distribución.
    - Las priorizaciones de las actividades realizadas por los grupos de expertos y practicantes tienen una distribución diferente.
2) Las columnas 1, 2 y 3 han sido obtenidas de la Tabla 15 del apartado anterior. Donde:
    - ID es el identificador de la actividad (columna 1).
    - X_A es la posición de la actividad en la priorización del grupo de expertos (columna 2).
    - X_B es la posición de la actividad en la priorización del grupo de practicantes (columna 3).
3) En la columna 4 se agregó la diferencia de $X_A - X_B$.
4) En la columna 5 se ha transformando la diferencia de $X_A - X_B$ a su valor absoluto.
5) Como paso intermedio se agregó la columna 5.1 en la cual se han ordenado los valores de la columna 5 de menor a mayor y se ha conservado el signo de la columna 4.
6) En la columna 6, por regla del método no se tomaron en cuenta aquellos elementos que dieron una diferencia de 0 en la resta de $X_A - X_B$. Enseguida los datos restantes son jerarquizados de menor a mayor considerando filas enlazadas[3] en los casos necesarios.

3 Las filas enlazadas se dan cuando dos o más de los datos tienen el mismo valor en secuencia. Se suman los valores de sus posiciones en la jerarquía y se divide entre el número de veces que se ha repetido el dato. Ejemplo: se tiene dos veces

7) La columna 6.1 ha sido agregada como apoyo pues sólo contiene una numeración en secuencia para ayudar en el cálculo de las filas enlazadas descritas en el punto anterior.

8) Los datos de la columna 7 son extraídos de la columna 6 con su signo proveniente de la columna 5.1

9) Enseguida se calcula la suma de los rangos con signo de la columna 7 en **W**, obteniendo así **W = 14**.

10) Posteriormente se calcula **N,** que es total de elementos del conjunto que han quedado luego eliminar los valores nulos. En este caso se eliminaron los primeros 6 del conjunto de 36 actividades en la columna 6, quedando así **N = 30**.

11) Enseguida se calcula la desviación estándar de la distribución con la siguiente fórmula:

    - w = N (N+1) (N+2) 6
    - w = 30 (31)(32) 6
    - w = 97.23

12) El siguiente paso es calcular el estadístico de prueba **Z** de la siguiente manera:

    - Z = (W - 0.5) w
    - Z = (14 - 0.5) 97.23
    - Z = 0.14

13) Se establece un nivel de significancia de 0.05 o 95%; entonces si la hipótesis es cierta, el valor de **Z** obtenido se encontrará entre -1.96 y 1.96 (debido a que el área bajo la curva normal entre esos valores es de 0.95).

14) Finalmente, si comparamos el valor de **Z** con el rango anterior tenemos que:

    - - 1.96 < Z < 1.96

---

el dato 1 en las posiciones 1 y 2 de la jerarquía, entonces se pondría su promedio en ambas posiciones; que sería: 1+2/2 = 1.5

- -1.96 < 0.14 < 1.96 ∴ Se acepta la hipótesis nula H_0 que indica que las priorizaciones de ambos conjuntos de actividades tienen una distribución similar. Esto con un margen de error de 0.05.

Tabla 16. Desarrollo del método de Wilcoxon

1	2	3	4	5	5.1	6	6.1	7
ID	**X_A**	**X_B**	**Original X_A - X_B**	**Valor absoluto X_A - X_B**	**X_A - X_B Ordenado con signo**	**Jerarquía del valor absoluto de X_A - X_B**	**Consecutivo de apoyo**	**Jerarquía con signos**
1	10	14	-4	4	0	--	--	--
2	36	36	0	0	0	--	--	--
3	15	30	-15	15	0	--	--	--
4	27	24	3	3	0	--	--	--
5	22	8	14	14	0	--	--	--
6	8	4	4	4	0	--	--	--
7	14	7	7	7	-1	1.5	1	-1.5
8	2	5	-3	3	1	1.5	2	1.5
9	6	6	0	0	-2	4	3	-4
10	18	32	-14	14	2	4	4	4
11	29	35	-6	6	2	4	5	4
12	1	1	0	0	-3	7.5	6	-7.5
13	24	11	13	13	3	7.5	7	7.5
14	11	2	9	9	3	7.5	8	7.5
15	4	13	-9	9	3	7.5	9	7.5
16	26	33	-7	7	-4	11	10	-11
17	23	12	11	11	-4	11	11	-11
18	7	16	-9	9	4	11	12	11
19	17	26	-9	9	-5	13	13	-13
20	12	17	-5	5	-6	14.5	14	-14.5
21	9	9	0	0	6	14.5	15	14.5
22	31	29	2	2	-7	16.5	16	-16.5

Tabla 16. Desarrollo del método de Wilcoxon (continuación)

1	2	3	4	5	5.1	6	6.1	7
ID	**X_A**	**X_B**	**Original X_A - X_B**	**Valor absoluto X_A - X_B**	**X_A - X_B Ordenado con signo**	**Jerarquía del valor absoluto de X_A - X_B**	**Consecutivo de apoyo**	**Jerarquía con signos**
23	20	19	1	1	7	16.5	17	16.5
24	25	22	3	3	-9	20	18	-20
25	32	21	11	11	-9	20	19	-20
26	19	10	9	9	-9	20	20	-20
27	33	27	6	6	9	20	21	20
28	28	28	0	0	9	20	22	20
29	35	23	12	12	11	23.5	23	23.5
30	13	15	-2	2	11	23.5	24	23.5
31	16	20	-4	4	12	25	25	25
32	21	18	3	3	13	26	26	26
33	3	25	-22	22	-14	27.5	27	-27.5
34	30	31	-1	1	14	27.5	28	27.5
35	5	3	2	2	-15	29	29	-29
36	34	34	0	0	-22	30	30	-30

4.9.2.4 Identificación del 20% de actividades con prioridad alta

En esta sección se muestran los pasos realizados para identificar las actividades del conjunto que serán catalogadas con prioridad alta. Dado que el conjunto de actividades está formado por 36 elementos, el 20% con prioridad alta buscado estaría conformado por 7 actividades.

En primera instancia se seleccionaron las 7 actividades con mayor calificación en la jerarquía presentada en la Tabla 13 de la sección 4.9.2.1 perteneciente al grupo de expertos como se muestra en la Tabla 17:

Tabla 17. Actividades con mayor calificación en la priorización del grupo de expertos

ID de la Actividad	Calificación (0 a 100)
2	92
29	80
36	67
27	66
25	66
22	62
34	61

Enseguida se seleccionaron las primeras 7 actividades con mayor calificación en la jerarquía presentada en la Tabla 14, perteneciente al grupo de practicantes como se muestra en la Tabla 18.

Tabla 18. Actividades con mayor calificación en la priorización de practicantes

ID de la Actividad	Calificación (0 a 100)
2	77
11	75
36	70
16	68
10	68
34	67
3	65

Las tablas anteriores proporcionaron los datos para formar la Tabla 19, que fueron colocados en las columnas 1, 3 y 4. En la columna 2 se colocó la descripción de la actividad para análisis posteriores.

Luego se analizaron las actividades prioritarias de ambos grupos. Donde se marcaron con color verde las actividades con identificadores 2, 36 y 34 como intersecciones de ambos.

Para analizar las actividades que no tuvieron intersecciones se incluyó la columna 6, en la cual se agregó la diferencia en calificaciones de las priorizaciones originales que se mostraron en las Tablas 13 y 14 de la sección 4.9.2.1

Para definir las otras 4 actividades faltantes, que completaran el 20% con prioridad alta, se seleccionaron las actividades restantes que fueron identificadas con prioridad alta por el grupo de expertos debido a que se conformó por miembros con mayor experiencia y preparación en Ingeniería de Software que el equipo de practicantes. Estas actividades han sido marcadas con color azul.

Las actividades en amarillo son aquellas que estuvieron dentro del conjunto de actividades prioritarias del grupo de expertos y que no estuvieron dentro del conjunto de actividades prioritarias del grupo de practicantes.

Tabla 19. Comparación de las 7 actividades con calificación más alta en la priorización de los dos grupos

1	2	3	4	5	6
Actividad	**Descripción de la actividad**	**Grupo de Expertos**	**Grupo de Practicantes**	**Coinciden**	**Diferencia en puntos de calificación**
2	Las metas del proyecto y del negocio son entendidas	Si	Si	Si	
29	Condiciones de prueba han sido incluidas en la información de los atributos de calidad	Si	No	No	8 más para el grupo de expertos
36	Riesgos en los requisitos de atributos de calidad han sido identificados y se ha obtenido una lista priorizada de escenarios de fallo	Si	Si	Si	
27	Prototipos rápidos han sido utilizados para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso	Si	No	No	8 más para grupo de expertos
25	Casos de prueba han sido incluidos en las historias de usuario o especificaciones de casos de uso	Si	No	No	15 más para el grupo de expertos

Tabla 19. Comparación de las 7 actividades con calificación más alta en la priorización de los dos grupos (continuación)

1	2	3	4	5	6
Actividad	**Descripción de la actividad**	**Grupo de Expertos**	**Grupo de Practicantes**	**Coinciden**	**Diferencia en puntos de calificación**
22	Narrativas para explorar la funcionalidad del sistema han sido elaboradas, incluyendo lo que sucede antes y después de cada acción	Si	No	No	1 más para el grupo de practicantes
34	Retroalimentación del producto es obtenida durante el desarrollo	Si	No	Si	
11	Las restricciones técnicas y de negocio han sido identificadas	No	No	No	16 más para el grupo de practicantes
16	El sistema o desarrollo ha sido clasificado de acuerdo a su importancia táctica o estratégica	No	No	No	13 más para el grupo de practicantes
10	Las reglas de negocio han sido consideradas como fuente de requisitos emergentes	No	No	No	20 más para el grupo de practicantes
3	Información sobre los usuarios ha sido recolectada y se han identificado los diferentes perfiles que juegan los involucrados en el proyecto	No	No	No	24 más para el grupo de practicantes

A partir de la tabla anterior tenemos que las actividades 2, 29, 36, 27, 25, 22 y 34 serán marcadas con prioridad alta dentro del conjunto.

Finalmente se obtuvo la lista de actividades catalogadas como se muestra en la Tabla 20, en la cual se marca con verde las actividades con prioridad alta y con azul las de prioridad baja.

Tabla 20. Conjunto de actividades catalogadas con prioridad alta-baja

ID	Actividad	Prioridad
1	La definición del problema elaborada es corta y nítida	Baja
2	**Las metas del proyecto y del negocio son entendidas**	**Alta**
3	Información sobre los usuarios ha sido recolectada y se han identificado los diferentes perfiles que juegan los involucrados en el proyecto	Baja
4	Los usuarios finales e incluso clientes han sido estudiados en su hábitat natural	Baja
5	La obtención de atributos de calidad ha sido incluida como parte del rol de arquitecto de software	Baja
6	Los atributos de calidad relevantes han sido identificados	Baja
7	Los drivers arquitecturales han sido identificados en los atributos de calidad y escenarios de funcionalidad	Baja
8	Se ha aprendido sobre los atributos de calidad comunes del dominio	Baja
9	El esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad ha sido estimado	Baja
10	Las reglas de negocio han sido consideradas como fuente de requisitos emergentes	Baja
11	Las restricciones técnicas y de negocio han sido identificadas	Baja
12	La lista de tecnologías aprobadas por la compañía ha sido revisada	Baja
13	Las dependencias con sistemas externos han sido revisadas	Baja
14	La plataforma de despliegue destino ha sido revisada	Baja
15	Las características del equipo son consideradas cuando se está arquitectando un sistema de software	Baja
16	El sistema o desarrollo ha sido clasificado de acuerdo a su importancia táctica o estratégica	Baja
17	Los principios de desarrollo y/o arquitectura han sido identificados y seleccionadas	Baja
18	Los riesgos del proyecto han sido identificados	Baja
19	Los riesgos del proyecto han sido priorizados	Baja
20	Riesgos en los requisitos han sido identificados	Baja
21	La lista de deseos de los involucrados sobre lo que necesitan que el sistema haga ha sido obtenida	Baja
22	**Narrativas para explorar la funcionalidad del sistema han sido elaboradas, incluyendo lo que sucede antes y después de cada acción**	**Alta**
23	Las restricciones y reglas de negocio han sido incluidas en historias de usuario	Baja
24	Los riesgos identificados han sido incluidos en la lista del producto o backlog	Baja

Tabla 20. Conjunto de actividades catalogadas con prioridad alta-baja (continuación)

ID	Actividad	Prioridad
25	**Casos de prueba han sido incluidos en las historias de usuario o especificaciones de casos de uso**	**Alta**
26	Las historias de usuario han sido priorizadas	Baja
27	**Prototipos rápidos han sido utilizados para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso**	**Alta**
28	Los atributos de calidad relevantes han sido priorizados	Baja
29	**Condiciones de prueba han sido incluidas en la información de los atributos de calidad**	**Alta**
30	Los escenarios de atributos de calidad han sido priorizados	Baja
31	La aparición de requisitos emergentes es soportada durante el desarrollo	Baja
32	Las restricciones han sido utilizadas en la prueba de escenarios de atributos de calidad	Baja
33	Las restricciones organizacionales han sido priorizadas	Baja
34	**Retroalimentación del producto es obtenida durante el desarrollo**	**Alta**
35	Todos los involucrados acuden a las mismas reuniones	Baja
36	**Riesgos en los requisitos de atributos de calidad han sido identificados y se ha obtenido una lista priorizada de escenarios de fallo**	**Alta**

4.9.2.5 Hallazgos

- Como se mostró en la tabla 20, sólo tres de las actividades fueron comunes para ambos grupos. Sin embargo, las 4 actividades restantes seleccionadas de la priorización del equipo de expertos muestran en su mayoría menor diferencia entre las calificaciones dadas en la priorización por ambos grupos de las 11 actividades que contiene.
- Las actividades 29 y 25 que hablan sobre casos de prueba en atributos de calidad e historias de usuario han sido identificadas por el grupo de expertos como actividades con relevancia dentro del manejo de requisitos de software. Aunque es poca la diferencia en la calificación obtenida en las priorizaciones de los dos grupos, estas actividades no aparecen dentro del 20% con mayor calificación del grupo de practicantes. Lo que puede ser indicador de que otras actividades reciben mayor

atención de los miembros del grupo y aspectos relacionados con actividades de pruebas pueden ser dejados de lado en su proceso de desarrollo.

- A diferencia de las actividades 29, 27, 25 y 22 que son actividades técnicas del desarrollo de software las actividades 3, 10, 11 y 16 marcadas con amarillo en la Tabla 19 y que han sido dejadas fuera del 20% con prioridad alta son actividades no técnicas en su mayoría.
- Respecto a las proposiciones e hipótesis, en la tabla 21 se resumen los hallazgos:

Tabla 21. Hallazgos sobre proposiciones e hipótesis

Pregunta de investigación	Proposición con hipótesis	Resultados	Hallazgos
PI1: ¿Las priorizaciones realizadas por los dos grupos son similares en su distribución?	**Proposición 1:** Si dos grupos de individuos priorizan las actividades, entonces se encontrará una relación entre ambas priorizaciones. **Hipótesis 1:** Diferentes grupos de individuos con diferentes niveles de experiencia en desarrollo de software pueden identificar las mismas actividades con prioridad alta y baja pertenecientes a un mismo conjunto.	**Proposición 1:** Verdadera **Hipótesis 1:** Verdadera	**Proposición 1:** Al analizar los dos grupos se encontró que tendieron a priorizar en lo general de la misma forma el conjunto de actividades. Con un 95% de probabilidad y un margen de error de 0.05. **Hipótesis 1:** Hubo diferencias significativas al identificar las actividades con prioridad alta. Especialmente el grupo de practicantes dejó de lado las actividades relacionadas a pruebas de software.
PI2: ¿Cuáles actividades forman el 20% que aportan el 80% del valor según el principio de Pareto, y que serán catalogadas con prioridad alta dentro del conjunto de actividades?	**Proposición 2**: El 20% de actividades catalogadas con prioridad alta en ambos grupos será el mismo.	**Proposición 2:** Falsa	**Proposición 2:** Se encontraron actividades con prioridad alta comunes a los dos grupos, sin embargo la priorización no fue exactamente igual.
PI3: ¿Cuáles actividades forman el 80% que serán catalogadas con prioridad baja y aportan el 20% del valor según el principio de Pareto, dentro del conjunto de actividades?	**Proposición 3**: El 80% de actividades catalogadas con prioridad baja de ambos grupos será el mismo.	**Proposición 3:** Falsa	**Proposición 3:** Al no ser igual el subconjunto de actividades con prioridad alta en los dos grupos; en consecuencia el subconjunto de actividades con prioridad baja tampoco es el mismo.

4.10. Estrategia de selección de casos y/o unidades de análisis

Para la realización de la encuesta en línea todas las personas involucradas se seleccionaron a partir de las relaciones existentes con el equipo responsable de esta investigación. Las unidades de análisis, en este caso cada individuo que participó en la priorización de las actividades fue seleccionado de acuerdo a sus características para ajustarse a las necesidades del caso de estudio.

Durante el estudio, los resultados obtenidos de una primera versión de la priorización de las actividades realizada con 5 diferentes grupos fue desechada debido a que estaba basada en un subconjunto de 9 elementos del conjunto de 36 actividades, lo cual no permitiría identificar con facilidad las actividades con mayor prioridad. Aunado a esto, no se empleó la herramienta *(Wiki Surveys)* utilizada con el grupo de expertos y las priorizaciones con los equipos de practicantes posteriores se hicieron utilizando tarjetas físicas, lo cual hacía imposible la comparación debido a la diferencia en los métodos con los que fueron extraídos los datos.

4.11 Estrategia de selección de datos

Debido a que se utilizó una herramienta elaborada por un tercero *(WiKi Surveys)* para realizar la encuesta en línea, los datos de la priorización de las actividades fueron obtenidos de manera condensada.

Como se mencionó en la sección 4.8 de este capítulo, se obtuvo una lista con las actividades y su calificación en una escala de 0 a 100 en un archivo CSV. Por lo cual, no fue necesaria la realización de otro tipo de estrategias relacionadas a la selección de datos.

4.12 Estrategia de replicación

En este caso de estudio no se intenta replicar algún trabajo previo. Por otro lado, debido a que se describe la fuente utilizada para obtener los datos, las características de los individuos que participaron en la encuesta en línea, el método utilizado para analizar los datos y la herramienta utilizada para validar la correctitud de los resultados obtenidos en el análisis, este caso de estudio puede ser replicado con facilidad.

4.13 Aseguramiento de la calidad, validación y disponibilidad

4.13.1 De los datos

Dado que la encuesta realizada se llevó a cabo utilizando una herramienta elaborada por un tercero y generó como producto una lista priorizada de actividades con la calificación obtenida, no hubo ningún tratamiento o transcripción de los datos en la recolección o antes de iniciar la fase de análisis de los datos.

4.13.2 Del análisis de los datos

Para asegurar que los resultados obtenidos de la aplicación de la prueba de Wilcoxon fueron correctos, se compararon con los resultados arrojados por la herramienta en línea *VassarStas: Website for Statistical Computation* (Lowry, 1998). Para esto se seleccionó el método de la prueba necesitada, en este caso la prueba de rangos de Wilcoxon. Además, se ingresaron los datos de las posiciones de cada actividad en la priorización provenientes de la Tabla 15. En la Tabla 22 se comparan los resultados obtenidos del experimento realizado manualmente con los resultados arrojados por la herramienta para comprobar la correctitud de los resultados de la prueba.

Tabla 22. Comparación de los datos obtenidos de la prueba de Wilcoxon realizada manualmente con los resultados obtenidos de la herramienta web de VassarStats

Variable	Resultados obtenidos en la prueba realizada manualmente	Resultados obtenidos en la herramienta web
W	14	14
N	30	30
w	97.23	
Z	0.14	0.14
P(1-tail)		0.4443
P(2-tail)		0.8887

4.14 Lecciones aprendidas

En esta sección se describen las principales lecciones aprendidas de la realización del caso de estudio.

4.14.1 Selección de los individuos para la aplicación de la encuesta

Como se mencionó anteriormente, los individuos que participaron en la realización de la encuesta para la priorización de las actividades fueron seleccionados en base a la red de contactos de los investigadores involucrados en el proyecto de tesis. Este aspecto permitió acelerar la recolección de los datos. Sin embargo, en las ocasiones en las que se necesitó contar con los grupos de manera presencial el reunirlos en el mismo lugar fue una tarea difícil. Para la aplicación de encuestas en línea, tal vez, una buena alternativa habría sido enviar los datos de acceso a la encuesta vía correo electrónico con instrucciones y que los participantes la llenaran por cuenta propia sin necesitar la presencia física.

4.14.2 Selección de las herramientas de soporte

En este caso de estudio se seleccionó la herramienta *Wiki Surveys* para llevar a cabo la encuesta debido a que cumple con ciertas características que encajan con el tipo de encuesta aplicada. Sin embargo, antes de seleccionar una herramienta desarrollada por terceros es importante revisar los datos que arrojará, los formatos en los cuales puede ser exportada la información y finalmente, el aspecto más importante es el preprocesamiento que se le realiza a los datos y cómo esto puede afectar los análisis posteriores de la información. Esto es importante debido a que es necesario conocer cómo funcionan las herramientas utilizadas para poder replicar los experimentos en el futuro, aun cuando no se tenga la herramienta disponible.

4.14.3 Método de recolección de datos

Para el grupo de expertos, la encuesta se realizó utilizando una herramienta en línea, mientras para el grupo practicantes se utilizaron tarjetas físicas y un método de priorización diferente al utilizado por la herramienta. Al tratar de comparar los resultados obtenidos se encontraron algunas incongruencias que llevaron a repetir la encuesta con el segundo grupo utilizando ahora la herramienta en línea que había sido utilizada con el primero. La lección aprendida en este punto, es que desde el diseño del caso de estudio se deben establecer los mismos mecanismos de recolección de datos para todos los objetos de estudio.

4.14.4 Definición del método de análisis de los datos

Aunque desde el inicio del diseño del caso de estudio se conocía que la herramienta en línea proporciona una lista de actividades priorizadas con sus calificaciones, no se profundizó en estudiar qué métodos de análisis de datos serían aplicables según las características de los datos obtenidos. Fue hasta que se obtuvieron los resultados, que se investigaron posibles técnicas a aplicar. Este aspecto debe ser definido en el diseño del caso de estudio y por ende antes de realizar la recolección de los datos.

5. RESULTADOS, CONCLUSIONES Y TRABAJO FUTURO

En este último capítulo se resume el trabajo realizado en este proyecto, qué tanto fueron cubiertos los objetivos definidos, las conclusiones obtenidas y finalmente algunas líneas de trabajo futuro que sería interesante explorar.

5.1. Resultados

El manejo de los requisitos es inherente al desarrollo de software. En este proyecto se ha recopilado un conjunto de actividades, técnicas, herramientas y artefactos relacionados a esta área que tienen como propósito ser de utilidad tanto para expertos, desarrolladores junior, así como estudiantes.

Para lograrlo se cumplieron varios objetivos que fueron planteados en el inicio del proyecto, mismos que son mencionados nuevamente a continuación, a la vez que se describe el grado en que fueron cubiertos.

- **Objetivo 1:** Identificar y analizar actividades relacionadas a ingeniería de requisitos sugeridas por los autores *J. Coplien y G. Bjørnvig, S. Brown y G. Fairbanks* en sus libros enfocados al desarrollo de arquitectura de software:
    - *Lean Architecture for Agile Software Development (2010)*
    - *Software Architecture for Developers (2016)*
    - *Just Enough Software Architecture (2010)*

Para lograr este objetivo se leyeron y analizaron detalladamente los tres libros anteriores y se identificaron actividades, técnicas, herramientas y artefactos. Esto ha sido desarrollado en la sección 3.1 del capítulo 3.

Como entregables de este objetivo se tienen tres matrices que se encuentran en hojas de cálculo llamadas *Anexo A*, *Anexo B* y *Anexo C* producto del análisis individual de cada enfoque de los autores, las cuales se pueden encontrar también en el archivo llamado ANEXOS1.xlsx albergado en la siguiente dirección web: https://github.com/mayanin/guia-de-actividades-de-ingenieria-de-requisitos.

- **Objetivo 2:** Clasificar y agrupar las actividades encontradas a partir del análisis de los tres enfoques revisados.

Las actividades de clasificación y agrupación se realizaron en conjunto y como acciones seguidas a la identificación y análisis parte del objetivo 1. Para lograrlo se utilizaron también matrices en hojas de cálculo, cuyo desarrollo es descrito en la sección 3.1 del capítulo 3.

Los entregables de este objetivo son las matrices que se encuentran en las hojas llamadas *Anexo D* y *Anexo E* dentro del archivo ANEXOS1.xlsx.

- **Objetivo 3:** Proponer una guía que integre además de actividades sugerencias de técnicas, herramientas y artefactos para complementar su aplicación.

Después de la realización de diversos prototipos, una primera versión completada de la guía ha sido agregada en la sección 3.7 del capítulo 3. Donde se muestra cómo quedó el diseño del conjunto de tarjetas que contienen cada una de las 36 actividades con su información relacionada y su modo de uso.

En el repositorio utilizado para albergar los entregables de este proyecto de tesis se han agregado también las tarjetas y el instructivo pertenecientes a la guía realizada. Estos se encuentran dentro de la carpeta nombrada Entregables.

- **Objetivo 4:** Validar la guía realizada por medio de un caso de estudio.

Específicamente con la realización del caso de estudio se comparó y validó la prioridad de las actividades en colaboración con dos grupos de individuos con diferentes niveles de conocimiento y experiencia en ingeniería de software. Esto ha sido abordado a profundidad en el capítulo 4.

Entre los entregables de este objetivo se encuentran los datos recabados y el análisis realizado a los mismos, los cuales se encuentran en las hojas llamadas *datos* y *análisis* dentro del archivo ANEXOS2.xlsx también disponible en el repositorio indicado en el objetivo 1.

- **Objetivo 5:** Publicar la guía en un sitio web disponible en internet para facilitar su diseminación de manera digital.

Por razones externas a este proyecto de tesis la diseminación de la guía de manera digital en un sitio web personalizado ha quedado en proceso y se espera en el futuro próximo ponerla a disposición de la comunidad en internet de manera gratuita.

Sin embargo en el repositorio utilizado para albergar los entregables de este proyecto de tesis se han agregado también las tarjetas y el instructivo pertenecientes a la guía realizada como se indicó en los entregables del objetivo 3.

5.2 Conclusiones

De manera general se han podido identificar diferentes tipos de conclusiones las cuales han sido desarrolladas en los siguientes puntos.

5.2.1 Sobre el proceso de generación de la guía

Esta etapa sin lugar a dudas ha sido la más compleja del proyecto debido a que involucró un trabajo intelectual intenso y atención en la lectura, identificación y clasificación de las actividades y demás elementos que las complementan. Entre más fuentes se tengan mayor será la complejidad y tiempo requerido, por este motivo desde el inicio del proyecto fueron definidas las fuentes. El no haberlas acotado de esta manera podría haber prolongado el trabajo de búsqueda de materia prima para la generación de la guía por un largo tiempo.

Una vez que fue obtenido el conjunto de actividades, fue necesario realizar actividades de diseño para definir de qué manera sería presentada la información y cómo podría lograrse obtener un producto que además de brindar información valiosa para sus usuarios fuera atractivo desde el punto de vista estético. Por esta razón fue necesaria la participación de un diseñador profesional. Entonces, podemos concluir que la elaboración de este tipo de trabajos no sólo se limita a ingenieros de software sino también se puede extender a otras áreas como en este caso el diseño gráfico profesional.

5.2.2 Sobre el desarrollo del caso de estudio

El desarrollo del caso de estudio ha sido también una actividad que ha dejado diversas lecciones aprendidas como:

La experiencia puede ser un factor fundamental al decidir cuáles actividades deberíamos o no realizar cuando se está desarrollando software o simplemente al decidir cuáles actividades son más valiosas para el proyecto en el que se está trabajando dentro de un conjunto de definido.

A pesar de que los dos grupos que intervinieron en el caso de estudio tenían diferentes niveles de experiencia, se encontró que de manera general tendieron a priorizar las

actividades de la misma forma, lo que en consecuencia ayudó a identificar de manera más clara el 20% de las actividades que aportará el 80% del valor en los proyectos de software.

Por otro lado, al comparar a fondo el 20% de actividades con prioridad alta de cada grupo se detectó que las actividades que no coincidían en prioridad para los expertos y los practicantes tenían que ver con pruebas de software. Los expertos asignaron una mayor prioridad a las pruebas, mientras que para los practicantes las pruebas quedaron en diferentes posiciones fuera del top 20%. Esto, como consecuencia puede mostrar signos de problemas en la ejecución de su trabajo habitual. Según Brooks (1995, p. 20) el tiempo de un desarrollador de software debería ser distribuido de la siguiente manera:

- ⅓ planificación
- ⅙ codificación
- ¼ pruebas de componentes
- ¼ pruebas de sistema

Según esta regla, la aplicación y desarrollo de diferentes tipos de pruebas podría llevarse hasta el 50% del tiempo dedicado al desarrollo de software.

5.2.3 Sobre el producto obtenido

Después de la realización de diversos prototipos se obtuvo una versión lista para publicar y diseminar en internet. Como se mencionó anteriormente, para generar esta última versión se ha necesitado la colaboración de un diseñador profesional. Por lo que, realizar una guía de éste ámbito no sólo necesita del trabajo de ingenieros de software sino un equipo de profesionales de diversas áreas.

A pesar de que no ha sido posible utilizar este juego de tarjetas en un proyecto real, el desarrollo del caso de estudio permitió ver el interés de los desarrolladores de software por contar con una herramienta que les facilite y disminuya el esfuerzo de obtener nuevo conocimiento que ha sido previamente recopilado y puesto en cada tarjeta.

5.2.4 Sobre el proyecto en general

Aunque se podría intuir por la secuencia en que ha sido documentado este proyecto de tesis que ha sido un proceso realizado de manera secuencial, no ha sido así. El desarrollar parte por parte la guía ha necesitado iterar diversas veces entre actividades para obtener finalmente un producto tangible de valor. El desarrollar un trabajo de este tipo necesita irse puliendo a través del tiempo.

De este proyecto de tesis se obtuvieron diversos beneficios entre los que se encuentran:

- La experiencia profesional de interactuar con otros profesionales y conocer sus puntos de vista sobre las actividades analizadas y el producto de esta tesis en general.
- El producto obtenido de este trabajo ha sido bien recibido por los grupos de individuos participantes en los ejercicios de validación realizados, lo que da la pauta a suponer que puede ser bien recibido por la comunidad al estar disponible en internet.
- Como cualquier cosa existente, el trabajo realizado puede ser mejorado de diversas maneras, como por ejemplo incluyendo más participantes en su desarrollo o extendiendo su alcance a otras etapas del ciclo de vida.
- Las restricciones han jugado un papel importante dentro de este proyecto, ya sean de tiempo, técnicas o monetarias deben tomarse en cuenta para establecer hasta donde se puede llegar.

Otro aspecto importante, es que debido a las diversas perspectivas abordadas por los autores de los libros analizados el conjunto de actividades puede utilizarse tanto en entornos de desarrollo ágil como tradicional. Además de servir como referencia para aquellos desarrolladores que no apliquen ninguna metodología en particular o no tengan conocimientos profundos del manejo de requisitos de software.

5.3 Trabajo futuro

Existen diversas áreas del proyecto que pueden ser exploradas como líneas de trabajo futuro. A continuación se mencionan algunas de ellas:

- **Alcance.** Como se mencionó anteriormente, el trabajo realizado podría aplicarse a otras etapas del ciclo de vida como diseño, construcción y pruebas de software.
- **Recursos.** La cantidad de recursos utilizados como fuentes de información podría ser ampliado a incluir más libros especializados en arquitectura ágil disponibles en el momento en que se realicen las nuevas investigaciones, publicaciones en revistas indexadas, blogs, etc. Esto, con la finalidad de alcanzar un mayor número de actividades, técnicas, herramientas y artefactos que hagan el contenido de la guía más completa.
- **Validación.** Promover el uso de la guía por equipos de desarrollo en proyectos reales para validar su utilidad y obtener retroalimentación acerca de su contenido, facilidad de uso y formato en el que ha sido presentada. Esto podría ser desarrollado a manera de casos de estudio ya que ofrecen una gran flexibilidad en el contexto de ingeniería de software y cada vez son más aceptados por la comunidad.
- **Difusión.** Publicación del contenido de la guía en internet y la obtención de retroalimentación de sus usuarios para la realización de análisis posteriores sobre su utilidad, facilidad de uso, etc. También se podría realizar un Objeto Digital de Aprendizaje en Materia de Arquitectura de Software o un taller.
- **Presentación.** El presentar la guía en formato de juego podría ser una opción para promover el interés en su uso de estudiantes y profesionales. Por otro lado, también podría cambiarse el formato de tarjetas físicas a una aplicación móvil o web.
- **Encuestas.** En este caso se compararon dos grupos de 17 individuos cada uno, divididos según su perfil. Por su puesto, podría hacerse más extensiva la comparación de los grupos buscando la participación de más profesionales del área de desarrollo de software haciendo crecer los tamaños de las muestras.

5.4 Otros productos de trabajo

Como parte de este proyecto de tesis se publicaron dos artículos, de los cuales se han agregado sus referencias a continuación:

Hernandez-Reveles, J., Sobrevilla-Dominguez, G., Velasco-Elizondo, P., & Soriano-Grande, S. (2016). Adding agile architecture practices to a Cyber-Physical System development. 2016 International Conference On Software Process Improvement (CIMPS). http://dx.doi.org/10.1109/cimps.2016.7802800.

Sobrevilla, G., Hernández, J., Velasco, P., & Soriano, S. (2017). Aplicando Scrum y Prácticas de Ingeniería de Software para la Mejora Continua del Desarrollo de un Sistema Ciber-Físico. Recibe - Revista Electrónica De Computación, Informática, Biomédica Y Electrónica, 6(1), 1-15. Disponible en: http://recibe.cucei.udg.mx/Recibe/index.php/Recibe/issue/view/16/showToc.

ANEXOS

En esta sección se incluyen los diferentes anexos referenciados a lo largo del documento de tesis.

Anexo A

Análisis del libro *Software Architecture for Developers* de Brown.

Pasos realizados:

1) Identificar prácticas, artefactos y técnicas y herramientas en el texto
2) Agregar las oración tal cual el autor en prácticas
3) Una vez completada la tabla identificar las actividades equivalentes y complementar la oración con ambas. para evitar prácticas repetidas

Nota. Esta tabla está pensada para poner tal cual escribe el autor una actividad, posteriormente en la matriz presentada en el Anexo D se descomponen las oraciones en las tareas, técnicas, herramientas y artefactos de manera más detallada.

No.	Actividades	Técnicas y Herramientas	Artefactos	Página (T1 = tomo 1, T2 = tomo 2)
1	Entender de las metas del proyecto y de negocio	No especificado	No especificado	T1, 13, 14
2	Obtener la *"lista de deseos"* de los involucrados acerca de lo que necesitan que el sistema haga	Aplicación de las técnicas: 1) Historias de usuario 2) Casos de uso 3) Especificaciones de requisitos tradicionales 4) Criterios de aceptación, etc.	Lista de deseos	T1, 55
3	Elaborar una lista de características o historias de usuario	No especificado	Lista de producto de Scrum	T1, 53
4	Priorizar requisitos funcionales, historias de usuario, etc.	1) Uso de la técnica MosCow *(Must have, Should have, Could have, and Won't have but would like)* para priorizar 2) Explicación las compensaciones o *"trade-offs"* al cliente puede ayudar a encontrar la mejor solución para el contexto dado	Historias de usuario	T1, 50
5	Incluir la obtención de atributos de calidad y restricciones como parte del rol de arquitecto de software para asegurar que son considerados	El rol puede ser responsabilidad de una persona o compartido por un grupo	No especificado	T1, 13, 14

Continuación Anexo A

No.	Actividades	Técnicas y Herramientas	Artefactos	Página (T1 = tomo 1, T2 = tomo 2)
6	Preguntar a los usuarios cuáles atributos de calidad son necesarios utilizando preguntas que nos brinden información acerca de ellos	1) Consideración de los siguientes atributos de calidad: desempeño, escalabilidad, disponibilidad, seguridad, recuperación de desastres, accesibilidad, monitoreo, auditoría, flexibilidad, extensibilidad, mantenibilidad, conformidad legal y/o regulatoria, internacionalización (multilenguaje) y localización (convenciones en monedas, fechas, etc.) 2) Preguntas como las siguientes pueden ayudar: ¿Cuánto tiempo puede tolerar el sistema fuera de línea?, ¿Qué pasa si el sistema falla durante horas de trabajo normal (9 a.m. a 6 p.m.)?, ¿Qué pasa si el sistema falla fuera de las horas normales de trabajo?	Lista de los atributos de calidad aplicables al sistema de software que se está construyendo	T1, 55, 56, 57, 58, 59
7	Aprender acerca de los atributos de calidad comunes en el dominio en el que se está trabajando y enfocarse en ellos al empezar a construir un nuevo sistema o modificar uno existente	No especificado	No especificado	T1, 58
8	Refinar la información obtenida sobre los atributos de calidad asociando por ejemplo, cantidades a los mismos evitando vaguedad en su definición	Preguntas como las siguientes pueden ayudar: ¿Cuántos usuarios concurrentes debería el sistema soportar en promedio?, ¿Cómo intervienen las horas pico?, ¿Qué tiempo de respuesta es considerado aceptable?, ¿El tiempo de respuesta debe ser el mismo en todas las partes del software o sólo para características específicas? además de preguntar sobre el número de usuarios concurrentes, volúmenes de datos y número máximo de respuestas	1) Conjunto específico de requisitos no funcionales, idealmente que se pueden cuantificar explícitamente. 2) Criterios de aceptación para realizar pruebas posteriormente	T1, 59, 60
9	Revisar si existe una lista aprobada de tecnologías por la compañía, a menudo restringen las diferentes tecnologías que la compañía tiene que mantener, operar y comprar licencias	No especificado	Lista de tecnologías aprobadas	T1, 62
10	Revisar si existen otros sistemas que se necesiten integrar con el sistema que se va a construir	Revisión de restricciones organizacionales dictando protocolos y tecnologías a utilizar para los puntos de integración	No especificado	T1, 62

Continuación Anexo A

No.	Actividades	Técnicas y Herramientas	Artefactos	Página (T1 = tomo 1, T2 = tomo 2)
11	Conocer la plataforma de despliegue destino, pues a menudo tiene fuerte impacto en las decisiones de tecnología	No especificado	Lista de restricciones sobre lo que el software puede o no hacer en la plataforma que lo alojará	T1, 63
12	Tomar en cuenta las características de los miembros del equipo cuando se está arquitectando un sistema de software	Conocer, ¿qué tan grande es el equipo de desarrollo?, ¿cuáles son sus habilidades?, ¿qué tan rápido puede escalar el equipo de desarrollo si es necesario?, ¿puedes requerir especialistas, consultoría o entrenamiento de ser necesario?, ¿tendrá el equipo de mantenimiento las mismas habilidades que el equipo de desarrollo?	No especificado	T1, 64
13	Tomar en cuenta si el sistema es parte de una implementación táctica o estratégica	No especificado	Lista de restricciones adicionales o removidas	T1, 64
14	Priorizar las restricciones dado que algunas son más importantes que otras, y puede utilizarse como una ventaja	No especificado	No especificado	T1, 65
15	Identificar los principios que se desean adoptar para introducir claridad y consistencia al código base. Pueden ser principios de desarrollo o arquitectura.	Revisar: 1) Estándares y convenciones de código 2) Uso de pruebas automatizadas 3) Herramientas de análisis estático 3) Uso de patrones arquitectónicos, etc.	No especificado	T1, 66
16	Decidir qué principios adoptar tomando en cuenta el tamaño, la complejidad y las restricciones del ambiente en que se desarrolla el proyecto	1) Estándares de código 2) Pruebas unitarias automatizadas 3) Herramientas de análisis estático, etc.	Una lista de principios puede ayudar a asegurar que todo el mundo está trabajando de la misma forma	T1, 68

Anexo B

Análisis del libro Lean Architecture for Agile Software Development de Coplien.

Pasos realizados:

1) Identificar prácticas, artefactos y técnicas y herramientas en el texto
2) Agregar las oración tal cual el autor en prácticas
3) Una vez completada la tabla identificar las actividades equivalentes y complementar la oración con ambas. para evitar prácticas repetidas

Nota. Esta tabla está pensada para poner tal cual escribe el autor una actividad, posteriormente en la matriz presentada en el Anexo D se de descomponen las oraciones en las tareas, técnicas, herramientas y artefactos de manera más detallada.

Actividades	Técnicas y Herramientas	Artefactos	Página
Recolectar información sobre los usuarios e identificar los diferentes perfiles que juegan los involucrados en el proyecto	Receta de los 4 pasos de Cohn para el Modelado de roles de usuario (Cohn 2004, capítulo 3): 1) Hacer una lluvia de ideas sobre el conjunto inicial de roles de usuario 2) Organizar el conjunto de roles de usuario obtenido 3) Consolidar los roles 4) Refinar los roles	Lista de perfiles de usuario (ejemplo: documentación de las características del usuario, edad, región geográfica, habilidades de lectura, etc.)	171
Iniciar la obtención historias de usuario y proceder a los casos de uso dirigidos por metas, incluyendo escenarios y motivaciones del usuario para la funcionalidad	Elaboración de una lista de historias de usuario, lista de características o escenarios para capturar historias de usuario	1) Historias de usuario 2) Características 3) Casos de uso y escenarios	42
Utilizar especificaciones de casos de uso o historias de usuario agregando, casos de prueba comúnmente al reverso de la historia (Cohn 2004, p. 7), además incluir restricciones y reglas de negocio como historias individuales	1) Agrupación de historias de usuario por áreas de negocio 2) Uso de herramientas para superar las limitaciones de texto de las tarjetas como Version One, Xplanner, Select Scope Manager (Cohn 2004, p. 179)	Casos de uso	167
Explorar la percepción del usuario de la forma del sistema utilizando prototipos en papel, esto puede ser una opción para obtener retroalimentación a un bajo costo y corto tiempo	Prototipos	Prototipos de papel	44,46

Continuación Anexo B

Actividades	Técnicas y Herramientas	Artefactos	Página
Escribir una definición de problema corta y nítida	Tomar como referencia la técnica de definición del problema de Jerry Weinberg: escribir la diferencia entre el estado actual y el estado deseado	Documentación del problema en una o dos sentencias	29
Poner expertos del dominio en un cuarto y realizar una lluvia de historias de usuario	Lluvia de ideas	Historias de usuario	169
Organizar las historias de usuario con una lista ordenada basada en las relaciones entre sí, su costo, el valor del negocio y las presiones de mercado o *"time to market"*	Uso de criterios como costo, valor del negocio y las presiones de mercado	Lista priorizada de historias	169
Obtener retroalimentación del producto para mantener el camino durante su desarrollo debido a los *"requisitos emergentes"*	Bucles de retroalimentación	Producto de software	169
Utilizar narrativas incluyendo lo que sucede antes y después de una acción para entender la motivación del usuario o del negocio	Uso de la combinación de personas y escenarios	1) Narrativa del problema 2) Narrativa de la visión	175
Utilizar guiones gráficos o *"storyboards"* para capturar la visión del flujo de uso de los usuarios y posteriormente validar sus asunciones sobre su comportamiento en la realidad	Capturar la visión del flujo de uso de los usuarios	Guiones gráficos	199
El personal de pruebas debe conocer los requisitos tan bien como los desarrolladores, por lo cual deberían atender las mismas reuniones	Organizar reuniones donde desarrolladores y personal de pruebas puedan acordar requisitos con las personas de negocio	No especificado	59
Estudiar a los usuarios finales e incluso clientes en su hábitat natural, para ganar conocimiento del dominio y aprender sobre el contexto en el cual los requisitos surgen	Diseño Contextual o "*Contextual Design"* (Beyer y Holtzablatt, 1998) ofrece las siguientes técnicas para explorar y capturar modelos mentales de usuario: 1) Investigación del contexto 2) Modelado de la forma de trabajo 3) Consolidación de la forma de trabajo, integrando los diagramas juntos para ver la forma de trabajo de todos los clientes 4) Rediseño de la forma de trabajo 5) Diseño del ambiente de usuario 6) Bosquejos y pruebas con clientes	1) Documentación del contexto 2) Diagramas 3) Bosquejos	60

Continuación Anexo B

Actividades	Técnicas y Herramientas	Artefactos	Página
Soportar requisitos emergentes en los casos de uso a través de desviaciones como extensiones, excepciones, alternativas, variaciones, manejo de errores y requisitos no funcionales	Desviaciones como: extensiones, excepciones, alternativas, variaciones, manejo de errores y requisitos no funcionales	Casos de uso	197
Considerar reglas de negocio como fuente de requisitos emergentes	Documentación de las reglas de negocio	Diccionario de reglas de negocio, glosario, diccionario de datos, o descripción de las reglas de negocio	197, 192

Anexo C

Análisis del libro Just Enough Software Architecture de Fairbanks.

Pasos realizados:

1) Identificar prácticas, artefactos y técnicas y herramientas en el texto
2) Agregar las oración tal cual el autor en prácticas
3) Una vez completada la tabla identificar las actividades equivalentes y complementar la oración con ambas. para evitar prácticas repetidas

Nota. Esta tabla está pensada para poner tal cual escribe el autor una actividad, posteriormente en la matriz presentada en el Anexo D se de descomponen las oraciones en las tareas, técnicas, herramientas y artefactos de manera más detallada.

Actividades	Técnicas y Herramientas	Artefactos	Página
Buscar en los requisitos aquellos que influenciarán en las decisiones de arquitectura	Una vez que se encuentren, preguntarse cómo el sistema hará eso, y si es algo en lo que su arquitectura ayudará o dificultará	Lista de requisitos	27
Elicitar riesgos en los atributos de calidad y obtener una lista priorizada de escenarios de fallo	Talleres de atributos de calidad o "Quality Atributte Workshops" y Cuestionarios de taxonomía de riesgos o Taxonomy-Based Questionnaire (TBQ) (Carr et al., 1993)	Lista de riesgos	41
Identificar riesgos en los requisitos (en la forma que tengan)	No especificado	Lista de riesgos	41
El equipo debe identificar riesgos del proyecto y anexarlos a la lista del producto para que el dueño de producto o *"Product Owner"* los pueda priorizar	Una forma de incorporar riesgos en el proceso ágil es convertir la lista del producto en lista de características y riesgos del producto. El dueño del producto agrega características y el equipo de software agrega riesgos técnicos. El equipo de software debe ayudar al dueño del producto a entender los riesgos técnicos y priorizar adecuadamente la lista de producto o "backlog"	1) Lista del producto 2) Lista de riesgos	57, 90
Identificar riesgos y priorizarlos	No especificado	Lista de características y riesgos	57
Identificar atributos de calidad relevantes y priorizarlos	No especificado	No especificado	70
Priorizar atributos de calidad unos de otros	No especificado	No especificado	142

Continuación Anexo C

Actividades	Técnicas y Herramientas	Artefactos	Página
Utilizar casos de uso, escenarios y diagrama de contexto para mostrar funcionalidad	Técnica de casos de uso, escenarios y diagrama de contexto	Conjunto de casos de uso, escenarios y diagramas de contexto	143
Describir requisitos de atributos de calidad utilizando la técnica ligera *"escenarios de atributos de calidad"*	Técnica ligera de escenarios de atributos de calidad	No especificado	150
Cada escenario de atributo de calidad debe ser calificado (o ponderado) tanto por las partes interesadas como por los desarrolladores	Utilizar alguna técnica de ponderación para obtener los escenarios relevantes	No especificado	151
Identificar los drivers arquitecturales utilizando ponderaciones como prioridad alta, media, baja	1) Aplicación de prioridad baja, media y alta como criterios de selección 2) Atributos de calidad alta, serán identificados como drivers arquitecturales	No especificado	151
Utilizar las restricciones para asegurarse que los escenarios de atributos de calidad pueden ser cumplidos	No especificado	Escenarios de funcionalidad	156
Utilizar escenarios de funcionalidad y casos de uso para expresar comportamiento del sistema	UML	Escenarios de funcionalidad	234
Utilizar restricciones o invariantes para especificar como el sistema debería o no debería ser	UML o texto	Lista de restricciones	239
Especificar condiciones de prueba para los atributos de calidad que se desee mediante escenarios de atributos de calidad	No especificado	Escenarios de funcionalidad	247
Obtener los escenarios de atributos de calidad	Taller de atributos de calidad o *"Quality Atributte Workshop"*	No especificado	250
Seleccionar los drivers arquitecturales de la lista de atributos de calidad y los escenarios de funcionalidad	No especificado	1) Lista de atributos de calidad 2) Escenarios de funcionalidad	250

Anexo D

En esta tabla se comparan las tareas de los tres autores, y se obtiene una genérica que englobe sus perspectivas. La columna tipo indica si la tarea es complementaria, al no repetirse por otro autor o es equivalente, si aparece dos o tres veces.

Pasos realizados:

- Una actividad se indicará como **complementaria** al conjunto cuando sólo haya sido mencionada por uno de los autores.
- Una actividad se indicará como **equivalente** al conjunto cuando haya sido mencionada por dos o más autores.
- La **actividad genérica** trata de englobar las perspectivas de los autores en una misma oración.

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
No encontrada	Escribir una definición de problema corta y nítida	No encontrada	Elaborar una definición de problema corta y nítida	Elaborar una definición de problema corta y nítida **para saber cuándo el trabajo ha terminado**	Complementaria
Entender las metas del proyecto y las metas de negocio	No encontrada	No encontrada	Entender los objetivos del proyecto y de negocio	Entender los objetivos del proyecto y de negocio **para establecer los resultados esperados del proyecto**	Complementaria
No encontrada	Recolectar información sobre los usuarios e identificar los diferentes perfiles que juegan los involucrados en el proyecto	No encontrada	Recolectar información sobre los usuarios	Recolectar información sobre los usuarios para su análisis posterior	Complementaria

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
No encontrada	Recolectar información sobre los usuarios e identificar los diferentes perfiles que juegan los involucrados en el proyecto	No encontrada	Identificar los diferentes perfiles de usuario que juegan los involucrados en el proyecto	Identificar los diferentes perfiles de usuario que juegan los involucrados en el proyecto para definir el conjunto de perfiles que intervendrán en el sistema de software a construir	Complementaria
No encontrada	Estudiar a los usuarios finales e incluso clientes en su hábitat natural, para ganar conocimiento del dominio y aprender sobre el contexto en el cual los requisitos surgen	No encontrada	Estudiar a los involucrados en su hábitat natural	Estudiar a los involucrados en su hábitat natural para ganar conocimiento del dominio y aprender sobre el contexto en el cual los requisitos surgen	Complementaria
Obtener la *"lista de deseos"* de los involucrados acerca de lo que necesitan que el sistema haga	No encontrada	No encontrada	Obtener la *"lista de deseos"* de los involucrados	Obtener la lista de deseos de los involucrados **para conocer lo que necesitan que el sistema haga**	Complementaria
No encontrada	Poner expertos del dominio en un cuarto y realizar una lluvia de historias de usuario	No encontrada	Realizar lluvia de historias de usuario en colaboración con expertos de dominio	Realizar lluvia de historias de usuario en colaboración con expertos de dominio para explorar el espacio de los requisitos	Complementaria

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
Elaborar una lista de características o historias de usuario	1) Iniciar la obtención historias de usuario y proceder a los casos de uso dirigidos por metas, incluyendo escenarios y motivaciones del usuario para la funcionalidad 2) Utilizar narrativas en las historias incluyendo lo que sucede antes y después de una acción para entender la motivación del usuario o del negocio	Utilizar escenarios de funcionalidad y casos de uso para expresar comportamiento del sistema	Incluir lo que sucede antes y después de cada acción en las historias de usuario	Incluir lo que sucede antes y después de cada acción en las historias de usuario **para entender la motivación del usuario o del negocio**	Equivalente
No encontrada	Utilizar especificaciones de casos de uso o historias de usuario agregando, casos de prueba comúnmente al reverso de la historia (Cohn 2004, p. 7)	No encontrada	Agregar criterios de aceptación a las historias de usuario	Agregar criterios de aceptación a las historias de usuario para establecer las condiciones que el producto de software debe satisfacer para ser aceptado el usuario	Complementaria
No encontrada	Incluir restricciones y reglas de negocio como historias de usuario individuales	No encontrada	Incluir restricciones y reglas de negocio como historias de usuario individuales	Incluir restricciones y reglas de negocio como historias de usuario individuales **para asegurar que las necesidades del negocio son consideradas dentro del proyecto**	Complementaria

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
No encontrada	No encontrada	1) Describir requisitos de atributos de calidad utilizando la técnica ligera escenarios de atributos de calidad 2) Obtener los escenarios de atributos de calidad utilizando técnicas como el taller de atributos de calidad o *"quality atributte workshop"*	Elaborar una lista de los atributos de calidad del sistema	Elaborar una lista de los atributos de calidad del sistema **para encontrar los atributos más difíciles e importantes a implementar**	Complementaria
No encontrada	No encontrada	Utilizar Talleres de Atributos de Calidad o "*Quality Atributte Workshops"* para elicitar riesgos en los atributos de calidad y obtener una lista priorizada de escenarios de fallo	Elicitar riesgos en los atributos de calidad	Elicitar riesgos en los atributos de calidad debido a que falta de entendimiento o la incompletitud de requisitos de atributos de calidad son un riesgo común	Complementaria
Priorizar requisitos funcionales, historias de usuario, etc.	Organizar las historias de usuario con una lista ordenada basada en las relaciones entre sí, su costo, el valor del negocio y el *"time to market"*	No encontrada	Priorizar los requisitos	Priorizar los requisitos **para identificar aquellos que aporten mayor valor al negocio**	Equivalente

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
No encontrada	1) Explorar la percepción del usuario de la forma del sistema utilizando prototipos en papel, esto puede ser una opción para obtener retroalimentación a un bajo costo y corto tiempo 2) Utilizar guiones gráficos o *"storyboards"* para capturar la visión del flujo de uso de los usuarios y posteriormente validar sus asunciones sobre su comportamiento en la realidad	Utilizar casos de uso y escenarios para mostrar funcionalidad	Utilizar prototipos rápidos para explorar la percepción del usuario sobre la forma del sistema y flujo de su uso	Utilizar prototipos rápidos para explorar la percepción del usuario sobre la forma del sistema y flujo el de su uso, **esto puede ser una opción para obtener retroalimentación a bajo costo y corto tiempo**	Equivalente
Incluir la obtención de atributos de calidad como parte del rol de arquitecto de software para asegurar que son considerados	No encontrada	No encontrada	Incluir la obtención de atributos de calidad como parte del rol de arquitecto de software	Incluir la obtención de atributos de calidad como parte del rol de arquitecto de software **para asegurar que son considerados**	Complementaria
Preguntar a los usuarios cuáles atributos de calidad son necesarios utilizando preguntas que nos brinden información acerca de ellos	No encontrada	Identificar atributos de calidad relevantes	Identificar atributos de calidad relevantes	Identificar atributos de calidad relevantes **para posteriormente realizar priorizaciones entre ellos**	Complementaria

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
No encontrada	No encontrada	Priorizar atributos de calidad relevantes	Priorizar atributos de calidad relevantes	Priorizar los atributos de calidad identificados como relevantes **para identificar aquellos que tendrán especial tratamiento durante el desarrollo de arquitectura**	Complementaria
No encontrada	No encontrada	1) Identificar los drivers arquitecturales utilizando ponderaciones como prioridad alta, media, baja 2) Seleccionar los drivers arquitecturales de la lista de atributos de calidad y los escenarios de funcionalidad	Identificar drivers arquitecturales en los atributos de calidad y escenarios de funcionalidad	Identificar drivers arquitecturales en los atributos de calidad y escenarios de funcionalidad, **estos servirán como punto de partida en el diseño de la arquitectura**	Complementaria
Aprender acerca de los atributos de calidad comunes en el dominio en el que se está trabajando y enfocarse en ellos al empezar a construir un nuevo sistema o modificar uno existente	No encontrada	No encontrada	Aprender acerca de los atributos de calidad comunes del dominio y enfocarse en ellos al empezar un nuevo sistema o modificar uno existente	Aprender acerca de los atributos de calidad comunes del dominio y enfocarse en ellos al empezar un nuevo sistema o modificar uno existente p**ara conocer los atributos de calidad que deberían ser considerados**	Complementaria

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
Refinar la información obtenida sobre los atributos de calidad asociando por ejemplo, cantidades a los mismos evitando así vaguedad en su definición	No encontrada	1) Especificar condiciones de prueba para los atributos de calidad que se desee mediante escenarios de atributos de calidad 2) Describir requisitos de atributos de calidad utilizando la técnica ligera escenarios de atributos de calidad	Refinar la información de los atributos de calidad especificando condiciones de prueba	Refinar la información de los atributos de calidad especificando condiciones de prueba para **evitar la vaguedad en su definición**	Equivalente
No encontrada	No encontrada	Cada escenario de atributo de calidad debe ser calificado (o ponderado) tanto por las partes interesadas como por los desarrolladores	Priorizar escenarios de atributos de calidad	Priorizar los escenarios de atributos de calidad **para la identificación de aquellos con mayor relevancia**	Complementaria
No encontrada	No encontrada	Cada escenario de atributo de calidad debe ser calificado (o ponderado) tanto por las partes interesadas como por los desarrolladores	Estimar el esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad	Estimar el esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad **para realizar estimaciones sobre entregas futuras**	Complementaria
No encontrada	Considerar reglas de negocio como fuente de requisitos emergentes	No especificada	Considerar reglas de negocio como fuente de requisitos emergentes	Considerar reglas de negocio como fuente de requisitos emergentes **para prever posibles cambios en el futuro**	Complementaria

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
No encontrada	Soportar requisitos emergentes en los casos de uso a través de desviaciones como extensiones, excepciones, alternativas, variaciones, manejo de errores y requisitos no funcionales	No encontrada	Soportar la aparición de requisitos emergentes durante el desarrollo	Soportar la aparición de requisitos emergentes durante el desarrollo **para aceptar cambios en el producto de software en construcción**	Complementaria
No encontrada	No encontrada	Utilizar restricciones o invariantes para especificar como el sistema debería o no debería ser	Identificar diferentes tipos de restricciones (técnicas y de negocio)	Identificar diferentes tipos de restricciones (técnicas y de negocio) **para especificar las características que debería o no cumplir el sistema a construir**	Complementaria
No encontrada	No encontrada	Utilizar las restricciones para asegurarse que los escenarios de atributos de calidad pueden ser cumplidos	Utilizar las restricciones en pruebas de escenarios de atributos de calidad	Utilizar las restricciones en pruebas de escenarios de atributos de calidad **para asegurar que pueden ser cumplidos**	Complementaria
Revisar si existe una lista aprobada de tecnologías por la compañía, a menudo restringen las diferentes tecnologías que la compañía tiene que mantener, operar y comprar licencias	No encontrada	No encontrada	Revisar la lista de tecnologías aprobadas por la compañía	Revisar la lista de tecnologías aprobadas por la compañía **pues a menudo restringen las diferentes tecnologías que la compañía tiene que mantener, operar y comprar licencias**	Complementaria

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
Revisar si existen otros sistemas que se necesiten integrar con el sistema que se va a construir	No encontrada	No encontrada	Revisar dependencias con sistemas externos	Revisar dependencias con sistemas externos **debido a que podrían existir sistemas que la organización necesite integrar al que estás construyendo**	Complementaria
Conocer la plataforma de despliegue destino, pues a menudo tiene fuerte impacto en las decisiones de tecnología	No encontrada	No encontrada	Conocer la plataforma de despliegue destino	Conocer la plataforma de despliegue destino **pues a menudo tiene fuerte impacto en las decisiones de tecnología**	Complementaria
Tomar en cuenta las características de los miembros del equipo cuando se está arquitectando un sistema de software	No encontrada	No encontrada	Tomar en cuenta las características de los miembros del equipo cuando se está arquitectando un sistema de software	Tomar en cuenta las características de los miembros del equipo cuando se está arquitectando un sistema de software **para prevenir la necesidad capacitación o recursos humanos**	Complementaria
Tomar en cuenta si el sistema es parte de una implementación táctica o estratégica	No encontrada	No encontrada	Tomar en cuenta si el sistema es parte de una implementación táctica o estratégica	Tomar en cuenta si el sistema es parte de una implementación táctica o estratégica **debido a que la respuesta a este pregunta puede quitar o agregar restricciones**	Complementaria

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
Priorizar las restricciones dado que algunas son más importantes que otras, y puede utilizarse como una ventaja	No encontrada	No encontrada	Priorizar las restricciones organizacionales	Priorizar las restricciones organizacionales **dado que algunas son más importantes que otras y puede utilizarse como ventaja**	Complementaria
Identificar los principios que se desean adoptar para introducir claridad y consistencia al código base. Pueden ser principios de desarrollo o arquitectura	No encontrada	No encontrada	Identificar principios de desarrollo y/o arquitectura a utilizar	Identificar principios de desarrollo y/o arquitectura a utilizar **para introducir claridad y consistencia al código base.**	Complementaria
Decidir qué principios adoptar tomando en cuenta el tamaño, la complejidad y las restricciones del ambiente en que se desarrolla el proyecto	No encontrada	No encontrada	Decidir qué principios de desarrollo y/o arquitectura se utilizarán	Decidir qué principios de desarrollo y/o arquitectura se utilizarán **pues la selección de estos depende del contexto en el que se encuentra.**	complementaria
No encontrada	Obtener retroalimentación del producto para mantener el camino durante su desarrollo debido a los *"requisitos emergentes"*	No encontrada	Obtener retroalimentación del producto durante su desarrollo	Obtener retroalimentación frecuente del producto **para evitar desviaciones debido a los requisitos emergentes**	Complementaria

Anexo D. Continuación

Brown - Software Architecture for Developers	Coplien - Lean Architecture for Agile Software Development	Fairbanks - Just Enough Software Architecture	Actividad genérica (explicando el qué)	Actividad genérica (explicando el qué y el porqué)	Tipo
No encontrada	El personal de pruebas debe conocer los requisitos tan bien como los desarrolladores, por lo cual deberían atender las mismas reuniones	No encontrada	Asegurar que todos los involucrados en el desarrollo acudan a las mismas reuniones	Asegurar que todos los involucrados en el desarrollo acudan a las mismas reuniones **debido a que todos deben conocer en el mismo grado de detalle los requisitos**	Complementaria
No encontrada	No encontrada	El equipo debe identificar riesgos del proyecto y anexarlos a la lista del producto para que el dueño del producto los pueda priorizar	Identificar riesgos del proyecto y priorizarlos en colaboración con el dueño del producto	Identificar riesgos del proyecto **para mantener monitoreados los riesgos que se puedan presentar durante el proyecto**	Complementaria
No encontrada	No encontrada	El equipo debe identificar riesgos del proyecto y anexarlos a la lista del producto para que el dueño del producto los pueda priorizar	Priorizar los riesgos en colaboración con el dueño del producto	Priorizar los riesgos en colaboración con el dueño del producto para identificar los riesgos más importantes del negocio	Complementaria
No encontrada	No encontrada	El equipo debe identificar riesgos del proyecto y anexarlos a la lista del producto para que el dueño del producto los pueda priorizar	Anexar los riesgos identificados a la lista del producto	Anexar los riesgos identificados a la lista del producto **para que el dueño del producto los pueda priorizar**	Complementaria
No encontrada	No encontrada	Identificar riesgos en los requisitos (en la forma que tengan)	Identificar riesgos en los requisitos	Identificar riesgos en los requisitos **para prevenir acciones correctivas así como estimar su probabilidad de impacto en el proyecto**	Complementaria

Anexo E

En esta matriz se desarrolla cada actividad incluyendo todos los elementos descritos para cada una.

Parte 1.

Actividad genérica (explicando sólo el qué)	**Reformulación de la actividad la lista de verificación**	**Finalidad**	**Objeto de la actividad**	**Etapa del proceso de ingeniería de requisitos**	**Rol**
Elaborar una definición del problema corta y nítida	La definición del problema **elaborada** es corta y nítida	La definición del problema ayudará a saber cuándo el trabajo de desarrollo ha terminado	Problema	Obtención y análisis	Equipo Scrum
Entender los objetivos del proyecto y del negocio	Las metas del proyecto y del negocio son **entendidas**	Establecimiento de los resultados esperados del proyecto	Meta	Obtención y análisis	Equipo Scrum
Recolectar información sobre los usuarios e identificar los diferentes perfiles de usuario que juegan los involucrados en el proyecto	Información sobre los usuarios ha sido **recolectada** y se han identificado los diferentes perfiles que juegan los involucrados en el proyecto	Definición del conjunto de perfiles que intervendrán en el sistema de software a construir	Usuarios	Obtención y análisis	Equipo de desarrollo
Estudiar a los usuarios finales e incluso clientes en su hábitat natural	Los usuarios finales e incluso clientes han sido **estudiados** en su hábitat natural	Obtención de conocimiento del dominio y aprendizaje sobre el contexto en el cual los requisitos surgen	Usuarios	Obtención y análisis	Equipo de desarrollo
Obtener la *"lista de deseos"* de los involucrados sobre lo que necesitan que el sistema haga	La lista de deseos de los involucrados sobre lo que necesitan que el sistema haga ha sido **obtenida**	Explorar el espacio de los requisitos	Lista de deseos	Especificación	Equipo Scrum
Elaborar narrativas para explorar la funcionalidad del sistema, incluyendo lo que sucede antes y después de cada acción	Narrativas para explorar la funcionalidad del sistema han sido **elaboradas**, incluyendo lo que sucede antes y después de cada acción	Entendimiento de la motivación del usuario o del negocio para una funcionalidad dada	Narrativas	Especificación	Equipo de desarrollo

Anexo E. Parte 1. Continuación

Actividad genérica (explicando sólo el qué)	Reformulación de la actividad la lista de verificación	Finalidad	Objeto de la actividad	Etapa del proceso de ingeniería de requisitos	Rol
Incluir casos de prueba en las historias de usuario o especificaciones de casos de uso	Casos de prueba han sido **incluidos** en las historias de usuario o especificaciones de casos de uso	Establecimiento de las condiciones que el producto de software debe satisfacer para ser aceptado el usuario	Casos de prueba	Validación	Equipo de desarrollo
Incluir restricciones y reglas de negocio como historias de usuario individuales	Restricciones y reglas de negocio han sido **incluidas** en historias de usuario	Asegurar que las necesidades del negocio son consideradas dentro del proyecto	Restricciones	Especificación	Equipo de desarrollo
Priorizar las historias de usuario	Las historias de usuario han sido **priorizadas**	Identificación de aquellas historias de usuario que aporten mayor valor al negocio	Historias de usuario	Validación	Dueño del producto + Equipo de desarrollo
Utilizar prototipos rápidos para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso	Prototipos rápidos han sido **utilizados** para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso	Obtención de retroalimentación a bajo costo y corto tiempo	Prototipo	Validación	Equipo de desarrollo
Incluir la obtención de atributos de calidad como parte del rol de arquitecto de software	La obtención de atributos de calidad ha sido **incluida** como parte del rol de arquitecto de software	Asegurar que los atributos de calidad son considerados en el diseño	atributos de calidad	Obtención y análisis	Equipo de desarrollo
Identificar atributos de calidad relevantes	Los atributos de calidad relevantes han sido **identificados**	Asegurar que el producto de software responde a los atributos de calidad comunes del dominio en el que surge	atributos de calidad	Obtención y análisis	Equipo de desarrollo
Priorizar atributos de calidad relevantes	Los atributos de calidad relevantes han sido **priorizados**	Identificación de aquellos atributos de calidad que tendrán especial tratamiento durante el desarrollo de arquitectura	atributos de calidad	Validación	Dueño del producto + Equipo de desarrollo
Utilizar prototipos rápidos para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso	Prototipos rápidos han sido **utilizados** para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso	Obtención de retroalimentación a bajo costo y corto tiempo	Prototipo	Validación	Equipo de desarrollo

Anexo E. Parte 1. Continuación

Actividad genérica (explicando sólo el qué)	Reformulación de la actividad la lista de verificación	Finalidad	Objeto de la actividad	Etapa del proceso de ingeniería de requisitos	Rol
Identificar drivers arquitecturales en los atributos de calidad y escenarios de funcionalidad	Los drivers arquitecturales han sido **identificados** en los atributos de calidad y escenarios de funcionalidad	Identificar los drivers arquitecturales que servirán como punto de partida en el diseño de la arquitectura	atributos de calidad	Obtención y análisis	Equipo de desarrollo
Aprender acerca de los atributos de calidad comunes del dominio	Se ha **aprendido** sobre los atributos de calidad comunes del dominio	Conocer los atributos de calidad que deberían ser considerados en el desarrollo del producto de software	atributos de calidad	Obtención y análisis	Equipo de desarrollo
Refinar la información de los atributos de calidad especificando condiciones de prueba	Condiciones de prueba han sido **incluidos** en la información de los atributos de calidad	Evitar vaguedad en la especificación de condiciones de prueba	Condiciones de prueba	Validación	Equipo de desarrollo
Priorizar escenarios de atributos de calidad	Los escenarios de atributos de calidad han sido **priorizados**	Identificación de los escenarios de atributos de calidad de mayor relevancia	atributos de calidad	Validación	Dueño del producto + Equipo de desarrollo
Estimar el esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad	El esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad ha sido **estimado**	Facilitar estimaciones sobre entregas futuras	atributos de calidad	Obtención y análisis	Equipo de desarrollo
Considerar reglas de negocio como fuente de requisitos emergentes	Las reglas de negocio han sido **consideradas** como fuente de requisitos emergentes	Prevención de cambios en el futuro	requisitos emergentes	Obtención y análisis	Dueño del producto + Equipo de desarrollo
Soportar la aparición de requisitos emergentes durante el desarrollo	La aparición de requisitos emergentes es **soportada** durante el desarrollo	Aceptación de cambios durante la construcción del producto de software	requisitos emergentes	Validación	Dueño del producto + Equipo de desarrollo
Identificar diferentes tipos de restricciones (técnicas y de negocio)	Restricciones técnicas y de negocio han sido **identificadas**	Establecimiento de las características que debería o no cumplir el sistema a construir	restricciones	Obtención y análisis	Dueño del producto + Equipo de desarrollo
Utilizar las restricciones en pruebas de escenarios de atributos de calidad	Las restricciones han sido **utilizadas** en la prueba de escenarios de atributos de calidad	Asegurar la consideración de las restricciones en la construcción del producto de software	restricciones	Validación	Equipo de desarrollo

Anexo E. Parte 1. Continuación

Actividad genérica (explicando sólo el qué)	**Reformulación de la actividad la lista de verificación**	**Finalidad**	**Objeto de la actividad**	**Etapa del proceso de ingeniería de requisitos**	**Rol**
Revisar la lista de tecnologías aprobadas por la compañía	La lista de tecnologías aprobadas por la compañía ha sido **revisada**	Conocimiento de las diferentes tecnologías que la compañía tiene que mantener, operar y comprar licencias	lista de tecnología	Obtención y análisis	Equipo de desarrollo
Revisar dependencias con sistemas externos	Las dependencias con sistemas externos han sido **revisadas**	Identificar si es necesario integrar sistemas externos con el que se está construyendo	sistema externo	Obtención y análisis	Equipo de desarrollo
Conocer la plataforma de despliegue destino	La plataforma de despliegue destino ha sido **revisada**	Prevención de impacto de la plataforma de despliegue en las decisiones de tecnología	Plataforma	Obtención y análisis	Equipo de desarrollo
Tomar en cuenta las características del equipo cuando se está arquitectando un sistema de software	Las características del equipo son **consideradas** cuando se está arquitectando un sistema de software	Prevención de la necesidad capacitación o recursos humanos	Scrum Master + Equipo de desarrollo	Obtención y análisis	Equipo de desarrollo
Tomar en cuenta si el sistema es parte de una implementación táctica o estratégica	El sistema o desarrollo ha **sido clasificado** de acuerdo a su importancia táctica o estratégica	Adición o eliminación de nuevas restricciones dependiendo del contexto y del tipo de producto a construir	Sistema	Obtención y análisis	Dueño del producto + Equipo de desarrollo
Priorizar las restricciones organizacionales	Las restricciones organizacionales han sido **priorizadas**	Identificación de las restricciones organizacionales más importantes	Restricciones	Validación	Dueño del producto + Equipo de desarrollo
Identificar y decidir qué principios de desarrollo y/o arquitectura se utilizarán	Los principios de desarrollo y/o arquitectura han sido identificados y seleccionados	Introducción de claridad y consistencia al código base	Principios de diseño	Obtención y análisis	Equipo de desarrollo
Obtener retroalimentación del producto durante su desarrollo	Retroalimentación del producto es **obtenida** durante el desarrollo	Evitar desviaciones en el proyecto debido a los requisitos emergentes	Retroalimentación	Validación	Scrum Team

Anexo E. Parte 1. Continuación

Actividad genérica (explicando sólo el qué)	Reformulación de la actividad la lista de verificación	Finalidad	Objeto de la actividad	Etapa del proceso de ingeniería de requisitos	Rol
Asegurar que todos los involucrados en el desarrollo acudan a las mismas reuniones	Todos los involucrados **acuden** a las mismas reuniones	Todos los involucrados deben conocer en el mismo grado de detalle los requisitos	Stakeholders	Validación	Scrum Master
Identificar riesgos del proyecto en colaboración con el dueño del producto	Los riesgos del proyecto han sido **identificados**	Conocer los diferentes riesgos a los que se puede enfrentar el proyecto durante su desarrollo	Riesgos	Obtención y análisis	Dueño del producto + Equipo de desarrollo
Priorizar los riesgos en colaboración con el dueño del producto	Los riesgos del proyecto han sido **priorizados**	Identificación de los riesgos más importantes del negocio	Riesgos	Obtención y análisis	Equipo de desarrollo
Anexar los riesgos identificados a la lista del producto o *backlog*	Los riesgos identificados han sido **incluidos** en la lista del producto o *backlog*	Facilitar la priorización de los elementos de la lista del producto	Riesgos	Especificación	Equipo de desarrollo
Identificar riesgos en los requisitos	Riesgos en los requisitos han sido **identificados**	Conocer los riesgos relacionados a los requisitos a los que se puede enfrentar el proyecto durante su desarrollo	Riesgos	Obtención y análisis	Equipo de desarrollo
Identificar riesgos y producir una lista priorizada de escenarios de fallo en los requisitos de atributos de calidad	Riesgos en los requisitos de atributos de calidad han sido identificados y se ha obtenido una lista priorizada de escenarios de fallo	Contrarrestar la falta de entendimiento o la incompletitud de requisitos de atributos de calidad	Riesgos	Validación	Equipo de desarrollo

Anexo E. Parte 2.

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos	* Técnica de definición del problema de **Jerry Weinberg:** consiste en escribir la diferencia entre el estado actual y el estado deseado. Ver "Are Your Lights On?: A Treatise on the Definition of Diverse Problems (1977)" * Análisis y diseño del problema con TRIZ		Procesadores de texto	Documentación de la descripción del problema en una o dos sentencias
Requisitos	Definir la meta del proyecto como un objetivo global que enuncie el resultado final del proyecto. Debería ser simple y comprensible. Gracias a ella, podrá conseguir la aprobación y guiar al equipo hacia una conclusión exitosa.	Recopilar información sobre la misión, visión y objetivos del negocio así como los objetivos del proyecto	Procesadores de texto	Documentación de las metas del proyecto y del producto
Requisitos	* Técnica de modelado de roles: perfiles de usuario **Nielsen** (2005) o personas **Coplien** (2010) * Receta de los 4 pasos de **Cohn** (2004) para el Modelado de roles de usuario: 1) Hacer una lluvia de ideas sobre el conjunto inicial de roles de usuario 2) Organizar el conjunto de roles de usuario obtenido 3) Consolidar los roles 4) Refinar los roles		Procesadores de texto	Conjunto de perfiles de usuario incluyendo por ejemplo: documentación de las características del usuario, edad, región geográfica, etc.

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos	* Obtención de requisitos basado en la observación (**Wiegers**, 2013) * Diseño Contextual o *Contextual Design* (**Beyer y Holtzablatt**, 1998) ofrece las siguientes técnicas para explorar y capturar modelos mentales de usuario: - Investigación del contexto - Modelado de la forma de trabajo - Consolidación de la forma de trabajo, integrando los diagramas juntos para ver la forma de trabajo de todos los clientes - Rediseño de la forma de trabajo - Diseño del ambiente de usuario - Bosquejos y pruebas con clientes		Herramientas de software para el modelado de BPMN (Business Process Model and Notation) como: - Activiti Modeler (Apache License) - Bonita BPM (GPLv2) - Imixs-BPMN (GPLv2) - jBPM (Apache License) - Modelio (GPLv3)	Bosquejos y/o diagramas de la forma de trabajo de los usuarios
Requisitos	* Lluvia de ideas sobre historias de usuario en colaboración con expertos de dominio * Técnicas para capturar las solicitudes del usuario: - Historias de usuario - Casos de uso - Especificaciones de requisitos tradicionales		* Tarjetas de papel * Procesadores de texto * Herramientas para la administración de proyectos como: - Trello (comercial) - JIRA (comercial) - Visual Studio Team Services (comercial) - VersionOne (comercial) - Select Scope Manager (comercial) - Taiga.io (GPL) - Tuleap (GPL)	Historias de usuario, casos de uso o documento de especificación de requisitos
Requisitos	Especificaciones de casos de uso		* Plantillas para especificación de casos de uso	Documentación de las especificaciones de casos de uso

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos	La fórmula Dado, Cuando, Entonces o *Given-When-Then* es una plantilla cuyo intento es guiar la escritura de pruebas de aceptación para una historia de usuario: - (Dado) algún contexto - (Cuando) alguna acción es llevada a cabo - (Entonces) un conjunto particular de consecuencias observables se deberían obtener	Este tipo de pruebas regularmente se agregan al reverso de la historia de usuario cuando se trabaja con un tablero físico	* Tablero Scrum o Kanban físico * Herramientas para la administración de proyectos como: - Trello (comercial) - JIRA (comercial) - Visual Studio Team Services (comercial) - VersionOne (comercial) - Select Scope Manager (comercial) - Taiga.io (GPL) - Tuleap (GPL)	Conjunto de historias de usuario o especificaciones de casos de uso incluyendo condiciones de prueba
Requisitos	Uso de la técnica de historias de usuario para capturar restricciones y/o reglas de negocio	Agrupación de historias de usuario por áreas de negocio	* Tablero Scrum o Kanban físico * Herramientas para la administración de proyectos como: - Trello (comercial) - JIRA (comercial) - Visual Studio Team Services (comercial) - VersionOne (comercial) - Select Scope Manager (comercial) - Taiga.io (GPL) - Tuleap (GPL)	Historias de usuario incluyendo restricciones y/o reglas de negocio
Requisitos	Técnicas de priorización: - Mapa de historias de usuario o *Story Mapping* - Juego del Árbol del Producto o Prune the Product Tree - La lancha rápida o *Speed Boat* - Priorización MoSCoW (*Must have* o debe tener, *Should have* o debería tener, *Could have* o pueden tener y W*on't have* o no tendrá... por el momento) - Compra de características o *Buy a feature*		Mapa de historias de usuario, Plantilla de árbol, Herramientas de software como: - idea engine (comercial), - decision engine (comercial) - TransparentChoice's AHP online tool (comercial), - easyAHP (libre)	Conjunto de requisitos o historias de usuario priorizadas

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos, Diseño y Construcción	Una clasificación de los prototipos a utilizar basado en funcionalidad y fidelidad es la siguiente: 1. Poco visual + Poca funcionalidad: Prototipos de papel hechos a mano. 2. Poco visual + Alta funcionalidad: Wireframes (esquemas de página o planos de pantalla). Prototipos visuales creados con HTML en wireframes interactivos 3. Altamente visual + Poca funcionalidad: Imágenes que muestran algunas capturas de pantalla juntas en el navegador web o impresión de capturas de pantalla para un prototipo de papel más bonito. 4. Altamente visual + Altamente funcional: Este prototipo está solo un paso antes del producto terminado y puede ser liberado como un producto mínimo viable para generar retroalimentación y probar su usabilidad.		* Herramientas para prototipos de baja fidelidad: - Balsamiq Mockups (comercial) - Moqups (comercial) - Pidoco (comercial) - WireframePro (comercial) * Herramientas para prototipos de alta fidelidad: - Axure (comercial) - UXPin (comercial) - Proto.io (comercial)	Prototipos
Requisitos	* Uso de matriz de responsabilidades detallando: rol, nombre de la persona y responsabilidades. * Definir de manera explícita (por escrito) la responsabilidad como una política en la etapa de diseño del tablero Kanban.		* Matriz de asignación de responsabilidades * Tarjetas estilo Post-it para la inclusión de políticas dentro del tablero Kanban	* Matriz de responsabilidades * Lista de verificación de políticas para la etapa de diseño

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos	Taller de atributos de calidad o *Quality Attribute Workshop*	* Lista de atributos de calidad que se pueden considerar: desempeño, escalabilidad, disponibilidad, seguridad, recuperación de desastres, accesibilidad, monitoreo, auditoría, flexibilidad, extensibilidad, mantenibilidad, conformidad legal y/o regulatoria, internacionalización (multilenguaje) y localización (convenciones en monedas, fechas, etc.). * Preguntas como las siguientes pueden ayudar: ¿Cuánto tiempo puedes tolerar el sistema fuera de línea?, ¿Qué pasa si el sistema falla durante horas de trabajo normal (9 a.m. a 6 p.m.)?, ¿Qué pasa si el sistema falla fuera de las horas normales de trabajo?	Formatos para llenado de escenarios de atributos de calidad	Conjunto de escenarios de atributos de calidad

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos	Las siguientes técnicas de priorización pueden ser de utilidad: - Diagrama red-araña o *spider-web* - Mapa de historias de usuario o *Story Mapping* - Juego del Árbol del Producto o *Prune the Product Tree* - La lancha rápida o Speed Boat - MoSCoW (*Must have* o debe tener, *Should have* o debería tener, *Could have* o puede tener y *Won't have* o no tendrá... por el momento) - Compra una característica o *Buy a feature* - Cuadrantes de manejo del tiempo de *Convey* *- Proceso Analítico jerárquico o Analytic Hierarchy Process*		* Diagrama de la telaraña impreso para trabajar con él en físico * Mapa de historias de usuario, plantilla de árbol o pizarrón * En digital: - idea engine (comercial) - decision engine (comercial) - TransparentChoice's AHP online tool (Comercial) - easyAHP (Libre)	Lista de atributos de calidad priorizada
Requisitos	* Uso de la lista de atributos de calidad y los escenarios de funcionalidad para hacer la selección * Aplicación de prioridad baja, media y alta como criterios de selección * Votación por rondas * Atributos de calidad alta, serán identificados como drivers arquitecturales		Tarjetas rotuladas con las frases "prioridad baja, prioridad media y prioridad alta" para realizar votaciones	Lista de drivers arquitecturales
Requisitos	Revisión del estado del arte sobre el dominio en el que se desenvuelve del proyecto en particular	Observación del ambiente donde el sistema de software a construir operará	Bibliotecas digitales como: - Google Scholar (Pública) - IEEEXplore (Privada) - ACM Digital Library (Privada)	Lista de atributos de calidad del dominio

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos	* Taller de atributos de calidad o Quality Attribute Workshop * La fórmula Dado, Cuando, Entonces o Given-When-Then es una plantilla cuyo intento es guiar la escritura de pruebas de aceptación para una historia de usuario: - (Dado) algún contexto - (Cuando) alguna acción es llevada a cabo - (Entonces) un conjunto particular de consecuencias observables se deberían obtener	* Preguntas como las siguientes pueden ayudar: ¿Cuántos usuarios concurrentes debería el sistema soportar en promedio?, ¿Cómo intervienen las horas pico?, ¿Qué tiempo de respuesta es considerado aceptable?, ¿El tiempo de respuesta debe ser el mismo en todas las partes del software o sólo para características específicas? además de preguntar sobre el número de usuarios concurrentes, volúmenes de datos y número máximo de respuestas. * Este tipo de pruebas a regularmente se agregan al reverso de la historia de usuario cuando se cuenta con un tablero físico.	* Formatos para llenado de escenarios de atributos de calidad * Herramientas para la administración de proyectos como: - Trello (Comercial) - JIRA (Comercial) - Visual Studio Team Services (Comercial) - VersionOne (Comercial) - Select Scope Manager (Comercial) - Taiga.io (GPL) - Tuleap (GPL) * Procesadores de texto	* Lista de atributos de calidad, idealmente que se pueden cuantificar de manera explícita * Criterios de aceptación para realizar pruebas posteriormente
Requisitos	- Juego del Árbol del Producto o *Prune the Product Tree* - La lancha rápida o *Speed Boat* - Priorización MoSCoW (*Must have* o debe tener, *Should have* o debería tener, *Could have* o puede tener y *Won't have* o no tendrá... por el momento) - Compra una característica o *Buy a feature* - Cuadrantes de manejo del tiempo de Convey - Proceso Analítico Jerárquico o *Analytic Hierarchy Process* - Votación en rondas		* Plantilla de árbol * Herramientas de software como: - idea engine (Comercial), - decision engine (Comercial) - TransparentChoice's AHP online tool (Comercial) - easyAHP (Libre)	Escenarios de atributos de calidad priorizados

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Construcción	Técnicas para estimar esfuerzo como: * Serie fibonacci y/o modificaciones de la serie (0, 1, 2, 3, 5, 8, 13, 21, etc). * Camisas o T-Shirts (xxs, xs, s, m, l, xl, xxl) * Múltiplos de 2 (2, 4, 6, 8, 10, 12, 14, etc.)		Aplicaciones en línea o descargables como: - Planningpoker.com - Planning Poker - Agile Planning Poker - Scrum Poker Cards - BEEVA Planning Poker.	* Escenarios de atributos de calidad estimados
Cualquier etapa	Historias de usuario o actualización de documento de especificación de requisitos para documentar nuevos requisitos	Uso del diccionario de reglas de negocio, glosario, diccionario de datos, o descripción de las reglas de negocio para descubrir nuevas restricciones	* Procesadores de texto o * Herramientas para la administración de proyectos como: - Trello (Comercial) - JIRA (Comercial) - Visual Studio Team Services (Comercial) - VersionOne (Comercial) - Select Scope Manager (Comercial) - Taiga.io (GPL) - Tuleap (GPL)	Historias de usuario o documento de especificación de requisitos actualizado
Cualquier etapa	Adición de extensiones, excepciones, alternativas, variaciones, manejo de errores y atributos de calidad a las historias de usuario o casos de uso.		* Procesadores de texto o * Herramientas para la administración de proyectos como: - Trello (Comercial) - JIRA (Comercial) - Visual Studio Team Services (Comercial) - VersionOne (Comercial) - Select Scope Manager (Comercial) - Taiga.io (GPL) - Tuleap (GPL)	Historias de usuario o casos de uso actualizados

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos	Recopilar información sobre los diferentes tipos de restricciones que influenciarán en el sistema de software a construir		* Procesadores de texto o * Herramientas para el modelado UML como: - Modelio (GPL & Comercial) - MySQL Workbench (GPL & Comercial) - Lucidchart (Comercial) - Microsoft Visio (Comercial)	Lista de restricciones técnicas y de negocio
Requisitos, construcción y pruebas	Escenarios de atributos de calidad		* Procesadores de texto * Formatos para llenado de escenarios de atributos de calidad	Escenarios de atributos de calidad actualizados
Requisitos	Comparación de las tecnologías a utilizar con la lista de tecnologías aprobadas por la compañía para verificar la viabilidad de su uso	Revisión de base de datos de gestión de la configuración	Procesadores de texto	Lista de tecnologías fuera de las aprobadas por la organización
Requisitos	* Revisión de restricciones organizacionales dictando protocolos y tecnologías a utilizar para los puntos de integración * Diagrama de contexto para modelar dependencias con sistemas externos		Procesadores de texto o herramientas para el modelado UML como: - Modelio (GPL & Comercial) - MySQL Workbench (GPL & Comercial) - Lucidchart (Comercial) - Microsoft Visio (Comercial)	* Lista de protocolos y tecnologías a utilizar * Diagrama de contexto del sistema
Requisitos	* Revisión de las características de la plataforma destino * Implementación de DevOps		Base de datos de gestión de la configuración	Lista de restricciones sobre lo que el software puede o no hacer en la plataforma que lo alojará

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos y Diseño	Liquidez de personal o *Staff liquidity*	Conocer: - ¿qué tan grande es el equipo de desarrollo?, - ¿cuáles son sus habilidades?, - ¿qué tan rápido puede escalar el equipo de desarrollo si es necesario?, - ¿puedes requerir especialistas, consultoría o entrenamiento de ser necesario?, - ¿tendrá el equipo de mantenimiento las mismas habilidades que el equipo de desarrollo?	* Matriz de habilidades	Lista de características a mejorar o trabajar del equipo necesarias para iniciar la construcción de un nuevo sistema
Requisitos		Identificar si el sistema es una implementación táctica o estratégica y las restricciones que esto implica	Procesador de textos	Lista de restricciones adicionales o removidas
Requisitos	Técnicas de priorización como: - Priorización MoSCoW (*Must have* o debe tener, *Should have* o debería tener, *Could have* o puede tener y *Won't have* o no tendrá... por el momento) - Cuadrantes de manejo del tiempo de Convey - Proceso Analítico Jerárquico o *Analytic Hierarchy Process*		* Pizarrón * Herramientas de software como: - idea engine (comercial), - decision engine (comercial) - TransparentChoice's AHP online tool (comercial), - easyAHP (libre)	Lista de restricciones priorizada
Diseño	* Considerar la complejidad del software que se está construyendo, además de las restricciones del ambiente, ayudarán a decidir qué principios adoptar. El contexto es siempre la clave.	* Tener una lista de principios puede ayudar a asegurar que todo el mundo dentro del equipo está trabajando de la misma manera. * Escuchar la retroalimentación de los miembros del equipo ayudará a decidir cuáles principios están funcionando y cuáles no.	No especificado	Lista de principios de diseño y/o arquitectura a adoptar

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Construcción	* Adopción de desarrollo incremental * Establecimiento de reuniones de entrega frecuentes durante el proyecto	En Scrum la retroalimentación se obtiene durante la reunión de revisión del sprint.	No especificado	Lista de cambios solicitados por el dueño del producto y/o usuarios finales
Cualquier etapa	Establecer con los involucrados propósitos, fechas, horas y duración de las reuniones		Calendarios en línea (Google Calendar, iCal, Live Calendar, Yahoo! Calendar)	Notificaciones electrónicas
Requisitos	* Realizar lluvias de ideas sobre riesgos del proyecto * Identificación de riesgos de la administración del proyecto y riesgos técnicos	Colaborar con el equipo de desarrollo para la identificación de riesgos técnicos	No especificado - pizarrón	Conjunto de historias de usuario que incluyen riesgos en su definición
Requisitos	* Cuadrantes de manejo del tiempo de Convey * Votación en rondas * Proceso Analítico Jerárquico o *Analytic Hierarchy Process* * Liquidez de personal o Staff liquidity * Uso de criterios de prioridad alta, media y baja para clasificar los riesgos.	El equipo de software debe ayudar al dueño del producto a entender los riesgos técnicos y priorizar adecuadamente la lista de producto	* TransparentChoice's AHP online tool (comercial) * easyAHP (libre)	Conjunto de historias de usuario sobre riesgos priorizadas
Requisitos	Convertir la lista del producto en lista de características y riesgos del producto		Herramientas para la administración de proyectos como: - Trello (Comercial) - JIRA (Comercial) - Visual Studio Team Services (Comercial) - VersionOne (Comercial) - Select Scope Manager (Comercial) - Taiga.io (GPL) - Tuleap (GPL)	Lista del producto actualizada

Anexo E. Parte 2. Continuación

Etapa del ciclo de vida	Técnicas	Consejos	Herramientas	Artefactos
Requisitos	* Lluvia de ideas * Structured SWIFT * Análisis de situaciones * Análisis de incidencias * El Análisis del Árbol de Fallas o *Fault tree analysis* * Análisis en forma de corbata o *The Bowtie Method*		Pizarrón	Lista de riesgos en los requisitos
Requisitos	* Taller de atributos de calidad o *Quality Attribute Workshop* * Cuestionarios de taxonomía de riesgos o *Taxonomy-Based Questionnaire* (TBQ) (Carr et al., 1993) * Lluvia de ideas * Análisis de situaciones * El Análisis del Árbol de Fallas o *Fault tree analysis* * Análisis en forma de corbata o *The Bowtie Method*		Pizarrón	Lista de riesgos y escenarios de fallo

Anexo F

Prototipo pre-entrega a diseñador.

La definición del problema elaborada es corta y nítida *(SG)*	
Objetivo: La definición del problema ayudará a identificar cuándo el trabajo de desarrollo ha terminado	
Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 
Técnicas: • Técnica de definición del problema de **Jerry Weinberg:** consiste en escribir la diferencia entre el estado actual y el estado deseado. Véase *Are Your Lights On?: A Treatise on the Definition of Diverse Problems* (1977) • Análisis y diseño del problema con TRIZ	
Herramientas: Procesadores de texto	
Artefactos: Documentación de la descripción del problema en una o dos sentencias	

1

Las metas del proyecto y del negocio son entendidas *(SG)*	
Objetivo: Establecimiento de los resultados esperados del proyecto	
Prioridad: Alta	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 
Técnicas: • Definir la meta del proyecto como un objetivo global que enuncie el resultado final del proyecto. Debería ser simple y comprensible. Gracias a ella, podrá conseguir la aprobación y guiar al equipo hacia una conclusión exitosa**.**	
Herramientas: Procesadores de texto	
Artefactos: Documentación de las metas del proyecto y del producto	

Consejo: Recopilar información sobre la misión, visión y objetivos del negocio así como los objetivos del proyecto.

2

Información sobre los usuarios ha sido recolectada y se han identificado los diferentes perfiles que juegan los involucrados en el proyecto *(E)*

Objetivo: Definición del conjunto de perfiles que intervendrán en el sistema de software a construir

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Receta de los 4 pasos de **Cohn** (2004) para el Modelado de roles de usuario:
  - Hacer una lluvia de ideas sobre el conjunto inicial de roles de usuario
  - Organizar el conjunto de roles de usuario obtenido
  - Consolidar los roles
  - Refinar los roles
- Técnica de modelado de roles: perfiles de usuario **Nielsen** (2005) o personas **Coplien** (2010)

Herramientas:
Procesadores de texto

Artefactos:
Conjunto de perfiles de usuario incluyendo por ejemplo: documentación de las características del usuario, edad, región geográfica, etc.

3

Los usuarios finales e incluso clientes han sido estudiados en su hábitat natural *(E)*

Objetivo: Obtención de conocimiento del dominio y aprendizaje sobre el contexto en el cual los requisitos surgen

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Diseño Contextual o *Contextual Design* (**Beyer y Holtzablatt**, 1998) ofrece las siguientes técnicas para explorar y capturar modelos mentales de usuario:
  - Investigación del contexto
  - Modelado de la forma de trabajo
  - Consolidación de la forma de trabajo, integrando los diagramas juntos para ver la forma de trabajo de todos los clientes
  - Rediseño de la forma de trabajo

- Diseño del ambiente de usuario
- Bosquejos y pruebas con clientes

- Obtención de requisitos basado en la observación (**Wiegers**, 2013)

Herramientas: Herramientas de software para el modelado de BPMN (Business Process Model and Notation) como:

- Activiti Modeler (Apache License)
- Bonita BPM (GPLv2)
- Imixs-BPMN (GPLv2)
- jBPM (Apache License)
- Modelio (GPLv3)

Artefactos:

Bosquejos y/o diagramas de la forma de trabajo de los usuarios

4

La lista de deseos de los involucrados sobre lo que necesitan que el sistema haga ha sido obtenida *(SG)*

Objetivo: Exploración del espacio de los requisitos

Prioridad: Baja	**Categoría:** Especificación
Etapa: Requisitos	**Roles:** 

Técnicas:

- Lluvia de ideas sobre historias de usuario en colaboración con expertos de dominio
- Técnicas para capturar las solicitudes del usuario:
  - Historias de usuario
  - Casos de uso
  - Especificaciones de requisitos tradicionales

Herramientas:

- Tarjetas de papel
- Procesadores de texto
- Herramientas para la administración de proyectos como:
  - Trello (comercial)
  - JIRA (comercial)
  - Visual Studio Team Services (comercial)
  - VersionOne (comercial)
  - Select Scope Manager (comercial)
  - Taiga.io (GPL)
  - Tuleap (GPL)

Artefactos:

Historias de usuario, casos de uso o documento de especificación de requisitos

21

Narrativas para explorar la funcionalidad del sistema han sido elaboradas, incluyendo lo que sucede antes y después de cada acción *(SG)*	
Objetivo: Entendimiento de la motivación de usuario o del negocio para una funcionalidad dada	
Prioridad: Alta	**Categoría:** Especificación
Etapa: Requisitos	**Roles:** 
Técnicas: • Especificaciones de casos de uso	
Herramientas: • Plantillas para especificación de casos de uso	
Artefactos: • Documentación de las especificaciones de casos de uso 22	

Casos de prueba han sido incluidos en las historias de usuario o especificaciones de casos de uso *(SG)*	
Objetivo: Establecimiento de las condiciones que el producto de software debe satisfacer para ser aceptado por el usuario	
Prioridad: Alta	**Categoría:** Validación
Etapa: Requisitos	**Roles:** 
Técnicas: • La fórmula Dado, Cuando, Entonces o *Given-When-Then* es una plantilla cuyo intento es guiar la escritura de pruebas de aceptación para una historia de usuario: ○ (Dado) algún contexto ○ (Cuando) alguna acción es llevada a cabo ○ (Entonces) un conjunto particular de consecuencias observables se deberían obtener	
Herramientas: • Tablero Scrum o Kanban físico • Herramientas para la administración de proyectos como: ○ Trello (comercial) ○ JIRA (comercial) ○ Visual Studio Team Services (comercial) ○ VersionOne (comercial) ○ Select Scope Manager (comercial) ○ Taiga.io (GPL) ○ Tuleap (GPL)	

Artefactos:
Conjunto de historias de usuario o especificaciones de casos de uso incluyendo condiciones de prueba

Consejo: Este tipo de pruebas regularmente se agregan al reverso de la historia de usuario cuando se trabaja con un tablero físico

25

Restricciones y reglas de negocio han sido incluidas en historias de usuario *(C)*

Objetivo: Asegurar que las necesidades del negocio sean consideradas dentro del proyecto

Prioridad: Baja	**Categoría:** Especificación
Etapa: Requisitos	**Roles:**

Técnicas:

- Uso de la técnica de historias de usuario para capturar restricciones y/o reglas de negocio

Herramientas:

- Tablero Scrum o Kanban físico
- Herramientas para la administración de proyectos como:
  - Trello (comercial)
  - JIRA (comercial)
  - Visual Studio Team Services (comercial)
  - VersionOne (comercial)
  - Select Scope Manager (comercial)
  - Taiga.io (GPL)
  - Tuleap (GPL)

Artefactos:
Conjunto de historias de usuario o especificaciones de casos de uso incluyendo condiciones de prueba

Consejo:
Agrupación de las historias de usuario por área de negocio

23

Las historias de usuario han sido priorizadas *(SG)*

Objetivo: Identificación de aquellas historias de usuario que aporten mayor valor al negocio

Prioridad: Baja	**Categoría:** Validación
Etapa: Requisitos	**Roles:** 

Técnicas:

- Técnicas orientadas a priorizar requisitos de software como:
  - Mapa de historias de usuario o *Story Mapping*
  - Juego del Árbol del Producto o Prune the Product Tree
  - La lancha rápida o *Speed Boat*
  - Priorización MoSCoW (*Must have* o debe tener, *Should have* o debería tener, *Could have* o puede tener y W*on't have* o no tendrá... por el momento)
  - Compra de características o *Buy a feature*
  - Proceso Analítico Jerárquico o *Analytic Hierarchy Process*

Herramientas:

- Mapa de historias de usuario
- Plantilla de árbol
- Herramientas de software como:
  - idea engine (comercial)
  - decision engine (comercial)
  - TransparentChoice's AHP online tool (comercial)
  - easyAHP (libre)

Artefactos:

Conjunto de requisitos o historias de usuario priorizadas

26

Prototipos rápidos han sido utilizados para explorar la percepción del usuario sobre la forma del sistema y el flujo de su uso *(SG)*

Objetivo: Obtención de retroalimentación a bajo costo y corto tiempo

Prioridad: Alta	**Categoría:** Validación
Etapa: Requisitos, diseño y construcción	**Roles:** 

Técnicas:

Una clasificación de los prototipos a utilizar basado en funcionalidad y fidelidad es la siguiente:

1. Poco visual + Poca funcionalidad: Prototipos de papel hechos a mano.
2. Poco visual + Alta funcionalidad: Wireframes (esquemas de página o planos de pantalla). Prototipos visuales creados con HTML en wireframes interactivos

3. Altamente visual + Poca funcionalidad: Imágenes que muestran algunas capturas de pantalla juntas en el navegador web o impresión de capturas de pantalla para un prototipo de papel más bonito.
4. Altamente visual + Altamente funcional: Este prototipo está solo un paso antes del producto terminado y puede ser liberado como un producto mínimo viable para generar retroalimentación y probar su usabilidad.

Herramientas:

- Herramientas para prototipos de baja fidelidad:
  - Balsamiq Mockups (comercial)
  - Moqups (comercial)
  - Pidoco (comercial)
  - WireframePro (comercial)
- Herramientas para prototipos de alta fidelidad:
  - Axure (comercial)
  - UXPin (comercial)
  - Proto.io (comercial)

Artefactos:

Prototipos

27

La obtención de atributos de calidad ha sido incluida como parte del rol de arquitecto de software *(A)*

Objetivo: Asegurar que los atributos de calidad son considerados en el diseño

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Uso de matriz de responsabilidades detallando: rol, nombre de la persona y responsabilidades.
- Definir de manera explícita (por escrito) la responsabilidad como una política en la etapa de diseño del tablero Kanban.

Herramientas:

- Matriz de asignación de responsabilidades
- Tarjetas estilo Post-it para la inclusión de políticas dentro del tablero Kanban

Artefactos:

- Matriz de responsabilidades
- Lista de verificación de políticas para la etapa de diseño

5

Los atributos de calidad relevantes han sido identificados *(A)*

Objetivo: Asegurar que el producto de software responde a los atributos de calidad comunes del dominio en el que surge

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Taller de atributos de calidad o *Quality Attribute Workshop*

Herramientas:
Formatos para llenado de escenarios de atributos de calidad

Artefactos:
Conjunto de escenarios de atributos de calidad

Consejos:

- Lista de atributos de calidad que se pueden considerar:
  - desempeño, escalabilidad, disponibilidad, seguridad, recuperación de desastres, accesibilidad, monitoreo, auditoría, flexibilidad, extensibilidad, mantenibilidad, conformidad legal y/o regulatoria, internacionalización (multilenguaje) y localización (convenciones en monedas, fechas, etc.)
- Preguntas como las siguientes pueden ayudar:
  - ¿Cuánto tiempo puedes tolerar el sistema fuera de línea?
  - ¿Qué pasa si el sistema falla durante horas de trabajo normal (9 a.m. a 6 p.m.)?
  - ¿Qué pasa si el sistema falla fuera de las horas normales de trabajo?

6

Los atributos de calidad relevantes han sido priorizados *(A)*

Objetivo: Identificación de aquellos atributos de calidad que tendrán especial tratamiento durante el desarrollo de arquitectura

Prioridad: Baja	**Categoría:** Validación
Etapa: Requisitos	**Roles:** 

Técnicas:

- Diagrama red-araña o *spider-web*
- Mapa de historias de usuario o *Story Mapping*
- Juego del Árbol del Producto o *Prune the Product Tree*
- La lancha rápida o Speed Boat
- MoSCoW (*Must have* o debe tener, *Should have* o debería tener, *Could have* o puede tener y *Won't have* o no tendrá... por el momento)

- Compra una característica o *Buy a feature*
- Cuadrantes de manejo del tiempo de *Convey*
- Proceso Analítico Jerárquico *o Analytic Hierarchy Process*

Herramientas:

- Diagrama de la telaraña impreso para trabajar con él en físico
- Mapa de historias de usuario, plantilla de árbol o pizarrón
- En digital:
  - idea engine (comercial)
  - decision engine (comercial)
  - TransparentChoice's AHP online tool (Comercial)
  - easyAHP (Libre)

Artefactos:

Lista de atributos de calidad priorizada

28

Los drivers arquitecturales han sido identificados en los atributos de calidad y escenarios de funcionalidad *(A)*

Objetivo: Identificar los drivers arquitecturales que servirán como punto de partida en el diseño de la arquitectura

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:**

Técnicas:

- Uso de la lista de atributos de calidad y los escenarios de funcionalidad para hacer la selección
- Aplicación de prioridad baja, media y alta como criterios de selección
- Votación por rondas
- Atributos de calidad alta, serán identificados como drivers arquitecturales

Herramientas:

Tarjetas rotuladas con las frases "prioridad baja, prioridad media y prioridad alta" para realizar votaciones

Artefactos:

Lista de drivers arquitecturales

7

Se ha aprendido sobre los atributos de calidad comunes del dominio *(A)*

Objetivo: Conocer los atributos de calidad que deberían ser considerados en el desarrollo del producto de software

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Revisión del estado del arte sobre el dominio en el que se desenvuelve el proyecto

Herramientas:

- Bibliotecas digitales como:
  - Google Scholar (Pública)
  - IEEEXplore (Privada)
  - ACM Digital Library (Privada)

Artefactos:

Lista de atributos de calidad comunes del dominio

8

Condiciones de prueba han sido incluidas en la información de los atributos de calidad *(A)*

Objetivo: Evitar vaguedad en la especificación de condiciones de prueba

Prioridad: Alta	**Categoría:** Validación
Etapa: Requisitos	**Roles:** 

Técnicas:

- Taller de atributos de calidad o Quality Attribute Workshop
- La fórmula Dado, Cuando, Entonces o Given-When-Then es una plantilla cuyo intento es guiar la escritura de pruebas de aceptación para una historia de usuario:
  - (Dado) algún contexto
  - (Cuando) alguna acción es llevada a cabo
  - (Entonces) un conjunto particular de consecuencias observables se deberían obtener

Herramientas:

- Formatos para llenado de escenarios de atributos de calidad en caso de utilizar taller de atributos de calidad
- Herramientas para la administración de proyectos como en el caso de criterios de aceptación en historias de usuario :
  - Trello (comercial)

- JIRA (comercial)
- Visual Studio Team Services (comercial)
- VersionOne (comercial)
- Select Scope Manager (comercial)
- Taiga.io (GPL)
- Tuleap (GPL)

- Procesadores de texto

Artefactos:

- Lista de atributos de calidad, idealmente que se pueden cuantificar de manera explícita
- Criterios de aceptación para realizar pruebas posteriormente

Consejos:

- Preguntas como las siguientes pueden ayudar:
  ¿Cuántos usuarios concurrentes debería el sistema soportar en promedio?, ¿Cómo intervienen las horas pico?, ¿Qué tiempo de respuesta es considerado aceptable?, ¿El tiempo de respuesta debe ser el mismo en todas las partes del software o sólo para características específicas? además de preguntar sobre el número de usuarios concurrentes, volúmenes de datos y número máximo de respuestas.
- Este tipo de pruebas a regularmente se agregan al reverso de la historia de usuario cuando se cuenta con un tablero físico.

29

Los escenarios de atributos de calidad han sido priorizados *(A)*

Objetivo: Identificación de los escenarios de atributos de calidad de mayor relevancia

Prioridad: Baja	**Categoría:** Validación
Etapa: Requisitos	**Roles:** 

Técnicas:

- Juego del Árbol del Producto o *Prune the Product Tree*
- La lancha rápida o *Speed Boat*
- Priorización MoSCoW (*Must have* o debe tener, *Should have* o debería tener, *Could have* o puede tener y *Won't have* o no tendrá... por el momento)
- Compra una característica o *Buy a feature*
- Cuadrantes de manejo del tiempo de Convey
- Proceso Analítico Jerárquico o *Analytic Hierarchy Process*
- Votación en rondas

Herramientas:

- Plantilla de árbol
- Herramientas de software como:
  - idea engine (comercial)
  - decision engine (comercial)
  - TransparentChoice's AHP online tool (comercial)
  - easyAHP (Libre)

Artefactos:
Escenarios de atributos de calidad priorizados

30

El esfuerzo necesario para el desarrollo de los escenarios de atributos de calidad ha sido estimado *(A)*

Objetivo: Facilitar estimaciones sobre entregas futuras

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Construcción	**Roles:**

Técnicas:

- Técnicas para estimar esfuerzo como:
  - Serie fibonacci y/o modificaciones de la serie (0, 1, 2, 3, 5, 8, 13, 21, etc).
  - Camisas o T-Shirts (xxs, xs, s, m, l, xl, xxl)
  - Múltiplos de 2 (2, 4, 6, 8, 10, 12, 14, etc.)

Herramientas:

- Aplicaciones en línea o descargables como:
- Planningpoker.com
- Planning Poker
- Agile Planning Poker
- Scrum Poker Cards
- BEEVA Planning Poker

Artefactos:
Escenarios de atributos de calidad estimados

9

Las reglas de negocio han sido consideradas como fuente de requisitos emergentes *(B)*

Objetivo: Prevención de cambios

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Cualquier etapa	**Roles:** 

Técnicas:

- Historias de usuario o actualización de documento de especificación de requisitos para documentar los requisitos emergentes

Herramientas:

- Procesadores de texto
- Herramientas para la administración de proyectos como:
  - Trello (comercial)
  - JIRA (comercial)
  - Visual Studio Team Services (comercial)
  - VersionOne (comercial)
  - Select Scope Manager (comercial)
  - Taiga.io (GPL)
  - Tuleap (GPL)

Artefactos:

Historias de usuario o documento de especificación de requisitos actualizado

Consejo:

Uso del diccionario de reglas de negocio, glosario, diccionario de datos, o descripción de las reglas de negocio para descubrir nuevas restricciones

10

Soportar la aparición de requisitos emergentes durante el desarrollo *(B)*

Objetivo: Aceptación de cambios durante la construcción del producto de software

Prioridad: Baja	**Categoría:** Validación
Etapa: Cualquier etapa	**Roles:** 

Técnicas:

- Adición de extensiones, excepciones, alternativas, variaciones, manejo de errores y atributos de calidad a las historias de usuario o casos de uso.

Herramientas:

- Procesadores de texto
- Herramientas para la administración de proyectos como:
  - Trello (comercial)
  - JIRA (comercial)
  - Visual Studio Team Services (comercial)
  - VersionOne (comercial)
  - Select Scope Manager (comercial)
  - Taiga.io (GPL)
  - Tuleap (GPL)

Artefactos:

Historias de usuario o casos de usuario actualizados

31

Restricciones técnicas y de negocio han sido identificadas *(C)*

Objetivo: Establecimiento de las características que debería o no cumplir el sistema a construir

Prioridad: Cualquier etapa	**Categoría:** Obtención y análisis
Etapa: Baja	**Roles:** 

Técnicas:

- Recopilar información sobre los diferentes tipos de restricciones que influenciarán en el sistema de software a construir

Herramientas:

- Procesadores de texto
- Herramientas para el modelado UML como:
  - Modelio (GPL & comercial)
  - MySQL Workbench (GPL & comercial)
  - Lucidchart (comercial)

- Microsoft Visio (comercial)

Artefactos:
Lista de restricciones técnicas y de negocio

11

Las restricciones han sido utilizadas en la prueba de escenarios de atributos de calidad *(C)*	
Objetivo: Asegurar la consideración de las restricciones en la construcción del producto de software	
Prioridad: Baja	**Categoría:** Validación
Etapa: Requisitos, construcción y pruebas	**Roles:**

Técnicas:

- Escenarios de atributos de calidad

Herramientas:

- Procesadores de texto
- Formatos para llenado de escenarios de atributos de calidad

Artefactos:
Escenarios de atributos de calidad actualizados

32

La lista de tecnologías aprobadas por la compañía ha sido revisada *(SG)*

Objetivo: Conocimiento de las diferentes tecnologías que la compañía tiene que mantener, operar y comprar licencias

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Comparación de las tecnologías a utilizar con la lista de tecnologías aprobadas por la compañía para verificar la viabilidad de su uso
- Revisión de base de datos de gestión de la configuración

Herramientas:
Procesadores de texto

Artefactos:
Lista de tecnologías fuera de las aprobadas por la organización

12

Las dependencias con sistemas externos han sido revisadas *(SG)*

Objetivo: Identificar si es necesario integrar sistemas externos con el que se está construyendo

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Revisión de restricciones organizacionales dictando protocolos y tecnologías a utilizar para los puntos de integración
- Diagrama de contexto para modelar dependencias con sistemas externos

Herramientas:

- Procesadores de texto o herramientas para el modelado UML como:
  - Modelio (GPL & comercial)
  - MySQL Workbench (GPL & comercial)
  - Lucidchart (comercial)
  - Microsoft Visio (comercial)

Artefactos:

- Lista de protocolos y tecnologías a utilizar
- Diagrama de contexto del sistema

13

La plataforma de despliegue destino ha sido revisada *(SG)*

Objetivo: Prevención de impacto de la plataforma de despliegue en las decisiones de tecnología

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Revisión de las características de la plataforma destino
- Implementación de DevOps

Herramientas:
Base de datos de gestión de la configuración

Artefactos:
Lista de restricciones sobre lo que el software puede o no hacer en la plataforma que lo alojará

14

Las características del equipo son consideradas cuando se está arquitectando un sistema de software *(SG)*

Objetivo: Prevención de la necesidad de capacitación o recursos humanos

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos y Diseño	**Roles:** 

Técnicas:

- Liquidez de personal o *Staff Liquidity*

Herramientas:
Matriz de habilidades

Artefactos:
Lista de características a mejorar o trabajar del equipo necesarias para iniciar la construcción de un nuevo sistema

15

El sistema o desarrollo ha sido clasificado de acuerdo a su importancia táctica o estratégica *(SG)*	
Objetivo: Adición o eliminación de nuevas restricciones dependiendo del contexto y del tipo de producto a construir	
Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 
Técnicas: Identificar si el sistema es una implementación táctica o estratégica y las restricciones que esto implica	
Herramientas: No aplica	
Artefactos: Lista de restricciones adicionales o removidas	

16

Las restricciones organizacionales han sido priorizadas *(C)*	
Objetivo: Identificación de las restricciones organizacionales más importantes	
Prioridad: Baja	**Categoría:** Validación
Etapa: Requisitos	**Roles:** 
Técnicas: • Técnicas de priorización como: ○ Priorización MoSCoW (*Must have* o debe tener, *Should have* o debería tener, *Could have* o puede tener y *Won't have* o no tendrá... por el momento) ○ Cuadrantes de manejo del tiempo de Convey ○ Proceso Analítico Jerárquico o *Analytic Hierarchy Process*	
Herramientas: • Pizarrón • Herramientas de software como: ○ idea engine (comercial), ○ decision engine (comercial) ○ TransparentChoice's AHP online tool (comercial) ○ easyAHP (libre)	
Artefactos:	

Lista de restricciones priorizada

33

Los principios de desarrollo y/o arquitectura han sido identificados y seleccionados *(SG)*

Objetivo: Introducción de claridad y consistencia al código base

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Diseño	**Roles:** 

Técnicas:

- Considerar la complejidad del software que se está construyendo, además de las restricciones del ambiente, ayudarán a decidir qué principios adoptar. El contexto es siempre la clave.

Herramientas:
No aplica

Artefactos:
Lista de principios de diseño y/o arquitectura a adoptar

Consejos:

- Tener una lista de principios puede ayudar a asegurar que todo el mundo dentro del equipo está trabajando de la misma manera
- Escuchar la retroalimentación de los miembros del equipo ayudará a decidir cuáles principios están funcionando y cuáles no

17

Retroalimentación del producto es obtenida durante el desarrollo *(SG)*

Objetivo: Evitar desviaciones en el proyecto debido a los requisitos emergentes

Prioridad: Alta	**Categoría:** Validación
Etapa: Construcción	**Roles:** 

Técnicas:

- Adopción de desarrollo incremental
- Establecimiento de reuniones de entrega frecuentes durante el proyecto

Herramientas:
No aplica

Artefactos:
Lista de cambios solicitados por el dueño del producto y/o usuarios finales

Consejo: En Scrum la retroalimentación del producto de software se obtiene durante la reunión de revisión del sprint

34

Todos los involucrados acuden a las mismas reuniones *(SG)*	
Objetivo: Todos los involucrados deben conocer en el mismo grado de detalle los requisitos	
Prioridad: Baja	**Categoría:** Validación
Etapa: Cualquier etapa	**Roles:** 

Técnicas:

- Establecer con los involucrados propósitos, fechas, horas y duración de las reuniones

Herramientas:
Calendarios en línea como Google Calendar, iCal, Live Calendar y Yahoo! Calendar

Artefactos:
Notificaciones electrónicas

35

Los riesgos del proyecto han sido identificados *(D)*	
Objetivo: Conocer los diferentes riesgos a los que se puede enfrentar el proyecto durante su desarrollo	
Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Realizar lluvias de ideas sobre riesgos del proyecto
- Identificación de riesgos de la administración del proyecto y riesgos técnicos

Herramientas:
Pizarrón

Artefactos:
Conjunto de historias de usuario que incluyen riesgos en su definición

Consejos:
Colaborar con el equipo de desarrollo para la identificación de riesgos técnicos

18

Los riesgos del proyecto han sido priorizados *(D)*

Objetivo: Identificación de los riesgos más importantes del negocio

Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 

Técnicas:

- Cuadrantes de manejo del tiempo de *Convey*
- Votación en rondas
- Proceso Analítico Jerárquico o *Analytic Hierarchy Process*
- Liquidez de personal o *Staff liquidity*
- Uso de criterios de prioridad alta, media y baja para clasificar los riesgos

Herramientas:

- TransparentChoice's AHP online tool (comercial)
- easyAHP (libre)

Artefactos:
Conjunto de historias de usuario sobre riesgos priorizadas

Consejo:
El equipo de software debe ayudar al dueño del producto a entender los riesgos técnicos y priorizar adecuadamente la lista de producto

19

Los riesgos identificados han sido incluidos a la lista de producto o *backlog (D)*	
Objetivo: Facilitar la priorización de los elementos de la lista del producto	
Prioridad: Baja	**Categoría:** Especificación
Etapa: Requisitos	**Roles:** 
Técnicas: • Convertir la lista del producto en lista de características y riesgos del producto	
Herramientas: • Herramientas para la administración de proyectos como: ○ Trello (comercial) ○ JIRA (comercial) ○ Visual Studio Team Services (comercial) ○ VersionOne (comercial) ○ Select Scope Manager (comercial) ○ Taiga.io (GPL) ○ Tuleap (GPL)	
Artefactos: Lista del producto actualizada 24	

Riesgos en los requisitos han sido identificados *(D)*	
Objetivo: Conocer los riesgos relacionados a los requisitos a los que se puede enfrentar el proyecto durante su desarrollo	
Prioridad: Baja	**Categoría:** Obtención y análisis
Etapa: Requisitos	**Roles:** 
Técnicas: • Lluvia de ideas • Structured SWIFT • Análisis de situaciones • Análisis de incidencias • El Análisis del Árbol de Fallas o *Fault tree analysis* • Análisis en forma de corbata o *The Bowtie Method*	
Herramientas: Pizarrón	

Artefactos:
Lista de riesgos en los requisitos

20

Riesgos en los requisitos de atributos de calidad han sido identificados y se ha obtenido una lista priorizada de escenarios de fallo *(D)*

Objetivo: Contrarrestar la falta de entendimiento o la incompletitud de requisitos de atributos de calidad

Prioridad: Alta	**Categoría:** Validación
Etapa: Requisitos	**Roles:**

Técnicas:

- Taller de atributos de calidad o *Quality Attribute Workshop*
- Cuestionarios de taxonomía de riesgos o *Taxonomy-Based Questionnaire* (TBQ) (Carr et al., 1993)
- Lluvia de ideas
- Análisis de situaciones
- El Análisis del Árbol de Fallas o *Fault tree analysis*
- Análisis en forma de corbata o *The Bowtie Method*

Herramientas:
Pizarrón

Artefactos:
Lista de riesgos y escenarios de fallo

36

Anexo G

Lista de verificación de los elementos del diseño de caso de estudio (P. Runeson et. al., 2012).

	Element	Example Questions Describing the element	Page	Done
1	Rationale	Why is the study being done?	24	x
2	Purpose	What is expected	24	x
3	The case	Overall, what is being studied?	26	x
3.1	Units of analysis	In more detail, what is being studied?	26	x
4	Theory	What is the theoretical frame of reference?	29	--
5	Research questions	What knowledge will be sought or expected to be discovered?	30	x
6	Propositions and hypothesis	What particular (causal) relationships are to be investigated?	31	x
7	Define concepts and measures	How are entities and attributes being defined and measured?	32	x
8	Methods of data collection	How will data be collected?	32	x
9	Methods of data analysis	How will data be analyzed?	33	x
10	Case selection strategy	How will cases (and units of analysis be identified and selected?	33	x
11	Data selection strategy	How will data be identified and selected? For example, who will be interviewed? What electronic data sources are available for use in the study?	35	x
12	Replication strategy	Is the study intended to literally replicate a previous study: or theoretically replicate a previous study or there no intention to replicate?		x
13	Quality assurance, validity and reliability	How will the data collected be checked for quality? How will the analysis be checked for quality?	36	x

REFERENCIAS

Árias, Á. (2015). *Aprende sobre la ingeniería de software* (p. 88). IT Campus Academy.

Bass, L. Clements, P. y Kazman, R. (2012). *Software Architecture in Practice* (3ra ed.) Addison-Wesley Professional.

Brooks, F. (1995). The Mythical Man-Month Essays on Software Engineering (20 ed., p. 20). Indiana: Addison-Wesley.

Brown, S. (2016). *Software Architecture for Developers* (p. 84). Leanpub.

Coplien, J., y Bjørnvig, G. (2010). *Lean Architecture for Agile Software Development*. Chichester: Wiley.

Fairbanks, G. (2010). *Just Enough Software Architecture* (p. 9, 10, 142). Colorado: Marshall & Brainerd.

Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A., & America, P. (2007). *A general model of software architecture design derived from five industrial approaches.* Journal Of Systems And Software, 80(1), 106-126. http://dx.doi.org/10.1016/j.jss.2006.05.024

International Standard ISO/IEC 12207. (2008) (2da ed., pp. 4,8).

Javier, F., & Izquierdo, P. (2009). Círculos de calidad (p. 91). Barcelona: Marcombo.

Kendall, K., y Kendall, J. (2005). *Análisis y diseño de sistemas* (6th ed., p. 658). México: Pearson Educación.

Lowry, R. (1998). VassarStats: Statistical Computation Web Site. Vassarstats.net. Recuperado el 2 de octubre del 2017, desde: http://vassarstats.net

Schwaber, K., & Shuterland, J. (2016). La Guía de Scrum (p. 6). Recuperado el 16 de Marzo de 2017, desde: https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish-European.pdf

Manifiesto por el Desarrollo Ágil de Software. (2001). Agilemanifesto.org. Recuperado el 16 de Marzo de 2017, desde: http://agilemanifesto.org/iso/es/manifesto.html

Meyer, B. (2014). *Agile!* (p. 2,4). Zurich: Springer International publishing.

Moya, F., González, C., Villa, D., Pérez, S., Redondo, M., Villanueva, F., García, M. (2013). *Desarrollo de videojuegos: técnicas avanzadas* (2da ed., p. 202). Ciudad Real: Universidad de Castilla La Mancha.

Murch, R. (2012). *The Software Development Lifecycle - A Complete Guide* (pp. 21, 28, 21, 49, 56, 73).

Noriega, R. (2017). *El proceso de desarrollo de software* (2da ed., p. 5). IT Campus Academy

Pérez, J., y Merino, M. (2008). *Definición de técnica.* Definición.de. Recuperado el 1 de Abril de 2017, desde: http://definicion.de/tecnica/

Pérez, J., y Merino, M. (2010). *Definición de herramienta*. Definición.de. Recuperado el 1 de Abril de 2017, desde: http://definicion.de/herramienta/

Runeson, P., Host, M., Reiner, A., & Regnell, B. (2012). Case Study Research in Software Engineering. Canadá: Wiley.

Salganik, M., & Levy, K. (2015). *Wiki Surveys: Open and Quantifiable Social Data Collection*. PLOS ONE, 10(5). http://dx.doi.org/10.1371/journal.pone.0123483

Software Engineering Institute. *Quality Attribute Workshop.* (2017). Software Engineering Institute | Carnegie Mellon University. Recuperado el 02 de Agosto de 2017, desde: http://www.sei.cmu.edu/architecture/tools/establish/qaw.cfm

Sommerville, I. (2005). *Ingeniería del Software* (7ma ed., pp. 69,70). Madrid: Pearson Educación, S. A.

Soria, I., y Condor, E. (2016). *Compendio de Ingeniería de Software* (p. 59).

The Standish Group International, Inc., (2016). *The chaos report* (p. 3).