PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

PROPUESTA DE METODOLOGÍA ÁGIL DE DESARROLLO DE SOFTWARE CON INTEGRACIÓN DE TAM – *Z-ÁGIL*

JUAN FRANCISCO GARCÍA TOLEDO

TESIS DE GRADO

MAGÍSTER EN INGENIERÍA INFORMÁTICA

DICIEMBRE 2012

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

PROPUESTA DE METODOLOGÍA ÁGIL DE DESARROLLO DE SOFTWARE CON INTEGRACIÓN DE TAM – *Z-ÁGIL*

JUAN FRANCISCO GARCÍA TOLEDO

Director de Tesis: José Miguel Rubio León
Programa: Magíster en Ingeniería Informática

DICIEMBRE 2012

Resumen

Las Metodologías Ágiles de desarrollo de software han demostrado ser altamente efectivas en proyectos con requerimientos y entornos cambiantes. A diferencia de las Metodologías Tradicionales que ponen un riguroso control sobre el proceso, las actividades y artefactos que se usarán, las Metodologías Ágiles ponen mayor énfasis y valor en el individuo, a la colaboración con el cliente y al desarrollo incremental con iteraciones cortas.

El siguiente trabajo propone la creación de una Metodología Ágil con integración del Modelo de Aceptación de la Tecnología (TAM). Con la inclusión de este modelo se espera obtener información sobre la aceptación, las percepciones de utilidad y usabilidad, datos que servirán como indicadores para realizar mejoras y correcciones a los sistemas durante su construcción.

Palabras clave: Desarrollo de Software, Metodologías Ágiles, Modelo de Aceptación de la Tecnología.

Abstract

The Agile Methodologies of software development has proven to be highly effective in projects with changing requirements and environments. Unlike the traditional methodologies that put rigorous control on the process, the activities and the artifacts that will be used, the Agile Methodologies place major emphasis and value in the individual, to the collaboration with client and to the incremental development with short iterations.

The following master thesis proposes the creation of an Agile Methodology integrated to the Technology Acceptance Model (TAM). With the incorporation of this model is expected to obtain information about the acceptance, the perceptions of usefulness and usability, data that will serve as indicators to realice improvements and corrections to the systems during his construction.

Keywords: Software Development, Agile Methodologies, Technology Acceptance Model.

Índice

Lista de Figuras

Lista de Tablas

1 Introducción

Para asegurar el éxito durante el desarrollo de software no es suficiente contar con notaciones de modelado y herramientas, hace falta un elemento importante: la metodología de desarrollo, la cual provee de una dirección a seguir para la correcta aplicación de los demás elementos.

Generalmente el proceso de desarrollo llevaba asociado un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades, artefactos, incluyendo modelado y documentación detallada. Este esquema para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de rigurosidad en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante y resulta difícil aplicar énfasis en el control de los procesos y la rigurosa definición de roles, actividades, artefactos, incluyendo modelado y documentación detallada que las Metodologías Tradicionales exige.

Ante las dificultades para utilizar Metodologías Tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo prescinden de las buenas prácticas de la ingeniería del software, asumiendo el riesgo que ello conlleva.

ZEKE es una empresa de consultoría y desarrollo de software a medida fundada en el año 2005. En los últimos 2 años ha registrado un rápido y sostenido crecimiento. La cantidad y complejidad de los proyectos no es la misma que en sus inicios, los clientes exigen mayor rigurosidad en los procesos y en la calidad de los productos. El proceso de desarrollo de software no se ha modificado para afrontar estos nuevos desafíos lo cual ha impactado considerablemente en la calidad de los productos liberados en la actualidad.

Este nuevo escenario ha hecho replantear a los directivos de ZEKE la efectividad del modelo actual de desarrollo. Las Metodologías Ágiles emergen como una posible respuesta para llenar ese vacío, en donde se cuentan con proyectos con un entorno cambiante, equipos de desarrollos pequeños, plazos reducidos, requisitos volátiles o basados en nuevas tecnologías. Las Metodologías Ágiles aportan una elevada simplificación sin renunciar a las prácticas esenciales para asegurar la calidad del producto.

Las Metodologías Ágiles son sin duda uno de los temas recientes en ingeniería de software que están acaparando gran interés. Prueba de ello es que se están haciendo un espacio destacado en la mayoría de conferencias celebrados en los últimos años. Es tal su impacto que actualmente existen 4 conferencias internacionales de alto nivel y específicas sobre el tema.

Hoy en día, el uso óptimo de las tecnologías de la información y comunicación dentro de las organizaciones es una necesidad, ante su influencia en la producción de bienes y servicios de calidad. Eso, asociado que estas tecnologías son cada vez más asequibles en el mercado, resaltando entonces la gestión para su uso adecuado en lugar de la inversión en ellas.

Existen varios modelos para medir la aceptación de la tecnología, entre ellos destaca el Modelo de Aceptación de la Tecnología (Davis, 1989), por ser un modelo altamente probado en

predecir el uso de las tecnologías de información y de comunicación. Es el modelo que goza de un mayor reconocimiento en la literatura sobre la adopción de innovaciones tecnológicas en esta última década, fue diseñado para realizar medidas evaluadoras de la calidad de los sistemas de información y de su ajuste de los requerimientos, por tanto, se utiliza para hacer predicciones de aceptación y de uso.

La aplicación de este tipo de modelo de aceptación, puede ser de gran utilidad ya que entrega valiosa información sobre la probabilidad de éxito de una TI en este caso un producto Software como también puede dejar en evidencia las mejoras que se debieran realizar al producto durante su desarrollo para mejorar la aceptación

Se propone la creación de una Metodología Ágil de desarrollo de software para ZEKE, que se adapte a las nuevas situaciones de la empresa y entregue a ella los modelos, procesos y herramientas para enfrentar los nuevos desafíos asegurando calidad, usabilidad y aceptación de sus productos mediante la integración del Modelo de Aceptación de la Tecnología.

2 Análisis de Objetivos

2.1 Objetivo general

Crear una Metodología Ágil de desarrollo de software con integración de TAM para la empresa ZEKE.

2.2 Objetivos específicos

Para el cumplimiento del objetivo general del proyecto, se debe lograr los siguientes objetivos específicos:

- Estudio del Modelo de Aceptación de la Tecnología y de Metodologías Ágiles de desarrollo de software.
- Estudio del actual modelo de proceso de desarrollo de software de ZEKE para identificación de deficiencias y aciertos.
- Creación de una Metodología Ágil de Desarrollo de Software con Integración de TAM
- Validación de la Metodología mediante la aplicación en el desarrollo de un proyecto real

2.3 Metodología y Plan de trabajo

El desarrollo de esta nueva Metodología ha sido trabajado en conjunto con los principales actores que participan en el actual proceso de desarrollo en ZEKE, se espera contar con el conocimiento y validación por parte de ellos.

Quién desarrolla este proyecto desempeña labores ligadas al desarrollo de software en ZEKE. Ambos vienen trabajando una relación laboral desde hace más de 2 años. ZEKE ha aceptado la propuesta de una nueva creación de una Metodología que ayude a mejorar su actual proceso de desarrollo de software, sus directivos están conscientes de que existe un problema con el desarrollo, vinculado principalmente al actual proceso con el cual se construye software.

Los directivos de la empresa, no han puesto límites en lo que ha tiempo y recursos se requiera para lograr los objetivos generales y específicos. Se pretende realizar dentro de la empresa solo trabajos prácticos como entrevistas y aplicaciones de encuestas, destinando el trabajo de investigación y análisis fuera de ella.

En la Figura 2.1 se detallan las actividades a realizar.

	Task Name	Duration	Start	Finish
1	**Estudios**	**15 days**	**Thu 01/03/12**	**Wed 21/03/12**
2	Estudio de Metodologias Agiles de desarrollo de software	10 days	Thu 01/03/12	Wed 14/03/12
3	Estudio del Modelo de Aceptacion de la Tecnología	5 days	Thu 15/03/12	Wed 21/03/12
4	**Compresión del actual proceso de desarrollo de software en ZEKE**	**26 days**	**Thu 22/03/12**	**Thu 26/04/12**
5	Proceso de estimación y planificación	2 days	Thu 22/03/12	Fri 23/03/12
6	Formación y reclutamiento de los equipos de trabajo	2 days	Mon 26/03/12	Tue 27/03/12
7	Proceso de Análisis	3 days	Wed 28/03/12	Fri 30/03/12
8	Proceso de Construcción	3 days	Mon 02/04/12	Wed 04/04/12
9	Proceso de Aseguramiento de la calidad	2 days	Thu 05/04/12	Fri 06/04/12
10	Proceso de transferencia, implantacion y cierre	4 days	Mon 09/04/12	Thu 12/04/12
11	Proceso de control y seguimiento de proyectos	5 days	Fri 13/04/12	Thu 19/04/12
12	Modelado de los procesos	5 days	Fri 20/04/12	Thu 26/04/12
13	**Identificación de deficiencias en el actual proceso de desarrollo de softwa**	**27 days**	**Fri 27/04/12**	**Mon 04/06/12**
14	**Búsqueda de deficiencias en los equipos de desarrollo**	**13 days**	**Fri 27/04/12**	**Tue 15/05/12**
15	Entrevista a jefes de proyecto	3 days	Fri 27/04/12	Tue 01/05/12
16	Entrevista a analistas	3 days	Wed 02/05/12	Fri 04/05/12
17	Entrevista a programadores	5 days	Mon 07/05/12	Fri 11/05/12
18	Entrevista a aseguradores de calidad	2 days	Mon 14/05/12	Tue 15/05/12
19	**Búsqueda de deficiencias en proyectos no exitosos**	**6 days**	**Wed 16/05/12**	**Wed 23/05/12**
20	Selección de proyectos no exitosos	2 days	Wed 16/05/12	Thu 17/05/12
21	Entrevista con equipos desarrolladores de los proyectos	3 days	Mon 21/05/12	Wed 23/05/12
22	**Determinacion de calidad de los sistemas desarrollados**	**5 days**	**Thu 24/05/12**	**Wed 30/05/12**
23	Satisfacción de los clientes	3 days	Thu 24/05/12	Mon 28/05/12
24	Evaluación de propiedades de los sistemas	1 day	Wed 30/05/12	Wed 30/05/12
25	Clasificación de las deficiencias	2 days	Thu 31/05/12	Fri 01/06/12
26	Análisis FODA	1 day	Mon 04/06/12	Mon 04/06/12
27	Desarrollo de nueva Metodología	24 days	Tue 05/06/12	Fri 06/07/12

Figura 2.1 Programación de actividades

3 Estado del Arte

3.1 Modelo de Aceptación de la Tecnología (TAM)

El Modelo de Aceptación de la Tecnología (TAM) es un modelo diseñado por Fred Davis (Davis, 1989) para realizar medidas evaluadoras de la calidad de los sistemas de información y de su ajuste a los requerimientos de las tareas a ejecutar. El TAM fue especialmente diseñado para predecir la aceptación de los sistemas de información por los usuarios en las organizaciones. Según Davis, el propósito principal de TAM es explicar los factores que determinan el uso de las TIC por un número importante de usuarios. El TAM sugiere que la utilidad y la facilidad de uso son determinantes en la intención que tenga un individuo para usar un sistema.

Este modelo se utiliza para predecir el uso de las TIC, basándose en dos características principales:

1. Utilidad percibida (Perceived Usefulness).

2. Facilidad de uso percibida (Perceived Ease of Use).

La utilidad percibida (PU) se refiere al grado en que una persona cree que usando un sistema en particular mejorará su desempeño en el trabajo, y la facilidad de uso percibida (PEOU) señala hasta qué grado una persona cree que usando un sistema en particular realizará menos esfuerzo para desempeñar sus tareas.

Ese modelo propone que las percepciones de un individuo en cuanto a la utilidad y la facilidad de uso percibidas de un sistema de información son concluyentes para determinar su intención de usar un sistema.

De acuerdo con este modelo, existen variables externas que influyen de manera directa en la PU y la PEOU. Por medio de esta influencia directa en ambas percepciones, las variables externas participan de forma indirecta en la actitud hacia el uso, la intención conductual para usar y la conducta de uso real. La PEOU tiene un efecto causal en la PU, además del efecto significativo de esta variable en la actitud del usuario (un sentimiento en favor o en contra) hacia el uso del sistema. El modelo TAM se puede observar en la Figura 3.1.

Aunque el TAM ayuda a conocer si una tecnología va a ser utilizada de manera óptima, es necesario identificar las variables externas que influyen de manera directa en la utilidad y la facilidad de uso percibidas por los usuarios de las TIC y determinar la relación que guardan con el resultado del uso de estas tecnologías.

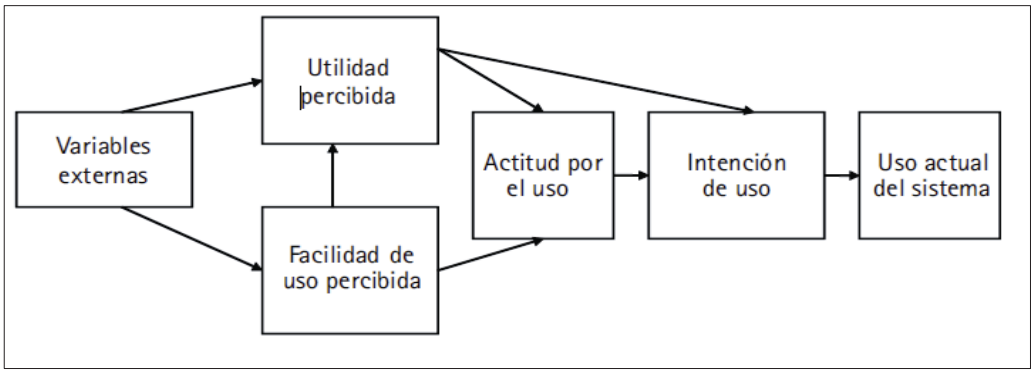


Figura 3.1 Modelo de aceptación de la tecnología

El propósito primario de TAM es indagar las consecuencias de los factores externos en cuanto a la utilidad y la facilidad de uso percibidas, para adelantar o predecir el uso de las TIC. Si bien el modelo TAM ayuda a conocer si una tecnología será utilizada de manera óptima, es necesario identificar las variables externas que inciden en ella, como las causantes de influir de manera directa en la utilidad y la facilidad de uso percibidas por los usuarios de las TIC y determinar la relación de dichas variables con el resultado de su uso.

EL TAM basa su funcionalidad en el uso de cuestionarios para medir la facilidad y usabilidad en el usuario en relación con la implementación de una herramienta. La formulación de los cuestionarios debe reflejar los siguientes aspectos:

a. Utilidad.

b. Facilidad de uso.

c. Actitud de uso.

d. Uso actual o intención de uso.

3.2 Metodologías Ágiles de Desarrollo

Las Metodologías Tradicionales de software han demostrado ser efectivas y necesarias en proyectos de gran tamaño, en donde se exige un alto grado de rigidez en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno de desarrollo es cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Ante las dificultades para utilizar Metodologías Tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo dejan de lado las buenas prácticas de la Ingeniería del Software, asumiendo el riesgo que ello conlleva. En este contexto, las Metodologías Ágiles emergen como una posible respuesta para llenar ese vacío metodológico. Por estar especialmente orientadas para proyectos pequeños, las Metodologías Ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación.

3.2.1 Camino hacia las Metodologías Ágiles

El modelo de cascada marcó un hito en la industria del software, de este se desprendió el análisis estructurado el cual fue uno de los precursores hacia nuevas y más complejas prácticas dentro de la Ingeniería de Software. El modelo de cascada surgió como una mejora al modelo de codificar y probar que era el predominante en los 70's a pesar de que existían ya modelos iterativos pero no estaban formalizados.

El modelo de cascada se hizo popular por el hecho de ser sencillo de poner en práctica. El inicio de una etapa no puede comenzar sin la finalización de la etapa predecesora, las fases del modelo son; el análisis de requisitos, diseño, implementación, pruebas e implantación. La rigidez del modelo exige contar con una definición clara y temprana de los requerimientos, ya que al presentar cambios se requiere un gran esfuerzo en re-trabajo.

Otro problema con el modelo ocurre con la integración de módulos. La fase de implementación del modelo requiere el desarrollo de los módulos en forma independiente con las correspondientes pruebas unitarias, y en la siguiente fase, se realizaba la integración de los mismos. Esto trae grandes inconvenientes debido a que todo estaba probado en forma unitaria sin interacción con los demás módulos. Siempre existía un costo asociado con la integración de los módulos que se veía reflejado en el tiempo del proyecto y calidad del mismo.

Para subsanar estas deficiencias fueron surgiendo diversos modelos denominados iterativos que proponían lidiar con la variabilidad del software mitigando los riesgos en forma temprana. El modelo en espiral creado por Barry Boehm (Boehm, 1986), de carácter iterativo en sus primeras fases, planteó la necesidad de efectuar al principio diversas iteraciones dirigidas a mitigar los riesgos más críticos en el proyecto mediante la realización de prototipos.

Una vez que esos prototipos son validados se suceden iteraciones del tipo: determinar objetivos, evaluar, desarrollar, planear. Una vez que se tenía el diseño detallado y validado por el cliente, se implementaba el software siguiendo las etapas de un modelo en cascada. Esta es una falla importante del modelo ya que no se acomoda a la posibilidad de cambios una vez que se inicia la construcción. Todas las críticas que se le hacían al modelo en cascada se aplican a estas fases del modelo en espiral.

Barry Boehm, consciente de aquello, publicó un artículo en el cual determina los 3 hitos críticos para ser utilizados en cualquier proyecto para poder planificar, controlar el progreso y dar visibilidad a los stakeholders. Estos hitos están relacionados con las etapas de avance que se van dando a lo largo de un proyecto de acuerdo a como ocurren las actividades de ingeniería (que componen los espirales del modelo en espiral) a las actividades de producción (que componen la construcción en cascada del software). Su impacto en la industria del software ha sido tan importante que uno de los procesos más utilizados en la actualidad, el RUP, los incorpora. Estos hitos son:

- Objetivos del Ciclo de Vida
- Arquitectura del Ciclo de Vida
- Capacidad de Operación Inicial

Los hitos se ven reflejados en las fases que dispone RUP, es así como el primer hito finaliza con la definición del alcance del software, la identificación de los stakeholders y el plan del desarrollo del sistema al finalizar la fase de Concepción. El segundo hito finaliza con el documento que detalla la arquitectura del sistema, la resolución de todos los riesgos críticos del proyecto, y el refinamiento de los objetivos y el alcance del sistema. La construcción se inicia a partir de este último hito, comenzando las fases más predecibles en cierta medida del desarrollo. El mismo corresponde al hito final de la fase de Elaboración según el RUP. El último de los hitos está asociado a la entrega del primer release del software, que incorpora la funcionalidad definida en la correspondiente iteración, a este nivel de avance ya se debe contar con artefactos para los usuarios como lo son los Manuales del Usuario y Manuales de Operaciones. Este hito se corresponde con el hito final de la fase de Construcción según el RUP.

RUP fue creado para que el mismo fuera un repositorio de todas las ideas vigentes y las denominadas buenas prácticas de la Ingeniería de Software. Sin embargo, al intentar abarcar proyectos de envergaduras tan dispares y cambiantes, RUP pierde la granularidad necesaria para describir en detalle uno de los factores más trascendentes de cualquier desarrollo de software: las personas.

Esta es una de las razones principales del advenimiento de las denominadas Metodologías Ágiles.

3.2.2 Origen de las Metodologías Ágiles

Hasta los 90's se pensaba que para la obtención de software a tiempo con el costo y la calidad requerida se debía contar con procesos de desarrollo altamente definidos, esta creencia fue perdiendo apoyo con el lanzamiento de una revolucionaria manera de desarrollar. Esta nueva manera de trabajo correspondía a una nueva Metodología denominada XP - eXtreme Programming (Beck & Andres, 2004).

XP a diferencia de las metodologías tradicionales, principalmente, pone más énfasis en la adaptabilidad que en la previsibilidad con la premisa de poder adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto a diferencia de intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Es así como que este tipo de metodologías fueron inicialmente llamadas “metodologías livianas”, sin embargo, aún no contaban con una aprobación, pues se le consideraba por muchos desarrolladores como meramente intuitiva. Luego, con el pasar de los años, en febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace formalmente el término “ágil” aplicado al desarrollo de software. En esta misma reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software con el objetivo de esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Tras esta reunión se creó The Agile Alliance, una organización sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y

ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía ágil.

3.2.3 El Manifiesto Ágil

El Manifiesto Ágil comienza enumerando los principales valores del desarrollo ágil (Penadés, Letelier, & Canós, 2006). Según el Manifiesto se valoran:

- Valorar más a los individuos y su interacción que a los procesos y las herramientas: Las personas son el principal factor de éxito de un proyecto software, este es, posiblemente, el principio más importante del manifiesto. Las personas sin conocimiento técnico y actitud adecuada, no producen resultados a pesar de contar con procesos definidos. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

- Valorar más el software que funciona que la documentación exhaustiva: La utilización de prototipos o partes ya desarrolladas de un sistema ofrece una retroalimentación que enriquece el proceso de codificación ya que genera nuevas ideas que son imposibles de concebir en un primer momento. Los documentos no pueden sustituir, ni pueden ofrecer la riqueza y generación de valor que se logra con la comunicación directa entre las personas y a través de la interacción con los prototipos. Por eso, siempre que sea posible debe preferirse, y reducir al mínimo indispensable el uso de documentación, que genera trabajo que no aporta un valor directo al producto.

- Valorar más la colaboración con el cliente que la negociación contractual: Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito. En el desarrollo ágil el cliente es un miembro más del equipo, que se integra y colabora en el grupo de trabajo. Los modelos de contrato por obra no encajan.

- Valorar más la respuesta al cambio que el seguimiento de un plan: Para un modelo de desarrollo que surge de entornos inestables, que tienen como factor inherente al cambio y la evolución rápida y continua, resulta mucho más valiosa la capacidad de respuesta que el seguimiento y aseguramiento de planes pre-establecidos. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo.

Principios del Manifiesto Ágil:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

4. Las personas del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

5. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

7. El software que funciona es la medida principal de progreso.

8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

10. La simplicidad es esencial.

11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

3.2.4 Tipos Metodologías Ágiles de Desarrollo de Software

3.2.4.1 XP- eXtreme Programming

XP es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado, centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Xp se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Beck & Andres, 2004).

A continuación se presentan algunas características esenciales de XP organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas.

A. Las Historias de Usuario. Son la técnica utilizada para especificar los requisitos del software tienen el mismo propósito que los casos de uso. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Las historias de usuario son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario. El tratamiento de las historias de usuario es muy dinámico y flexible.
   Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Beck en su libro (Beck & Andres, 2004) presenta un ejemplo de ficha, el contenido puede ser: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con la fecha, estado cosas por terminar y comentarios. A efectos de planificación, las historias pueden ser de una a tres semanas de tiempo de programación. Las historias de usuario son descompuestas en tareas de programación (task card) y asignadas a los programadores para ser implementadas durante una iteración.

B. Roles XP. Los roles de acuerdo con la propuesta original de Beck son:

   - Programador: El programador escribe las pruebas unitarias y produce el código del sistema.

   - Cliente: Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

   - Tester: Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

- Tracker: Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

- Coach: Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

- Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.

- Gestor: Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

C. Ciclo de vida de XP. El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. El ciclo de vida de XP se enfatiza en el carácter interactivo e incremental del desarrollo, una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas que en el caso de XP corresponden a un conjunto de historias de usuarios. Las iteraciones son relativamente cortas ya que se piensa que entre más rápido se le entreguen desarrollos al cliente, más retroalimentación se va a obtener y esto va a representar una mejor calidad del producto a largo plazo, ver Figura 3.2.

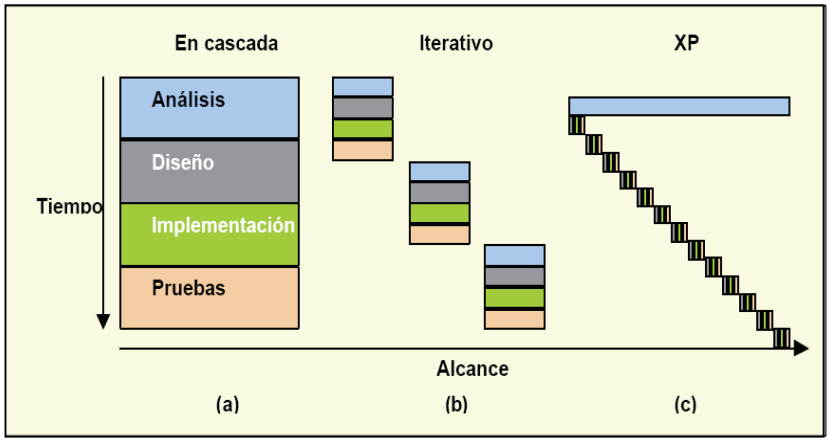


Figura 3.2 Desarrollo iterativo en XP. Fuente: (Galván & Torres, 2010)

Existe una fase de análisis inicial orientada a programar las iteraciones de desarrollo y cada iteración incluye diseño, codificación y pruebas, fases superpuestas de tal manera que no se separen en el tiempo. Las fases en las que se subdivide el ciclo de vida XP son las siguientes:

1. Fase de la exploración: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas,

tecnologías y prácticas que se utilizarán en el proyecto. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2. Fase del planeamiento: Se priorizan las historias de usuario y se acuerda el alcance del release. Los programadores estiman cuánto esfuerzo requiere cada historia y a partir de allí se define el cronograma. La duración del cronograma del primer release no excede normalmente dos meses. La fase de planeamiento toma un par de días. Se deben incluir varias iteraciones para lograr un release. El cronograma fijado en la etapa de planeamiento se realiza a un número de iteraciones, cada una toma de una a cuatro semanas en ejecución. La primera iteración crea un sistema con la arquitectura del sistema completo. Esto es alcanzado seleccionando las historias que harán cumplir la construcción de la estructura para el sistema completo. El cliente decide las historias que se seleccionarán para cada iteración. Las pruebas funcionales creadas por el cliente se ejecutan al final de cada iteración. Al final de la última iteración el sistema está listo para producción.

3. Fase de producción: Requiere prueba y comprobación extra del funcionamiento del sistema antes de que éste se pueda liberar al cliente. En esta fase, los nuevos cambios pueden todavía ser encontrados y debe tomarse la decisión de si se incluyen o no en el release actual. Durante esta fase, las iteraciones pueden ser aceleradas de una a tres semanas. Las ideas y las sugerencias pospuestas se documentan para una puesta en práctica posterior por ejemplo en la fase de mantenimiento. Después de que se realice el primer release productivo para uso del cliente, el proyecto de XP debe mantener el funcionamiento del sistema mientras que realiza nuevas iteraciones.

4. Fase de mantenimiento: Requiere de un mayor esfuerzo para satisfacer también las tareas del cliente. Así, la velocidad del desarrollo puede desacelerar después que el sistema esté en producción. La fase de mantenimiento puede requerir la incorporación de nueva gente y cambiar la estructura del equipo.

5. Fase de muerte: Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

D. Prácticas XP. La principal suposición que se realiza en XP es la posibilidad de disminuir la curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue con las siguientes prácticas.

- El juego de la planificación. Hay una comunicación frecuente entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.

- Entregas pequeñas. Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema.

- Metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema.

- Diseño simple. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.

- Pruebas. La producción de código está dirigida por las pruebas unitarias. Éstas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.

- Refactorización. Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios.

- Programación en parejas. Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas.

- Integración continúa. Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

- Cliente in-situ. El cliente tiene que estar presente y disponible todo el tiempo para el equipo.

- Estándares de programación. Implementación de estándares de programación para mantener el código legible.

La mayoría de las prácticas propuestas por XP no son novedosas sino que en alguna forma ya habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

3.2.4.2 Scrum

Scrum es un modelo de desarrollo ágil caracterizado por adoptar una estrategia de desarrollo incremental, basar la calidad del resultado más en el conocimiento de las personas en equipos auto-organizados que en la calidad de los procesos empleados (Schwaber, 2009). Al principio del proyecto se define el Product Backlog, que contiene todos los requerimientos funcionales y no funcionales que deberá satisfacer el sistema a construir. Los mismos estarán especificados de acuerdo a las convenciones de la organización ya sea mediante: features, casos de uso, diagramas de flujo de datos, incidentes, tareas, etc. El Product Backlog será definido durante reuniones de planeamiento con los stakeholders. A partir de ahí se definirán las iteraciones, conocidas como Sprint, en las que la aplicación irá evolucionando. Cada Sprint tendrá su propio Sprint Backlog que será un subconjunto del Product Backlog con los requerimientos a ser construidos en el Sprint correspondiente. La duración recomendada del Sprint es de un mes.

Dentro de cada Sprint el Scrum Master (equivalente al Líder de Proyecto) llevará a cabo la gestión de la iteración, convocando diariamente al Scrum Daily Meeting que representa una reunión de avance diaria de no más de 15 minutos con el propósito de tener realimentación sobre las tareas de los recursos y los obstáculos que se presentan. Al final de cada Sprint, se realizará un Sprint Review para evaluar los artefactos construidos y comentar el planeamiento del próximo Sprint.

Como se puede observar en la Figura 3.3, la metodología resulta sencilla definiendo algunos roles y artefactos que contribuyen a tener un proceso que maximiza el feedback para mitigar cualquier riesgo que pueda presentarse.

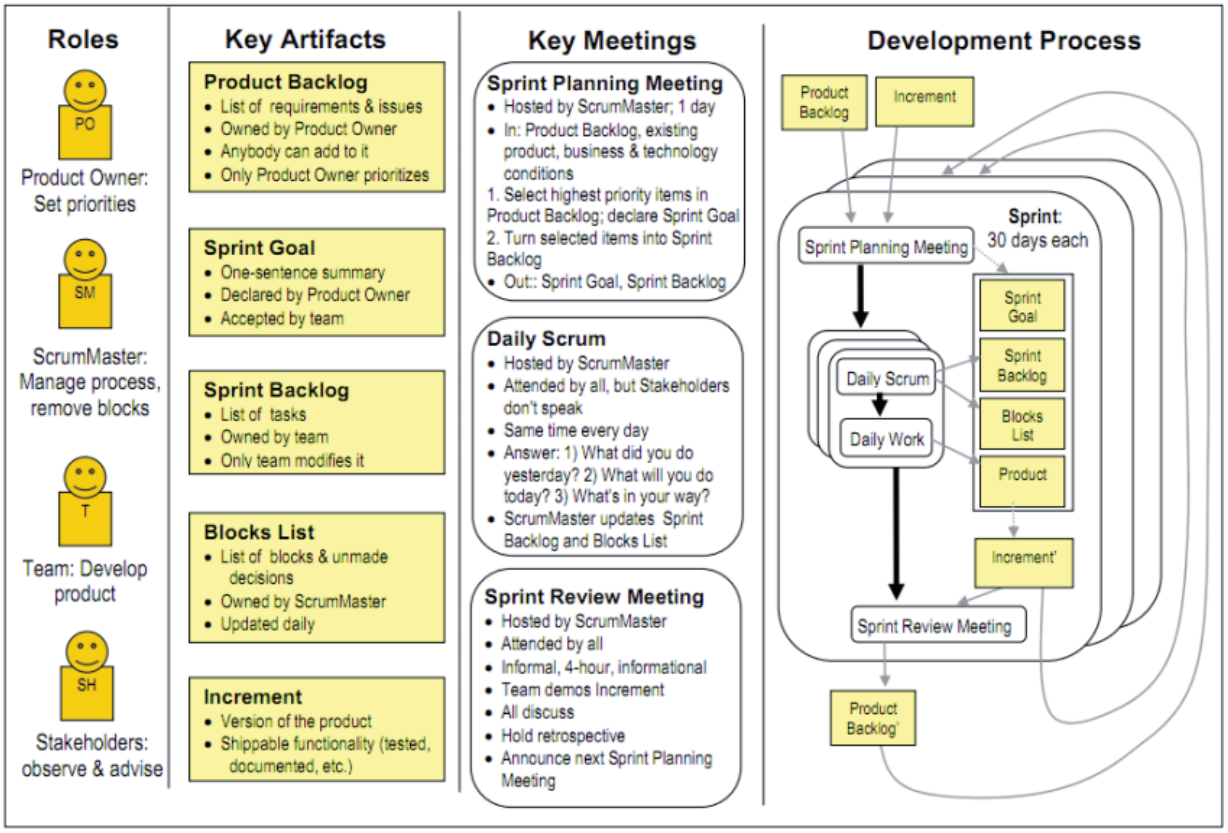


Figura 3.3 Elementos de Scrum. Fuente: (Silvera & de Hormaechea, 2012)

Aunque Scrum surgió como modelo para el desarrollo de productos tecnológicos, también se utiliza en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software.

La intención de Scrum es la de maximizar la realimentación sobre el desarrollo pudiendo corregir problemas y mitigar riesgos de forma temprana. Su uso se está extendiendo cada vez más dentro de la comunidad de Metodologías Ágiles, siendo combinado con otras – como XP – para completar sus carencias. Cabe mencionar que Scrum no propone el uso de ninguna práctica de desarrollo en particular; sin embargo, es habitual emplearlo como un framework ágil de administración de proyectos que puede ser combinado con cualquiera de las metodologías mencionadas. A continuación se definen los Roles y el Ciclo de Vida de Scrum.

A. Roles en Scrum. Scrum define 7 roles, los cuales son:

- Product Owner: Representa la voz del cliente. Se asegura de que el equipo Scrum trabaja de forma adecuada desde la perspectiva del negocio. El Product Owner escribe historias de usuario, las prioriza, y las coloca en el Product Backlog.

- Scrum Master: El Scrum es facilitado por un ScrumMaster, cuyo trabajo primario es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint. El ScrumMaster no es el líder del equipo (porque ellos se auto-organizan), sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. El

ScrumMaster se asegura de que el proceso Scrum se utiliza como es debido. El ScrumMaster es el que hace que las reglas se cumplan.

- Scrum Team: El equipo tiene la responsabilidad de entregar el producto. Un pequeño equipo de 3 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc.).

- Stakeholders: Se refiere a la gente que hace posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su producción. Sólo participan directamente durante las revisiones del sprint.

- Managers: Es la gente que establece el ambiente para el desarrollo del producto.

- Usuarios.

La dimensión del equipo total de Scrum no debería ser superior a diez ingenieros. Si hay más, lo más recomendable es formar varios equipos divididos en Scrums de Scrum.

B. Ciclo de Vida de Scrum:

1. Pre-Juego - Planeamiento: El propósito es establecer la visión, definir expectativas y asegurarse la financiación. Las actividades son la escritura de la visión, el presupuesto, el registro de acumulación o retraso (Backlog) del producto inicial y los ítems estimados, así como la arquitectura de alto nivel, el diseño exploratorio y los prototipos. El registro de acumulación es de alto nivel de abstracción.
2. Pre-Juego - Montaje (Staging): El propósito es identificar más requerimientos y priorizar las tareas para la primera iteración. Las actividades son planificación, diseño exploratorio y prototipos.

3. Juego o Desarrollo: Es donde se desarrollan los Sprint, iteraciones de 30 días aprox. Las reglas de esta fase son sencillas, se distribuyen las tareas para cada miembro del equipo, se trabaja duro y se intentan conseguir el objetivo. Todos los miembros del equipo han de participar en una reunión diaria que en ningún caso deberá exceder los 30 minutos, llamada Sprint-meeting. En esta reunión cada desarrollador debe dar respuesta a tres preguntas: que hizo desde la última reunión, que dificultades se están teniendo en el desarrollo de la tarea, que va a hacer hasta la próxima reunión diaria.

4. Pos-Juego: El propósito es el despliegue operacional. Las actividades, documentación, entrenamiento, mercadeo y venta.

Algunos textos sobre Scrum establecen una arquitectura global en la fase de pre-juego; otros dicen que no hay una arquitectura global en Scrum, sino que la arquitectura y el diseño emanan de múltiples corridas. No hay una ingeniería del software prescripta para Scrum; cada quien puede escoger entonces las prácticas de automatización, inspección de código, prueba unitaria, análisis o programación en pares que le resulten adecuadas.

3.2.4.3 Dynamic Systems Development Method (DSDM)

Es la metodología más antigua de las auto-denominadas ágiles surgidas por el Consorcio Ágil, formado originalmente por 17 miembros fundadores en enero de 1994. El objetivo del Consorcio era producir una metodología de dominio público que fuera independiente de las herramientas y que pudiera ser utilizado en proyectos de tipo RAD (Rapid Application Development). El Consorcio, tomando las mejores prácticas que se conocían en la industria y la experiencia traída por sus fundadores, liberó la primera versión de DSDM a principios de 1995. A partir de ese momento el método fue bien acogido por la industria, que empezó a utilizarlo y a capacitar a su personal en las prácticas y valores de DSDM. Debido a este éxito, el Consorcio comisionó al Presidente del Comité Técnico, Jennifer Stapleton, la creación de un libro que explorara la realidad de implementar el método.

Próxima a los métodos formales, de hecho la implantación de un modelo DSDM en una organización la lleva a alcanzar lo que CMMI Institute consideraría un nivel 2 de madurez. Sin embargo los aspectos que DSDM reprocha a CMMI son:

- Aunque es cierto que CMMI se ha desarrollado con éxito en algunas organizaciones, lo que funciona bien en unos entornos no tiene por qué servir para todos.

- CMMI no le da al diseño la importancia que debería tener.

- CMMI plantea un foco excesivo en los procesos, olvidando la importancia que la industria del software tiene: el talento de las personas.

- El tener procesos claramente definidos no es sinónimo de tener buenos procesos.

Dado el enfoque hacia proyectos de características RAD esta metodología encuadra perfectamente en el movimiento de Metodologías Ágiles. La estructura del método fue guiada por estos nueve principios:

1. El involucramiento del usuario es imperativo.

2. Los equipos de DSDM deben tener el poder de tomar decisiones.

3. El foco está puesto en la entrega frecuente de productos.

4. La conformidad con los propósitos del negocio es el criterio esencial para la aceptación de los entregables.

5. El desarrollo iterativo e incremental es necesario para converger hacia una correcta solución del negocio.

6. Todos los cambios durante el desarrollo son reversibles.

7. Los requerimientos están especificados a un alto nivel.

8. El testing es integrado a través del ciclo de vida.

9. Un enfoque colaborativo y cooperativo entre todos los interesados es esencial.

El equipo mínimo de DSDM es de dos personas y puede llegar a seis, pero puede haber varios equipos en un proyecto. El mínimo de dos personas involucra que un equipo consiste de un programador y un usuario. El máximo de seis es el valor que se encuentra en la práctica. DSDM se ha aplicado a proyectos grandes y pequeños. La precondición para su uso en sistemas grandes es su partición en componentes que pueden ser desarrollados por equipos normales. A continuación se definen los Roles y Ciclo de vida de DSDM.

A. Roles en DSDM. DSDM define 15 roles, aquí se presentan los más importantes:

- Executive Sponsor: Es el rol de la organización que tiene autoridad y responsabilidad financiera relacionada al proyecto, por lo tanto el máximo poder en la toma de decisiones.

- Visionary: El trabajo del visionario es el encargado de asegurar que se satisfacen las necesidades de negocio, es decir, que desde el principio, lo requisitos esenciales se encuentran y que el proyecto sigue la dirección correcta para cumplir dichos requisitos. Es el rol con la visión más precisa sobre los objetivos del negocio del sistema y del proyecto y, probablemente, aquel que tiene la idea inicial de la construcción del sistema.

- Ambassador User: El usuario embajador debe ser miembro del grupo de usuarios, que espera utilizar el sistema, pues este rol tiene como función aportar el conocimiento de este grupo del proyecto y difundir información de los avances del proyecto al resto de los usuarios. De esta forma se asegura una correcta retroalimentación de los usuarios. Se ofrece conocimiento del negocio y define los requisitos del software.

- Advisor User: Consejero del usuario embajador. Este rol se emplea cuando el rol de usuario embajador no es suficiente para expresar todas las opiniones o puntos de vista importantes de los usuarios sobre un punto del proyecto.

- Project Manager: El Project Manager coordina todos los aspectos de la gestión del proyecto, esto debe incluir tanto aspectos comerciales como técnicos, desde el establecimiento de las bases del proyecto a la implantación de la solución. Es vital que asuma esta responsabilidad en toda la duración del proyecto.

- Technical Coordinator: Responsable tanto de la calidad técnica como del control técnico del proyecto, por ejemplo el uso de software de gestión de configuración y el cumplimiento de estándares técnicos. Está encargado de mantener la arquitectura.

- Developer and Senior developer: Son los únicos roles definidos para desarrolladores, por esto, este rol incluye a todo el personal de desarrollo, sean diseñadores, programadores o tester. La diferencia entre desarrollador y desarrollador senior es que

los segundos tienen gran experiencia en las tareas que realizan, por lo que suelen dirigir el equipo.

B. Ciclo de vida de DSDM. DSDM define cinco fases en la construcción de un sistema – ver Figura 3.4. Las mismas son:

- Estudio de factibilidad: El estudio de factibilidad es una pequeña fase que propone DSDM para determinar si la metodología se ajusta al proyecto en cuestión.

- Estudio del negocio: Durante el estudio del negocio se involucra al cliente de forma temprana, para tratar de entender la operatoria que el sistema deberá automatizar. Este estudio sienta las bases para iniciar el desarrollo, definiendo las features de alto nivel que deberá contener el software.

- Iteración del modelo funcional: Se inician las iteraciones en las que se describirán en detalle las características definidas en la fase anterior, planeando un modelo previo que dé solución aceptable a la problemática. Esta es la etapa de diseño.

- Iteración del diseño y construcción: Se realizara el diseño de los mismos codificando el modelo diseñado, se construirán los componentes el manual de usuario y técnico.

- Implementación: Entrega del producto al cliente o usuario final. Cuando éste de su aprobación se implantará el sistema en producción.

La primera y segunda fase son secuenciales y, por lo tanto, realizadas una única vez al principio para analizar la factibilidad desde el punto de vista del negocio del desarrollo, las demás fases presentan las características de los modelos iterativos e incrementales.

DSDM denomina las iteraciones como "timeboxes". Una "timebox" dura un periodo predefinido de tiempo y la iteración debe finalizar dentro de la "timebox" que suele durar desde unos días hasta unas pocas semanas. Sin embargo, lo que diferencia a DSDM de otros modelos son los principios alrededor de los cuales se estructura, tratados en el apartado anterior, y que hacen énfasis en los equipos de desarrollo, en el feedback con el cliente y en las entregas frecuentes de productos.

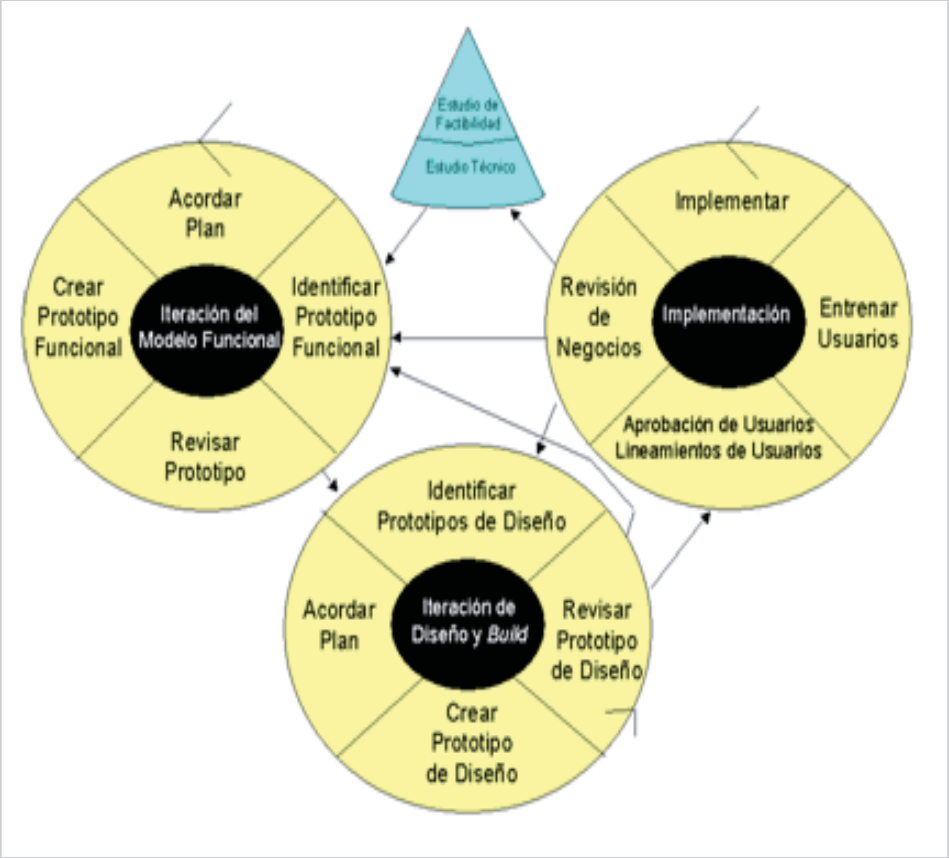


Figura 3.4 Fases de desarrollo en DSDM. Fuente: (Peinado & Vázquez, 2010)

3.2.5 ¿Por qué usar Metodologías Ágiles?

Tomando las ideas de la Tabla 3-1, es posible indicar que las Metodologías Tradicionales presentan los siguientes problemas a la hora de abordar proyectos:

- Existen unas costosas fases previas de especificación de requisitos, análisis y diseño. La corrección durante el desarrollo de errores introducidos en estas fases será costosa, es decir, se pierde flexibilidad ante los cambios.
- El proceso de desarrollo está ajustado a plazos de tiempos determinados.
- El desarrollo es más lento. Es difícil para los desarrolladores entender un sistema complejo en su globalidad.

Las Metodologías Ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes. Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación. Dividir el trabajo en módulos abordables minimiza los fallos y el coste. Las metodologías ágiles presentan diversas ventajas, entre las que podemos destacar:

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo.
- Entrega continua y en plazos breves de software funcional.
- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Importancia de la simplicidad, eliminado el trabajo innecesario.
- Atención continua a la excelencia técnica y al buen diseño.
- Mejora continua de los procesos y el equipo de desarrollo.

Tabla 3-1 Metodologías Agiles vs Metodologías Tradicionales

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios.
Proceso menos controlado, con pocos principios.	Proceso más controlado, con numerosas normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos y roles.	Más artefactos y roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

3.2.6 Conclusiones

Las Metodologías Tradicionales históricamente han intentado abordar la mayor cantidad de situaciones del contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y de requisitos cambiantes.

Las Metodologías Ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos por su simplicidad en el aprendizaje como en puesta en marcha, permitiendo a los pequeños grupos de desarrollo concentrarse en la tarea de construir software fomentando prácticas de fácil adopción y en un entorno ordenado que permiten que los proyectos finalicen exitosamente.

Las características de los proyectos para los cuales las Metodologías Ágiles han sido especialmente pensadas se ajustan a un amplio rango de proyectos de desarrollo de software; aquellos en los cuales los equipos de desarrollo son pequeños, con plazos reducidos, requisitos cambiantes, y/o basados en nuevas tecnologías.

A pesar de las continuas críticas que las Metodologías Ágiles sufren, son usadas por muchas grandes empresas y se han utilizado en grandes sistemas.

3.3 Integración de TAM en Metodologías Ágiles

Como se describió en el capítulo de TAM, el modelo sirve para evaluar la aceptación de los sistemas de información por los usuarios. Para ello se basa en dos características importantes de los sistemas; Utilidad percibida (PU) y la Facilidad de uso percibida (PEOU).

Según el modelo (Figura 3.1) se pueden desprender las siguientes conclusiones: La Facilidad de uso percibida tiene un efecto directo y significativo sobre la Utilidad percibida, ya que en igualdad de condiciones, un sistema que es fácil de usar se traduce en un sistema más útil ya que aumenta el rendimiento en el trabajo (mayor utilidad).

Si el sistema es de fácil uso, el usuario se vuelve más productivo en su trabajo por la mayor facilidad de uso, entonces el usuario se convierte en un usuario más productivo. Por lo tanto, las características del sistema pueden influir indirectamente afectando la utilidad debido a la facilidad de uso.

Un sistema que es fácil de usar, es un sistema usable, un sistema en el cual en diseño y funcionalidad cuentan con las características de un sistema usable, las siguientes características corresponden a las heurísticas de usabilidad de Nielsen (Nielsen, 1994).

1. Visibilidad del estado del sistema. El sistema debe siempre mantener a los usuarios informados del estado del sistema, con una realimentación apropiada y en un tiempo razonable.

2. Utilizar el lenguaje de los usuarios. El sistema debe hablar el lenguaje de los usuarios, con las palabras, las frases y los conceptos familiares, en lugar de que los términos estén orientados al sistema.

3. Control y libertad para el usuario. Los usuarios eligen a veces funciones del sistema por error y necesitan a menudo una salida de emergencia claramente marcada, esto es, salir del estado indeseado sin tener que pasar por un diálogo extendido. Es importante disponer de deshacer y rehacer.

4. Consistencia y estándares. Los usuarios no deben tener que preguntarse si las diversas palabras, situaciones, o acciones significan la misma cosa. En general siga las normas y convenciones de la plataforma sobre la que está implementando el sistema.

5. Prevención de errores. Es importante prevenir la aparición de errores que mejor que generar buenos mensajes de error.

6. Minimizar la carga de la memoria del usuario. El usuario no debería tener que recordar la información de una parte del diálogo a la otra. Es mejor mantener objetos, acciones, y las opciones visibles que memorizar.

7. Flexibilidad y eficiencia de uso. Las instrucciones para el uso del sistema deben ser visibles o fácilmente accesibles siempre que se necesiten. Los aceleradores no vistos por el usuario principiante, mejoran la interacción para el usuario experto de tal manera que el sistema puede servir para usuarios inexpertos y experimentados. Es importante que el sistema permita personalizar acciones frecuentes.

8. Los diálogos estéticos y diseño minimalista. No deben contener la información que sea inaplicable o se necesite raramente. Cada unidad adicional de la información en un diálogo compite con las unidades relevantes de la información y disminuye su visibilidad relativa.

9. Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores. Que los mensajes de error se deben expresar en un lenguaje claro, se debe indicar exactamente el problema, y deben ser constructivos.

10. Ayuda y documentación. Aunque es mejor si el sistema se puede usar sin documentación, puede ser necesario disponer de ayuda y documentación. Ésta tiene que ser fácil de buscar, centrada en las tareas del usuario, tener información de las etapas a realizar y que no sea muy extensa.

Por tanto, mediante la utilización de TAM, en parte, es posible medir la usabilidad de un sistema y con esto poder lograr evaluar su aceptación.

La integración de TAM a una Metodología de desarrollo iterativa puede resultar de gran utilidad, por ejemplo, en las iteraciones de construcción se podrían realizar correcciones al sistema en base a la información entregada por la aplicación del modelo, esto aseguraría que el producto final contará con altos grados de aceptación y satisfacción por parte del usuario final.

En la actualidad no existe literatura en la cual se haya planteado la integración de TAM a alguna Metodología de desarrollo de software, esto no quiere decir que no se haya o no se esté aplicando, sino más bien, no existe como modelo formal de integración y aplicación.

3.4 Caso de Estudio: Empresa ZEKE

3.4.1 Inicio y Crecimiento

ZEKE es una empresa informática Chilena, formada en el año 2005 por un grupo de jóvenes profesionales, comenzó siendo solo una idea de emprendimiento, en la actualidad se ha convertido en empresa líder en el desarrollo de servicios informáticos de la región.

Como empresa de servicios informáticos y consultoría tecnológica, tiene como principal objetivo brindar soluciones integrales en informática, principalmente el desarrollo de proyectos WEB, utilizando las últimas tecnologías disponibles en el mercado.

ZEKE cuenta en la actualidad con importantes clientes que han depositado su confianza en la empresa; algunos de ellos son: Ministerio de educación, Ministerio de salud y el Ministerio de planificación, por nombrar algunos.

El año 2009 resulta clave para ZEKE, es en este año donde la empresa comienza a registrar un crecimiento sostenido y seguro. La cantidad de proyectos adjudicados en dicho año, trajo consigo la necesidad de contar con una mayor dotación de profesionales (ver Figura 3.5 y Figura 3.6). Esta mayor dotación permitió identificar y crear nuevos roles para que participaran en el desarrollo de los proyectos de aquel entonces, esto llevó a la creación de áreas ligadas al desarrollo, que antes no existían, por ejemplo: área de análisis, programación y aseguramiento de la calidad, ver Figura 3.7.

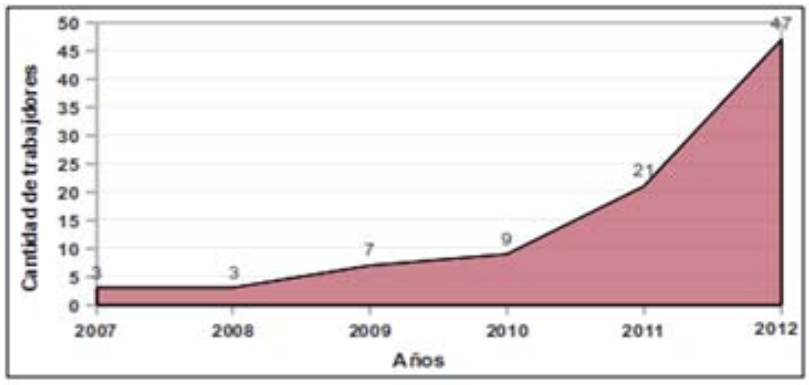


Figura 3.5 Cantidad de trabajadores por año

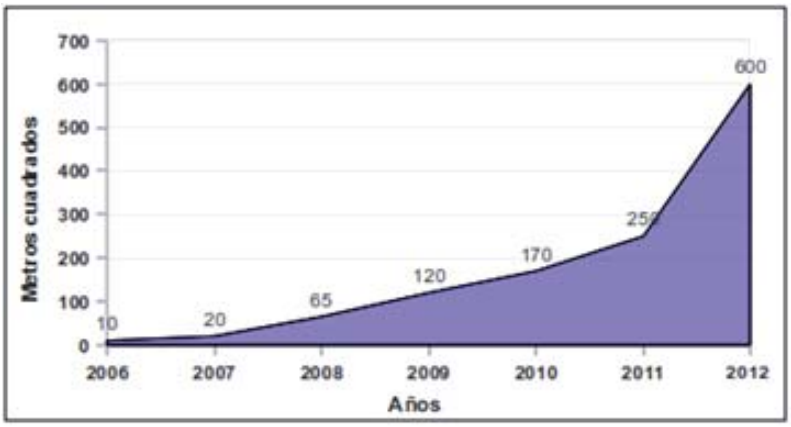


Figura 3.6 Crecimiento en infraestructura

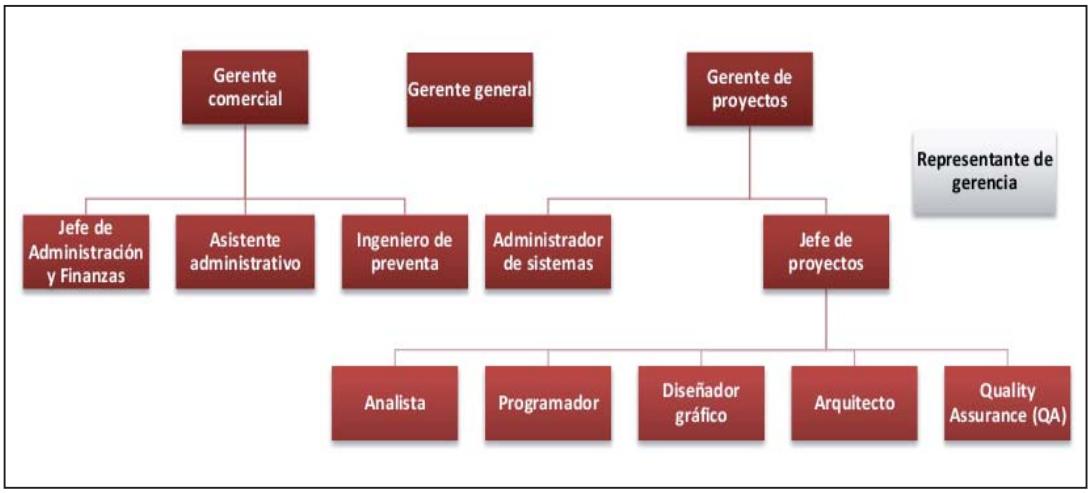


Figura 3.7 Organigrama de ZEKE

A comienzo del año 2010 ZEKE logra pertenecer al selecto grupo de empresas informáticas que se adjudican el nuevo Convenio Marco de Desarrollo de Software que ChileCompra estableció, tras este acuerdo, ZEKE pasó ser considerada como proveedor de primera opción cuando un organismo público requiera contratar servicios tecnológicos. Gracias a la adjudicación de este convenio es que ZEKE comienza el trabajo con importantes clientes del sector público para el desarrollo de proyectos cada vez más grandes y ambiciosos.

El mayor crecimiento de la empresa trajo consigo la formalización de los procesos. Para mantener el prestigio ganado en sus primeros años y para hacer frente a los nuevos desafíos que traía el Convenio Marco, se hizo imperante regular y sistematizar los procesos internos de la empresa. Es como de esta manera que ZEKE, a comienzo del año 2011, se certifica bajo la norma ISO 9001:2008 a manos del Grupo Bureau Veritas.

3.4.2 Equipos de Desarrollo

3.4.2.1 Características de las Personas y Áreas de Desarrollo

ZEKE cuenta en la actualidad con una dotación de 42 personas ligadas al área de la ingeniería, informática y telecomunicaciones. Entre profesionales y técnicos, los trabajadores desarrollan labores en áreas determinadas por la Gerencia de proyectos, esto se realiza mediante dos procesos:

A. Un proceso de reclutamiento el cual identifica las aptitudes y condiciones del profesional para desempeñarse en algún área; esta selección es realizada mediante el uso de entrevistas y pruebas técnicas. Se busca en las personas que ingresan a la empresa características específicas para desempeñar algún cargo. Dichas características han sido determinadas por la experiencia en el transcurso de estos años.

B. Otra manera, corresponde a las observaciones del desempeño de los trabajadores que llevan más tiempo en la empresa. Aquellos que muestran condiciones necesarias pueden ser ascendidos a cargos con mayor responsabilidad. Es como de esta manera que personas que realizan la función de análisis pueden ejercer en algún momento el cargo de Jefatura de algún proyecto o cómo los programadores pueden realizar la labor de análisis.

Las áreas de desarrollo corresponden a las comúnmente empleadas en la Ingeniería de software; estas son áreas de análisis, desarrollo (programación) y aseguramiento de calidad, transversal a ellas está la jefatura del proyecto (Figura 3.7). A pesar que la empresa continua creciendo y cada vez más son las personas que ingresan a trabajar, se ha mantenido una distribución similar a lo largo de los años en relación a la cantidad de trabajadores por área, ver Figura 3.8.

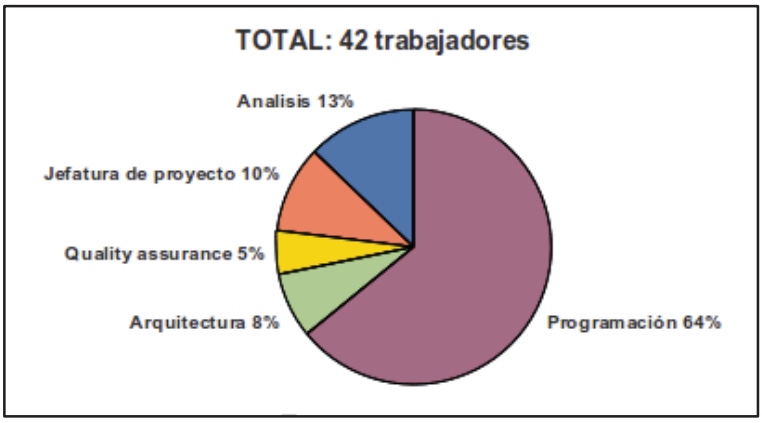


Figura 3.8 Cantidad de trabajadores por área

3.4.2.2 Descriptores de Cargo

Cada cargo exige tener competencias específicas para desarrollarlo, los descriptores de cargo son una herramienta útil para difundir este conocimiento, en ellos se muestra las habilidades duras y blandas que deben tener las personas que trabajan en la empresa. Además, identifican las responsabilidades que deben cumplir las personas que desempeñan el cargo en el proceso de desarrollo, esto resulta útil al momento de identificar quien tiene las obligaciones sobre alguna actividad.

3.4.3 Política de Calidad

La Gerencia General de ZEKE establece que la Política de Calidad que representa la empresa contiene los siguientes compromisos:

- Entregar productos y servicios de calidad, cumpliendo con las especificaciones y requerimientos de sus clientes.
- Facilitar el intercambio de conocimientos técnicos al interior de la Organización.
- Cumplir los requisitos normativos de la Norma Internacional ISO 9001:2008, y las normativas legales vigentes aplicables a su rubro.
- Mejorar continuamente el desempeño de sus procesos.

3.4.4 Sistema de Gestión de Calidad

ZEKE ha implementado y mantiene un sistema de gestión de calidad basado en la Norma ISO 9001:2008, el propósito, servir como base sólida para el ordenamiento, planificación, control y mejora continua de los procedimientos internos de le empresa otorgando dinamismo, flexibilidad y evidencia de cumplimiento del compromiso de servicio contraído con el Cliente.

Entre los diversos procedimientos que norma el Sistema de gestión de calidad, se encuentran los ligados con el desarrollo de software, estos son: Procedimiento de diseño y desarrollo y Procedimiento de Construcción, dichos procedimientos identifican roles, responsabilidades y artefactos que son solicitados para cada proyecto de desarrollo a medida que ZEKE gestiona.

3.4.5 Proceso de Desarrollo de Software en ZEKE

ZEKE, bajo su sistema de gestión de calidad dispone de 2 procedimientos ligados con el desarrollo de software, el Procedimiento de diseño y desarrollo y el Procedimiento de construcción. Estos procedimientos fueron construidos con el propósito de contar con un marco de desarrollo estándar para la empresa que proporcionara las actividades, roles y responsabilidades aplicables a los desarrollos a medida. Los diversos proyectos con los cuales

ZEKE trabaja, hacen que ambos procedimientos sean perfeccionados de forma periódica buscando siempre la obtención de calidad como objetivo final.

3.4.5.1 Procedimiento de Diseño y Desarrollo

Este procedimiento tiene como fin describir y estandarizar las actividades mediante las cuales ZEKE transforma los requerimientos de sus clientes en especificaciones de un producto y/o servicio.

1. Reunión de Planificación Interna. El Gerente de Proyectos asigna los RRHH para el proyecto adjudicado, si producto de esta asignación determina contratar personal o servicios adicionales, debe solicitarlo al Gerente Comercial para que aplique el procedimiento correspondiente. El Gerente Comercial presenta la planificación inicial, Carta Gantt. Se firmará una minuta de esta reunión, con los acuerdos tomados respecto del equipo que participará en el proyecto y de la planificación inicial.

2. Inicio del Proyecto. El Gerente de Proyectos se reúne con el Jefe de Proyecto designado y revisan todos los aspectos del proyecto, identificando plazos, recursos y riesgos iniciales, entre otros. El Jefe de Proyecto configura las hojas de control e invita a cada uno de los integrantes del equipo de trabajo. Esta hoja de control del proyecto contiene una descripción general del proyecto (integrantes, documentos a generar, control de cambios, control de construcción). Al aceptar la invitación, los integrantes se dan por enterado de su participación en el proyecto.

3. Levantamiento. En caso de corresponder, el Jefe de Proyecto coordina las reuniones de levantamiento con el cliente. Para cada reunión existe una minuta, validada por el cliente en un plazo no mayor a 48 horas. El Analista, dependiendo del tipo de proyecto, y con toda la información recogida de esta etapa, documentará el levantamiento de requerimientos según lo definido en la hoja de control del proyecto, el que debe ser validado por el cliente según el plazo definido en el proyecto.

   En caso de no presentar observaciones en el plazo acordado en la reunión de planificación con el cliente, los productos de trabajo elaborados se dan por validados y se continúa con la siguiente etapa. Esta acción es comunicada con el cliente.

4. Análisis. El Jefe de proyecto en conjunto con el Analista decide la manera más adecuada de documentar un análisis, por ejemplo:
   - Diagramas de procesos
   - Especificación de casos de uso
   - Mockups
   - Prototipos no funcionales
   - Otros

   El Jefe de proyecto actualiza la hoja de control indicando los documentos de análisis y diseño del proyecto y la disponibilidad al equipo de trabajo, el Analista confecciona

la especificación acordada, la cual es revisada por el Jefe de proyecto y validada por el cliente.
En caso de no presentar observaciones en el plazo acordado en la reunión de planificación con el cliente, los productos de trabajo elaborados se dan por validados y se continúa con la siguiente etapa.

5. Diseño. El analista es el responsable del diseño del producto, en uno o más documentos definidos en la hoja de control. En la elaboración de este documento podrán participar además el Arquitecto y el Programador. Este documento es revisado por el Jefe de Proyecto.

   El Diseñador gráfico, dependiendo del tipo de proyecto, realiza una propuesta gráfica, la que debe ser aprobada por el cliente. Cabe destacar que se solicita dicha aprobación solamente en el caso de proyectos desarrollados en su totalidad por ZEKE, quedando fuera aquellos que se realicen por consorcio.

   En caso de corresponder, el Arquitecto debe proponer los cambios en la arquitectura del proyecto, incluidas las modificaciones que pudieran surgir durante la etapa de análisis. Éstas deberán ser aprobadas por el cliente.

   En caso de no presentar observaciones en el plazo acordado en la reunión de planificación con el cliente, los productos de trabajo elaborados se dan por validados y se continúa con la siguiente etapa.

6. Esfuerzos de Construcción. En paralelo a la etapa de análisis, el Jefe de Proyectos, utilizando la información proveniente de la planificación definitiva y dependiendo del tipo de proyecto realiza una estimación con los "esfuerzos de construcción", mediante el cual se determinarán las horas de duración en fase de construcción, en base a la que posteriormente se evalúa el cumplimiento de plazos o tiempos asociados a estas actividades.

   El instrumento a utilizar para medir el esfuerzo asociado a la construcción debe ser definido por el Jefe de Proyecto.

3.4.5.2 Procedimiento de Construcción

Este procedimiento es aplicable una vez que se ha generado toda la documentación y finalizado la etapa de Diseño y Desarrollo. En este procedimiento se detallarán todas las etapas que conforman el proceso de construcción de Sistemas, desde el inicio con las tareas de programación, hasta las actividades de liberación o aprobación del producto por parte de los clientes.

1. Inicio del Proceso. El Jefe de Proyecto, dependiendo de la complejidad de las actividades a realizar, definirá e informará a los programadores respecto de las prioridades y las secuencias de trabajo. Además, entregará la planificación del proyecto (al menos la parte de construcción) al equipo de trabajo para que todos tengan

idea respecto de los tiempos involucrados en el desarrollo. En esta instancia, cabe destacar que dichas prioridades y secuencia de trabajo pueden sufrir modificaciones en el transcurso de toda la etapa construcción, dependiendo de la situación actual en el desarrollo. Dichas modificaciones serán informadas debidamente al equipo de trabajo.

Cada programador tomará un caso de uso (si es que el tipo de proyecto requería su elaboración) o especificación del requerimiento y trabajará de acuerdo al detalle de plazos indicado en la planilla de control más adecuada para el proyecto, de acuerdo a los elementos o módulos de trabajo sobre los cuales se requiera hacer un seguimiento. La decisión sobre cuál utilizar será responsabilidad del Jefe de proyecto. Una vez finalizada esta actividad, se registrará su término en la planilla de control correspondiente.

2. Certificación Interna. La primera actividad a realizar en esta etapa es responsabilidad del Jefe de proyecto, a quien le corresponde decidir el tipo de herramienta a utilizar para registrar las incidencias que sean detectadas por el QA. Dicha herramienta podría ser diferente de un proyecto a otro, dependiendo de las necesidades de cada cual.

   El QA confecciona el plan de pruebas para el sistema basándose para ello en las especificaciones de los casos de uso. Luego, a medida que las especificaciones a construir (casos de uso, requerimiento u otro elemento de trabajo) van siendo terminados, el QA comienza a ejecutar las pruebas correspondientes al funcionamiento de cada uno, o un grupo de ellos, dependiendo del tipo de proyecto.

   En caso que se detecten errores, estos deben ser registrados en la herramienta indicada por el Jefe de proyecto, a objeto que el programador pueda reparar los errores detectados, respetando el tiempo asignado para correcciones en la Carta Gantt del proyecto. Posteriormente, el QA vuelve a revisar para descartar nuevos errores. Sin embargo en el caso que los errores se repitan, o producto de la revisión se evidencien nuevos errores, se deberá repetir el procedimiento de registro en el Sistema de gestión de incidencias, y será el Programador el responsable de corregir.

3. Certificación Externa. Una vez que el producto se encuentra preparado para su entrega, el cliente debe aprobar mediante la realización de pruebas, la conformidad del producto.

   La primera actividad a realizar es responsabilidad del Jefe de proyecto, a quien le corresponde decidir el tipo de herramienta a utilizar para registrar las incidencias que sean detectadas por el QA.

   En caso que en esta etapa el cliente detecte errores, deberá informarlos a ZEKE mediante la herramienta indicada por el Jefe de proyecto. Se realizan las correcciones necesarias y se presenta el producto nuevamente al cliente para su liberación y aprobación final.

4. Cambios de Requerimientos. Esta sección del procedimiento documentado se activa cuando el cliente plantea cambios de requerimientos como producto de la certificación externa o por otra situación antes de la entrega final del producto y luego de ya avanzada la construcción.

   Se recibe el requerimiento de parte del cliente, incluyéndolo en la planilla de control en la hoja de Cambios de requerimientos. Esta actividad es realizada por el analista. Luego, se pueden tomar 2 cursos de acción:

   - Si los requerimientos están lo suficientemente claros, se pasan directamente al programador para que los desarrolle.
   - Si no están del todo claros, entonces el analista elabora una pequeña especificación del cambio, la cual es aprobada por el Jefe de proyecto, quien valida si se construye o no, dependiendo del impacto que pueda tener.

3.5 Problemas con el Desarrollo de Software

Con la creación e implementación de los procedimientos para el desarrollo se pensó que se obtendrían resultados favorables en la gestión de los procesos, la calidad de este y la calidad del producto, sin embargo esto no ha sido del todo real.

Ha habido proyectos en los cuales han existido atrasos considerables, si bien es cierto, algunos han sido por una planificación demasiado optimista, existe otro conjunto los cuales contando con una correcta estimación y una planificación realizada en base a proyectos similares se ha sufrido de igual manera atrasos de gran envergadura, lo que ha llevado a realizar esfuerzos extraordinarios por parte de los equipos para cumplir los hitos y los entregables. Se ha conocido internamente en la empresa a este fenómeno como los "Proyectos sin fin". Estos tipos de proyectos cuya fecha de término es negociada cada vez por cada iteración de corrección utilizan valiosos recursos que son requeridos por otros proyectos, el principal, el recurso humano. Al no contar con una fecha de término definitiva los programadores y analistas comienzan el trabajo en nuevos proyectos, el tiempo ocioso en el equipo en cuyo proyectos se encuentra en el proceso de Certificación externa es grande, más aún si este proceso se extiende de forma prolongada, eso lleva a que tanto programadores como analistas deben redoblar esfuerzos.

Otro problema importante tiene relación con los productos construidos y liberados, en este último se ha apreciado la carencia de calidad de ellos, esto se refleja principalmente en el cumplimiento de requerimientos y algunos atributos propios de los sistemas informáticos que denotan calidad.

Una vez construido un producto y siguiendo el procedimiento de construcción en la fase de validación por parte del cliente, el tiempo estimado y planificado al inicio del proyecto para esa tarea, no se está cumpliendo. El cliente reporta su insatisfacción con la implementación de sus requerimientos, el producto es analizado, corregido, probado internamente y liberado para la aprobación del cliente; esta iteración de corrección algunas veces resulta ser tediosa, los analistas hacen saber que los cambios, que algunas veces se solicitan, no fueron levantados ni

menos analizados en el proceso de análisis; por otro lado, los programadores muestran su malestar al no tener un sistema escalable y menos de fácil mantención.

3.6 Identificación de Deficiencias en el Proceso Actual de Desarrollo.

El siguiente capítulo tiene por objetivo realizar un estudio sobre los procedimientos de desarrollo: Procedimiento de diseño y desarrollo y el Procedimiento de construcción, con la finalidad de detectar deficiencias que actúen en contra de la calidad de los procesos y los productos. Estudiando las deficiencias se puede construir un nuevo modelo de proceso desarrollo mejorado que asegure calidad en ambos ámbitos.

3.6.1 Metodología de Trabajo

Para la identificación de factores que hacen deficiente el actual proceso de desarrollo con sus dos procedimientos, primero se realizó un estudio a nivel de los implicados en el desarrollo, esto son; analistas, programadores, jefes de proyecto y QA's. En base a entrevistas grupales y personales y la aplicación de cuestionarios se conoció la realidad de sus labores y cómo ellos reconocieron factores que hacen deficiente sus trabajos.

Se obtuvieron factores extraídos en base al estudio de los proyectos reconocidos y catalogados como no exitosos. Para ello, se recurrió al uso de información histórica de la empresa para determinar qué factores hicieron que un determinado proyecto tuviera esas características.

Para concluir la identificación, se cruzarán todos los factores obtenidos, los cuales fueron definidos, catalogados y priorizados para ser utilizados al momento de construir el modelo de proceso de desarrollo de software que integre esta metodología.

3.6.2 Búsqueda de Deficiencias en el Equipo de Desarrollo

3.6.2.1 Identificación y Clasificación de Factores Desencadenantes de Deficiencias

Para la identificación de factores a este ámbito, se creó un formulario tipo encuesta el cual fue repartido a todos los integrantes de todas las áreas que componen el proceso de desarrollo de software en la empresa (ver Tabla 3-2, Tabla 3-3 y Tabla 3-4). El objetivo fue que cada trabajador reconociera el factor e indicara el nivel de impacto que tiene éste en la calidad de su trabajo. Los factores fueron obtenidos entrevistando a algunos trabajadores de cada área para que ellos identificaran los propios de su ámbito.

Se construyó un documento para cada trabajador de cada área, en cada documento, estuvieron plasmados los factores obtenidos de los puntos anteriores. Los documentos

solicitaban a los trabajadores que indicaran el nivel de impacto o severidad que tienen cada factor en su trabajo y que resta calidad al mismo. Además de esto, se requería que se especificara la frecuencia con que ese factor era percibido en los proyectos que participan.

Para medir el impacto o severidad se utilizó un escala de 1 a 5, en donde 5 es el mayor nivel y 1 el menor, esto es, que si un factor ha tenido nivel 5, significa que ha influido directa o considerablemente en contra de la calidad de su trabajo.

Para medir la frecuencia se utilizó una escala nominal de Alto, Medio y Bajo. Que fueron traducidos a valores numéricos de 3, 2 y 1 respectivamente.

El motivo para solicitar estos dos indicadores es la obtención de un indicador maestro que permitiera priorizar el estudio y corrección de los factores más críticos, este indicador es el grado de Criticidad, que corresponde al nivel de severidad multiplicado por la frecuencia. Para determinar qué factores eran de importancia de estudio, se utilizó el valor 5 de criticidad como la frontera de interés, de esta forma se centra la atención solo en los factores cuya criticidad estuviese por encima de esta línea y que son los de mayor relevancia.

Tabla 3-2 Factores desencadenantes de deficiencia Proceso de Análisis

Factor	Descripción
Personal mediocre	En el equipo de analistas. Baja capacidad para la resolución de problemas, baja productividad, poca experiencia, etc.
Personal problemático	En el equipo del proyecto. Personas que no aceptan críticas, ayuda, observaciones, generan conflicto, etc.
Falta de participación de los implicados	La no participación activa de todos los participantes del proyecto.
No existencia de estándar de trabajo	El uso de herramientas, diseño de interfaces, modelos, sin contar con una norma.
Control insuficiente	Poco control del Jefe de proyecto para detectar a tiempo los signos de posibles retrasos.
Falta dominio técnico sobre la tecnología	No conocer las posibilidades y límites de la tecnología con la cual se trabaja al momento de diseñar y analizar las soluciones.
Bajo control de calidad	Falta de revisión y validación del jefe de proyecto de los diversos documentos generados.
Falta de participación del cliente	Poca disposición a resolución de temas, colaboración, tiempos de respuesta, ayuda en general.
Cambios de requerimientos	
Clientes poco claros	Clientes poco claros con la solución que requieren, cambiantes en sus requerimientos.
Planificación excesivamente optimista	Tiempo estimado para el levantamiento de requerimientos, trabajo en los casos de uso, modelos de datos, análisis en general, demasiado optimista.

Tabla 3-3 Factores desencadenantes de deficiencia Proceso de Construcción

Factor	Descripción
Personal mediocre	En el equipo de analistas. Baja capacidad para la resolución de problemas, baja productividad, poca experiencia, etc.
Personal problemático	En el equipo del proyecto. Personas que no aceptan críticas, ayuda, observaciones, generan conflicto, etc.
Falta de participación de los implicados	La no participación activa de todos los participantes del proyecto.
Falta de un programador líder	Programador líder que sea capaz de dirigir el equipo, tomar decisiones técnicas, guiar la programación.
Falta de conocimiento de la arquitectura	Conocimientos de los lenguajes, Frameworks, ciclos de vida, etc.
Control insuficiente	Poco control del Jefe de proyecto para detectar a tiempo los signos de posibles retrasos.
Programación a destajo	Programadores son libres para codificar, sin límites, ni restricciones. Falta de una norma de programación.
Control de riesgos insuficiente	Escasa gestión de riesgos: Plan de contingencia y mitigación deficiente.
Análisis deficiente	Diseño deficiente, modelos incompletos, casos de uso inconsistentes, etc.
Bajo control de calidad	Bajo control de calidad del código fuente e implementación de los casos de uso.
Planificación excesivamente optimista	Tiempo estimado para el desarrollo de los requerimientos demasiado optimista.

Tabla 3-4 Factores desencadenantes de deficiencia Proceso de QA

Factor	Descripción
Personal problemático	En el equipo del proyecto. Personas que no aceptan críticas, ayuda, observaciones, generan conflicto, etc.
Falta de conocimientos técnicos	Poco conocimiento sobre la arquitectura del sistema evaluado por lo que las pruebas no son del todo completas.
Planificación optimista	Tiempo estimado para el desarrollo de los requerimientos demasiado optimista.
Herramientas deficientes	Herramientas insuficientes para realizar pruebas.
Falta de conocimientos del negocio	Poco conocimiento sobre el negocio lo que implica que el desarrollo del plan de pruebas y la ejecución son incompletas.
Sistema no apto para pruebas	Sistemas no apto para el comienzo de las pruebas.

3.6.2.2 Análisis de Resultados - Proceso de Análisis

En la Figura 3.9, se pueden apreciar 3 factores estrechamente relacionados que se encuentran en el sector de interés de estudio, estos son: Clientes poco claros, Cambio de requerimientos, Falta de participación del cliente.

La relación de estos 3 factores está en que cada uno es el cliente el que provoca la aparición del factor. Según la reunión con los analistas en donde se estudiaron estos resultados, se

concluyó que contar con Clientes Poco Claros más la Falta de Participación Activa, resultan ser factores claves para realizar un correcto análisis, esto resulta obvio. Al no contar con una participación activa del cliente el análisis o que este no esté claro con la solución que requiere, el trabajo de análisis comienza a desarrollarse muchas veces en base supuestos, los casos de uso resultantes en definitiva son deficientes, inconsistentes e incompletos.

Un trabajo de captura de requerimientos, para que sea de calidad, en primer lugar se requiere que el cliente esté claro de la solución que necesita, no se puede construir un producto software que satisfaga a un cliente si éste no fue claro con lo que requería de el al momento de realizar las reunión con los equipos de análisis; en segundo lugar, es estrictamente necesario la participación activa en el proyecto sobre todo a la etapa de levantamiento y análisis. La participación activa del cliente en la etapa de análisis, es que éste disponga del tiempo necesario para la correcta obtención de sus requerimientos y la atención de eventuales dudas que pudieran surgir en el proceso. Según inspección de Standish Group se descubrió que la razón número uno de que los proyectos de Sistemas de Información tuviesen éxito es la implicación del usuario (The Standish Group International, 1994;1995). Los proyectos que no implican al usuario desde el principio corren el riesgo de que no se comprendan los requerimientos del proyecto, y son vulnerables a que se consuma tiempo en prestaciones que más tarde retrasarán el proyecto.

Dejando de lado los factores ligados al cliente, existe otro factor también por encima de la frontera de interés de estudio. Y que tiene relación con los mencionados más arriba, es el Bajo Control de Calidad. Los documentos generados en la fase de análisis son principalmente casos de usos, modelos de datos y diseño de interfaces; existen otros como diagramas de secuencia y modelo de clases pero corresponden a casos muy puntuales. En la reunión sostenida con los analistas, indicaron que este factor, el no contar con una instancia de revisión y validación de sus artefactos tiene directa implicancia con la calidad del producto software que se entrega al cliente y las modificaciones que resultan una vez liberado las primeras versiones. El procedimiento de análisis detalla como responsables de revisión y validación de todos los productos generados en análisis a los Jefes de Proyecto y al cliente, por ende, si los analistas no reconocen que existen validaciones de sus trabajos es porque no se está realizando o se está haciendo de forma deficiente. Los motivos de por qué está ocurriendo esto se tratará en las conclusiones de este capítulo.

Las especificaciones de los distintos casos de uso que son creados por los analistas, incluyen en algunas ocasiones interacciones entre los actores y el sistema que son reflejadas en wireframes, alguna de estas interacciones tienen un detallado nivel de comportamiento. Los analistas cuando construyen estas interacciones no poseen el conocimiento técnico sobre los alcances de la tecnología con la cual el sistema de desarrollará, esto induce a que existan casos de uso los cuales el tiempo en su desarrollo (programación) tome más de lo estimado o que simplemente se desechen y reconstruyan, todo esto porque a los programadores se les dificulta o simplemente no pueden dar abasto con la especificación por culpa de la tecnología en la cual desarrollan. Si el factor de Falta Dominio Técnico sobre la tecnología está latente traerá consigo un impacto en la planificación del proyecto, por el tiempo adicional a lo estimado en el caso en que un caso de uso tuviese que ser reescrito para ser soportado en la tecnología de la arquitectura o por el tiempo que tome su implementación en la programación.

El factor de Control insuficiente hace referencia a las habilidades del Jefe de proyecto para detectar los retrasos en el proyecto que pudiesen comenzar en el trabajo de análisis, si bien este

factor tiene mayor relación con las habilidades de gestión de los Jefes de proyecto si está estrechamente vinculado con el factor de Bajo Control de Calidad.

Por último, los analistas hicieron notar la falta de un procedimiento formal de análisis y documentación en el factor de No Existencia de Estándar. Por lo general los proyectos de gran envergadura cuentan con más de un analista, al no existir una norma de cómo efectuar un correcto análisis y cómo este es reflejado en los distintos modelos, provoca la existencia de casos de uso inconsistentes, si se suma a ello la existencia del factor de Bajo Control de Calidad, producirá que los programadores encuentren, en el mejor de los casos, el error y que tenga que ser corregido o que los programadores los implementen trayendo consigo un sistema inconsistente.

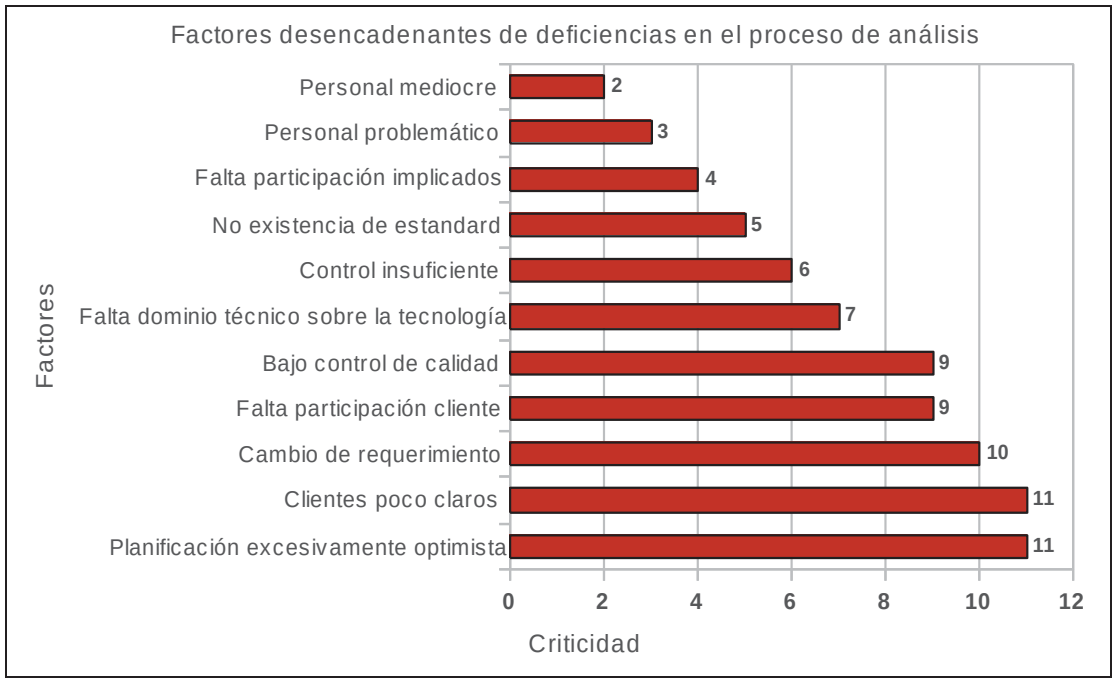


Figura 3.9 Factores desencadenantes de deficiencias en el proceso de análisis

3.6.2.3 Análisis de Resultados - Proceso de Construcción

Como se aprecia en la Figura 3.10, los programadores coinciden en que uno de los principales factores que hacen su trabajo deficiente son los cambios de requerimientos. El factor de Cambios de Requerimientos identificado como principal factor que provoca deficiencia en la calidad del trabajo en programación puede resultar extraño a primera vista, ya que los cambios de requerimientos son inherentes a un proyecto de desarrollo de software, se estima que los proyectos de software tienen 25% de cambios en los requerimientos, y ZEKE cuenta con un procedimiento formal para trabajar cuando se presenta algún cambio de requerimiento, entonces ¿Por qué es reconocido como un factor que resta calidad al trabajo de programación?.

En la reunión con el grupo de programadores se les mostró el trabajo efectuado y los resultados obtenidos, en aquella ocasión se discutió el factor de Cambio de requerimientos, los resultados fueron sorprendentes:

- El procedimiento de ZEKE para enfrentar un cambio de requerimientos fue validado por los programadores, el procedimiento funciona, no necesita de corrección.

- El problema radica en que al momento de implementar los cambios, algunas ocasiones los programadores se enfrentan a la modificación de trabajo no realizado por ellos, trabajo que en la mayoría de las veces es deficiente en calidad en código. Un programador que se enfrenta a esta situación debe lidiar aparte del trabajo de implementar el cambio de requerimiento con la corrección de errores no previstos surgidos de la propia modificación. Cabe recordar que los equipos de proyecto están integrados por alrededor de 4 programadores.

Se desprende que el impacto de la realización de un cambio de requerimientos va por el lado del esfuerzo que toma la modificación de código de baja calidad y la incertidumbre que genera el cambio en el sistema a nivel global. Como los códigos son de baja calidad, esto es, código poco robusto, flexible y escalable, los programadores deben redoblar esfuerzos para dar soporte tanto para el cambio del requerimiento como a las correcciones que deben realizar a un trabajo ya efectuado con anterioridad.

La baja calidad del código está estrechamente relacionado con otros factores de mayor criticidad, el Bajo Control de Calidad, Programación a Destajo, Falta de Conocimiento de la Arquitectura y Falta de un Programador Líder. Todos ubicados por encima de la frontera de interés.

El actual procedimiento de construcción no contempla la revisión de código o la revisión de implementación de los casos de usos (factor de Bajo Control Calidad), se asume que los programadores realizan un buen trabajo solo por el hecho que las funcionalidades programadas responden a las entradas con las salidas esperadas, no existe una instancia de auditoria de código o revisión de la implementación de los casos de uso a nivel de codificación, si se le suma a ello que los programadores no tienen normas, guías o restricciones al momento de codificar (factor de Programación a Destajo) implica que indudablemente se obtendrá un producto con calidad de código deficiente. Los programadores reconocen esto, y están al tanto de los efectos colaterales que trae, como es un cambio de requerimiento.

Por debajo del factor de la Programación a Destajo, existe otro factor, también por encima de la frontera de interés que tiene cierta relación con este último y la baja de calidad del código, es la Falta de Conocimiento de la Arquitectura. El factor de Falta de Conocimiento de la Arquitectura tiene relación con el conocimiento del programador sobre todo lo relacionado con la arquitectura del sistema que se desarrolla, esto es, lenguajes de programación. Frameworks, ciclos de vida, etc. Los programadores indicaron en la reunión, que algunos proyectos es necesario contar con un mayor conocimiento sobre la tecnología en la cual desarrollarán, antes de comenzar su trabajo. Algunos proyectos ya sea por exigencias del cliente o por la propuesta de solución de la empresa, deben ser desarrollados con tecnología que algunos casos no son dominio de todo el equipo, el mayor problema de esta poca experiencia, es que la programación no resulta de calidad, el desconocimiento lleva al programador a realizar tareas por flujos

alternativos conocidos por estos y no por los que exigen las tecnologías con las cuales trabajan. Si a este factor de desconocimiento se le suma el factor de Bajo Control de Calidad, se estará muy cerca de la obtención de un producto deficiente.

El último factor relacionado con la baja calidad del código es la Falta de un Programador Líder. Los equipos de programación pueden ser estructurados de diversas maneras. ZEKE en su procedimiento construcción no tiene una norma para estructurar los equipos de desarrollo, en la actualidad estos son conformados solo en base la disponibilidad de tiempo de cada programador (que se cierra un proyecto) o la integración de nuevos programadores mediante el reclutamiento. Los miembros que conforman los equipos de programadores bajo estas condiciones, en algunos casos, quedan muy dispares en conocimientos técnicos, obviamente no es posible que un programador nuevo alcance a adquirir todo el conocimiento o experiencia en unas cuantas semanas, sin embargo, si se contara con un programador líder que guie la programación para estos últimos casos, tome la última decisión en conflictos de cómo enfrentar problemas y haga revisión periódica de los códigos fuentes, ayudaría bastante con la calidad.

Otro factor que resulta interesante tiene relación con el proceso de análisis, es el factor de Análisis Deficiente. Lo programadores indicaron, que gran parte de los documentos generados por los analistas, esto son; casos de uso, modelo de datos, diseño de interfaces están inconsistentes e incompletos. Al contar con una documentación con esas características, el trabajo de programación resulta defectuoso y por ende resta calidad al producto al no cumplir con los requerimientos. Como se mostró en el punto anterior, los analistas reconocen que sus trabajos no son validados ni certificados por nadie, reconocen a esto como una necesidad imperante ya que están conscientes de lo estragos que trae esto una vez que sus documentos pasan a ser trabajados por los programadores.

Una programación realizada utilizando documentos de análisis inconsistentes e incompletos solo serán detectados en las fases finales del proyecto, cuando el producto es puesto a prueba para obtener la validación del cliente, esto es, en el proceso de Certificación externa el costo de implementar un cambio en el producto a este nivel de construcción trae como efecto el rehacer trabajo de análisis, programación y pruebas, esta repetición de trabajo adicional cuesta de 10 a 100 veces más de lo que habría costado si se hubiera hecho apropiadamente desde el principio (Papaccio & Boehm, 1988).

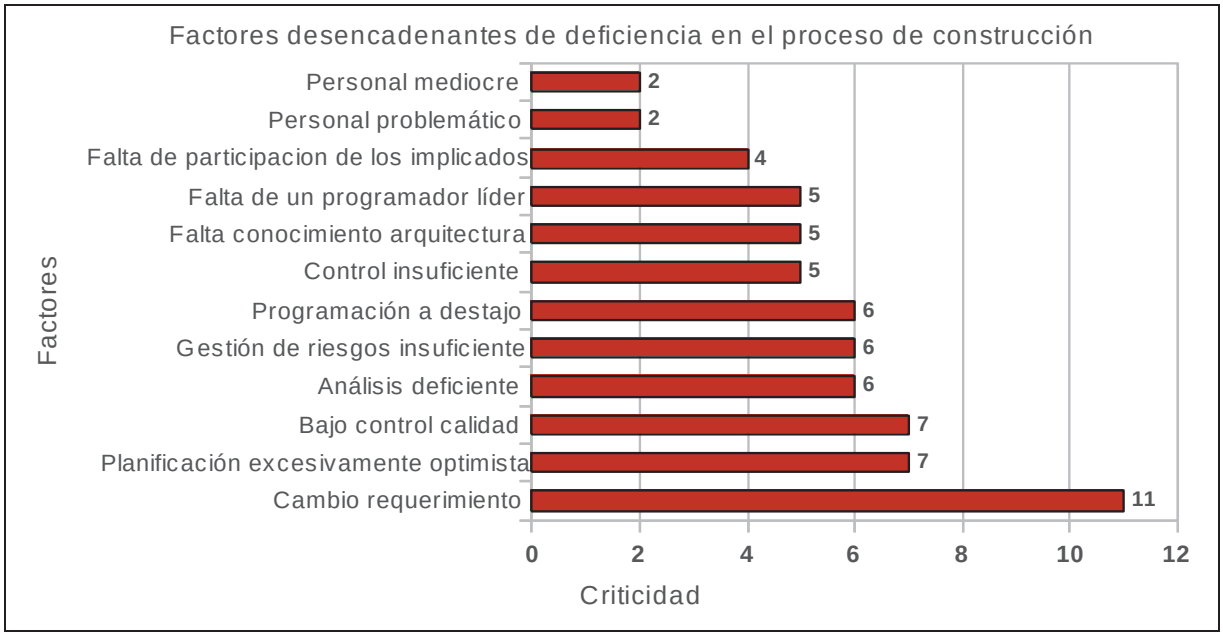


Figura 3.10 Factores desencadenantes de deficiencia en el proceso de construcción

3.6.2.4 Análisis de Resultados - Proceso de Aseguramiento de Calidad

Como lo indica el procedimiento de Certificación interna, una vez construido el sistema o alguna parte de él, este es validado por algún QA asignado al proyecto. El QA diseña pruebas tomando las especificaciones de los casos de uso, si el módulo pasa las pruebas, es aprobado por el QA y cumple con el hito de la certificación interna.

En la reunión efectuada con las personas de QA, se logró identificar que el trabajo que ellos realizan tiene 2 deficiencias importantes que hacen que su trabajo no sea de calidad y que se reflejan en la Figura 3.11.

En primer lugar, el QA requiere de las especificaciones de los casos de uso para construir los planes de pruebas que se le aplicarán al sistema una vez que este termine, al parecer estos artefactos no son del todo suficiente para crear los planes., el factor de Falta de Conocimiento del Negocio así lo refleja. En la reunión efectuada con las personas que se desarrollan como QA indicaron que los planes creados por ellos en ocasiones no contienen todas las validaciones de negocio que se requieran por no contar con el conocimiento completo y acabado del mismo, esto conlleva a que algunos sistemas se les hagan pruebas de certificación que son incompletas, esto resulta gravísimo para el correcto funcionamiento del sistema ya que son potenciales fuentes de errores.

El último factor corresponde a Sistema No Apto Para Pruebas. Los QA's indicaron que algunos sistemas no pueden ser certificados por que son recibidos con errores de programación o con implementaciones incompletas que imposibilitan ejercer algún tipo de pruebas. En este caso, la responsabilidad de que esto no ocurra recae en los programadores, falta en ellos ejercer

un mayor control de calidad de su trabajo antes de liberar el producto para ser sometidos a pruebas.

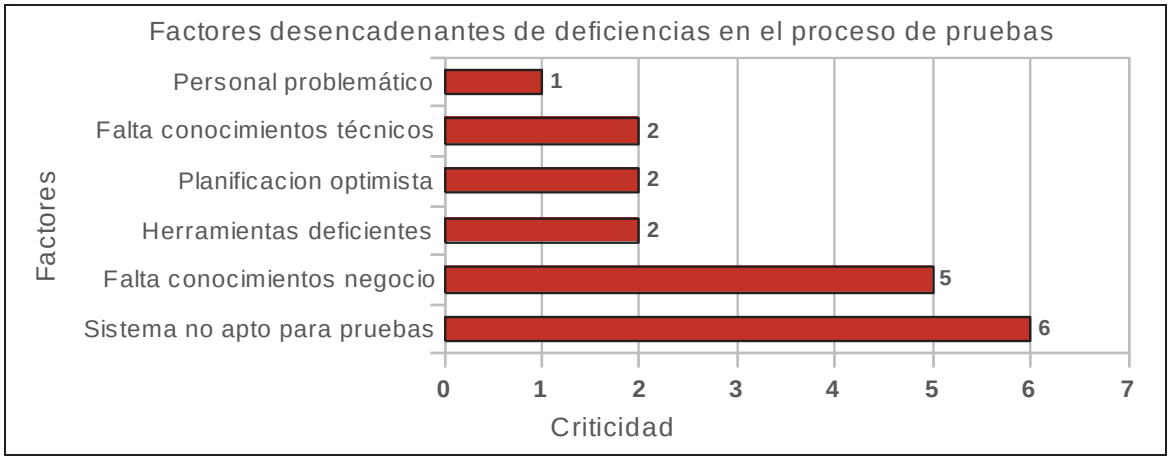


Figura 3.11 Factores desencadenantes de deficiencias en el proceso de QA

3.6.2.5 Conclusiones y Propuestas de Mejoras para Factores de Deficiencias

Los factores desencadenes de deficiencia de calidad estudiados en el proceso de Análisis son posible agruparlos en tres categorías; los ligados con la interacción o participación del cliente, los de trabajo propio de los analistas y los de gestión de la calidad.

Para la primera categoría se encuentran con los factores de Clientes Poco Claros y Falta de Participación del Cliente, como se mencionó en el capítulo donde se analizaron esos factores, la no integración o participación del cliente/usuario desde el inicio del proyecto son propensos a sufrir retrasos y cambios de requerimientos. Obtener un requerimiento como es debido en el primer paso normalmente cuesta de 50 a 200 veces menos que si se espera a las fases de construcción o mantenimiento (Papaccio & Boehm, 1988). Un proyecto normal tiene un 25% de cambios en requerimientos. Sin embargo, para evitar que estos factores se hagan presentes en el inicio o desarrollo de un proyecto puede resultar sumamente difícil.

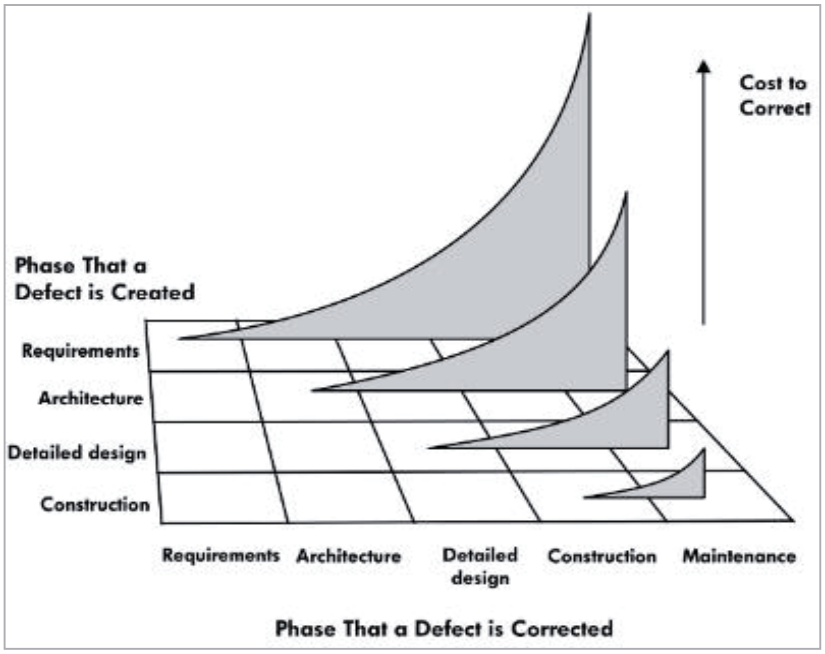


Figura 3.12 Costo de realizar correcciones. Fuente: (McConnell, 2004)

Hacer partícipe a los clientes a los largo de todo el proyecto, incluido las fases en donde su participación no es tan requerida, es sumamente complejo, según la estadísticas que se manejan dentro de ZEKE, en 6 de cada 10 proyectos, el cliente es un ente activo dentro de éste. La motivación de un cliente/usuario puede verse afectada por un sin número de otros factores externos al proyecto, como por ejemplo: presiones políticas, percepción de utilidad, afinidad con la empresa, etc. Será responsabilidad de las caras visibles de los proyectos; Jefes de proyecto y Analistas, detectar posibles signos que indiquen lejanía del cliente, analizarlos y revertirlos.

Para obtener éxito en la gestión de requerimientos a pesar de contar con Clientes poco claros, será útil la aplicación de métodos diferentes de forma que sea posible escoger los más apropiados, pueden ser:

- Métodos de análisis: Análisis estructurado, análisis estructurado de datos y análisis orientado a objetos.

- Métodos de modelado: Diagramas de clases, diagramas de flujo de datos, diagramas entidad-relación, notación del diccionario de datos y diagramas de estado-transición.

- Métodos de comunicación: Desarrollo Conjunto de Aplicaciones (JAD), prototipado de la interfaz de usuario y métodos generales de entrevistas.

En la segunda categoría están inmersos los factores de Falta Dominio Técnico Sobre la Tecnología y No Existencia de Estándar. La falta de dominio técnico sobre la tecnología repercute en que el análisis y en el resultado de estos, los casos de uso, suelen no estar acorde con la tecnología con la cual se desarrolla el proyecto, provocando un consumo de tiempo mayor de lo estimado en la implementación. Aunque el descriptor de cargo para Análisis habla de un Analista Técnico, en el caso de que no lo fuere, es necesario que el proceso de análisis sea apoyado por alguien quien resuelva dudas con respecto a la tecnología, pudiendo actuar también como validador de los casos de uso antes de que estos lleguen a los programadores, esta función la podría desarrollar el Arquitecto del sistema. Con respecto a la no existencia de estándar para análisis, esto tiene relación con proyectos en los cuales se cuentan con más de un analista y en donde se hace necesario contar con una guía que normalice la documentación, para ello sería útil crear una instancia de reunión entre el Jefe de proyecto y Analistas y cuyo resultado fuese un documento que norme como enfrentar el análisis del proyecto y principalmente como documentar los casos de uso, añadiendo límites y restricciones. La finalidad es conseguir que el sistema sea desarrollado de forma consistente al momento de las implementaciones de validaciones, mensajes e interacciones del usuario con el sistema.

En la última categoría, se encuentran el Control Insuficiente y Bajo Control de Calidad. El Control Insuficiente tiene relación con la habilidades del Jefe de Proyecto para detectar posibles riesgos o problemas y el Bajo Control de Calidad a la casi nula revisión de los casos de uso, modelos o cualquier tipo de artefacto que en esta fase se genere, ambos factores se relacionan con la no existencia de instancias de revisión y validación del trabajo realizado en Análisis. Por tanto, es necesario crearlas, pueden ser reuniones de revisión periódicas en donde participen tantos los Analistas como el Jefe de Proyecto contando también con la participación del Arquitecto, si se requiere.

El Cambio de Requerimiento fue identificado como el principal factor que genera deficiencia en el trabajo en fase de Construcción. Se determinó que esto se debía a la baja calidad del código fuente que se está generando y los estragos que produce la modificación o implementación de nuevos requerimientos, los demás factores como el Bajo Control de Calidad, Programación a Destajo, Falta de Conocimiento de la Arquitectura, Falta de un Programador Líder agudizan aún más esta deficiencia.

La Falta de Conocimiento de la Arquitectura puede ser resuelto mediante reuniones de capacitación para los programadores de parte de los Arquitectos del sistema antes de iniciar la programación; en ellas se deberán instruir a los primeros sobre todo lo relacionado con la arquitectura con la cual trabajarán: límites, restricciones, posibilidades, etc. Dichas reuniones podrían dar como resultado algún tipo de Manual normativo o de standard de desarrollo.

Con respecto a la necesidad de contar con un programador líder. Este programador líder debiese ser un programador provenido desde el mismo equipo de programación, el cual cuya trayectoria y experiencia hagan merecedor de aquel nuevo rol. Algunas responsabilidades podrían ser: coordinar el equipo, resolver problemas complejos, tomar decisiones técnicas, etc. Además seria él el indicado para llevar a cabo el control de calidad del código, velando por mantener una alta calidad, mediante la auditoria y el refactoring continuo.

Otro factor, identificado por los programadores, el cual afecta la calidad de su trabajo es contar con un Análisis Deficiente, los documentos generados por los analistas, estos son; casos

de uso, modelo de datos, diseño de interfaces, etc. En ocasiones están inconsistentes o incompletos. Para corregir esta deficiencia, los artefactos generados en el proceso de análisis requieren ser revisados y validados antes de que sean desarrollados por los programadores. Es necesario crear estas instancias de revisión y validación en análisis.

Analizando los resultados para el área de Aseguramiento de Calidad, se indica que los casos de uso no son lo suficientemente útiles para construir planes de prueba lo suficientemente completos para asegurar que el producto estuviese libre de errores. Era necesario contar con un conocimiento global y más acabado del negocio y no funcionalidades determinadas como son las que muestran los casos de uso. Para superar esta deficiencia es necesario que el o los QA's se integren al proceso de Análisis; participando de reuniones de captura de requerimientos, revisando los Artefactos, entre otros.

En la Tabla 3-5 se hace resumen de las soluciones para cada deficiencia descrita.

Tabla 3-5 Propuestas de mejoras para factores de deficiencias

Área	**Factor**	**Solución**	
		ID	**Descripción**
Análisis	Clientes poco Claros	S0	Utilización de métodos específicos o combinación de estos que permitan una óptima obtención de requerimientos.
	Falta de participación del cliente	S1	Utilizar estrategias que reviertan estas condiciones.
	No existencia de estándar	S2	Reunión de coordinación pre-análisis, en donde participen los Analistas, Jefes de proyectos y Arquitectos. Creación de un Manual Normativo de Análisis para el proyecto.
	Control insuficiente Bajo control de calidad	S3	Reuniones periódicas de revisión y validación de los trabajos en donde participen Analistas, Arquitectos y Jefes de proyectos.
	Falta dominio técnico sobre la tecnología	S4	Inclusión del Arquitecto en el proceso de Análisis para apoyo y validación de documentación.
Construcción	Falta de programador líder Bajo control de calidad	S5	Creación de un nuevo rol. Programador líder, quien coordine el equipo, resuelva problemas complejos, tome decisiones técnicas y vele por mantener una alta calidad mediante la auditoria del código fuente y el refactoring continuo.
	Falta de conocimiento de la arquitectura	S6	Reuniones de capacitación para los programadores de parte de los Arquitectos del sistema antes de iniciar la programación.
	Programación a destajo	S7	Creación de Manual de mejores prácticas de programación o un Manual que estandarice la programación en el proyecto.
	Análisis deficiente	S8	Resuelto en S3
Aseguramiento de calidad	Falta de conocimiento del negocio	S9	Integración del QA en el proceso de Análisis, revisión de Artefactos.
	Sistema no apto para pruebas	S10	Resuelto en S5

3.6.3 Análisis de Indicios en Proyectos No Exitosos

ZEKE desde sus inicios, ha desarrollado diversos proyectos de software, para diversos clientes, con propósitos y tecnologías muy distintas, esto se mantuvo por largo tiempo. En este último tiempo las características de los proyectos están convergiendo a una misma tecnología y en ocasiones muy similares en los propósitos.

El siguiente estudio tiene como finalidad analizar proyectos que resultaron no ser exitosos, con el fin de detectar factores tanto internos como externos que pudieron influir en que los proyectos terminaran en esas condiciones.

ZEKE cuenta con una manera de identificar un proyecto exitoso, aparte de que el producto software haya cumplido los requerimientos del cliente y haber contado con las características de un software de calidad, ZEKE suma a ello, la periocidad de compra del mismo cliente en un transcurso de años determinado. Esto es, que si un cliente vuelve a adquirir los servicios de ZEKE para el desarrollo de un nuevo proyecto en un periodo de no más de 3 años contando desde la finalización de un proyecto anterior para el mismo cliente, se considerará que este último trabajo resulto ser un éxito.

De la cartera de proyectos se seleccionaron dos que resultaron ser considerados como proyectos no exitosos. Entre las características se encuentran: el incumplimiento de los requerimientos del usuario, atrasos considerables en las fechas de entrega y baja calidad de los sistemas (sistemas poco estables, mantenible y robustos).

Para ambos proyectos que resultaron ser no exitosos, existen deficiencias que fueron comunes en el desarrollo de ambos. Estas deficiencias pueden ser homologados con los factores desencadenantes de deficiencia descubiertos en el proceso de desarrollo (ver 3.6.2 Búsqueda de Deficiencias en el Equipo de Desarrollo) de esta forma, los puntos pueden ser llevados al factor que los contiene como se aprecia en la Tabla 3-6.

Tabla 3-6 Correspondencia de deficiencias entre proyectos y equipo de desarrollo

Deficiencias en proyectos no exitosos	Deficiencias en equipo de desarrollo
No se contó con instancias de revisión y validación para los procesos y artefactos generados en análisis y construcción.	Bajo control de calidad.
Análisis deficiente.	Análisis deficiente.
La poca experiencia de los programadores.	Personal mediocre.
Falta de capacitación de las tecnologías de las arquitecturas.	Falta de conocimiento de la arquitectura.
Libre programación e implementación de los casos de uso.	Programación a destajo, bajo control de calidad y falta de un programador líder.
Subestimación y programación demasiado optimista de las actividades de análisis y construcción.	Programación demasiado optimista.
Ejecución de pruebas para certificación interna deficiente.	Bajo control de calidad, falta de conocimientos del negocio y sistema no apto para pruebas.

Como se pudo apreciar en las tabla de arriba, las deficiencias presentadas en los proyectos no exitosos tienen una correspondencia con los factores desencadenantes de deficiencia más críticos para las tres áreas del desarrollo, por tanto, se valida la identificación por los distintos implicados en el proceso de desarrollo.

4 Propuesta de Metodología Ágil de Desarrollo de Software con Integración de TAM

4.1 Bases de la nueva propuesta de Metodología de Desarrollo

Los problemas con el desarrollo mostrados en los puntos anteriores tienen una implicancia directa sobre el bienestar económico de la empresa, los proyectos que sufren de retrasos incrementan los costos y los proyectos que no satisfacen a los clientes traen consigo una pérdida de la reputación sobre el trabajo, profesionalismo y calidad como empresa.

Una imagen corporativa sólida es un incentivo para la venta de productos y servicios. Ayuda a la empresa a contratar a los mejores empleados, atrae inversionistas, genera confianza entre los públicos internos y externos (Blauw, 1994). Una imagen corporativa firme crea un valor agregado a una empresa y asegura que esté un paso adelante de sus competidores (Brinkerhof, 1990).

Si bien es cierto, no se ha presentado el caso en el cual un cliente haya tomado algún tipo de medida legal en contra de la empresa por el no cumplimiento de los acuerdos o bases de alguna licitación, sí han existido casos en donde a causa de la mala gestión del proyecto, incluyendo esto, el proceso de desarrollo, clientes han manifestado explícitamente su descontento y rechazo al trabajo de la empresa por la mala calidad de los procesos y sus productos, este malestar ha traído consigo la desconfianza de potenciales nuevos clientes de adquirir los servicios de ZEKE. Todo indica que si se siguen manteniendo los actuales procesos y procedimientos de desarrollo no se estaría muy alejado de esta medida.

Como se demostró en los capítulos anteriores, el proceso de desarrollo de software creado bajo la norma ISO 9001:2008, no está dando buenos resultados para la gran cantidad, diversidad y complejidad de los actuales proyectos que ZEKE está gestionando. Existen numerosas deficiencias que abarcan desde los procesos de levantamiento hasta las fases finales de prueba. La mayoría de las deficiencias tienen relación con la imposibilidad de gestionar de forma correcta los cambios de requerimientos, la obtención de calidad de los productos y la falta revisión de los artefactos.

Como se muestra en la Tabla 3-5, las soluciones para los factores de deficiencia más críticos. Dichas soluciones permitieron identificar qué requerimientos eran necesarios incorporar a esta propuesta de Metodología de desarrollo y de esta manera asegurar que el ciclo de vida no poseerá todos aquellos elementos que fueron fuentes de deficiencia de calidad en el proceso de desarrollo anterior.

Si antes se utilizó el modelo de cascada clásico éste se ha reemplazado por uno iterativo e incremental agrupado en 4 fases. En esta propuesta de Metodología de desarrollo se pone énfasis en las revisiones de los artefactos y entregables, la gestión de los riesgos y la participación del cliente en el desarrollo.

Si en el actual proceso de desarrollo, las actividades de revisión eran insuficientes, en esta propuesta se hace hincapié en contar con nuevas instancias de revisión y validación, para ello, se han redefinido las actividades del rol de QA's y creado nuevos roles, el Programador Líder y el Tester.

Un punto importante a destacar es la gran responsabilidad que debe tener el rol QA en esta nueva propuesta. La participación del QA en el actual proceso de desarrollo es solo la de confección y ejecución de planes de prueba para la certificación interna; en esta propuesta, el QA deberá velar por la calidad de todo el proceso, revisando y validando la mayoría de los artefactos que se generan en todas las fases, interactuando continuamente con todos los miembros del proyecto, haciendo cumplir las entregas y los compromisos adoptados.

Con respecto a la incorporación del cliente al desarrollo, en el actual proceso, solo existen 2 instancias de participación del cliente; una primera en el levantamiento de requerimientos y una última correspondiente a la obtención de su certificación del producto. Por tanto, durante todo el esfuerzo de construcción el cliente desconoce el proceso, no existe visibilidad del producto por lo que los cambios de requerimientos aparecen al final del proceso de desarrollo cuando el sistema está por completo construido cuando el cliente logra visualizar el sistema para su certificación. Con esta propuesta, se integra al cliente en la programación desde el inicio disminuyendo los esfuerzos del trabajo asociado a los cambios de requerimientos, esto se logra liberando pequeñas entregas que son revisadas y validadas por el mismo, por lo que él está al tanto de la evolución del producto, desde pequeños módulos funcionales en un principio hasta llegar al sistema por completo al final de la construcción.

Analizando el actual enfoque de desarrollo de software y esta nueva propuesta, es posible posicionar ambas en un mapa de formalidad, como se muestra en la Figura 4.1. En la ilustración es posible apreciar la ubicación de varias metodologías ubicadas según el nivel de rigurosidad o formalidad de sus procesos. La línea va desde la Burocracia, representada por procesos rígidos y definidos hasta el más mínimo nivel de detalles, y el de Adhocracia, que representa el desarrollo caótico sin ningún proceso ni visibilidad sobre el estado y el rumbo de los proyectos.

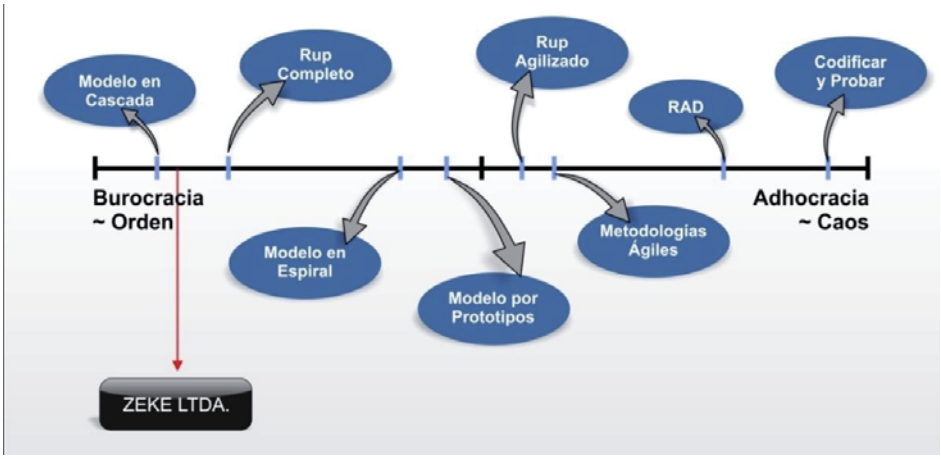


Figura 4.1 Ubicación del actual proceso de desarrollo de ZEKE

Según la ilustración, los procesos ubicados hacia la izquierda pretenden definir todas las actividades, roles, entregables, y demás aspectos relacionados en un proyecto de software, para generar resultados predecibles a lo largo del tiempo. El actual proceso de desarrollo de ZEKE, inspirado en el Modelo en Cascada ejemplifica este enfoque al ser un modelo orientado a documentos, en donde el progreso se mide en base a un conjunto de fases bien definidas y al grado de avance de la documentación correspondiente. Sin embargo, y como ya se ha descrito, este modelo no se está ajustando a los nuevos desafíos de la empresa, principalmente por la envergadura y complejidad de los nuevos proyectos trayendo consigo todos los problemas descritos en los capítulos anteriores.

Esta propuesta de metodología, inspirada en los principios y practicas ágiles pretende corregir estas deficiencias, transformando del antiguo proceso, trasladándolo desde una orientación puramente burocrática a un nuevo punto, más cercano a los Metodologías Ágiles (ver Figura 4.2), poniendo énfasis en las entregas tempranas e incorporación del cliente, ambas practicas rechazadas por el actual modelo de desarrollo.

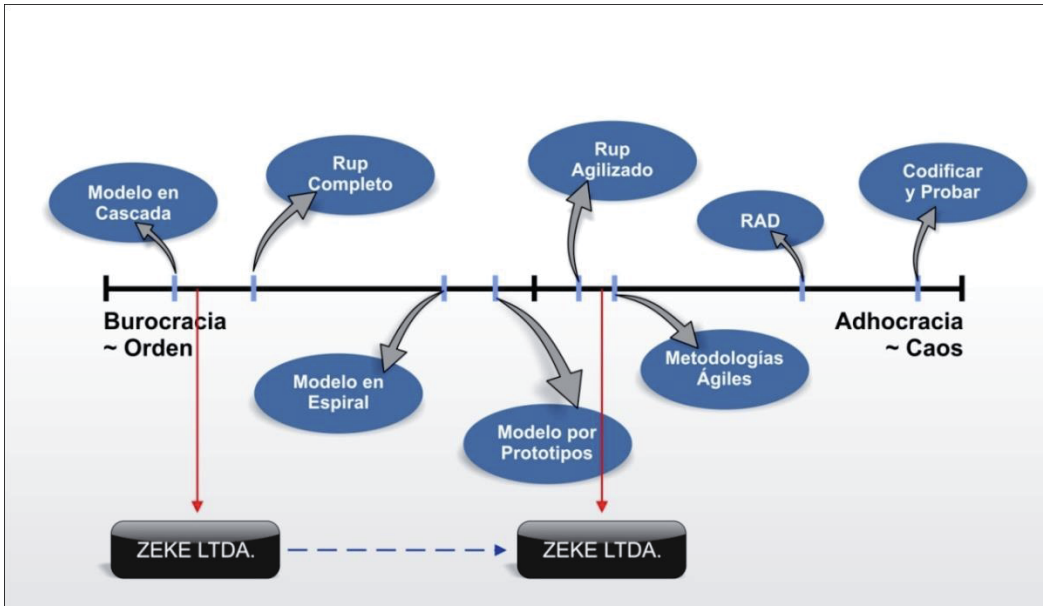


Figura 4.2 Nueva ubicación del proceso de desarrollo de ZEKE

En la ilustración de más arriba, es posible apreciar la posición en la cual se ubicaría esta propuesta de metodología. Como se muestra en la imagen, se pretende desplazar el actual proceso, ubicado del rincón de la esquina izquierda, asociados a metodologías tradicionales o rígidas de la línea hacia el centro, introduciendo una que mezcle aspectos de las metodologías ágiles, tales como XP, SCRUM, entre otras.

La siguiente propuesta de Metodología de desarrollo de software, denominada en adelante *Z-ÁGIL*, ha sido concebida utilizando la guía propuesta por Alistair Cockburn (Cockburn, 1998) que desglosa una metodología en 9 elementos, enumerados a continuación:

1. Roles: Las descripciones a los trabajos que se solicitan en las búsquedas laborales cuando se necesitan tomar gente.
2. Destrezas: Las destrezas que presupone poseen los recursos que llevan a cabo el proyecto. Por ejemplo: social, proactivo, hábil en el manejo de herramientas.
3. Técnicas: Las técnicas que se espera que las personas utilicen en su trabajo diario.
4. Herramientas: Las herramientas que las personas usan a diario.
5. Asignación de Tareas: Cómo son asignadas las tareas a los diferentes roles.
6. Equipos: La manera en que cómo agrupan a las personas.
7. Entregables (Artefactos): Lo que se quiere que cada persona o equipo entregue a otra persona o equipo.
8. Estándares: Lo que está o no permitido en el diseño y los entregables.
9. Actividades: Las actividades e hitos de los diferentes equipos, cómo se integran para producir los entregables finales.

En las próximas secciones se describen cada uno de los elementos listados, cumpliendo con ello las bases que toda Metodología de Desarrollo de Software debiese poseer.

4.2 Principios Agiles adoptados e Integración de TAM

Para dar solución a las deficiencias declaradas en los capítulos anteriores y creando consistencia con las bases de esta nueva propuesta metodológica es que se han integrado principios y valores de Metodologías Ágiles, como se aprecia en la Tabla 4-1.

Tabla 4-1 Prácticas y principios asimilados

Característica Adoptada	**Metodología(s) Fuente(s)**
Buena comunicación entre los integrantes del equipo de desarrollo	XP
Cliente in-situ	XP
Entregas tempranas y frecuentes	XP y DSDM
Pruebas unitarias por parte de los programadores	XP
Comunicación entre el equipo por sobre el exceso de documentación.	XP
Desarrollo iterativo e incremental	UP, XP y DSDM
Todos los cambios durante el desarrollo son reversibles	DSDM
El testing es integrado a través del ciclo de vida	DSDM
Los requerimientos están especificados a un alto nivel	DSDM
Integración continua	XP y DSDM

Con respecto a la inclusión de TAM. TAM es integrado como herramienta, su uso se materializa como actividad concreta dentro del ciclo de vida de *Z-Ágil* cuyo significativo aporte es la entrega de información para la realización de ajustes o cambios al software en fase de construcción.

4.3 Roles, Destrezas y Actividades

Los roles definen ese conjunto cohesivo de actividades realizadas y artefactos mantenidos que son llevados a cabo por personas o por grupos de personas. No sólo se refieren a personas internas al desarrollo del software, sino que también involucran a los usuarios u otras personas que se vean afectadas por el proyecto, cualquiera de los denominados stakeholders.

4.3.1 Jefe de proyecto

- Descripción: Profesional encargado de gestionar el proyecto; la realización de seguimientos y controles de avance.

- Actividades: Reuniones de planificación, la identificación de riesgos, la aplicación y análisis de las encuestas de aceptación TAM, coordinación de entregas al cliente, la verificación que el proceso de desarrollo marche en lo presupuestado, entre otras actividades.

- Destrezas: Liderazgo, capacidad de trabajo bajo presión y trabajo en equipos, autónomo, metódico y proactivo.

4.3.2 Analista

- Descripción: Profesional encargado de levantar los requerimientos del cliente y generar toda la documentación necesaria para que el equipo de programación construya los requisitos, para ello, las actividades: Construcción del Informe de Especificación de Requerimientos, Descriptores de Requerimientos. Además de generar documentación anexa para apoyo al cliente como lo es el Manual de usuario.

- Destrezas: Capacidad de análisis, de trabajo en equipo y bajo presión, metódico relaciones interpersonales.

4.3.3 Programador líder

- Descripción: Programador con vasta experiencia en el desarrollo de software, capaz de liderar equipos y resolver conflictos. Su objetivo es asistir al equipo de programación en cualquier problema que exista en la implementación de los requerimientos, la utilización de las herramientas o las dudas que pudiesen existir sobre las formas.

- Actividades: Encargado de solucionar aquellos requerimientos que sean identificados como complejos o estén asociados a potenciales riesgos que pudiesen afectar la calidad del producto o la programación del proyecto.

  El programador líder tiene la responsabilidad de velar por la calidad del código y las implementaciones de los casos de uso de todo el equipo desarrollador, deberá efectuar auditorias de código de forma continua, informar a los desarrolladores sobre las deficiencias que pudiesen existir, ayudando y asistiendo en las correcciones si se necesitase.

- Destrezas: Liderazgo, capacidad de trabajo bajo presión y trabajo en equipos, autónomo, metódico y proactivo.

4.3.4 Programador

- Descripción: Profesional cuya función es la implementación o programación de los Descriptores de Requerimientos generados por el Analista.

- Actividades: El Analista genera el Artefacto denominado Descriptor de Requerimiento que es la entrada para el trabajo del programador, los programadores deben entregar el requerimiento construido según las especificaciones del Descriptor.

- Destrezas: Autodidacta, metódico, capacidad de trabajo bajo presión y trabajo en equipo.

4.3.5 Arquitecto

- Descripción: Profesional encargado del desarrollo y difusión de Arquitectura de software.

- Actividades: Proponer Arquitecturas que satisfagan los requerimientos de la solución, construir los Documentos de Arquitectura de Software y si el Jefe del Proyecto lo determina, realizar capacitaciones a los programadores, también debe efectuar revisiones de los Descriptores de Requerimientos generados por el Analista para verificar que estas especificaciones se ajustan a la Arquitectura que el propuso.

- Destrezas: Capacidad de análisis e investigación, autónomo y autodidacta.

4.3.6 QA

- Descripción: Profesional encargado de asegurar la calidad de todo el proceso de desarrollo. Tendrá un rol importante en la metodología ya que debe velar por que se cumplan los compromisos, las fechas establecidas, fechas de entrega y por sobre todo velar por la calidad de los artefactos. En el ciclo de vida de la metodología, el QA tiene

actividades definidas de revisión y validación de documentos generados en análisis pudiendo aceptar o rechazar las confecciones, esto no lo inhabilita para ejercer revisiones de otros artefactos o documentos generados por otros roles aparte del Analista.

- Actividades: Revisión del Informe de Especificación de Requerimientos y Descriptores de Requerimientos.

- Destrezas: Capacidad de análisis, metódico, capacidad de trabajo bajo presión y relaciones interpersonales.

4.3.7 Tester

- Descripción: Profesional encargado de realizar pruebas a los Release que se van generando por iteración en la fase de construcción.

- Actividades: Por cada nuevo Release que se genera en cada iteración de construcción el Tester debe validar que la implementación sea la correcta, esto es, que se cumplan las reglas de negocio descritas en los Descriptores de requerimientos.

- Destrezas: Capacidad de análisis y metódico.

4.4 Artefactos

La mayoría de las Metodologías Ágiles hacen hincapié en que el desarrollo debe centrarse en las personas, sus relaciones y por sobre todo, en un producto que funcione. La excesiva documentación es enemiga del agilísimo ya que provoca esfuerzos innecesarios en la creación y la mantención de estos, tiempos que pudiesen ser aprovechados en cambio, en mantener una constante comunicación entre los desarrolladores y el cliente. A pesar de esto, es necesario contar con un mínimo de documentación que sirva de comunicación, acuerdo y gestión, motivos sin los cuales un proyecto no llegaría al éxito.

En la Tabla 4-2 se enumeran los artefactos utilizados por *Z-ÁGIL*. Los artefactos marcados como “Opcional” pueden no ser incluidos en el desarrollo, todo dependerá de la naturaleza del proyecto, dimensiones, tecnologías o características del cliente. Los artefactos indicados como “Obligatorio”, corresponden a los entregables mínimos requeridos para todo desarrollo a medida que utilicen *Z-ÁGIL*, siendo las bases en la que se sustenta.

Tabla 4-2 Obligatoriedad de cumplimiento de artefactos

Artefacto	Obligatoriedad
Descriptor de Visión y Alcance	Obligatorio
Informe de Especificación de Requerimientos	Obligatorio
Documento de Arquitectura de Software	Opcional
Descriptor de Riesgos	Opcional
Descriptor de Requerimiento	Obligatorio
Encuesta TAM	Obligatorio
Revisión de Ajuste	Obligatorio
Manual de Usuario	Obligatorio
Manual de Instalación	Obligatorio
Release	Obligatorio

4.4.1 Descriptor de Visión y Alcance

El Descriptor de Visión y Alcance es un documento cuyo propósito es reflejar tanto el problema por el cual se está realizando el proyecto como la solución que ZEKE está proponiendo, lo importante de este documento es el hecho que refleja el límite o alcance que el sistema va a tener con respecto al problema que se quiere solucionar, por lo que juega un importante rol ya que delimita el proyecto y sistema, dejando en claro lo que construirá y lo que no se abordará. La plantilla de este documento se puede ver en el Anexo I:Plantilla de Descriptor de Visión y Alcance.

4.4.2 Informe de Especificación de Requerimientos

Documento que contiene la definición de todos los requerimientos que se desean tener en el producto. Es construido por el Analista en base a las reuniones que sostiene con las contrapartes de negocio. Contiene una descripción de alto nivel de cada uno de los requerimientos, las reglas de negocio asociadas a ellos y la identificación de los actores del sistema. Este informe es acompañado de otros artefactos como los son el Modelo de Dominio, Modelo de Datos y Diagramas de Estado.

El propósito de este documento es reflejar el entendimiento de los requerimientos entre el Analista y la contraparte de negocios, sirviendo como común acuerdo antes de iniciar la programación, de esta manera el cliente valida y acepta, que lo mostrado en el informe corresponderá a lo que se construirá.

La plantilla de este documento se puede apreciar en el Anexo J: Plantilla Informe de Especificación de Requerimientos.

4.4.3 Documento de Arquitectura de Software

Documento confeccionado por el Arquitecto de Software. Contiene una descripción técnica de los componentes y sus relaciones en donde se especifican los distintos aspectos técnicos y funcionales.

Este documento es construido en base a vistas, que representan las dimensiones del sistema:

- Vista funcional: Permite resumir cuáles son las funciones principales del sistema. También permite hacer un vínculo explícito entre los aspectos funcionales del sistema (casos de uso, historias, etc.) y explicar por qué son importantes para la arquitectura. La vista funcional responde a la pregunta ¿Qué hace el software?

- Vista lógica: Permite presentar la estructura del sistema a través de sus componentes de software y sus interacciones.

- Vista de infraestructura: Se utiliza para describir el hardware físico y redes en las que el software será implementado.

- Vista de seguridad: Proporciona una manera de explicar cómo la arquitectura implementa los requisitos de seguridad como la autenticación, autorización, confidencialidad de datos, etc.

- Vista de datos: Corresponde al modelo relacional y diccionario de datos del sistema. Útil para conocer los aspectos técnicos de cómo se almacenen los datos y cómo son tratados por el motor de base de datos que los gestiona.

La plantilla de este documento se puede apreciar en el Anexo K:Plantilla de Documento de Arquitectura de Software.

4.4.4 Descriptor de Riesgos

Documento creado por el Jefe del Proyecto con el propósito de mantener un listado de los riesgos del proyecto. A cada riesgo del listado se le asocian los parámetros mostrados en la Tabla 4-3. El listado de los riesgos es dinámico, variará conforme se avance en el proyecto, en cada fase de desarrollo existe una instancia de revisión de los riesgos, estas instancias llevan consigo la identificación de nuevos riesgos asociados a la fase que deben ser ingresados en el listado más el cálculo del grado de exposición de todos.

El Jefe del Proyecto por cada revisión del Descriptor de Riesgos debe determinar la activación de los planes de contingencia o mitigación de los riesgos según el grado de exposición que muestran al momento de la revisión. Quedará a criterio del Jefe del Proyecto establecer el

límite del grado de exposición que haga la diferencia de la aplicación de uno u otro plan, todo en función de la materialización del riesgo.

Tabla 4-3 Valores posibles de los parámetros para evaluación de riesgos

Parámetro	Descripción	Valores posibles
Probabilidad	Como escala para determinar la probabilidad de que el riesgo se materialice.	1,2,3
Impacto	Como escala para determinar el impacto de los efectos que puede causar el riesgo sobre el proyecto en caso de materializarse.	1,2,3,4 y 5
Exposición	En función de la calificación de atributos (probabilidad e impacto), se obtiene la exposición al riesgo según la siguiente formula: *E = Probabilidad * Impacto*	
Categoría	Corresponden a los aspectos relacionados con el proyecto	Las Personas, el Proceso, el Producto, la Tecnología.
Plan de contingencia	Corresponden a todas las acciones a realizar para disminuir la probabilidad que el riesgo se materialice.	
Plan de mitigación	Corresponden a todas las acciones a realizar para disminuir el impacto en el proyecto luego que el riesgo se ha materializado.	

La plantilla de este documento se puede apreciar en el Anexo L:Plantilla Descriptor de Riesgos.

4.4.5 Descriptor de Requerimiento

Documento que contiene la descripción de un requerimiento determinado que debe ser implementado. En la descripción se incluyen las reglas de negocio que deben ser desarrolladas, por ejemplo: entrada, salidas, validaciones, restricciones, actores involucrados, relación con los Modelos de Dominio y Datos e información asociado al proyecto, como lo son; tiempos de desarrollo, nombre del programador encargado entre otras.

Este Artefacto corresponde a la selección de un Caso de uso del Informe de Especificación de Requerimientos, sin embargo, los Casos de uso contenidos en el informe difieren de los Casos de uso normales, la diferencia radica en que los Casos de Uso del informe contienen una cantidad de escenarios o pantallas estrechamente relacionadas que el programador debe realizar para dar por concluido el desarrollo del Caso de uso. La agrupación de pantallas en los Casos de uso entrega mayor detalle del negocio al programador permitiendo que éste obtenga una mejor visión del problema y del sistema.

La plantilla de este documento se puede apreciar en el Anexo M:Plantilla de Descriptor de Requerimiento.

4.4.6 Encuesta TAM

La encuesta TAM es un artefacto generado por el Jefe del Proyecto útil para la aplicación de TAM y la obtención de resultados. Este documento tipo encuesta es aplicado al Cliente para que evalúe el sistema, la encuesta debe reflejar tanto la utilidad percibida como la facilidad de uso del sistema, características necesarias para la aplicación del modelo, por tanto las preguntas deben ir orientadas siempre en esos aspectos.

Como se mostrará en el uso de este artefacto en el ciclo de vida de la metodología, la aplicación de la Encuesta TAM no se hace al sistema entero si no a grupos de requerimientos implementados por iteraciones de construcción, por lo que el Jefe del Proyecto debe rediseñar las encuestas por cada nueva iteración pues contendrá un nuevo grupo de requerimientos necesarios a evaluar que se consolidan con otros ya evaluados.

La plantilla de este documento se puede apreciar en el Anexo P:Plantilla de Encuesta TAM.

4.4.7 Revisión de Ajuste

Documento confeccionado por el Jefe del Proyecto posterior a la aplicación de la Encuesta TAM y el análisis de esta contiene las correcciones y ajustes que se deben realizar a los requerimientos que ya han sido programados. Con el análisis de la aplicación de TAM, el Jefe de Proyecto puede determinar que existen requerimientos cuya implementación ha sido deficiente o que se requiere realizar ajustes en ellos para la obtención de un mayor grado de aceptación por parte del cliente.

Como se mostró en el punto 3.1 Modelo de Aceptación de la Tecnología (TAM), el modelo se basa en dos características para determinar el grado de aceptación del sistema; la utilidad percibida y la facilidad de uso percibida. El Jefe del Proyecto debe determinar qué requerimientos o que elementos en ellos influyen negativamente en ambas percepciones y que afectan la intención de usar el sistema, utilizando para ello los indicadores obtenidos en la Encuesta TAM.

Como se explicó en el punto anterior, la aplicación de la Encuesta TAM es realizada al finalizar cada iteración de construcción, por tanto, la confección del documento de Revisión de Ajuste debe ser creado en las mismas condiciones, esto queda más claro en punto 4.5.3 Fase de Construcción.

La plantilla de este documento se puede apreciar en el Anexo N:Plantilla de Revisión de Ajuste.

4.4.8 Manual de Usuario

Documento que contiene las especificaciones de uso del sistema, creado por el Analista tiene el propósito de servir como instrumento de guía y de consulta a los usuarios del sistema.

4.4.9 Manual de Instalación

Documento elaborado por el Programador Líder que sirve de guía para la instalación y puesta en marcha del sistema. Contiene una detallada especificación sobre el funcionamiento de los componentes software y hardware requeridos y cómo éstos deben ser configurados para la instalación y uso correcto del sistema.

4.4.10 Release

Corresponde a una versión del sistema que contiene un conjunto de requerimientos construidos y cuya finalidad es ser probado por el cliente y mediante la aplicación de la Encuesta TAM (ver 4.4.6 Encuesta TAM) para evaluar el grado de aceptación mediante las Percepción de Utilidad, Facilidad de Uso, Actitud e Intención de Uso.

4.5 Ciclo de Vida

El ciclo de vida describe el desarrollo de software, desde la fase inicial hasta la fase final. El propósito es definir las distintas fases y flujos que se requieren para garantizar que el software cumpla los requisitos por el cual fue concebido.

Partiendo por el hecho que:

A. Es muy costoso rectificar los errores que se detectan tarde dentro de la fase de implementación (Papaccio & Boehm, 1988)

B. Los principios ágiles que han sido asimilados por *Z-Ágil* (ver 4.2 Principios Agiles adoptados e Integración de TAM)

C. La integración del TAM para la obtención de información sobre la aceptación y las percepciones de utilidad y usabilidad

El ciclo de vida de *Z-Ágil* se ha divido en un único ciclo de vida con 4 fases:

- Fase de Concepción: Conformar el equipo y asignar responsabilidades, realizar análisis de riesgos y definir el alcance.

- Fase de Exploración: Capturar y documentar los requerimientos, diseñar parte de la solución.

- Fase de Construcción: Diseñar y construir el producto.

- Fase de Implantación y Transferencia: Finalizar el proyecto, entregar el producto y la documentación generada.

Esta organización por fases tiene el propósito de organizar trabajos comunes y definir hitos claros de finalización e inicio entre fases, poniendo énfasis a la revisión de los entregables (QA-Tester) y la participación del cliente en el desarrollo. Por ello existen gran número de actividades que giran en torno a estos 2 aspectos.

El desarrollo de todas las fases, excluyendo la de Concepción, son realizadas en iteraciones cortas, cada iteración termina con la entrega de algún artefacto o la realización de alguna actividad de tipo hito (ver Figura 4.3). Las actividades, responsables y entregables de cada iteración variará dependiendo de la fase en que se encuentre el proyecto.

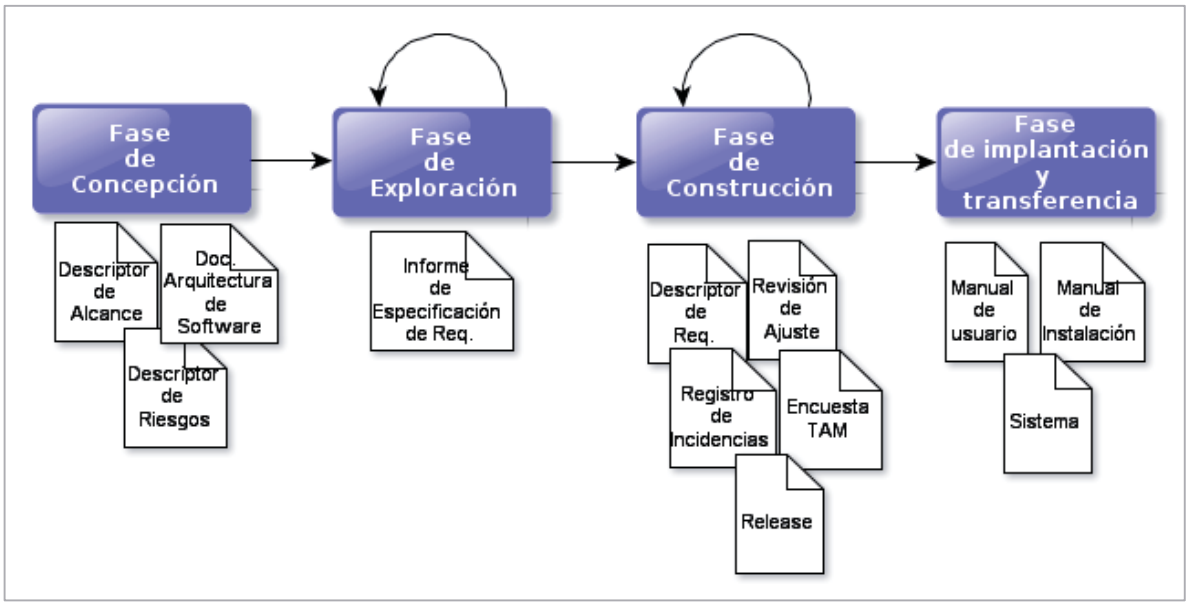


Figura 4.3 Artefactos generados por las fases

4.5.1 Fase de Concepción

En esta fase se realiza un análisis del proyecto de manera global, determinando alcances, identificando riesgos y restricciones.

El Jefe del Proyecto debe determinar el alcance que tendrá el proyecto, según el problema que se quiere resolver. Para ello debe contar con la mayor información posible sobre el problema, puede tomar como base soluciones anteriores que hayan resuelto algún problema similar para conocer cómo se enfrentó el desarrollo del proyecto y coger datos útiles que pudiesen servir para que sean aplicados al actual. El artefacto que se genera en este análisis es el Descriptor de Alcance (ver 4.4.1 Descriptor de Visión y Alcance).

El Jefe de Proyecto puede determinar los posibles riesgos asociados al desarrollo del proyecto, dicho análisis debe incluir planes de contingencia y mitigación. El artefacto resultante de dicha identificación será el Descriptor de Riesgos (ver 4.4.4 Descriptor de Riesgos).

Para el caso que el proyecto requiera más de un analista, el Jefe del Proyecto debe coordinar en conjunto con ellos, la manera en la cual se va a enfrentar el análisis, esto es, de

cómo se realizará la captura de requerimientos y su documentación, como también las herramientas con las cuales los analistas tendrán que trabajar.

Entre los miembros del equipo de programadores, el Jefe del Proyecto tiene que seleccionar el responsable de oficiar como el Programador Líder del equipo. Dicha selección tiene que ser comunicada a los desarrolladores indicando la función y alcance que tendrá él en el desarrollo del proyecto.

El Arquitecto según la especificación del Descriptor de Alcance, puede construir el Documento de Arquitectura de Software (ver Anexo J: Plantilla Informe de Especificación de Requerimientos), si fuese el caso, este documento tiene que quedar a disposición del equipo de programación y debe servir como guía maestra para el desarrollo.

Según la arquitectura de la solución que se ha propuesto y si el Jefe de Proyecto lo determina, es posible realizar una inducción sobre la arquitectura al equipo de programadores, dicha inducción debe incluir aparte entre las características de la arquitectura ejemplos prácticos de cómo resolver problemas comunes que también quedarán plasmados en el Documento de Arquitectura de Software.

En la Figura 4.4 se hace resumen de las actividades y artefactos de la Fase de Concepción

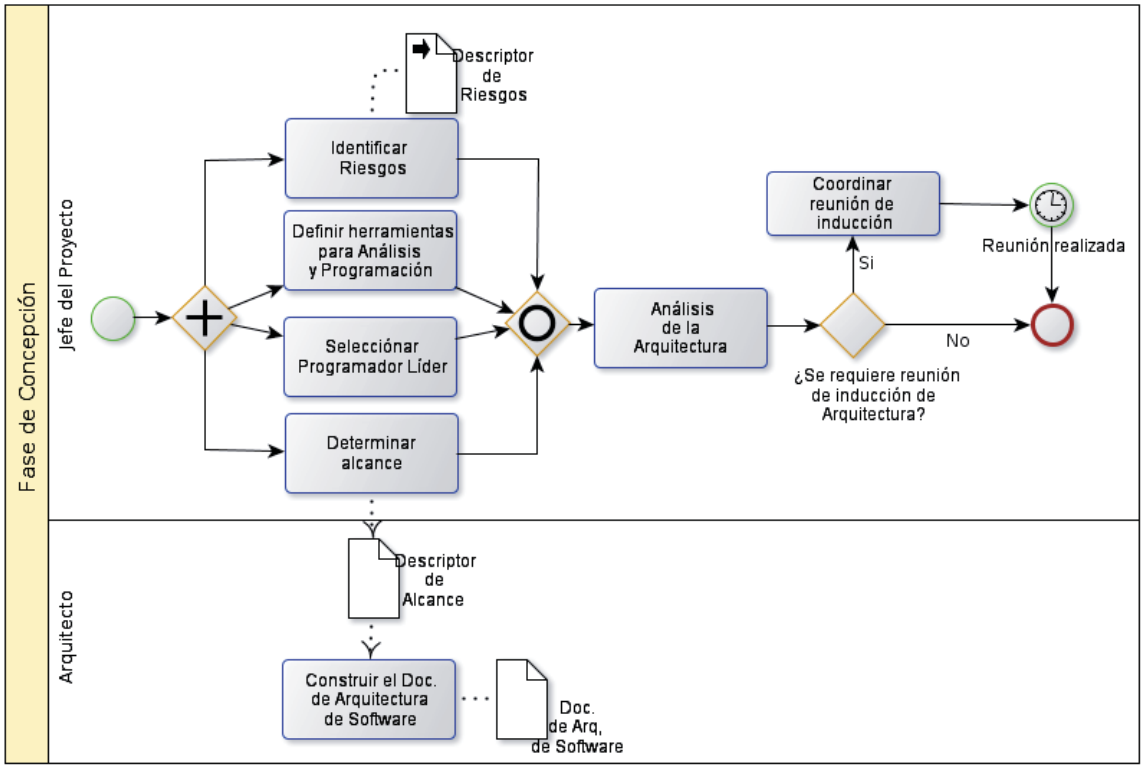


Figura 4.4 Fase de Concepción

4.5.2 Fase de Exploración

Fase en la cual el equipo de análisis realiza el levantamiento de requerimientos, esta fase da inicia con la reunión de kickoff y termina con el hito de entrega del artefacto de Informe de Especificación de Requerimientos (ver 4.4.2 Informe de Especificación de Requerimientos).

Esta fase es desarrollada en varias iteraciones, al menos 2 y no más de una semana de duración. Cada iteración consiste en al menos 2 reuniones con las contrapartes de negocio, cada iteración finaliza con una construcción parcial del Informe de Especificación de Requerimientos. En cada inicio de una nueva iteración se revisan y validan con las contrapartes de negocio los requerimientos obtenidos en las reuniones anteriores de las iteraciones pasadas. Cada nueva iteración resultará en un incremento en el Informe de Especificación de Requerimientos.

Si se ha definido que se realizarían evaluaciones de riesgos en la fase de concepción, cada iteración de esta fase debe incluir también una revisión del Descriptor de Riesgos formulado en la fase anterior. La revisión consiste en re-calcular los grados de exposición (probabilidad * impacto) de los riesgos ya identificados y la inclusión de nuevos obtenidos según el análisis de cada reunión. El proceso de re-calcular los riesgos y la incorporación de nuevos tiene que hacerse en conjunto con el Jefe de Proyecto, si en este estudio se determina que existen riesgos que potencialmente pueden volverse realidad será necesario la activación de los planes de mitigación y contingencia.

Con el Informe de Especificación de Requerimientos construido en su totalidad, el QA asignado al proyecto realiza una revisión del documento para determinar el correcto desarrollo de los artefactos. Si el QA encuentra que unos de estos documentos resulta ser deficiente, informa al Jefe del Proyecto de las deficiencias para que éste en conjunto con el Analista determinen la validez del error, corrigiendo el Analista los errores si es que los hubiere. Si el QA apruebas los artefactos informa al Jefe del proyecto del éxito de la revisión.

Con el visto bueno del QA se da a conocer a las contrapartes de negocio el Informe de Especificación de Requerimientos para la obtención de su aprobación. Mediante la aprobación de la contraparte se certifica que se ha logrado identificar, dimensionar y comprender sus requerimientos.

En la Figura 4.5 se hace resumen de las actividades y artefactos de la Fase de Exploración

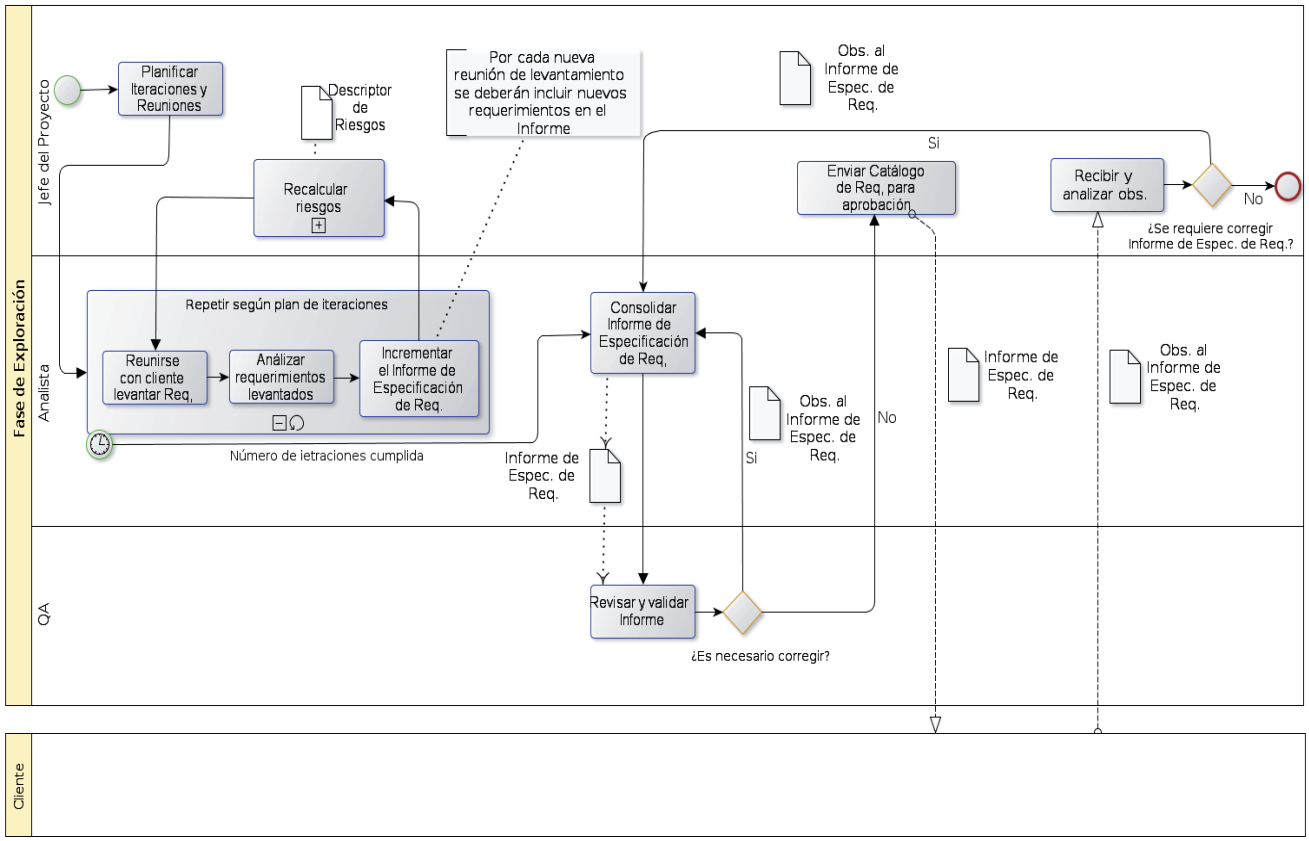


Figura 4.5 Fase de Exploración

4.5.3 Fase de Construcción

Fase en donde se construyen los requerimientos. Se inicia con una reunión de inducción del negocio al equipo de programación y con una priorización de los requerimientos a desarrollar. Esta fase finaliza con la aceptación del producto. La fase es desarrollada en iteraciones que consisten en la implementación de un conjunto de requerimientos, cada iteración finaliza con la satisfacción y aceptación del cliente con respecto al conjunto de requerimientos realizados.

El Jefe del Proyecto selecciona un grupo de requerimientos del Informe de Especificación de Requerimientos que serán implementados en la iteración. Cada requerimiento es formalizado en un Descriptor de Requerimiento (ver 4.4.5 Descriptor de Requerimiento), documento confeccionado por el Analista para que sea desarrollado por el programador asignado. Dicho documento contiene tanto la descripción del requerimiento como la especificación de todas las reglas de negocio asociadas a él. El grupo de Descriptores de requerimiento es sometido a una doble validación. En primer lugar es revisado por el Arquitecto del sistema quien examina a grandes rasgos que las reglas de negocio se ajusten a la arquitectura de la solución, para luego ser revisado por el QA para asegurar que cada Descriptor de Requerimiento del grupo, se ajusta a los Modelos de Dominio y de Datos y que el documento en sí, cumpla con toda la especificaciones necesarias para la correcta implementación por parte de los programadores. Tanto como el Arquitecto como el QA pueden rechazar el trabajo del Analista, si es el caso, el Analista debe tomar en consideración las observaciones de ambos y re-hacer el trabajo.

El Jefe del Proyecto asigna los Descriptores de Requerimiento a cada programador. Por cada finalización de la implementación de un Descriptor de Requerimiento, el Programador Líder revisa y valida el código, pudiendo rechazar el trabajo si este no cumple el documentos de las buenas practicas o si la manera con la cual el código del Descriptor fue desarrollado resulta ser deficiente. Con el visto bueno de la implementación del Descriptor de Requerimiento, este es integrado al servidor de integración continua.

Con el conjunto de Descriptores de Requerimientos programados, el Tester revisa y valida las implementaciones, sometiéndolo a pruebas basadas en las reglas de negocio descritas en cada Descriptor de Requerimiento. Para el caso que la revisión de alguna implementación no sea satisfactoria, el Tester registra el detalle del problema en el Registro de Incidencias utilizando para ello el Gestor de Incidencia.

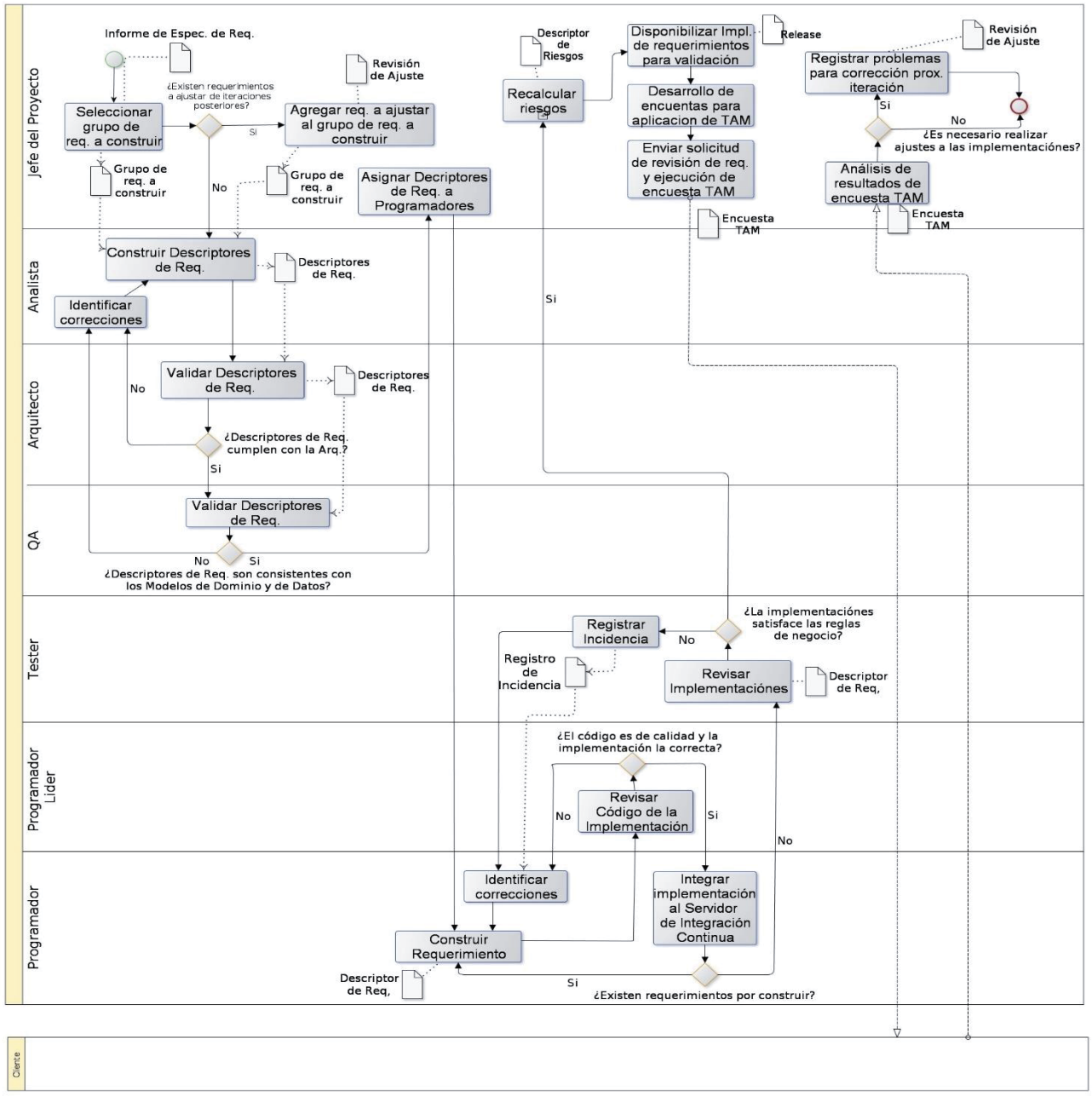
Con el conjunto de Descriptor de Requerimiento implementados y validados por el Tester, el Jefe del Proyecto somete el sistema a pruebas de satisfacción y aceptación por parte del cliente. El jefe del proyecto construye la Encuesta TAM (ver 4.4.6 Encuesta TAM) sobre los requerimientos desarrollados, este conjunto de requerimientos a ser evaluados corresponderán a una versión del sistema, Release (ver 4.4.10 Release). Si la prueba de aceptación aplicada al Release resulta insatisfactoria, se identifican las deficiencias y se integran a la próxima iteración de construcción como requerimientos necesarios a ajustar; este registro queda en la Revisión de Ajuste (ver 4.4.7 Revisión de Ajuste).

Si se ha definido que se realizarían evaluaciones de riesgos en la fase de concepción, cada iteración de esta fase debe incluir también una revisión del Descriptor de Riesgos para re-calcular los grados de exposición (probabilidad * impacto) de la lista de los riesgos identificados más la integración de nuevos que son identificados por el Programador Líder. Si en conjunto se determina que existe la posibilidad de algunos riesgos puedan volverse realidad, el Jefe del Proyecto debe activar los planes de mitigación y contingencia de cada uno de los riesgos potenciales.

Las iteraciones finalizan cuando ya no quedan más grupos de requerimientos a desarrollar. El producto integrado es liberado para la revisión del cliente. Las observaciones del cliente también son ingresadas en el Registro de Incidencias y al igual que las revisiones internas, el Jefe del Proyecto estima la magnitud del error en costo de tiempo y recursos involucrados, y asigna a su resolución al o los programadores más la inclusión del Analista si es que se necesitase. Con el visto bueno del cliente la Fase de construcción termina.

En la Figura 4.6 se hace resumen de las actividades y artefactos para la realización de una iteración de la Fase de Construcción

Figura 4.6 Fase de Construcción - Ejemplo de iteración de construcción



4.5.4 Fase de Implantación y Transferencia

Con la construcción del producto finalizado, el Analista confecciona los Manuales de usuario (ver 4.4.8 Manual de Usuario) y el Programador Líder el Manual de instalación (ver 4.4.9 Manual de Instalación). Ambos son revisados por el QA, con su aprobación el Jefe del Proyecto cierra el proyecto guardando toda la documentación generada en el desarrollo para su utilización futura. El Jefe de Proyecto hace entrega al cliente de los siguientes artefactos:

- Informe de Especificación de Requerimientos
- Código fuente y Aplicación
- Descriptor de Alcance, si es que se construyó.
- Documento de Arquitectura de Software, si es que se construyó.
- Manuales de usuario e instalación.

4.6 Matrices RACI de Actividades y Artefactos

Una matriz de asignación de responsabilidades RACI, es un cuadro que muestra el personal asignado a cada actividad en un proyecto y las relaciones entre los integrantes del equipo de proyecto. En cada cuadro de la matriz se identifica el tipo de relación entre cuatro posibles tipos:

- R: Responsable
- A: Aprobador
- C: Consultado
- I: Informado

En la Tabla 4-4 de detalla la actividad de generación de los artefactos en relación a la responsabilidad de los roles.

Tabla 4-4 Matriz RACI de Artefactos

Artefacto/Roles	J. Proyecto	Analista	Arquitecto	Programador	Programador Líder	QA	Tester	Cliente
Generar Descriptor de Alcance	**R**	**I**	**I**	**I**	**I**	**I**	**I**	**CI**
Generar Informe de Especificación de Requerimientos	**A**	**R**	**I**	**I**	**I**	**I**	**I**	**ACI**
Generar Documento de Arquitectura de Software	**A**	**I**	**R**	**CI**	**I**	**I**	**I**	**CI**
Generar Descriptor de Riesgos	**R**	**CI**	**CI**	**CI**	**CI**	**CI**	**CI**	**CI**
Generar Descriptor de Requerimiento	**I**	**R**	**I**	**I**	**I**	**A**	**I**	**I**
Generar Encuesta TAM	**R**	**I**	**I**	**I**	**I**	**I**	**I**	**I**
Generar Revisión de Ajuste	**R**	**I**	**I**	**I**	**I**	**I**	**I**	**ACI**
Generar Manual de Usuario	**A**	**R**	**I**	**I**	**I**	**I**	**I**	**I**
Generar Manual de Instalación	**A**	**I**	**C**	**I**	**R**	**I**	**I**	**I**
Generar Release	**R**	**I**	**I**	**I**	**C**	**I**	**I**	**I**

4.7 Herramientas

4.7.1 Servidor de integración continúa

La integración continua es un modelo informático que consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes. Entendemos por integración la compilación y ejecución de test de todo un proyecto.

Los desarrolladores pueden detectar y solucionar problemas de integración de forma continua, evitando el caos de última hora cuando se acercan las fechas de entrega. Entre las ventajas de disponer de un servidor de integración continua son;

- Disponibilidad constante de una build para pruebas, demos o lanzamientos anticipados.

- La ejecución inmediata de las pruebas unitarias.

- la monitorización continua de las métricas de calidad del proyecto.

4.7.2 Servidor de versiones

Un servidor de versión, es un sistema que facilita la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente específico). Un sistema de control de versiones proporciona:

- Mecanismo de almacenamiento de los elementos que deba gestionar.

- Posibilidad de realizar cambios sobre los elementos almacenados.

- Registro histórico de las acciones realizadas con cada elemento.

4.7.3 Gestor de incidencias

Una incidencia, llevado a la informática, es cualquier evento que no forma parte de la operación normal de un sistema y que causa, o puede causar una interrupción, o una reducción de la calidad del mismo. Un Gestor de Incidencias es una herramienta que permite incorporar las mejores prácticas en la gestión de incidencias. Con un Gestor de Incidencias es posible:

- Registrar la incidencia: quién informa del problema, síntomas, equipo involucrado, etc.

- Clasificar la incidencia y asignar el trabajo a realizar a un grupo de soporte o a un técnico.

- Investigar la causa de la incidencia y compararla con otras incidencias parecidas.

- Documentar la solución, anexar ficheros con información relacionada y cerrar la incidencia.

- Comunicar automáticamente al usuario el estado de su solicitud a través del e-mail y/o portal de soporte.

- Elaborar informes, que ayuden a conocer qué está sucediendo y a mejorar el proceso.

4.7.4 TAM

Si bien el TAM es un modelo, la integración a esta propuesta de metodología lo ha convertido en una herramienta que entrega información útil para medir la aceptación y posibilitar la identificación de correcciones en plena construcción del sistema. Disminuyendo los riesgos de la realización de correcciones en etapas tardías del proyecto.

5 Validación de *Z-ÁGIL* en la Aplicación de un Proyecto

El siguiente capítulo tiene como finalidad poner a prueba esta propuesta metodológica denominada *Z-ÁGIL*. Para ello ZEKE, ha dispuesto su aplicación en el proyecto denominado SCP- Sistema de Certificación de Procesos Ambientales para el Ministerio del Medio Ambiente.

A continuación, se describe a grandes rasgos la problemática a resolver en el proyecto para luego dar paso a la obtención de resultados.

5.1 Descripción General del Proyecto

1. Situación Actual

El Ministerio del Medio Ambiente (en adelante MMA) es una Secretaría de Estado de Chile encargada de colaborar con el Presidente de la República en el diseño y aplicación de políticas, planes y programas en materia ambiental, así como en la protección y conservación de la diversidad biológica y de los recursos naturales renovables e hídricos, promoviendo el desarrollo sustentable, la integridad de la política ambiental y su regulación normativa. También es su rol informar a la ciudadanía del estado del medio ambiente, su evolución y las consecuencias de las acciones que el hombre desarrolla sobre él, con el objeto de crear conciencia y promover el involucramiento de la sociedad en la problemática ambiental.

Una de las áreas que abarca el MMA para cumplir con su misión, es la Certificación de Procesos Ambientales para municipalidades, colegios y jardines.

2. Descripción del Problema

Los programas de certificación tienen como finalidad incentivar las acciones destinadas a difundir la importancia de una cultura para la sustentabilidad, promover los valores y conservación del medio ambiente en materias como el reciclaje, el ahorro energético y de agua, el desarrollo de instrumentos que fomenten la participación de los vecinos, mejorando su desempeño ambiental y la calidad de vida de los habitantes que convergen en un territorio colindante.

En el marco de lo anterior, se requiere contar con un modelo de gestión que ordene y haga costo efectivo la intervención del Estado y actores involucrados en materia de certificación ambiental, a través de diversos mecanismos e instrumentos, tanto para la postulación, proyecto y certificación ambiental de las entidades solicitantes; además, se requiere de información unificada, fidedigna, accesible y pública, sobre los centros certificados medioambientales. Es por ello que se requiere contar con un Sistema de Certificación Ambiental que permita llevar este control y, así mismo, informar a la ciudadanía sobre los proyectos o iniciativas locales para el cuidado del medio ambiente.

3. Objetivos del Negocio

- Permitir realizar reportes automatizados para satisfacer demandas nacionales.
- Responder a las expectativas y demandas de la ciudadanía, poniendo a su disposición la información relativa a los procesos de certificación.
- Fomentar la trasparencia, eficiencia y simplicidad de los procedimientos para la correcta ejecución de los procesos que deben formar parte del sistema de certificación ambiental.
- Obtener índices de gestión para ayudar a la toma de decisiones.

4. Objetivos del Proyecto

- Implementar una solución tecnológica con la capacidad de cumplir con todos los requerimientos planteados por el Ministerio del Medio Ambiente.
- Las entregas comprometidas se realizarán respetando todos los plazos y costos del proyecto.
- Realizar la transferencia tecnológica a los profesionales del Ministerio del Medio Ambiente que permita la correcta operación de la plataforma.
- Respeto y resguardo de la confidencialidad de la información que pueda ser recibida desde el Ministerio del Medio Ambiente y que sea útil en el desarrollo del sistema.

5. Objetivos de Sistema

- Centralización de la información.
- Simplicidad del sistema con interfaces amigables para minimizar errores y mejorar la experiencia de usuario.
- Utilizar los mecanismos de seguridad apropiados para proteger la información y datos de accesos no autorizados (a personas no autorizadas).
- Construir una aplicación robusta, modular y mantenible, que sea escalable, segura, capaz de manejar alta concurrencia.
- Dar cumplimiento a las normativas vigentes del Ministerio del Medio Ambiente.
- Asegurar la calidad del sistema a través de pruebas unitarias y de integración, así minimizar errores de programación.
- Soporte a los browsers más utilizados.

5.2 Planificación y preparativos

El proyecto fue identificado internamente como SCP (Sistema de Certificación de Procesos Ambientales). El proyecto fue estimado en una duración de 6 meses, dándose inicio formal el 01/09/2012 con la reunión de kickoff, en donde se presentó el equipo de trabajo y la manera en cómo se abordaría el proyecto informando que se aplicaría una nueva metodología de trabajo. Se detalló a grandes rasgos las nuevas características que se aplicarían: entregas tempranas y continuas una vez iniciada la construcción y la participación mediante el desarrollo de cuestionarios (aplicación de TAM) para evaluar el sistema.

5.2.1 Planificación y Entregas

Las fechas acordadas de las entregas para este proyecto, se listan en la Tabla 5-1.

Tabla 5-1 Fechas de inicio y término de las fases

Fase	Fecha de inicio estimada	Fecha finalización estimada	Entregable
Concepción	10/09/2012	27/09/2012	Informe de Visión y Alcance del sistema
Exploración	28/09/2012	29/10/2012	Informe de Especificación de Requerimientos
Construcción	30/10/2012	17/03/2013	Entrega del sistema para pruebas finales de aceptación
Impl. y Transf	18/03/2013	27/03/2013	Entrega de documentación final

5.2.2 Conformación del Equipo de Desarrollo

El equipo de desarrollo del proyecto se puede apreciar en la Tabla 5-2.

Tabla 5-2 Equipo de desarrollo del proyecto SCP

Rol	Cantidad
Jefes de Proyecto	1
Analistas	1
Programadores	3 (uno de ellos realizó el rol de Programador Líder)
Arquitecto	1
QA	1
Tester	1

5.2.3 Selección de Artefactos y Actividades a Realizar

En el capítulo de 4.4 Artefactos, se listaron todos los artefactos que se incluyen en *Z-Ágil*. No todos los artefactos son obligatorios, todo dependerá de la naturaleza o complejidad de la solución o por la determinación del Jefe de Proyecto. Para el caso de este proyecto, se utilizaron los artefactos listados en la Tabla 5-3.

Tabla 5-3 Artefactos de *Z-Ágil* utilizados para SCP

Artefacto	**Fase donde se genera**	**Rol encargado**
Descriptor de Alcance	Concepción	Jefe de Proyectos
Informe de Especificación de Requerimientos	Exploración	Analista
Descriptor de Requerimiento	Construcción	Analista
Encuesta TAM	Construcción	Jefe de Proyectos
Revisión de Ajuste	Construcción	Jefe de Proyectos
Release	Construcción	Jefe de Proyectos
Manual de Usuario	Implantación y Transferencia	Analista
Manual de Instalación	Implantación y Transferencia	Programador Líder

5.2.4 Capacitación sobre *Z-Ágil* al Equipo del Proyecto

Antes de dar inicio al proyecto con la *Z-ÁGIL*, se requirió educar a los miembros del equipo sobre sus nuevas responsabilidades y actividades a realizar bajo este nuevo marco de desarrollo. Para ello, se hicieron una serie de reuniones de capacitación a los miembros. En las primeras reuniones se les detalló los problemas existentes con el actual modelo de desarrollo, las encuestas aplicadas a cada área y los resultados obtenidos, introduciendo el tema de las metodologías ágiles como alternativas viables que podrían dar solución a nuestros problemas. Con esto terminado, se presentó *Z-Ágil* como alternativa de solución, se explicó el ciclo de vida, los roles y artefactos más las actividades y responsabilidades particulares de cada rol, poniendo énfasis en la naturaleza independiente e innovadora de esta metodología, ya que si bien, está inspirada en prácticas ágiles la integración de TAM marca la diferencia.

5.3 Desarrollo del Proyecto SCP con Z-ÁGIL

5.3.1 Desarrollo de la Fase de Concepción

Una vez iniciado el proyecto con la reunión de kickoff, comenzó la Fase de Concepción, en esta fase se conformó el equipo que trabajó en el proyecto, se definió la arquitectura, herramientas de análisis y programación.

El equipo de programación estuvo conformado por 3 personas (ver Tabla 5-2), todos programadores con experiencias en la arquitectura y herramientas que se utilizaron. El jefe de proyecto seleccionó de entre ellos a quien sería el Programador Líder, dándole a conocer a este sus actividades y responsabilidades con respecto a la evaluación de los trabajos de sus pares una vez iniciada la construcción en la Fase de Construcción.

Como se contaba con un equipo de programación con experiencia en la arquitectura, el Jefe de Proyecto desestimó la realización de las actividades asociadas a la arquitectura, no se realizó ni el documento de arquitectura de software y ni las reuniones de inducción de la arquitectura. Además del manejo de la arquitectura por parte de los programadores, dentro de la empresa existe variada documentación sobre ésta lo cual legitimó aún más la decisión.

El Jefe de Proyecto, construyó el Descriptor de Visión y Alcance del Sistema, entregando copias a cada miembro y enviando una copia a las contrapartes del proyecto.

Con la conformación del equipo terminada más el documento de Visión y Alcance del Sistema, finalizaron las actividades de la Fase de Concepción, concluyendo esta etapa.

5.3.2 Desarrollo de la Fase de Exploración

Con la Fase de Concepción terminada, se dio inicio a la Fase de Exploración.

Como se puede apreciar en la Tabla 4-4, es en esta fase donde se realizaron las actividades necesarias para capturar, analizar y documentar los requerimientos de los usuarios.

El Jefe de Proyecto planificó un total de 8 reuniones de levantamiento, todas realizadas en las dependencias de la institución (Ministerio del Medio Ambiente), en cada una de ellas se contó con la participación del Jefe de Proyecto y Analista de ZEKE más la presencia de las contrapartes técnicas y de negocio, Jefes de Proyecto y usuarios finales del MMA respectivamente.

El analista confeccionó el Informe de Especificación de Requerimiento, documento obligatorio y único artefacto de *Z-Ágil* para esta fase, descrito como artefacto en 4.4.2 Informe de Especificación de Requerimientos. Este documento fue creado en las fechas estipuladas y entregado a las contrapartes para su revisión. En este documento quedó de manifiesto todos los requerimientos de los usuarios a ser implementados, agrupados en 4 módulos, ver Figura 5.1.

Figura 5.1 Módulos del SCP

1. Módulo Diseñador

Módulo que posibilita la creación y mantención de Procesos de Certificación además de otras funciones como la gestión de plantillas de mensajes.

2. Módulo Certificación

Módulo que permite a un centro certificarse en algún Proceso de Certificación mediante la ejecución de diversas actividades. La participación abarca desde usuarios interesados en la certificación como de usuarios internos del Ministerio del Medio Ambiente.

3. Módulo de Administración

Módulo destinado a la gestión de usuarios del sistema y a la obtención de información sobre las diversas interacciones que se realizan dentro de éste.

4. Módulo de Notificaciones

Módulo que posibilita la comunicación interna y externa de los usuarios del sistema mediante el envío y recepción de mensajes.

Siguiendo el proceso de interacción para esta fase (Figura 4.5), el documento fue revisado y validado por el QA asignado al proyecto, sin antes, realizar una serie de observaciones a los modelos que en el Informe se generan: Diagramas de estado, Modelo de Dominio y Modelo de Caso de uso. Cabe destacar, que el Rol de QA ya existía en el antiguo modelo de desarrollo, sin embargo, las actividades que este rol realizaba en él, era solo de testing, actividades que en *Z-Ágil* están asociadas al rol de Tester. Esto implicó que la persona encargada de realizar de QA recibiera clases sobre los distintos modelos UML que están integrados en el Informe, con énfasis en la interacción entre ellos y en la búsqueda de inconsistencias con respecto a las líneas de validaciones y cambios de estado.

Con el término de la confección del documento y las aprobaciones internas ejecutadas, el informe fue enviado a las contrapartes del Ministerio para su aprobación. Una vez recibido el informe, el MMA realizó observaciones que fueron acogidas e incorporadas al Informe. Esta dinámica de recepción de observaciones y de corrección, produjo un atraso de 20 días en la finalización de esta fase, lo que llevó a la recalendarización del proyecto. Esta situación pudo haberse manejado de manera distinta, por ejemplo, disminuyendo la recepción de observaciones, excluyendo aquellas que no se ajustaban al alcance del proyecto, para ello, se debió haber utilizado el documento de Descriptor de Visión y Alcance, generado en la Fase de Concepción. Este error puede explicarse por la falta de experiencia del Jefe de Proyecto en el manejo de *Z-Ágil.*

A pesar de que la fase no finalizara en lo planificado, el Informe de Especificación de Requerimientos fue aprobado por las contrapartes. Con la aprobación formal, se dio por concluida la Fase de Exploración.

5.3.3 Desarrollo de la Fase de Construcción

Según lo descrito en el Informe de Especificación de Requerimientos, el proyecto fue divido en módulos (ver punto anterior). Cada módulo contenía un conjunto de requerimientos estrechamente relacionados entre sí (requerimiento del módulo) y distantes de otros requerimientos pertenecientes a otros módulos.

Siguiendo el modelo del proceso para esta fase (Figura 4.6), la primera actividad a desarrollar, correspondió a la selección del grupo de requerimientos a construir. Sin embargo, los requerimientos ya se encontraban delimitados al ser reunidos por módulos en el Informe de Especificación de requerimientos, bajo esto, se seleccionaron los Módulos de Diseño y Certificación como primera iteración de construcción, seguido de los Módulos de Admi9nistración y Notificación como segunda iteración.

Por cada conjunto de módulos se debió desarrollar las actividades descritas por el ciclo de vida de *Z-ÁGIL* para esta fase, las actividades para la fase de construcción), estas son:

a) Construir/modificar descriptores de requerimientos

b) Validar descriptores de requerimientos

c) Asignar descriptores de requerimientos

d) Construir o modificar construcción de requerimientos

e) Revisar código implementación de requerimientos

f) Probar implementación de requerimientos

g) Disponibilizar implementación de requerimientos

h) Confeccionar encuestas TAM

i) Analizar resultados TAM y confeccionar Doc. Revisión de Ajuste

Como se puede apreciar en el listado, la primera actividad a ser realizada correspondió al desarrollo de los descriptores de requerimiento, cada descriptor, confeccionado por el Analista, reflejó un requerimiento a implementar del módulo, cada descriptor fue validado por el QA del proyecto, posterior asignación de cada descriptor a los programadores. Las implementaciones también fueron probadas y validadas, por el Programador Líder y el Tester. Con las implementaciones probadas, el Jefe del Proyecto, construyó las encuestas TAM y disponibilizó el ambiente para las pruebas efectuadas por las contrapartes, el resumen de los resultados de las evaluaciones y los ajustes que se realizaron se muestran en los siguientes puntos.

5.3.3.1 Preparación de encuestas TAM

Como se describió en el punto anterior y siguiendo el flujo de pasos para esta fase de construcción, se realizaron 2 agrupaciones de requerimientos, el primero, los requerimientos asociados a los Módulos de Diseño y Certificación y segundo los requerimientos pertenecientes a los Módulos de Administración y Notificaciones. Para cada uno de estos agrupamientos se realizó la encuesta mostrada en el ANEXO P:Plantilla de Encuesta TAM.

Para obtener las mediciones correspondientes que entrega TAM, cada una de esas preguntas debió reflejar los aspectos de Utilidad, Usabilidad, Actitud e Intención de uso. Las preguntas fueron tomadas de fuentes oficiales y codificadas para realizar una mejor manipulación, ver ANEXO O:Constructos e ítems de TAM.

Para realizar la valoración de los ítems de cada variable, se utilizó la modalidad de escala de Likert con 5 puntos. La escala Likert es una escala psicométrica comúnmente utilizada en cuestionarios y es la escala de uso más amplio en encuestas para la investigación. Al responder a una pregunta de un cuestionario elaborado con la técnica de Likert, se especifica el nivel de acuerdo o desacuerdo con una declaración, para este caso los usuarios debieron encerrar en un círculo el valor que estimaban correcto. Los niveles se aprecian en la Tabla 5-4.

Tabla 5-4 Definición de los niveles para la escala de Likert de 5 puntos

Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
1	2	3	4	5

Las encuestas fueron aplicadas dentro del Ministerio del Medio a los usuarios finales del SCP, estos usuarios corresponden a dos usuarios por región del país, donde 28 usuarios tienen el rol de usuarios regional y 2 usuarios poseen el rol administrador dentro del SCP. El resumen de la aplicación del estudio se observa en la Tabla 5-5.

Tabla 5-5 Detalle del estudio

Muestra	30 individuos
Unidad muestral	Usuarios finales del SCP
Ámbito de estudio	Ministerio del Medio Ambiente, Santiago, Chile.
Método de recogida de información	Realización de encuesta
Fecha de trabajo de campo	Diciembre 2012 - Enero 2013

5.3.3.2 Planteamiento de las hipótesis de TAM

La literatura abordada permite suponer que el TAM es un modelo válido para explicar el comportamiento de aceptación de la tecnología en este caso para el SCP. La Teoría de Aceptación de Tecnología permite establecer las hipótesis mostradas en la Tabla 5-6.

Tabla 5-6 Hipótesis de TAM. (Davis, 1989)

H1	La Facilidad Percibida de uso influye sobre la Actitud hacia el Uso
H2	La Facilidad Percibida de Uso influye sobre la Utilidad Percibida
H3	La Utilidad Percibida de Uso influye sobre la Actitud hacia el uso
H4	La Utilidad Percibida de Uso influye sobre la Intención de usar
H5	La Actitud hacia el Uso influye sobre la Intención de Usar

La Figura 5.2 muestra el modelo TAM con las hipótesis interrelacionando. El modelo se puede explicar como sigue: aumentar la percepción de facilidad de uso aumenta la percepción de utilidad de las TI, y al aumentar estas dos percepciones aumenta la Actitud de uso. Un aumento de la Actitud de uso está directamente relacionado con la Intención de Uso y con ello el uso real del sistema.

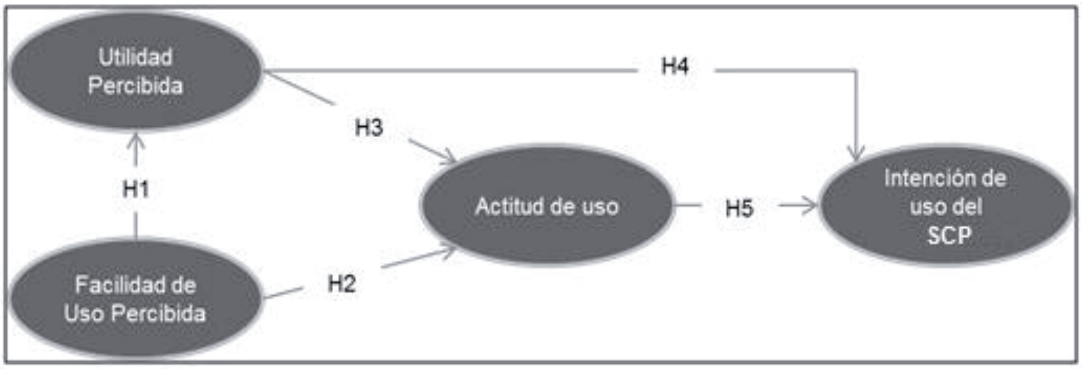


Figura 5.2 Numeración de hipótesis de los autores. Fuente: (Davis, 1989)

5.3.3.3 Generalidades del método estadístico PLS - Partial Least Square

Los modelos de ecuaciones estructurales (SEM, por su nombre en inglés, Structural Equation Models) permiten la estimación de cadenas de relaciones causales, definidas teóricamente, entre variables no observables o latentes a partir de métodos estadísticos. Las variables latentes a su vez son formadas o reflejadas por variables observables o manifiestas provenientes de encuestas. De las variables manifiestas se extrae información para la estimación de las variables latentes, la puntuación obtenida en las variables latentes es utilizada en la estimación de las relaciones causales propuestas. De esta manera es posible cuantificar variables no observables y estimar tanto la dirección como la magnitud del impacto con otras variables no observables.

PLS (Partial Least Square) es uno de los enfoques para la estimación de SEM. Su objetivo es la predicción en el análisis causal. Al usar PLS, se analiza cuan bien se relacionan los indicadores con su constructo y se verifica si las relaciones "hipotetizadas" en el modelo teórico, se cumplen en el trabajo empírico. Para efectos de los cálculos se utilizó esta técnica mediante el uso del software SmartPLS Versión: 2.0.M3 (http://www.smartpls.de/). SmartPLS es una aplicación gráfica para el modelado de caminos con variables latentes (LVP) que permite la creación y estimación de modelos PLS, en ella se modelaron los path y las relaciones entre las variables y los ítems a valorar, como se muestra en la Figura 5.3.

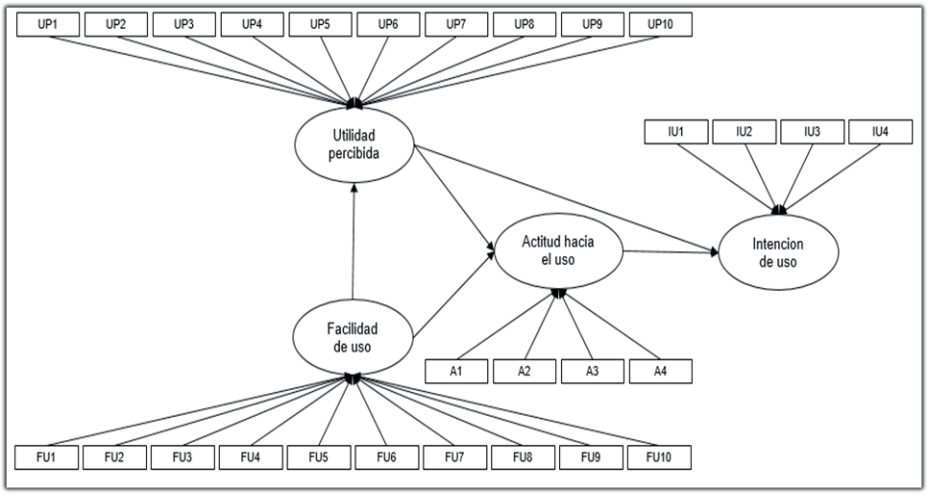


Figura 5.3 Plantilla de modelo TAM en SmartPLS

5.3.3.4 Procedimiento para verificar la fiabilidad y validez de los cálculos

El procedimiento para la evaluación de la confiabilidad y la validez del modelo se realiza dividiendo el modelo en dos modelos: el modelo de medida y el modelo estructural.

El modelo de medida es aquel que representa las relaciones entre las variables manifiestas y sus indicadores, mientras que el modelo estructural comprende las relaciones entre las variables latentes, ver Figura 5.4.

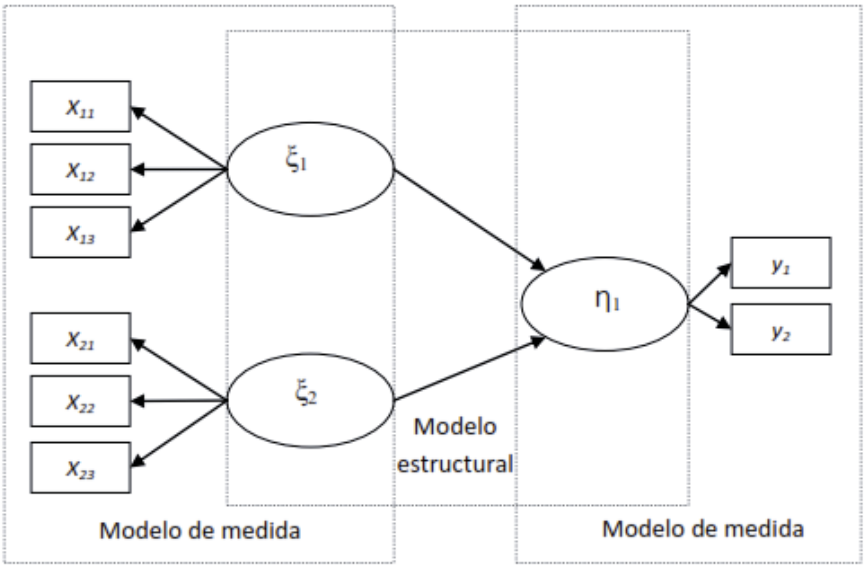


Figura 5.4 Modelo estructural y modelo de medida en PLS

El procedimiento consiste en analizar el modelo de medida, revisando las correlaciones entre la variable latente y sus indicadores, la varianza de los indicadores extraída por la variable latente, así como la correlación de los indicadores con las variables latentes a las que no pertenecen. Una vez satisfechos lo criterios de confiabilidad y validez del modelo de medida, entonces es posible analizar el modelo estructural. En el modelo estructural se verifica que la dirección y el tamaño estimado corresponda al esperado teóricamente, la significancia estadística de los coeficientes obtenidos en las relaciones estructurales, así como el nivel alcanzado en la explicación del fenómeno.

A continuación se describirán los métodos utilizados para analizar la confiabilidad y la validez del modelo de medida:

a) ***Fiabilidad individual del ítem*:** Se valora examinando las cargas (o correlaciones) de los indicadores con su respectivo constructo. Para considerar que las relaciones son fuertes, es decir, para aceptar a un indicador como integrante de un constructo, las mediciones de estas cargas deben ser superiores a 0.7 (Bagozzi & Yi, 1988).

b) ***Fiabilidad de constructo:*** Comprueba la consistencia de todos los indicadores al medir el concepto. Se valora inspeccionando el Alpha de Cronbach. El Alpha de Cronbach determina la consistencia interna de una escala analizando la correlación media de una variable con todas las demás que integran dicha escala. Cuanto más se acerque el coeficiente a la unidad, mayor será la consistencia interna de los indicadores en la escala evaluada. Se considera que 0.7 es un valor aceptable para el Alfa de Cronbach (Churchill, 1979).

c) ***Validez convergente:*** Permite determinar si los indicadores de los constructos miden realmente el mismo concepto, por lo cual se requiere que estén altamente correlacionados. Para determinar la validez convergente se utiliza la varianza extraída media (AVE), que expresa la cantidad de la varianza que un constructo obtiene de sus indicadores. Se recomienda que su valor sea superior a 0.5 (Fornell & Larcker, 1981). con lo que se establece que más del 50 por ciento de la varianza del constructo es debida a sus indicadores.

En la Tabla 5-7Tabla 5-8 se presentan los criterios de aceptación para la evaluación del modelo de medida.

Tabla 5-7 Criterios de aceptación para evaluación modelo de medida

	Fiabilidad individual del ítem	**Fiabilidad del constructo**	**Validez convergente**
Criterio	$> 0{,}7$	$\alpha > 0{,}7$	$AVE > 0{,}5$

Al verificarse satisfactoriamente la confiabilidad y validez del modelo de medida se realiza la valoración del modelo estructural, gracias a este estudio se consigue obtener los parámetros suficientes para poder aceptar o no las hipótesis planteadas. Para poder admitir o no las hipótesis se deben tener en cuenta los siguientes parámetros:

a) ***Coeficiente de varianza explicada, R^2.*** Indica la cantidad de varianza del constructo que se explica por medio de las variables que lo predicen. Se establecen como valores adecuados de la varianza explicada aquellos iguales o mayores que 0.1, valores inferiores indican un bajo nivel predictivo de la variable latente dependiente.

b) ***Coeficiente path, β.*** Indica en qué medida las variables predictivas contribuyen a la varianza explicada de las variables dependientes, evalúa el nivel de significancia de las relaciones entre los constructos. Para ser considerados prácticamente significativos, los coeficientes deben alcanzar al menos un valor de 0.2, e idealmente situarse por encima de 0.3.

c) ***T-Student*:** Determinar la estabilidad de las estimaciones. Se calcula mediante la técnica de remuestreo Bootstrap. Los resultados de esta prueba permiten comprobar la firmeza de las hipótesis planteadas en los modelos. Para ello los valores del coeficiente $T\ de\ Student$ que se obtengan deben ser mayores que el valor del estadístico $T\ de\ Student$ de infinitos grados de libertad.

En la Tabla 5-8 se presentan los criterios de aceptación para la evaluación del modelo estructural.

Tabla 5-8 Criterios de aceptación para evaluación modelo estructural

	R^2	β	$T\ de\ Student$
Criterio	$> 0{,}1$	$> 0{,}2$	$T > 1.65\ para\ p < 0{,}10$ $T > 1{,}96\ para\ p < 0{,}05$ $T > 2.58\ para\ p < 0{,}01$

5.3.3.5 Evaluación de TAM y ajustes los Módulos de Certificación y Diseño

El Módulo Diseñador tiene como propósito brindar todas las herramientas necesarias para que los usuarios administradores del SCP sean capaces de crear y mantener procesos de Certificación, procesos por los cuales los centros son partícipes mediante el Módulo de Certificación con el fin de obtener las certificaciones ambientales.

La Tabla 5-9 muestra la media y desviación estándar de los ítems de las variables del modelo de investigación para el caso de este módulo.

Tabla 5-9 Media y Desv. Tip. para los Ítems de los Módulos de Certificación y Diseño

Ítem	Media	Desv. Típ.
UP1	4,07	0,85
UP2	4,3	0,69
UP3	3,77	0,84
UP4	3,33	0,94
UP5	3,73	0,96
UP6	4,17	0,93
UP7	4,2	0,79
UP8	4,13	0,88
UP9	3,97	0,95
UP10	4,13	0,76

Ítem	Media	Desv. Típ.
FU1	3,7	0,97
FU2	3,9	1,04
FU3	3,17	1,16
FU4	3,37	1,2
FU5	3,47	1,02
FU6	3,43	0,92
FU7	3,2	0,95
FU8	3,6	0,95
FU9	3,73	0,89
FU10	4,03	0,84

Ítem	Media	Desv. Típ.
A1	3,9	0,87
A2	3,83	0,82
A3	4	0,86
A4	4	0,82

Ítem	Media	Desv. Típ.
IU1	4,03	0,75
IU2	4,07	0,77
IU3	4,17	0,73
IU4	4,07	0,81

Como se indica en el capítulo anterior, el primer paso del proceso para valorar la fiabilidad y validez del modelo es evaluar el modelo de medida, para luego probar las relaciones estructurales entre las variables latentes en el modelo estructural. En tal sentido se ejecutó el programa SmartPLS; para realizar esta evaluación, se determinaron los índices que miden la fiabilidad de cada indicador con su constructo (cargas factoriales), la fiabilidad del constructo (confiabilidad compuesta) y la validez convergente (varianza extraída media AVE). Los valores se pueden observar en la Tabla 5-10.

Tabla 5-10 Análisis del modelo de medida - Módulos de Certificación y Diseño

Constructo	Indicadores	Carga factorial	Confiabilidad compuesta	AVE
Actitud de uso	A1	0,898	0,913	0,725
	A2	0,805		
	A3	0,909		
	A4	0,787		
Facilidad de uso	FU1	0,732	0,94	0,616
	FU10	0,679		
	FU2	0,854		
	FU3	0,673		
	FU4	0,802		
	FU5	0,860		
	FU6	0,909		
	FU7	**0,496**		
	FU8	0,851		
	FU9	0,897		
Intención de uso	IU1	0,744	0,911	0,721
	IU2	0,925		
	IU3	0,926		
	IU4	0,786		
Utilidad Percibida	UP1	0,794	0,912	0,513
	UP10	0,745		
	UP2	**0,467**		
	UP3	0,751		
	UP4	0,775		
	UP5	0,712		
	UP6	0,690		
	UP7	0,729		
	UP8	0,826		
	UP9	**0,602**		

Se eliminaron los indicadores denominados FU7, UP2 y UP9 cuyas cargas factoriales estuvieron muy por debajo a 0.7 (ver Tabla 5-7). Se realizó el cálculo nuevamente sin los indicadores descritos. En Tabla 5-11 se pueden apreciar que los valores obtenidos en los índices AVE, Confiabilidad Compuesta, Alfa de Cronbachs y Comunidad reflejan que los ítems consultados son de razonable fiabilidad, superando los criterios de aceptación de la Tabla 5-7, en donde: el cálculo del Alfa de Cronbach fuera superior a 0,7, la Confiabilidad Compuesta superior a 0,7 y la Varianza Extraída Media (AVE) superar a 0,5.

Tabla 5-11 Análisis del modelo de medida ajustado - Módulos de Certificación y Diseño

Variable	AVE	Confiabilidad Compuesta	R^2	Alfa de Cronbach	Comunalidad
Utilidad Percibida	0,588	0,919	0,131	0,902	0,588
Facilidad de Uso Percibida	0,663	0,946	0,000	0,935	0,663
Actitud de Uso	0,725	0,913	0,712	0,872	0,725
Intención de Uso	0,726	0,913	0,224	0,881	0,726

Una vez garantizadas la fiabilidad y la validez del modelo de medida, se procedió a contrastar las hipótesis. La Tabla 5-12 muestra el resultado del proceso de *Bootstrap* para el cálculo de la fiabilidad de los caminos estructurales. Como se puede apreciar, al observar los valores de β, los $Estadísticos\ T$ y los R^2 superan los valores de referencia de los criterios de aceptación de la Tabla 5-8.

Tabla 5-12 Coeficientes de las relaciones - Módulos de Certificación y Diseño

Relación	β	$Estadístico\ T$	Significancia
Facilidad de uso → Actitud de Uso	0,703	45,434	***
Facilidad de uso → Utilidad Percibida	0,361	16,716	***
Utilidad Percibida → Actitud de Uso	0,277	12,447	***
Utilidad Percibida → Intención de Uso	-0,316	6,714	***
Actitud de Uso → Intención de Uso	0,558	18,098	***
$* p < 01, \quad ** p < 0.05, \quad *** p < 0.001$			

Los resultados confirman todas las relaciones que se establecieron en el modelo de investigación, confirmando a su vez las hipótesis establecidas, ver Tabla 5-13.

Tabla 5-13 Contraste de las hipótesis - Módulos de Certificación y Diseño

Hipótesis	Contraste
H1:La Facilidad Percibida de uso influye sobre la Actitud hacia el Uso	Soportada
H2:La Facilidad Percibida de Uso influye sobre la Utilidad Percibida	Soportada
H3:La Utilidad Percibida de Uso influye sobre la Actitud hacia el uso	Soportada
H4:La Utilidad Percibida de Uso influye sobre la Intención de usar	Soportada
H5:La Actitud hacia el Uso influye sobre la Intención de Usar	Soportada

La representación gráfica del modelo contrastado se puede observar en la Figura 5.5.

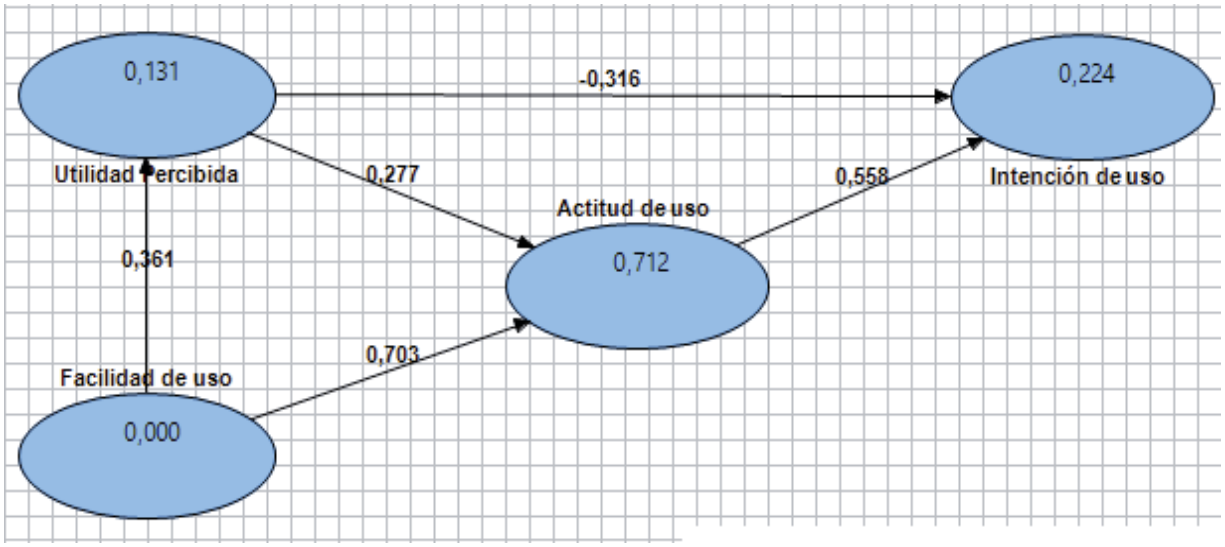


Figura 5.5 Modelo TAM contrastado en Módulos de Certificación y Diseño

Realizando el análisis del modelo, se puede interpretar que los usuarios del Ministerio del Medio Ambiente tienen una alta intención de uso del sistema, esto se evidencia puesto que la Intención de uso es explicada significativamente por la Actitud de Uso y éste a su vez es explicado significativamente por sus antecedentes, Utilidad Percibida y Facilidad de Uso, todos los coeficientes con valores por encima de 0,2, ver los criterios de aceptación en Tabla 5-8.

La Facilidad de Uso se constituye como el principal antecedente de la Actitud de Uso, por lo tanto, podemos determinar con certeza que este módulo posee características de diseño que hacen del sistema fácil de usar y agradable de interactuar por los usuarios finales que influyeron sobre la Actitud de Uso. Punto importante a destacar refiere al hecho que el Módulo Diseñador en conjunto con el Módulo de Certificación, representan las principales funcionalidades del SCP, por tanto, contar con los excelente valores obtenidos en la aplicación de TAM en una primera iteración correspondió a un acierto en los procesos de análisis y diseño contenidos en la Fase de Exploración de *Z-Ágil.*

De igual manera, la Facilidad de Uso explica la Utilidad Percibida con un nivel de significancia menor a las otras relaciones pero igualmente válido.

A pesar de que las variables predichas explican a las variables estimadas de manera significativa, la Utilidad Percibida ejerció una influencia inversa hacia la Actitud de Uso, valor -0,3, esto significa que un aumento en la Utilidad Percibida disminuye la Intención de Uso, esta relación requirió de un análisis mayor ¿existieron características del sistema que provocaron que sucediera esto o más bien se debió a otros factores externos no medidos? Para responder este cuestionamiento se realizó una reunión con los usuarios finales a modo de foro abierto de discusión donde se debatió este punto, se llegó a la conclusión que los usuarios finales si bien, percibían que este módulo del sistema era de fácil uso, no percibían utilidad en él, esto se debió principalmente que en sus procesos diarios de trabajo, realizan el trabajo de guiar y evaluar a los centros que participan de alguna certificación ambiental como también trabajan en la confección de nuevos procesos de certificación ambientales. Estos trabajos los realizan de manera eficiente y sin contratiempos, tomando este antecedente, es posible culpar a la

resistencia al cambio como el fenómeno que podría explicar esta situación. La mayoría de las empresas se sienten cómodas realizando negocios, tal y como siempre lo han realizado. Atravesar el umbral hacia el cambio de los procesos internos o formas nuevas de hacer negocios, producto de las innovaciones, coloca a estas organizaciones en un área de incomodidad que no siempre están dispuestas a afrontar, bien por temor a que las cosas no salgan igual, o peor aún, que salgan mal (Tapscott & Caston, 2000).

Las generaciones recientes, tienen una gran facilidad para adoptar el uso de las tecnologías, puesto que prácticamente han nacido disfrutando de sus beneficios. Estas generaciones jóvenes tienen una considerable brecha tecnológica con respecto a las generaciones mayores, la brecha se traduce principalmente en la actitud para enfrentar las nuevas tecnologías y con ello los cambios. En el caso de los usuarios finales del SCP, es posible encasillar en esta última generación, reacia al cambio. Sin embargo, para efectos de la aplicación y validación de *Z-Ágil*, la resistencia al cambio corresponde a un factor fuera del manejo y alcance de esta metodología.

5.3.3.6 Evaluación de TAM y ajustes a los Módulos de Administración y Notificaciones

El Módulo de Administración tiene como finalidad la gestión de usuarios del sistema como la obtención de información sobre las diversas interacciones que se realizan dentro de éste a modo de auditoría. En cambio el Módulo de Notificación posibilita la comunicación interna y externa de los usuarios del sistema mediante el envío y recepción de mensajes.

La Tabla 5-14 muestra la media y desviación estándar de los ítems de las variables del modelo de investigación para el caso de estos módulos.

Tabla 5-14 Media y Desv. Típ. para los Ítems de los Módulos de Administración y Notificaciones

Ítem	Media	Desv. Típ.
UP1	4,4	0,61
UP2	4,4	0,66
UP3	3,97	0,66
UP4	3,67	0,94
UP5	4	0,89
UP6	4,4	0,8
UP7	4,3	0,69
UP8	4,4	0,71
UP9	4,07	0,96
UP10	4,37	0,75

Ítem	Media	Desv. Típ.
FU1	3,8	0,95
FU2	4,03	0,91
FU3	3,3	1,13
FU4	3,5	1,15
FU5	3,6	0,95
FU6	3,57	0,84
FU7	3,2	0,95
FU8	3,73	0,85
FU9	3,73	0,89
FU10	4,1	0,83

Ítem	Media	Desv. Típ.
A1	4,2	0,79
A2	4,1	0,75
A3	4,3	0,74
A4	4,2	0,75

Ítem	Media	Desv. Típ.
IU1	4,17	0,73
IU2	4,2	0,75
IU3	4,27	0,73
IU4	4,2	0,79

De la misma manera como se trabajó en la iteración de construcción anterior para la valoración de la fiabilidad y validez del modelo, fue evaluar el modelo de medida, para luego probar las relaciones estructurales entre las variables latentes en el modelo estructural. Se ejecutó el programa SmartPLS, para realizar esta evaluación, se determinaron los índices que miden la fiabilidad de cada indicador con su constructo (cargas factoriales), la fiabilidad del constructo (confiabilidad compuesta) y la validez convergente (varianza extraída media AVE). Los valores se pueden observar en la Tabla 5-15.

Tabla 5-15 Análisis del modelo de medida - Módulos de Administración y Notificaciones

Constructo	Indicadores	Carga factorial	Confiabilidad compuesta	AVE
Actitud de Uso	A1	0,86	0,889	0,668
	A2	0,74		
	A3	0,88		
	A4	0,78		
Facilidad de Uso Percibida	FU1	0,78	0,916	0,537
	FU10	0,67		
	FU2	0,8		
	FU3	0,6		
	FU4	0,79		
	FU5	0,83		
	FU6	0,89		
	FU7	**0,25**		
	FU8	0,82		
	FU9	0,69		
Intención de Uso	IU1	0,77	0,912	0,723
	IU2	0,92		
	IU3	0,92		
	IU4	0,77		
Utilidad Percibida	UP1	0,76	0,853	0,388
	UP10	0,75		
	UP2	**0,29**		
	UP3	**0,4**		
	UP4	0,71		
	UP5	0,68		
	UP6	0,71		
	UP7	**0,41**		
	UP8	0,81		
	UP9	**0,42**		

Se ajustó el modelo de medida eliminando aquellos indicadores que poseían una carga factorial menor a 0,7 (ver Tabla 5-7) por ser poco significativos respecto al constructo dado. Por esta razón, se suprimieron los indicadores denominados FU7, UP2, UP3, UP7 y UP9 cuyas cargas factoriales estuvieron muy por debajo a 0.7.

Se realizó el cálculo nuevamente sin los indicadores descritos, en Tabla 5-16 se pueden apreciar que los valores obtenidos en los índices AVE, Confiabilidad Compuesta, Alfa de Cronbachs y Comunidad reflejan que los ítems consultados son de razonable fiabilidad, superando los criterios de aceptación de la Tabla 5-7, en donde: el cálculo del Alfa de Cronbach fuera superior a 0,7, la Confiabilidad Compuesta superior a 0,7 y la Varianza Extraída Media (AVE) superar a 0,5.

Tabla 5-16 Análisis del modelo de medida ajustado - Módulos de Administración y Notificaciones

Variable	AVE	Confiabilidad Compuesta	Alfa de Cronbach	R^2	Comunalidad
Utilidad Percibida	0,576	0,890	0,857	0,1	0,576
Facilidad de Uso Percibida	0,589	0,927	0,914	0,0	0,589
Actitud de Uso	0,668	0,889	0,833	0,5	0,668
Intención de Uso	0,721	0,911	0,879	0,2	0,721

Una vez garantizadas la fiabilidad y la validez del modelo de medida, se procedió a contrastar las hipótesis. La Tabla 5-17 muestra el resultado del proceso de *Bootstrap* para el cálculo de la fiabilidad de los caminos estructurales. Como se puede apreciar, al observar los valores de β, los $Estadísticos\ T$ y los R^2 superan los valores de referencia de los criterios de aceptación de la Tabla 5-8.

Tabla 5-17 Coeficientes de las relaciones - Módulos de Administración y Notificaciones

Relación	β	$Estadístico\ T$	Significancia
Facilidad de uso → Actitud de Uso	0,552	21,040	***
Facilidad de uso → Utilidad Percibida	0,224	8,259	***
Utilidad Percibida → Actitud de Uso	0,348	14,237	***
Utilidad Percibida → Intención de Uso	0,162	5,223	***
Actitud de Uso → Intención de Uso	0,370	9,527	***
$* p < 01\ ** p < 0.05\ *** p < 0.001$			

Los resultados confirman todas las relaciones que se establecieron en el modelo de investigación, confirmando a su vez las hipótesis establecidas, ver Tabla 5-18.

Tabla 5-18 Contraste de las hipótesis - Módulos de Administración y Notificaciones

Hipótesis	Contraste
H1:La Facilidad Percibida de uso influye sobre la Actitud hacia el Uso	Soportada
H2:La Facilidad Percibida de Uso influye sobre la Utilidad Percibida	Soportada
H3:La Utilidad Percibida de Uso influye sobre la Actitud hacia el uso	Soportada
H4:La Utilidad Percibida de Uso influye sobre la Intención de usar	Soportada
H5:La Actitud hacia el Uso influye sobre la Intención de Usar	Soportada

La representación gráfica del modelo contrastado se puede observar en la Figura 5.6.

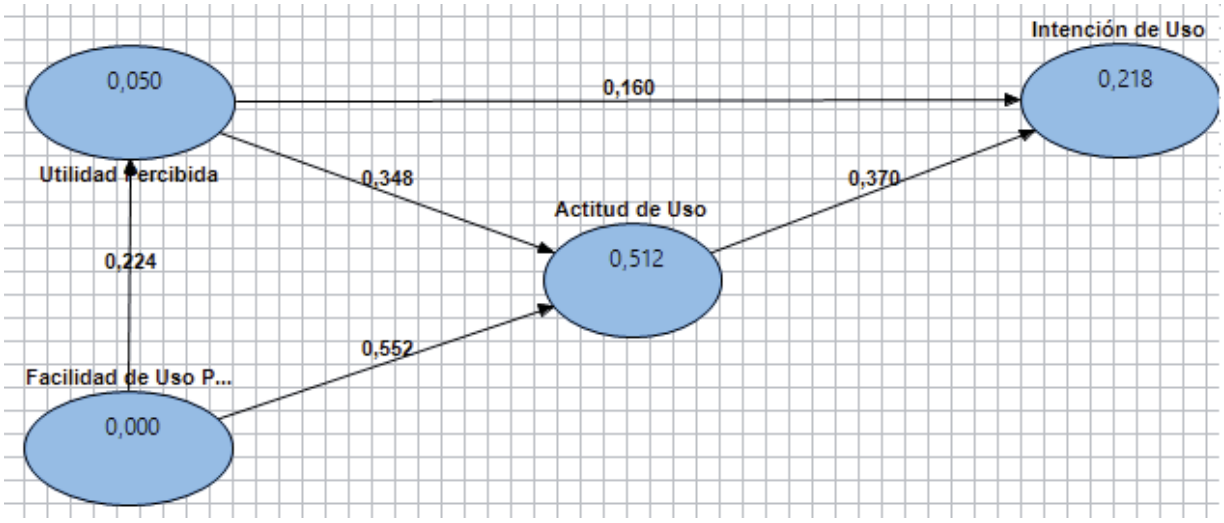


Figura 5.6 Modelo TAM contrastado en Módulo de Administración y Notificaciones

A diferencia de los resultados obtenidos en los módulos de Diseño y Certificación, el análisis del modelo para este módulos arrojó que los usuarios del Ministerio del Medio Ambiente tienen una Intención de Uso normal, esto es no sobresaliente, tomando como antecedente la Actitud de Uso, esto se evidencia puesto que la Actitud de Uso explica a la Intención de Uso con un valor de 0.370, esto se considera significativo al situarse por encima del valor 0.2 establecido en Tabla 5-8.

Con respecto a la Actitud de Uso, ambos antecedentes directos, la Facilidad de Uso Percibida y la Utilidad Percibida, explican su relación, es decir, ambos antecedentes ejercen una relación significativa sobre la Actitud. Punto importante a destacar es la relación que ejerce la Facilidad de Uso sobre la Actitud de Uso, este valor es el de mayor significancia en el modelo, esto puede interpretarse como que la usabilidad del sistema ejerce una alta influencia sobre la Actitud de Uso, es decir, el sistema cuenta con características de diseño que incitan a los usuarios a obtener una predisposición mayor a la utilización del sistema.

De igual manera, la Facilidad de Uso explica la Utilidad Percibida con un nivel de significancia menor a las otras relaciones pero igualmente valido, >0,2.

Por tanto, todas las relaciones que se reflejan en el modelo son significativas, cada variable predicha explica a las variables dependientes, todas a excepción de la Utilidad Percibida sobre la Intención de Uso.

Al igual que la iteración de construcción anterior, se realizó una reunión con los usuarios finales para discutir los resultados obtenidos, específicamente la relación poca significativa entre la Utilidad Percibida y la Intención de Uso. En dicha reunión se logró establecer que algunas funcionalidades del sistema requerían contar con características adicionales que facilitaran la labor de los usuarios administradores en la plataforma delegando al sistema tareas repetitivas, adicionalmente, se agregaron características no consideradas en la Fase de Exploración. En la Tabla 5-19 se puede apreciar una muestra de requerimientos que sufrieron cambios.

Tabla 5-19 Muestra de Req. a ajustar en Módulos de Administración y Notificaciones

Nro.	Módulo	Requerimiento	Propósito	Ajustes
1	Notificaciones	Escenario: Nuevo mensaje a usuarios Postulante	Automatizar la alerta de avisos de inicio de etapa a los usuarios que participan de los procesos de certificación	5,3 y 1 día antes que la fecha de inicio de la Etapa se cumpla, crear un mensaje con estado No leído y enviarlo vía mail e interno en el sistema a todos los usuarios que tengan asociado el Proceso de Certificación y que se encuentren con la cuenta activa. Este envío incluye a los usuarios Postulantes que hayan participado del Proceso en otro periodo y este se encuentre con estado Suspendida.
2	Notificaciones	Escenario: Nuevo mensaje a usuarios Postulante	Automatizar la alerta de avisos de términos de etapa a los usuarios que participan de los procesos de certificación	5,3 y 1 días antes de que la fecha de término de la Etapa se cumpla, crear un mensaje con estado No leído y enviarlo vía mail e interno en el sistema a todos los usuarios que tengan asociado el Proceso y que se encuentren con la cuenta activa. Este envío incluye a los usuarios Postulantes que hayan participado del Proceso en otro periodo y este se encuentre con estado Suspendida.
3	Administración	Escenario: Gestión de Usuarios	Permitir llevar el registro de todas las actividades que realicen las personas que utilizan el sistema	Se debe llevar un registro de todas las acciones que realizan las personas en el sistema, esto se aplica para todos los roles de la plataforma. La información almacenada no se debe eliminar y el acceso solo debe estar habilitado para el usuario Administrador.

Con el listado de ajustes, se confeccionó el Documento de Ajuste (ver 4.4.7 Revisión de Ajuste). Estos requerimientos fueron re-programados en una nueva iteración de construcción.

Con las iteraciones de construcción finalizadas y todos los ajustes implementados, el sistema fue liberado para la revisión del cliente. Las observaciones del cliente fueron ingresadas en el Registro de Incidencias y resueltas por los programadores. Con el visto bueno del cliente, la Fase de construcción finalizó.

5.3.4 Desarrollo de la Fase de Implantación y Transferencia

Con la construcción del sistema concluido, el Analista confeccionó los Manuales de usuario y el Programador Líder el Manual de instalación, ambos revisados por el QA; con su aprobación el Jefe del Proyecto cerró el proyecto guardando toda la documentación generada en el desarrollo para su utilización futura. En la Tabla 5-3 se listaron los artefactos construidos para el desarrollo del proyecto del SCP, el Jefe de Proyecto hizo entrega al cliente de los siguientes artefactos:

- Informe de Especificación de Requerimientos
- Descriptor de Alcance
- Manuales de usuario e instalación.

Además de esto, se hizo la transferencia de los códigos fuentes y aplicativos para que fueran puestos en ambiente de QA Ministerial y luego en ambiente productivo.

5.3.5 Análisis de Resultados de la validación de *Z-Ágil*

En primer lugar, no se esperaba lograr tener aceptación inmediata de la aplicación de *Z-Ágil* por parte del equipo de trabajo, este temor estaba fundado por el rechazo que podría haber causado el cambio de modelo, tomando en consideración la cantidad de años que llevaban los miembros del equipo trabajando bajo un mismo marco de desarrollo, con herramientas, artefactos, modelos, etc., conocidos, aceptados y valorados. Sin embargo, no existió tal rechazo, lo que posibilitó que las actividades asignadas según sus roles a los miembros del equipo se desarrollaran en conformidad a los tiempos establecidos.

Uno de los cambios más notorios que pudieron percibir el Analista y por sobre todo los Programadores, fue la manera en cómo se abordó la documentación de los requerimientos de usuario en el Informe de Especificación de Requerimientos. En el antiguo modelo de desarrollo, estos eran documentados como Casos de uso, con la estructura comúnmente utilizada para ello: Casos de uso de alto nivel y bajo nivel más la descripción detallada de las interacciones con validaciones, flujos normales y flujos alternativos. Bajo *Z-Ágil*, los requerimientos son agrupados por módulos y cada requerimiento contiene uno o más escenarios de interacción cada programador es responsable de la realización de un requerimiento. Este cambio fue bien valorado por el equipo de programación ya que les permitió tener una visión más global del sistema (incluso desde los primeros requerimientos que se construían) y no tan aislado como cuando desarrollaban Casos de uso.

En la Fase de Construcción, el equipo logró implementar de manera correcta las iteraciones de construcción. Los miembros fueron capaces de realizar las actividades que sus roles demandaban, es así como los programadores fueron capaces de implementar los Descriptores de Requerimientos y a utilizar la herramienta de integración continua correctamente, herramienta nueva para ellos. Por otro lado, el Programador Líder, QA y Tester realizaron las tareas de revisión y validación sin objeción alguna, valorando todas estas nuevas instancias de revisión.

Con respecto a la participación de los usuarios finales, como se indicaba en las interacciones de construcción, los usuarios tuvieron participación activa durante del desarrollo del proyecto, como primera instancia en la captura y definición de requerimientos en la Fase de Exploración para luego en la Fase de Construcción probando el sistema, contestando las encuestas de TAM y en los foros de discusión para la captura de mejoras al sistema.

Desde una perspectiva práctica, los usuarios del Ministerio del Medio Ambiente tienen una alta intención de uso del SCP, los mismos que evalúan positivamente en relación a la Facilidad de Uso, la Actitud de Uso e Intención de Uso. Sin embargo, fue mediante la aplicación de TAM que se logró determinar que los usuarios se mostraban reacios al cambio, indicador obtenido en Fase de Construcción gracias a la aplicación de TAM, de otra forma esta información se hubiese sabido cuando el sistema estuviese en operación.

6 Mejoras a futuro

Luego de realizar la validación de *Z-Ágil* fue posible detectar una serie de mejoras a realizar al proceso en general, estas mejoras guardan relación con la mejora integración de TAM y el proceso de captura de la información entregada por el modelo.

Cuando se aplicó *Z-Ágil* al desarrollo del SCP, la aplicación de TAM arrojó resultados que permitieron obtener valiosa información sobre las percepciones de utilidad y de usabilidad del SCP, en las 2 iteraciones de construcción, las Percepciones de Usabilidad y Facilidad de Uso fueron antecedentes claves para determinar el uso real del sistema. Se concluyó que el sistema poseía características de diseño que hacían del sistema usable y amigable a la interacción de los usuarios, sin embargo, ¿si esto no hubiese sido tal?, ¿cómo se habrían determinado las mejoras a realizar al diseño si el antecedente de Facilidad de Uso no hubiese sido influyente en la Intención de Uso?

Cuando se determinó que los usuarios finales no sentían que la utilidad del sistema tuviese injerencia en la intención, se tomó la decisión de realizar un foro de discusión para recabar mayor información sobre esta situación, se logró determinar que era producto de la resistencia al cambio lo que producía una baja Intención de Uso producto de la Percepción de Utilidad.

Ya sea una baja Intención de Uso producto de malos antecedentes de Percepción de Utilidad o Percepción de Facilidad de Uso, es necesario mejorar la manera en la cual se recoge, procesa y se determinan las acciones correctivas al diseño del sistema y que son materializadas en las iteraciones de construcción. Para recabar información en la aplicación en el SCP, se realizó un foro en donde mediante la discusión abierta se obtuvo información sobre los antecedentes que fallaron en el modelo, sin embargo, esta manera de capturar información puede ser engorrosa y los resultados obtenidos errados, puesto que es una discusión abierta donde los participantes pueden verse influenciado por sus pares. Para evitar esto, se propone como mejora la utilización de formularios en línea donde los usuarios puedan evaluar las pantallas o interfaces del módulo o conjunto de requerimientos que se evaluó. Esta evaluación sería realizada por los usuarios finales al ponderar las interfaces y el comportamiento indicado además las mejoras a realizar, el Documento de Revisión de Ajuste se construirá a partir del análisis de los resultados de las evaluaciones de las pantallas, tomando en consideración aquellas pantallas que obtuvieron en promedio una baja calificación por ejemplo.

Otra mejora guarda relación con la manera de la aplicación de las encuetas de TAM. Como se describió en la Tabla 5-5Tabla 5-5 Detalle del estudio, las encuestas fueron realizadas en dependencias del Ministerio utilizando para ello papel, la mejora a aplicar va por automatizar este proceso utilizando algún tipo de herramienta que permita la confección y publicación de las encuestas como la obtención de los datos procesados de manera automática.

7 Conclusión

La integración de TAM como herramienta para "medir" la aceptación de un sistema en fases tempranas de construcción, esto es, sin conocerse la globalidad del sistema y aun así entregar resultados que permitieron realizar mejoras en el diseño es algo nuevo en lo que respecta a la aplicación del Modelo de Aceptación de la Tecnología, por lo mismo, se sientan precedentes en este aspecto al convertir al TAM en una herramienta perteneciente a una Metodología de desarrollo de software.

Con respecto a la integración de *Z-Ágil* dentro de la organización, un punto importante a destacar fue la aceptación y rápida asimilación del nuevo modelo. Con la redefinición de los roles, actividades y las nuevas herramientas en *Z-Ágil*, se esperaba contar con cierta resistencia al cambio, sin embargo, esto no sucedió, lo que posibilitó que las actividades asignadas según sus roles a los miembros del equipo se desarrollaran en conformidad a los tiempos establecidos.

En *Z-Ágil* se distingue la posibilidad de realizar modificaciones al sistema en Fase de construcción recogiendo observaciones directas de los usuarios finales. En el antiguo proceso de desarrollo, se avanzaba a través de él de manera lineal, finalizando etapas para iniciar otras. Las modificaciones o mejoras que se realizaban al sistema ocurrían en fases tardías del desarrollo, cuando el sistema se encontraba en las etapas de prueba y aceptación de los clientes; era en este tiempo donde el cliente podía observar la implementación de sus requerimientos y con ello, realizar sus aprensiones que luego se convertían en modificaciones. El costo de realizar este tipo de ajustes a este nivel de avance en el proyecto cuesta entre 50 a 200 veces más que si se realizara en fases tempranas de levantamiento de requerimientos o diseño (Papaccio & Boehm, 1988). Con la aplicación de *Z-Ágil*, las observaciones de los usuarios son capturados en la etapa de codificación en la Fase de Construcción, complementadas con la valiosísima información que aporta la aplicación de TAM a cada release que se evalúa, los ajustes al sistema son materializados en la misma Fase de Construcción lo que reduce considerablemente los costos.

Las observaciones que se obtienen luego de aplicar el TAM y que se convierten en mejoras, sin duda alguna permitieron perfeccionar el diseño del sistema impactando en el uso final, sin embargo, la manera en cómo se obtiene esta información es un punto en el cual se debe mejorar. En la validación de *Z-Ágil* en el proyecto del SCP se realizaron reuniones con los usuarios finales en una especie de foro, bajo esta modalidad se obtuvieron las observaciones de los usuarios que se ingresaron como ajustes al inicio de una nueva iteración de construcción. La manera en cómo se recogió y procesó esta información, como la decisión de qué requerimientos agrupar para construir un release y la forma en como automatizar la aplicación de las encuestas de TAM se enmarcan como mejoras y trabajos futuros.

A la fecha, el SCP se encuentra en operación en http://scp.mma.gob.cl/, recibiendo solicitudes y entregando certificaciones ambientales a diversos centros del País. La resistencia mostrada en el uso del sistema en Fase de Construcción ha sido superada, el sistema está siendo utilizado por los mismos usuarios que participaron en el desarrollo éste.

El Ministerio del Medio Ambiente sigue en contacto con la empresa proveedora para la realización de mejoras y la incorporación de nuevas características, extendiendo de esta manera las funcionalidades originales del sistema, adoptándolo a nuevos requerimientos y desafíos.

8 Bibliografía

Bagozzi, R., & Yi, Y. (1988). On the evaluation of structural equation models. *Journal of the Academy of Marketing Science, 16*(1), pp. 74-94.

Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change* (Segunda ed.). Addison-Wesley Professional.

Bhattacherjee, A. (2001). An empirical analysis of the antecedents of electronic commerce service continuance. *Decision support systems, 32*(2), pp. 201-214.

Blauw, E. (1994). *Het corporate image: over imago en identiteit.* Amsterdam: De Viergang.

Boehm, B. (1986). A Spiral Model of Software Development and Enhancement. *SIGSOFT Softw. Eng. Notes*, pp. 11-24.

Brinkerhof, J. (1990). *Corporate Image Als Concurrentiewapen.* Holland Harvard Review.

Cheong, J., & Park, M. (2005). Mobile internet acceptance in Korea. *Internet Research, 15*(2), pp 125-140.

Churchill, G. (1979). A paradigm for developing better measures of marketing constructs. *Journal of Marketing Research*, pp. 64-73.

Cockburn, A. (1998). *Surviving object-oriented projects: a manager's guide.* Addison Wesley.

Davis, F. (1989). *Perceived usefulness, perceived ease of use, and user acceptance of information technology.*

Fornell, C., & Larcker, D. (1981). Evaluating structural equation models with unobservable variables and measurement error. *Journal of Marketing Research, 18*(1), pp. 39-50.

Galván, Á., & Torres, J. (17 de Noviembre de 2010). *WikiCE.* Obtenido de WikiCE: http://osl2.uca.es/wikiCE/index.php?title=Extreme_programming

George, J. (2004). The theory of planned behavior and Internet purchasing. *Internet research, 14*(3), pp. 198-212.

McConnell, S. (2004). *Code Complete* (Segunda ed.). Microsoft Press.

Nielsen, J. (1994). Heuristic evaluation. En J. Nielsen , & R. Mack, *Usability Inspection Methods* (págs. 25-62). New York: John Wiley & Sons.

Papaccio, P. N., & Boehm, B. W. (1988). *Understanding and Controlling Software Costs.* IEEE Transactions on Software Engineering.

Peinado , S., & Vázquez, F. (17 de Noviembre de 2010). *WikiCE*. Obtenido de WikiCE: http://osl2.uca.es/wikiCE/index.php/Dynamic_Systems_Development

Penadés, C., Letelier, P., & Canós, J. (2006). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica administrativa ISSN-e 1666-1680, 5*(26).

Schwaber, K. (2009). *Agile Project Management with Scrum.* Microsoft Press.

Silvera, F., & de Hormaechea, S. (29 de Junio de 2012). *http://revista2.linti.unlp.edu.ar*. Obtenido de http://revista2.linti.unlp.edu.ar: http://revista2.linti.unlp.edu.ar/tesinas/tesis69.pdf

Tapscott, D., & Caston, A. (2000). *Cambio de Paradigmas Empresariales.* Mc Graw Hill.

Taylor, S., & Todd, P. A. (1995). Understanding Information Technology Usage: A Test of Competing Models. *Information Systems Research, 6*(2), pp. 144-176.

The Standish Group International. (1994;1995). *The Chaos Report.* Obtenido de http://www.cs.nmt.edu/~cs328/reading/Standish.pdf

The Standish Group International. (2000). *The Chaos Report.*

ANEXOS

A: Formulario de Evaluación de Factores de Deficiencia en Análisis

BUSQUEDA DE DEFICIENCIAS EN EL PROCESO DE DESARROLLO
DEFICIENCIAS EN EL EQUIPO DE DESARROLLO
ANALISTA

INSTRUCCIONES:

En base a los proyectos que usted ha trabajado como **Analista**, complete lo siguiente:

S: Nivel de **SEVERIDAD**. Indica el grado de influencia que ha tenido el factor en su desempeño y que ha restado calidad a su trabajo. Utilice la escala de 1 a 5. En donde 5 es el mayor nivel y 1 el menor, esto es, que si un factor ha tenido nivel 5 significará que ha influido directa y considerablemente en contra de la calidad de su trabajo.

F: FRECUENCIA. Frecuencia que ha visto presente ese factor en su trabajo.
A: para Alto, **M:** para medio y **B** para Bajo.

Nombre del proyecto: Puede señalar de forma alternativa el nombre de algún Proyecto en donde estuvo presente el factor y cree que es considerable analizar cómo se gestó el desarrollo.

FACTOR	DESCRIPCIÓN	S	F	PROYECTO
Control insuficiente	Poco control del JP para detectar a tiempo los signos de posibles retrasos.			
Planificación excesivamente optimista	Tiempo estimado para el desarrollo de los levantamientos, trabajo en CU, modelo de datos, análisis en general, demasiado optimista.			
Bajo control de calidad	Falta de revisión y validación del JP de los diversos documentos generados.			
Clientes poco claros	Clientes poco claros con la solución que requieren, cambiantes en sus requerimientos.			
Falta de participación del cliente	Poca disposición a resolución de temas, colaboración, tiempos de respuesta, ayuda en general.			
Falta de participación de los implicados	Falta de participación activa de todos los miembros del proyecto.			
Personal problemático	En el equipo del proyecto. Personas que no aceptan críticas, ayuda, observaciones, generan conflicto, etc.			
Personal mediocre	En el equipo de Analistas. Baja capacidad para la resolución de problemas, baja productividad, poca experiencia, etc.			
Cambios de requerimientos				
No existencia de un estándar de trabajo	Uso de herramientas, diseño de interfaces, modelos, sin contar una norma.			
Falta dominio técnico sobre la tecnología	No conocer las posibilidades y límites de la tecnología con la cual se trabaja al momento de diseñar y analizar las soluciones.			

B: Formulario de Evaluación de Factores de Deficiencias en Programación

BUSQUEDA DE DEFICIENCIAS EN EL PROCESO DE DESARROLLO
DEFICIENCIAS EN EL EQUIPO DE DESARROLLO
PROGRAMADOR

INSTRUCCIONES:

En base a los proyectos que usted ha trabajado como **Programador**, complete lo siguiente:

S: Nivel de **SEVERIDAD**. Indica el grado de influencia que ha tenido el factor en su desempeño y que ha restado calidad a su trabajo. Utilice la escala de 1 a 5. En donde 5 es el mayor nivel y 1 el menor, esto es, que si un factor ha tenido nivel 5 significará que ha influido directa y considerablemente en contra de la calidad de su trabajo.

F: FRECUENCIA. Frecuencia que ha visto presente ese factor en su trabajo.
A: para Alto, **M:** para medio y **B** para Bajo.

Nombre del proyecto: Puede señalar de forma alternativa el nombre de algún Proyecto en donde estuvo presente el factor y cree que es considerable analizar cómo se gestó el desarrollo.

FACTOR	DESCRIPCIÓN	S	F	PROYECTO
Análisis deficiente	Diseño deficiente, modelos incompletos, CU inconsistentes, etc.			
Control insuficiente	Poco control del JP para detectar a tiempo los signos de posibles retrasos. Desconocimiento el estado de avance del proyecto.			
Planificación excesivamente optimista	Tiempo estimado para el desarrollo de los CU demasiado optimista.			
Gestión de riesgos insuficientes	Escasa gestión de riesgos: Plan de Contingencia y mitigación deficiente.			
Bajo control de calidad	Bajo control de la calidad de los códigos e implementaciones de CU.			
Programación a destajo	Programadores son libres para codificar, sin limitéis, ni restricciones. Falta de una norma de programación.			
Personal problemático	En el equipo del proyecto. Personas que no aceptan críticas, ayuda, observaciones, generan conflicto, etc.			
Falta de participación de los implicados	La no participación activa de todos los participantes del proyecto.			
Personal mediocre	En el equipo de programadores. Baja capacidad para la resolución de problemas, baja productividad, poca experiencia, etc.			
Cambios de requerimientos				
Falta de conocimiento de la Arquitectura	Conocimiento del lenguaje, Frameworks, ciclos de vida, etc.			
Falta de un programador líder	Líder que sea capaz de dirigir el equipo, tomar decisiones y guiar la programación			

C: Formulario de Evaluación de Factores de Deficiencia en Aseguramiento de Calidad

BUSQUEDA DE DEFICIENCIAS EN EL PROCESO DE DESARROLLO
DEFICIENCIAS EN EL EQUIPO DE DESARROLLO
ASEGURADOR DE CALIDAD

INSTRUCCIONES:

En base a los proyectos que usted ha trabajado como **QA**, complete lo siguiente:

S: Nivel de **SEVERIDAD**. Indica el grado de influencia que ha tenido el factor en su desempeño y que ha restado calidad a su trabajo. Utilice la escala de 1 a 5. En donde 5 es el mayor nivel y 1 el menor, esto es, que si un factor ha tenido nivel 5 significará que ha influido directa y considerablemente en contra de la calidad de su trabajo.

F: FRECUENCIA. Frecuencia que ha visto presente ese factor en su trabajo.
A: para Alto, **M:** para medio y **B** para Bajo.

Nombre del proyecto: Puede señalar de forma alternativa el nombre de algún Proyecto en donde estuvo presente el factor y cree que es considerable analizar cómo se gestó el desarrollo.

FACTOR	DESCRIPCIÓN	S	F	PROYECTO
Personal problemático	En el equipo del proyecto. Personas que no aceptan criticas, ayuda, observaciones, generan conflicto, etc.			
Falta de conocimiento del negocio	Poco conocimiento sobre el negocio lo que implica que las pruebas sean incompletas y se recurra a ayuda de otros miembros del equipo.			
Sistema no apto para el comienzo de pruebas	Sistemas inestables al momento de comenzar pruebas.			
Falta de conocimientos técnicos	Poco conocimiento sobre la arquitectura del producto evaluado por lo que las pruebas no son del todo integras.			
Herramientas deficientes	Herramientas insuficientes, falta de licencias, características de las máquinas, etc.			

D: Descriptor de Cargo de Analista

	DESCRIPTOR DE CARGO	ZEKE
Generalidades		
Empleado		
Nombre del cargo	ANALISTA TECNICO	
Departamento al que pertenece	DESARROLLO DE SOFTWARE	
Dependencia del cargo	JEFE DE PROYECTO	
Supervisión del cargo	NO APLICA	
Reemplazante del cargo	ANALISTA TECNICO	
Perfil del cargo		
Formación	TÉCNICO INFORMÁTICO	
Años de experiencia laboral	NO REQUIERE	
Características personales deseables	CAPACIDAD DE ANALISIS, METODICO, CAPACIDAD DE TRABAJO BAJO PRESION, RELACIONES INTERPERSONALES.	
Idiomas indispensables	ESPAÑOL	
Idiomas deseables	INGLES TECNICO	
Principales actividades del cargo		

CONCEPCIÓN DE PROYECTO

- ELABORAR DOCUMENTOS DE ANALISIS Y DISEÑO EN FUNCION DE LOS PROYECTOS
- PREPARAR MATERIAL DE CAPACITACION DEL NUEVO SISTEMA
- ELABORAR MANUALES DE USUARIO
- DESARROLLAR ESPECIFICACIONES DE REQUERIMIENTOS
- CREAR PLANES DE PRUEBA

GESTIÓN DE PROYECTO

- REALIZAR TRASPASO FORMAL DE UNIDADES DE TRABAJO DE ANÁLISIS A PROGRAMADOR

COORDINACIÓN Y COMUNICACIÓN

- PARTICIPAR EN REUNIONES DE LEVANTAMIENTO DE REQUERIMIENTOS
- REALIZAR CAPACITACIÓN A CLIENTES (USUARIOS)
- ELABORAR Y ENVIAR MINUTAS A CLIENTES

E: Descriptor de Cargo de Arquitecto

	DESCRIPTOR DE CARGO	ZEKE
Generalidades		
Empleado		
Nombre del cargo	ARQUITECTO DE SOFTWARE	
Departamento al que pertenece	DESARROLLO DE SOFTWARE	
Dependencia del cargo	JEFE DE PROYECTO	
Supervisión del cargo	NO APLICA	
Reemplazante del cargo	ARQUITECTO DE SOFTWARE	
Perfil del cargo		
Formación	INGENIERO EJECUCION INFORMATICO	
Años de experiencia laboral	2	
Características personales deseables	CAPACIDAD DE ANALISIS, TRABAJO BAJO PRESION, AUTONOMIA, AUTODIDACTA.	
Idiomas indispensables	ESPAÑOL	
Idiomas deseables	INGLES TECNICO	
Principales actividades del cargo		

DESARROLLO DE PROYECTO

- CONCEPTUALIZAR ARQUITECTURAS DE SOFTWARE PARA NUEVOS PROYECTOS
- REALIZAR PRUEBAS DE CONCEPTOS DE ARQUITECTURA
- DEFINIR NORMAS DE DESARROLLO PARA ARQUICTURAS ARQUITECTURAS MÁS FRECUENTES
- IMPLEMENTAR ARQUITECTURAS

GESTIÓN DE PROYECTO

- CONTROLAR APLICACIÓN DE NORMAS DE DESARROLLO EN LOS PROYECTOS
- NORMALIZAR Y BUSCAR PERMANENTEMENTE METODOS QUE PERMITAN OPTIMIZAR LAS TAREAS DE CODIFICACIÓN Y PROGRAMACIÓN

COORDINACIÓN Y COMUNICACIÓN

- PARTICIPAR EN REUNIONES DE ESTIMACION DE PROYECTOS DE SOFTWARE
- DIFUNDIR LAS MEJORES PRACTICAS DE PROGRAMACION Y USO DE FRAMEWORKS MEDIANTE CHARLAS, DEFINICION DE NORMAS DE PROGRAMACION Y ARTICULOS PUBLICITARIOS EN LAS HERRAMIENTAS COLABORATIVAS DE ZEKE.
- EVALUAR TECNICAMENTE POSTULANTES

LIDERAZGO Y DIRECCIÓN

- CONOCER COMPETENCIAS TECNICAS DE CADA PROGRAMADOR
- EVALUAR PERIODICAMENTE TRABAJO DE PROGRAMADORES MEDIANTE REVISIONES DE CODIGOS
- VALIDAR CRITERIOS USADOS POR PROGRAMADORES
- APOYAR TECNICAMENTE A PROGRAMADORES EN CASO DE DIFICULTADES CON EL USO DE HERRAMIENTAS O FRAMEWORKS.

F: Descriptor de Cargo de Jefe de Proyectos

	DESCRIPTOR DE CARGO	ZEKE
Generalidades		
Empleado		
Nombre del cargo	JEFE DE PROYECTOS	
Departamento al que pertenece	DESARROLLO	
Dependencia del cargo	GERENTE DE PROYECTOS	
Supervisión del cargo	ANALISTA, PROGRAMADORES, ARQUITECTO	
Reemplazante del cargo	ANALISTA TÉCNICO	
Perfil del cargo		
Formación	INGENIERO CIVIL INFORMATICO.	
Años de experiencia laboral	2	
Características personales deseables	LIDERAZGO, TRABAJO BAJO PRESION, AUTONOMO, METODICO, PROACTIVO, TRABAJO EN EQUIPO.	
Idiomas indispensables	ESPAÑOL	
Idiomas deseables	INGLES TECNICO	
Principales actividades del cargo		

COORDINACIÓN Y COMUNICACIÓN

- COMUNICAR PROGRAMACIÓN DE PAGOS AL JEFE DE ADMINISTRACIÓN Y FINANZAS
- COMUNICAR MENSUALMENTE LA PLANIFICACION DE ACTIVIDADES AL EQUIPO
- COMUNICAR AL CIERRE DE LA FASE DE DISEÑO EL PLAN DE CONSTRUCCION
- COORDINAR Y CONTROLAR EQUIPOS DE TRABAJO
- REPORTAR AL GERENTE DE PROYECTOS LAS DIFICULTADES Y NECESIDADES ASOCIADAS AL PROYECTO
- ELABORAR INFORMES DE AVANCE Y MANTENER INFORMADO AL CLIENTE
- REPORTAR AL GERENTE DE PROYECTOS SEMANALMENTE EL AVANCE DE LOS PROYECTOS Y RIESGOS ASOCIADOS
- IDENTIFICAR DE FORMA PROACTIVA LOS RIESGOS ASOCIADOS AL PROYECTO

LIDERAZGO Y DIRECCIÓN

- ASEGURAR QUE EL EQUIPO DEL PROYECTO CUENTE CON TODOS LOS RECURSOS NECESARIOS PARA DESEMPEÑAR SU LABOR
- CONTROLAR AVANCE EJECUCION DE LAS TAREAS DE PROGRAMACION

GESTIÓN DE PROYECTO

- VELAR POR EL CUMPLIMIENTO DE LOS PLAZOS Y COSTOS ESTABLECIDOS
- ACTUALIZAR CARTA GANTT

GESTIÓN COMERCIAL

- NEGOCIAR REQUERIMIENTOS CON CLIENTES
- HACER UN SEGUIMIENTO DE LOS COMPROMISOS CONTRAIDOS CON EL CLIENTE Y DEL CLIENTE
- GESTIONAR APROBACIÓN DE HITOS Y FACTURAS ASOCIADA

G: Descriptor de Cargo de Programador

	DESCRIPTOR DE CARGO	ZEKE
Generalidades		
Empleado		
Nombre del cargo	PROGRAMADOR	
Departamento al que pertenece	DESARROLLO DE SOWTFARE	
Dependencia del cargo	JEFE DE PROYECTO	
Supervisión del cargo	NO APLICA	
Reemplazante del cargo	PROGRAMADOR	
Perfil del cargo		
Formación	Técnico Universitario en Informática	
Años de experiencia laboral	No es necesaria	
Características personales deseables	Autodidacta, metódico	
Idiomas indispensables	Español	
Idiomas deseables	Inglés Técnico	
Principales actividades del cargo		

COORDINACION Y COMUNICACIÓN

- REPORTAR AL JEFE DE PROYECTO
- ASEGURAR LA COORDINACIÓN Y COMUNICACIÓN CON EL DISEÑADOR GRÁFICO
- REPORTAR AL ARQUITECTO DUDAS TECNICAS O DE ARQUITECTURA
- REPORTAR AL ANALISTA DUDAS DE NEGOCIO
- PARTICIPAR EN REUNIONES CON EL CLIENTE CUANDO LE SEA SOLICITADO
- COMUNICAR AVANCES DEL PROYECTO

GESTIÓN DE PROYECTO

- COMPLETAR BITACORA DE HORAS DIARIAS POR PROYECTO
- COMPLETAR BITACORA DE CONTROL DE IMPLEMENTACION
- BRINDAR SOPORTE EN EL MODELAMIENTO DE DATOS
- IDENTIFICAR DE FORMA PROACTIVA LOS RIESGOS ASOCIADOS AL PROYECTO

DESARROLLO DE PROYECTO

- EJECUTAR PLANES DE PRUEBA
- PROGRAMAR CASOS DE USO/PROTOTIPOS U OTRO SEGÚN LA ARQUITECTURA DEFINIDA
- ESTIMAR TIEMPOS DE DESARROLLO PARA LA ELABORACIÓN DE PROPUESTAS TECNICAS
- APOYAR LABORES DE INSTALACIÓN
- ELABORAR MANUALES TECNICOS DE INSTALACIÓN

H: Descriptor de Cargo de Asegurador de la Calidad

	DESCRIPTOR DE CARGO	ZEKE
Generalidades		
Empleado		
Nombre del cargo	QA	
Departamento al que pertenece	Desarrollo de software	
Dependencia del cargo	GERENTE DE PROYECTOS	
Supervisión del cargo	NO APLICA	
Reemplazante del cargo	NO APLICA	
Perfil del cargo		
Formación	TECNICO INFORMATICO	
Años de experiencia laboral	NO REQUIERE	
Características personales deseables	CAPACIDAD DE ANALISIS, METODICO, CAPACIDAD DE TRABAJO BAJO PRESION, RELACIONES INTERPERSONALES.	
Idiomas indispensables	ESPAÑOL	
Idiomas deseables	INGLES TECNICO	
Principales actividades del cargo		

DESARROLLO DE PROYECTO

- REVISIÓN DE DOCUMENTOS DE ANÁLISIS Y DISEÑO
- DISEÑO DE PLANES DE PRUEBA
- DISEÑO DE CASOS DE PRUEBA
- EJECUCIÓN DE CASOS DE PRUEBA
- CONFECCIÓN DE MANUALES DE SISTEMA
- PRUEBA DE MANUALES DE INSTALACIÓN
- DISEÑO DE PRUEBAS DE STRESS
- EJECUCIÓN DE PRUEBAS DE STRESS
- REVISIÓN DE EVIDENCIA DE CUMPLIMIENTO DE SISTEMA DE GESTIÓN DE CALIDAD

I: Plantilla de Descriptor de Visión y Alcance

Descriptor de Visión y Alcance para <Nombre del Sistema>

ZEKE LTDA.
<Número Versión>

1. Enunciado del Problema

1.1. Necesidad

<Descripción detallada de la necesidad>

1.2. Problema y Motivación

<Descripción detallada del problema>

2. VISIÓN

<Descripción detallada de la solución, poniendo énfasis en las mejoras que se obtendrán con su implementación>

2.1. Objetivos del Negocio

<Listar los objetivos del negocio>

2.2. Objetivos del Proyecto

<Listar los objetivos del proyecto>

2.3. Objetivos del Sistema

<Listar los objetivos del sistema o aplicación que se construirá>

3. Alcances

Con la finalidad de clarificar su presentación, los hemos subdividido en alcances del proyecto y alcances específicos de la solución para <Nombre del cliente>.

3.1. Alcances del Proyecto

<Listar los alcances del proyecto, haciendo énfasis en que lo listado será lo que se trabajará durante la duración del proyecto, si alguna actividad no está presente en el listado ésta no se realizará >

3.2. Alcances Específicos del sistema

<Listar los alcances del sistema, haciendo énfasis en que lo listado serán las funcionalidades que el sistema contendrá, si alguna funcionalidad no está presente en el listado ésta no se realizará >

3.3. Fases del Proyecto y sus Entregables

La implementación de la plataforma descrita en el capítulo anterior requiere de la ejecución de ciertas actividades y entregables, éstos se encuentran detallados a continuación.

- **Fase de Concepción:** <Breve descripción de la fase><Listado de entregables>
- **Fase de Exploración :** <Breve descripción de la fase><Listado de entregables>
- **Fase de Desarrollo:** <Breve descripción de la fase><Listado de entregables>
- **Fase de Implantación y Pruebas:**<Breve descripción de la fase><Listado de entregables>

3.4. Roles del Proyecto y Responsables

3.4.1. Definición de Roles

Rol	Responsabilidad
Jefe de Proyecto	<Descripción de las funciones del Jefe de Proyecto en el proyecto>
Analista	<Descripción de las funciones del Analista en el proyecto>
Programador	<Descripción de las funciones del Programador en el proyecto>
Tester	<Descripción de las funciones del Tester en el proyecto>
QA	<Descripción de las funciones del QA en el proyecto>
Tester	<Descripción de las funciones del Tester en el proyecto>
Arquitecto	<Descripción de las funciones del Arquitecto en el proyecto>

3.4.2. Responsables

A continuación se asignan las personas a cada uno de los roles definidos en la metodología, y adaptados a las necesidades del proyecto en particular. En algunos casos se asigna más de un rol a la misma persona, principalmente debido a la magnitud del proyecto.

El equipo de trabajo y la asignación de roles quedó constituido de acuerdo al siguiente cuadro:

Rol	Responsable
Jefe de Proyecto	<Nombre del responsable>
Analistas	<Nombre del responsable>
Programador	<Nombre del responsable>
Tester	<Nombre del responsable>
QA	<Nombre del responsable>
Arquitecto	<Nombre del responsable>

3.5. Carta Gantt General

<Carta Gantt del Proyecto>

J: Plantilla Informe de Especificación de Requerimientos

Informe de Especificación de Requerimientos para <Nombre del Sistema>

ZEKE LTDA.

<Número Versión>

1. Introducción

A continuación se presenta la Especificación de Requerimientos para <Nombre de Sistema>, documento que expresa el resultado de la actividad de análisis llevada a cabo por <Nombre Cliente> y ZEKE.

El objetivo de esta especificación es **validar** el análisis realizado, desde la siguientes perspectivas:

- El alcance del sistema: todo lo que está descrito en este documento será lo que constituya el sistema en su versión final, y lo que no esté en este documento no estará en la versión final del sistema.

- La Lógica de Negocio: la lógica de negocio expresada tanto en los eventos de cada escenario, como en las validaciones, será fielmente implementada, y cualquier diferencia o cambios posteriores no serán considerados dentro de los tiempos y plazos del proyecto original.

Es necesario resaltar que estas restricciones obedecen a la necesidad de mantener el orden, control y planificación del proyecto, situación beneficiosa tanto para <Nombre de la Empresa> y para ZEKE, no implicando esto la imposibilidad técnica de incorporar modificaciones o nuevos requerimientos en el futuro, administrados como Proyectos de Control de Cambios.

La estructura del documento es la siguiente:

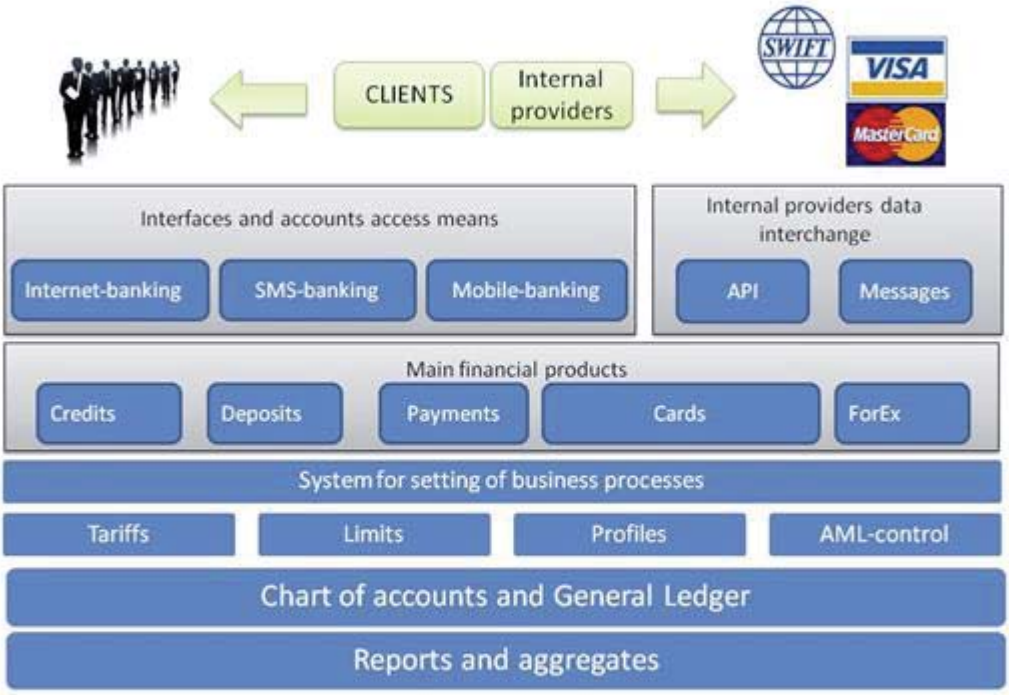
- El sistema se ha dividido en Módulos, cada uno contiene funcionalidades afines, que permiten desarrollar una parte de la lógica total del sistema. La división y su lógica se detalla en la sección 2.
- Por cada módulo, se ha identificado un conjunto de procesos de negocio y los actores que intervienen. Los procesos de negocio son llamados Casos de Uso, y se describen en detalle en la sección 3.
- Cada Caso de Uso se compone de uno o más escenarios, para los cuales se entrega un diagrama de navegación, y por cada escenario, un prototipo más la descripción de sus acciones.

Se solicita poner énfasis en las siguientes validaciones:

- Actores de Casos de Uso. Un actor puede verse como un perfil dentro del sistema. Si no están todos los esperados, se debe informar como resultado de la validación del documento, e indicar los faltantes.
- Prototipos de Escenarios. Los escenarios que se construirán serán equivalentes a los prototipados, con diferencias propias de los estilos finales, pero es en esta instancia en que se debe validar: navegación, disposición de elementos de los formularios (layout) , e información disponible pantalla.
- Reglas de Negocio: Para cada acción que es posible hacer en los escenarios, se especifican acciones, que tienen descritas las reglas del negocio del sistema. Esta es la instancia para validar que está lógica está correcta.

2. Módulos del Sistema

<Diagrama de Módulos del Sistema>



(ejemplo de descomposición funcional de un sistema, en un diagrama no estructurado)

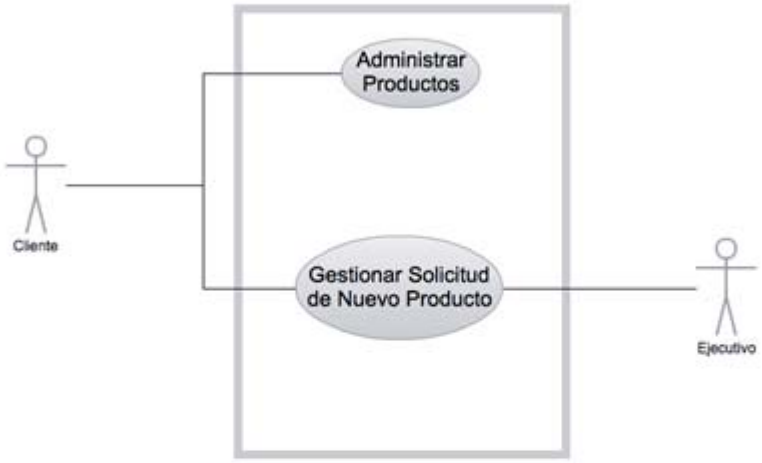
<Descripción breve de cada módulo>

3. Especificación de Requerimientos para <Nombre de Sistema>

3.1 Módulo 1: <Nombre Módulo>

3.1.1 Diagrama de Casos de Uso del Módulo <Nombre Módulo>

<Diagrama de Casos de Uso del Módulo>



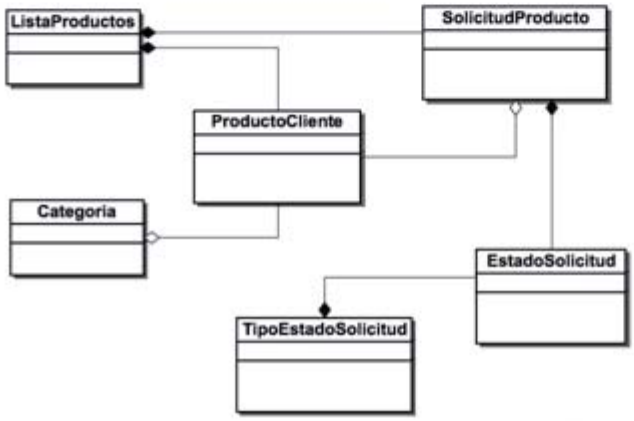
(ejemplo de descomposición funcional de un sistema, en un diagrama no estructurado)

3.1.2 Descripción de Negocio del Módulo <Nombre Módulo>

<Breve descripción en lenguaje coloquial del Módulo>

3.1.3 Modelo de Dominio del Módulo <Nombre Módulo>

<Diagrama de Clases con el Modelo de Dominio del Módulo>



3.1.4 Diagramas de Estado de Entidades de Negocio

<Diagrama de Estado por cada entidad>

(ejemplo de diagrama de estados para la entidad "solicitud")

3.1.5 Desarrollo de Casos de Uso

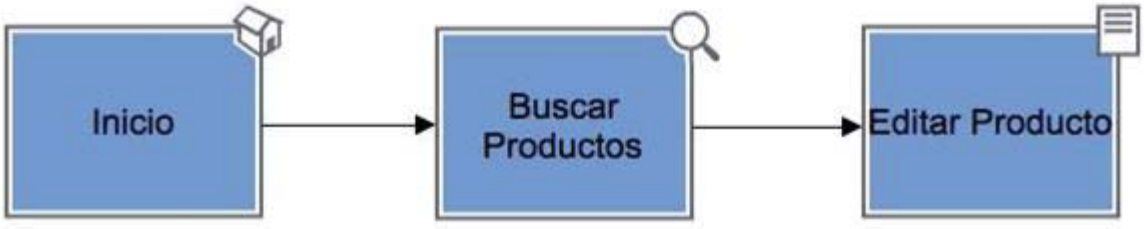
3.1.5.1 Caso de Uso: "MOD01-CU01 Administrar Productos"

3.1.5.1.1 Propósito

<Breve descripción en lenguaje coloquial del negocio desarrollado por el Caso de Uso>

3.1.5.1.2 Escenarios

<Diagrama que permita identificar a todos los escenarios del caso de uso y su navegación>



(ejemplo de diagrama de escenarios y navegación)

3.1.5.1.3 Escenario 1: Buscar Productos

<Prototipo/Maqueta del escenario>

Acción 1: Inicialización

- Se debe validar que <condición de negocio>
- Se deben obtener todos los X que <condición de negocio> y desplegarlos, ordenados por Y.
- Si <condición de negocio> se debe permitir <opciones habilitadas>

Acción 2: Selección de Lista de Producto

- Al seleccionar una Lista de Producto, se debe prellenar el filtro de Categorías con todas las Categorías que tienen los Productos actualmente en la Lista.

Acción 3: Editar Producto

- Validar que se haya seleccionado previamente un X
- Continuar con el Escenario 2, con el X seleccionado.

Acción Z: Confirmar Edición de X

- Validaciones
  - Validar que se hayan ingresado los datos obligatorios
  - Validar que la fecha de creación sea menor o igual a la actual
- Cambios de Estado
  - Se modifican los datos del X, con los valores ingresados.
  - El nuevo X debe quedar asociado al Y seleccionado.
- Acciones Adicionales
  - Se debe mostrar mensaje de creación exitosa
  - Se debe dejar un registro en el log de la creación
  - Se debe enviar un correo electrónico al usuario informando la creación

K: Plantilla de Documento de Arquitectura de Software

Documento de Arquitectura de Software para <Nombre del Sistema>

ZEKE LTDA.
<Número Versión>

1. Vista de Funcional

Permite resumir cuáles son las funciones principales del sistema. Para reflejar estos aspectos se deben utilizar los siguientes modelos de UML: casos de uso, dominio y estados.

1.1. Modelo de Casos de Uso

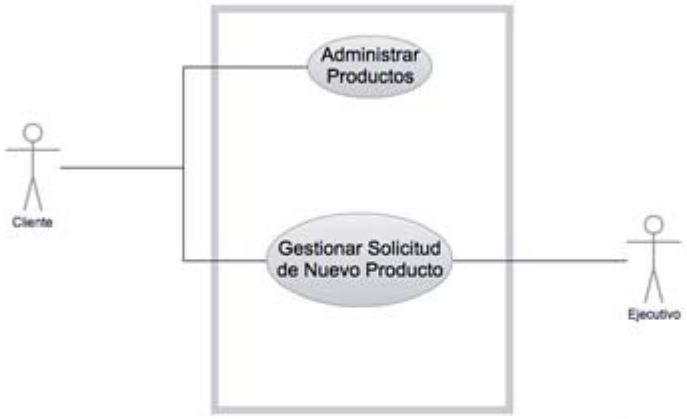


Figura 1 Ejemplo de modelo de caso de uso

1.2. Modelo de Dominio

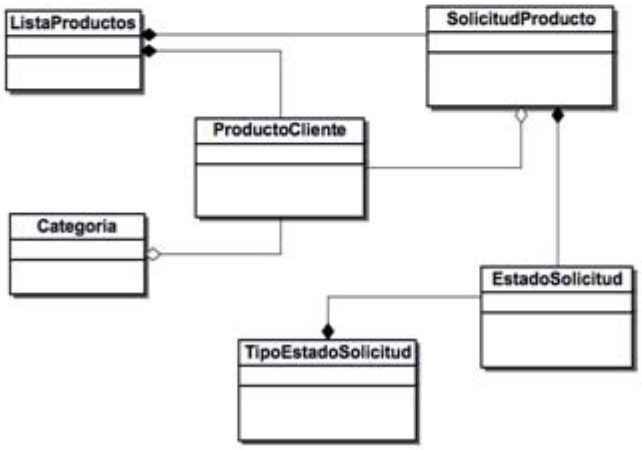


Figura 2 Ejemplo modelo de dominio

1.3. Modelo de Estado

Figura 3 Ejemplo modelo de estados

2. Vista Lógica

Permite presentar la estructura del sistema a través de sus componentes de software y sus interacciones. Para representar gráficamente esto, se debe utilizar diagramas de componentes de UML.

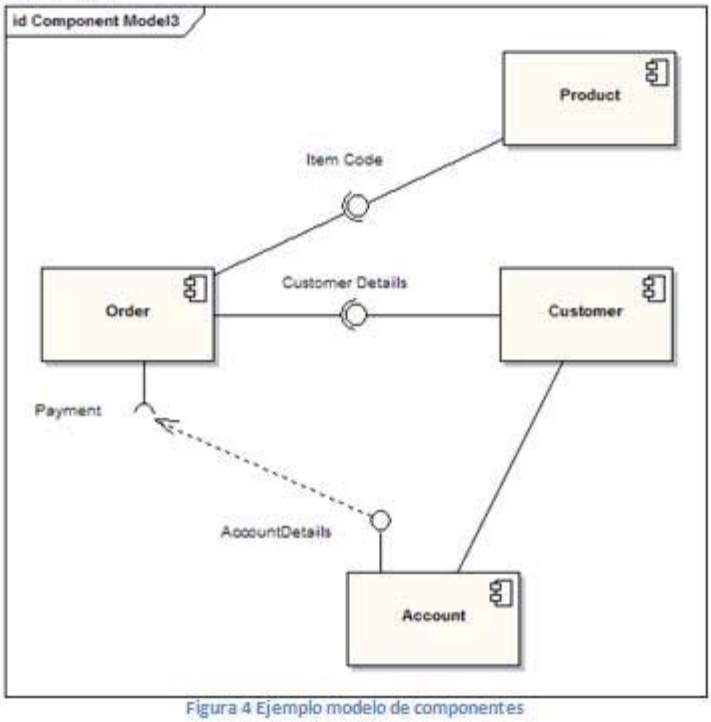


Figura 4 Ejemplo modelo de componentes

3. Vista Infraestructura

Se utiliza para describir el hardware físico, redes, comunicaciones, etc., en las que el software será implementado. Para este propósito es posible utilizar diagramas de despliegue de UML.

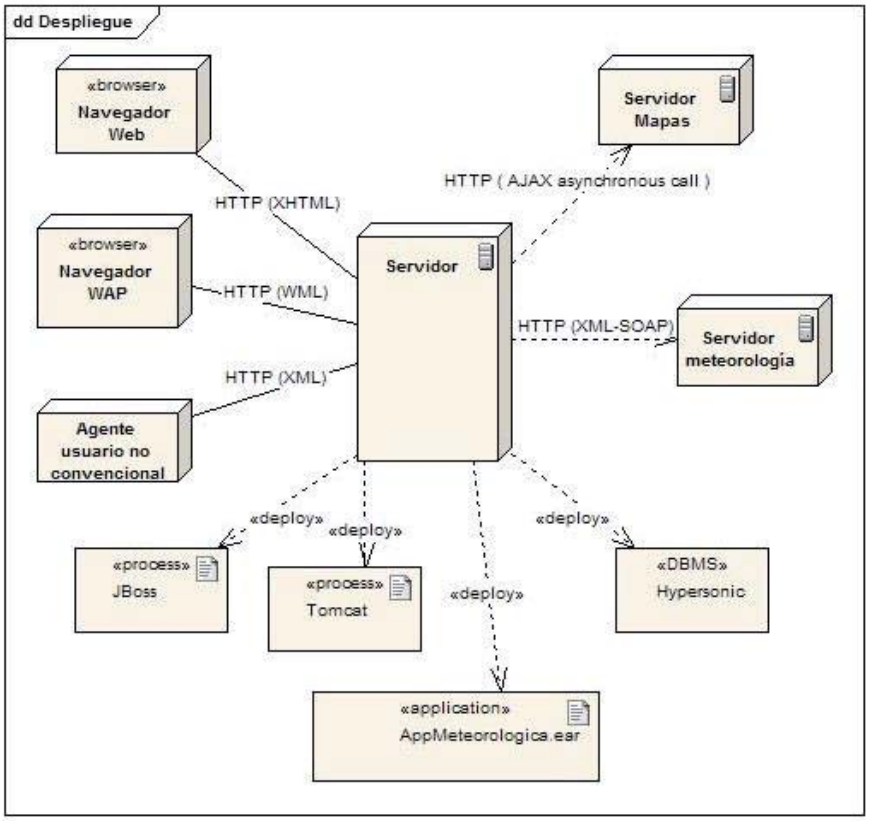


Figura 5 Ejemplo de diagrama de despliegue

4. Vista de Seguridad

Proporciona una manera de explicar cómo la arquitectura implementa los requisitos de seguridad como la autenticación, autorización, confidencialidad de datos, etc.

5. Vista de Datos

Corresponde al modelo relacional y diccionario de datos del sistema. Útil para conocer los aspectos técnicos de cómo se almacenen los datos y cómo son tratados por el motor de base de datos que los gestiona.

L: Plantilla Descriptor de Riesgos

Descriptor de Riesgos para <Nombre del Sistema>

ZEKE LTDA.
<Número Versión>

1. Riesgos del Proceso

1.1. Lista de riesgos

ID Riesgo	Descripción	Probabilidad	Impacto	Exposición
PROC<correlativo>	<Descripción>	[1,2,3]	[1,2,3,4,5]	*(Probabilidad*Impacto)*

1.2. Plan de Mitigación

ID Riesgo	Descripción del plan
PROC<correlativo>	<Descripción de plan de acción para disminuir la probabilidad de que riesgo se materialice>

1.3. Plan de Contingencia

ID Riesgo	Descripción del plan
PROC<correlativo>	<Descripción de plan de acción para minimizar el impacto del riesgo materializado>

2. Riesgos de las Personas

2.1. Lista de riesgos

ID	Descripción	Probabilidad	Impacto	Exposición
PER<correlativo>	<Descripción>	[1,2,3]	[1,2,3,4,5]	*(Probabilidad*Impacto)*

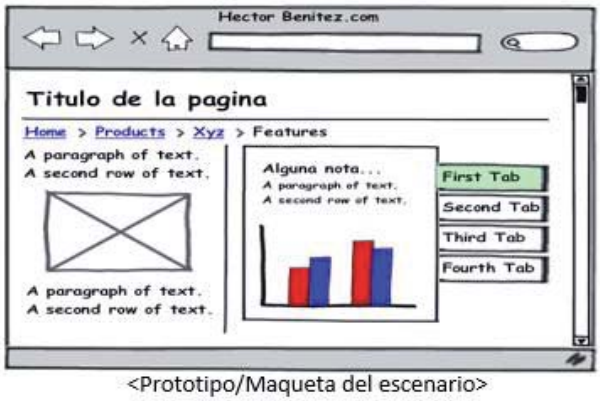
2.1. Plan de Mitigación

ID Riesgo	Descripción del plan
PER<correlativo>	<Descripción de plan de acción para disminuir la probabilidad de que riesgo se materialice>

2.2. Plan de Contingencia

ID Riesgo	Descripción del plan
PER<correlativo>	<Descripción de plan de acción para minimizar el impacto del riesgo materializado>

M: Plantilla de Descriptor de Requerimiento

DESCRIPTOR DE REQ: "REQMOD<*Número del Módulo*>-REQ<*Nombre del Requerimiento*>	
Información del Requerimiento	
ID	REQMOD<*Número del Módulo*>-REQ<*Nombre del Requerimiento*>
Propósito	<*Descripción general de lo que se va a construir*>
Requerimiento	<*Nombre del Requerimiento*>
Módulo	<*Nombre del Módulo*>
Actores	<*Lista de los actores del sistema*>
Responsabilidad	
Programador	<*Nombre del programador encargado del desarrollo*>
Fecha de inicio	<*Fecha de inicio*>
Fecha de término	<*Fecha de término*>
Escenarios de Navegación	
 <Diagrama que permita identificar a todos los escenarios del requerimiento y su navegación, en este caso, correspondería a 3 escenarios>	
Escenario 1	
 <Prototipo/Maqueta del escenario>	
Nombre	<*Nombre del escenario*>
Validaciones	*a) Inicialización* • <Se deben listar todas la acciones requeridas para que el escenario sea cargado y permita una correcta visualización e interacción del usuario, por ejemplo: cargar la lista de regiones en la selección de regiones, cargar el nombre del usuario en el sistema, etc.>

	b) Acción *<Nombre del objeto de la interfaz que permite interacción entre el sistema y el usuario ejemplo: Botón buscar, Botón Guardar, etc>* • *Validaciones* o *<Listar todas las validaciones requeridas previa ejecución de la acción, por ejemplo: validar que haya seleccionado campos obligatorios>* • *Cambios de estado* o *<Listar todos los cambios de estado producidos en las clases por la ejecución de la acción. Para esto, debe apoyar por el Modelo de Dominio del Módulo>* • *Acciones adicionales* • *<Listar todas las acciones adicionales posterior ejecución de la acción, por ejemplo: mensajes de éxito o error.>*

N: Plantilla de Revisión de Ajuste

REVISIÓN DE AJUSTE: "REVMOD<*Número del Módulo*>-REQ<*Nombre del Requerimiento*>	
Información del Ajuste	
ID	REVMOD<*Número del Módulo*>-REQ<*Nombre del Requerimiento*>
Propósito	<*Descripción general de lo que se requiere ajustar en el requerimiento*>
Requerimiento	<*Nombre del Requerimiento*>
Módulo	<*Nombre del Módulo*>
Responsabilidad	
Programador	<*Nombre del programador encargado del ajuste*>
Fecha de inicio	<*Fecha de inicio*>
Fecha de término	<*Fecha de término*>
Lista de Ajustes	
a) <*Descripción de lo que se quiere cambiar, por ejemplo: "El titulo de la página debiese estar en negrita acompañada del logo del gobierno, como se muestra en la imagen a continuación"*> 	

O: Constructos e ítems de TAM

CONSTRUCTO	ESCALA		FUENTE
FACILIDAD DE USO	FU1	Encuentro que el uso del sistema SCP es sencillo	(Davis, 1989)
	FU2	Aprender a operar el sistema SCP es fácil para mi	
	FU3	El hecho de interactuar con el sistema SCP es satisfactorio	
	FU4	Al usar el sistema SCP estoy seguro que podré hacer en el sistema lo que necesito realizar	
	FU5	El SCP es adaptable y flexible para interactuar con él	
	FU6	Es fácil para mi recordar cómo realizar tareas en el SCP	
	FU7	Interactuando con el SCP no se requiere mucho esfuerzo mental	
	FU8	Mi interacción con SCP es clara y comprensible	
	FU9	Me parece que no se necesita un gran esfuerzo para convertirse en experto en el uso del SCP	
	FU10	En general, creo que el uso del SCP es de fácil uso	

CONSTRUCTO	ESCALA		FUENTE
UTILIDAD PERCIBIDA	UP1	El uso del SCP mejora la calidad de mi trabajo	(Davis, 1989)
	UP2	El uso del SCP me entrega un mayor control sobre mi trabajo	
	UP3	El uso del SCP me permite realizar mis tareas con mayor rapidez	
	UP4	En los momentos críticos de mi trabajo el SCP se adapta a mis requerimientos	
	UP5	El uso del SCP me permite aumentar la productividad de mi trabajo	
	UP6	El uso del SCP me permite aumentar el rendimiento de mi trabajo	
	UP7	El uso del SCP me permite hacer más trabajo que de otra manera no podría hacer	
	UP8	El uso del SCP mejora la eficacia de mi trabajo	
	UP9	El uso del SCP hace que sea más fácil hacer mi trabajo	
	UP10	En general creo que el SCP es útil para mi trabajo	

CONSTRUCTO	ESCALA		FUENTE
ACTITUD HACIA EL USO	A1	Es una idea que me gusta	(Taylor & Todd, 1995), (Bhattacherjee, 2001) y (George, 2004)
	A2	Me parece una idea inteligente	
	A3	Es una buena idea	
	A4	Me parece una experiencia positiva	

CONSTRUCTO	ESCALA		FUENTE
INTENSION DE USO	IU1	Pienso que utilizaré el SCP en los próximos años	(Taylor & Todd, 1995) y (Cheong & Park, 2005)
	IU2	Considero que utilizaré el SCP al máximo	
	IU3	Recomendaría el uso del SCP a otros	
	IU4	Pienso que el uso del SCP vale la pena	

P: Plantilla de Encuesta TAM

	Muy en desacuerdo	**En desacuerdo**	**Ni de acuerdo ni en desacuerdo**	**De acuerdo**	**Totalmente de acuerdo**
Encuentro que el uso del sistema SCP es sencillo	1	2	3	4	5
Aprender a operar el sistema SCP es fácil para mi	1	2	3	4	5
El hecho de interactuar con el sistema SCP es satisfactorio	1	2	3	4	5
Al usar el sistema SCP estoy seguro que podré hacer en el sistema lo que necesito realizar	1	2	3	4	5
El SCP es adaptable y flexible para interactuar con el	1	2	3	4	5
Es fácil para mi recordar cómo realizar tareas en el SCP	1	2	3	4	5
Interactuando con el SCP no se requiere mucho esfuerzo mental	1	2	3	4	5
Mi interacción con SCP es clara y comprensible	1	2	3	4	5
Me parece que no se necesita un gran esfuerzo para convertirse en experto en el uso del SCP	1	2	3	4	5
En general, creo que el uso del SCP es de fácil uso	1	2	3	4	5

	Muy en desacuerdo	**En desacuerdo**	**Ni de acuerdo ni en desacuerdo**	**De acuerdo**	**Totalmente de acuerdo**
Es una idea que me gusta	1	2	3	4	5
Me parece una idea inteligente	1	2	3	4	5
Es una buena idea	1	2	3	4	5
Me parece una experiencia positiva	1	2	3	4	5

	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
El uso del SCP mejora la calidad de mi trabajo	1	2	3	4	5
El uso del SCP me entrega un mayor control sobre mi trabajo	1	2	3	4	5
El uso del SCP me permite realizar mis tareas con mayor rapidez	1	2	3	4	5
En los momentos críticos de mi trabajo el SCP se adapta a mis requerimientos	1	2	3	4	5
El uso del SCP me permite aumentar la productividad de mi trabajo	1	2	3	4	5
El uso del SCP me permite aumentar el rendimiento de mi trabajo	1	2	3	4	5
El uso del SCP mi permite hacer más trabajo que de otra manera no podría hacer	1	2	3	4	5
El uso del SCP mejora la eficacia de mi trabajo	1	2	3	4	5
El uso del SCP hace que sea más fácil hacer mi trabajo	1	2	3	4	5
En general creo que el SCP es útil para mi trabajo	1	2	3	4	5

	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Pienso que utilizaré el SCP en los próximos años	1	2	3	4	5
Considero que utilizaré el SCP al máximo	1	2	3	4	5
Recomendaría el uso del SCP a otros	1	2	3	4	5
Pienso que el uso del SCP vale la pena	1	2	3	4	5