UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

COMPARACIÓN DE TÉCNICAS PARA LA ESPECIFICACIÓN, VERIFICACIÓN Y VALIDACIÓN DE REQUERIMIENTOS; ESTRUCTURADAS EN UN MARCO COMÚN UTILIZANDO EL ALPHA REQUIREMENTS DE ESSENCE

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRA EN INGENIERÍA (COMPUTACIÓN)

PRESENTA:
INGRID ROXANA SÁNCHEZ VÁZQUEZ

TUTORA:
M. EN C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ,
FACULTAD DE CIENCIAS, UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

MÉXICO, D. F. JUNIO 2015

Tabla de contenido

Introducción

Requerimientos de Software es reconocida como un área de la Ingeniería de Software, la cual cuenta con siete subáreas que son: Fundamentos, Proceso, Obtención, Análisis, Especificación, Validación y Consideraciones Prácticas [1].

Durante la especificación de requerimientos el ingeniero de software documentará los requerimientos para posteriormente con ayuda del proceso de validación y verificación tratar de asegurar que el sistema que se está desarrollando cumpla con lo determinado y con lo que el usuario espera.

Hoy en día existen diversas técnicas para apoyar a los procesos de especificación, validación y verificación de requerimientos. La selección de estas técnicas se basa, generalmente, en las preferencias personales o en las prácticas existentes en la organización [2].

Por otro lado el Software *Engineering Method and Theory* (SEMAT)[3] y la *Object Management Group* (OMG)[4] en un esfuerzo para re-fundamentar la Ingeniería de Software presenta *Essence*[5]que es una propuesta para establecer un núcleo para métodos de desarrollo de software.

Essence consta de un núcleo y un lenguaje para definir métodos, prácticas y elementos del núcleo. Dentro del núcleo se encuentran las Alphas que son una representación básica de las cosas con las que se puede trabajar al desarrollar software. Proveen una descripción del tipo de elementos que el equipo puede manejar, producir y usar en el proceso de desarrollo, mantenimiento y soporte de sistemas de software)[6].

Una de estas *Alphas* es el *Alpha Requirements* que tiene una estructura de estados que son: concebido, acotado, coherente, aceptable, dirigido y cumplido. Durante el desarrollo de un sistema de software los requerimientos deberán ir progresando mediante el cambio de un estado a otro. Para evaluar el estado y el progreso de los requerimientos, *Essence* provee de listas de verificación para cada estado.

El objetivo principal de este trabajo es realizar la comparación de las técnicas de Casos de Uso, Historias de usuario, Caso de Uso 2.0, *Features* y *Theme* para la especificación, verificación y validación de requerimientos funcionales; y estructurar dichas técnicas utilizando como marco común el *Alpha Requirements* de *Essence*. Con lo cual se aporten elementos que faciliten la selección de una técnica para la especificación, verificación y validación de requerimientos, acorde con las necesidades del problema y las técnicas empleadas.

Estructura de la tesis

Este trabajo de tesis está compuesto por cuatro capítulos, conclusiones, trabajo a futuro y referencias bibliográficas. En el capítulo 1 se presentan los conceptos generales sobre qué es un requerimiento, especificación, validación y verificación de requerimientos. En el capítulo 2 se da una breve introducción a lo que es *Essence*, el *Alpha Requirements* y los estados que lo componen. En el capítulo 3 se presentan las definiciones, características, así como, una

propuesta de plantilla para documentar, verificar y validar requerimientos con cada una de las técnicas para la especificación, validación y verificación de requerimientos como son: Casos de Uso, Historias de Usuario, Casos de Uso 2.0, *Features* y *Theme*. En el capítulo 4 se realiza la comparación de las técnicas para la especificación, validación y verificación de requerimientos. Finalmente se presentan las conclusiones y el trabajo futuro.

Capítulo 1. Conceptos Generales

1.1 Requerimiento de Software

"Un requerimiento es una propiedad que debe ser exhibida para resolver algún problema del mundo real"[1].

Todos los requerimientos de software es que deben ser verificables, aunque esto pueda resultar costoso, para ello deben ser expresados con la mayor claridad, es decir sin ambigüedades y, cuando sea el caso, cuantificarlos; especialmente cuando se habla de requerimientos no funcionales.

Por su naturaleza los requerimientos de software se dividen en:

- Funcionales: describen las funciones o flujos de negocio que el software debe ejecutar.
- No funcionales: son los que actúan para limitar la solución, estos son a veces conocidos como las restricciones y requerimientos de calidad.

Es importante tener una comprensión del problema a resolver, lo cual se logra después de una serie de interacciones entre seres humanos donde: identificarán a los interesados y se establecerá la relación entre el equipo de desarrollo y el cliente.

Al tener esa comprensión sobre el problema será crítico informar el alcance del proyecto, en el cual se debe proporcionar una descripción del software y su objetivo dando prioridad a que las necesidades más importantes del negocio sean las primeras en entregarse para evitar gastar tiempo en requerimientos de poca importancia. También se debe buscar que el software sea escalable y extensible en la mayoría de las ocasiones, permitiendo que nuevos requerimientos no contemplados en las primeras listas sean aceptados.

Típicamente los requerimientos tienen diversas fuentes y es importante que todas sean identificadas y evaluadas; las principales son las siguientes[1]:

- Objetivos. Estos dan el motivo del software, a menudo son escritos vagamente y el ingeniero del software deberá poner atención a la evaluación del valor y el costo de los objetivos.
- Conocimiento del dominio. Proporciona contexto sobre el problema a resolver y permitirá que el ingeniero de software identifique conceptos relevantes del dominio.
- Interesados. Es importante que el ingeniero de software atienda los "puntos de vista" de los diversos grupos de interesados a fin de que pueda identificarlos y representarlos para que lo entregado al cliente cumpla con las expectativas y necesidades de los involucrados.
- Reglas de negocio. Estas son declaraciones que definen o limitan algún aspecto de la estructura o el comportamiento de la propia organización, expresan el flujo de negocio.
- Ambiente operacional. Los requerimientos se derivan del medio ambiente en que el software se ejecutará, por lo que es importante buscarlos activamente ya que pueden afectar la viabilidad y costes del software.
- Ambiente organizacional. El software típicamente apoya el proceso de negocio y el ingeniero de software debe poner atención a este proceso, además de la estructura cultural y política de la organización para que el nuevo software no force un cambio no planificado en el proceso de negocio.

Los requerimientos se pueden clasificar de la siguiente manera[1]:

- Requerimientos Funcionales. Son las capacidades o funciones del software.
- Requerimientos No Funcionales. Son los que actúan para obligar a llegar a la solución pero no son parte integral del software. Cabe hacer notar, que no siempre es así ya que en muchas ocasiones estos se convierten en requerimientos funcionales [7].
- Características Inesperadas. Son requerimientos que se toman pero no pueden comprobarse hasta que esté funcionando el software en condiciones reales.
- Requerimientos de Producto. Se refiere a establecer los parámetros del problema a solucionar para ser traducido a un software.
- Requerimientos de Proceso. Se refiere a la parte técnica y a lo que se va a utilizar para construir el software (lenguaje de programación, por ejemplo).
- Prioridad. Cuanto más alta sea la prioridad, más esencial es el requerimiento para satisfacer las necesidades.
- Alcance. Se refiere al grado en que un requerimiento afecta al software o alguno de sus componentes.
- Volatilidad/estabilidad. Se refiere a la probabilidad de que un requerimiento cambie.
- Requerimientos del Sistema y de Software. Se refiere a todo lo que necesita el software para funcionar desde el punto de vista de hardware, software, recurso humano, información, instalaciones, servicios, etc. Los requerimientos de software se derivan de los del sistema.

1.2 Especificación de Requerimientos de Software

La especificación de requerimientos es el proceso en el cual el ingeniero de software escribe los requerimientos de usuario y del sistema en un documento[7]. Este documento que también es llamado Especificación de Requerimientos de Software y debe poder ser revisado sistemáticamente, evaluado y aprobado, permitiendo así, disminuir el rediseño en etapas posteriores ya que proporciona una base para hacer una correcta estimación de costos, tiempos y riesgos en el desarrollo de software.

Los requerimientos se deben describir de manera que sean comprensibles para los usuarios del sistema, que no tienen un conocimiento técnico detallado, usando el lenguaje natural apoyados con tablas sencillas, formularios y diagramas intuitivos y especificando sólo el comportamiento externo del sistema. El documento de requerimientos no debe incluir detalles de la arquitectura o el diseño del sistema.

1.3 Validación y Verificación de Requerimientos de Software

En este proceso se trata de asegurar que el sistema que se está desarrollando cumpla con lo determinado durante la especificación, así como con lo que el usuario espera de él.

Validación:

- Trata de comprobar si se está construyendo un producto que satisfaga las necesidades del usuario haciendo lo que él espera[8].
- Comparación de los requerimientos especificados [9].

Verificación:

- Busca comprobar que producto cumpla con los requerimientos especificados [10].
- Comprobación de idoneidad para uso esperado[9].

Entre los objetivos de este proceso de control se encuentran los siguientes puntos:

- Detectar y corregir los defectos tan pronto como sea posible.
- Disminuir los riesgos, las desviaciones sobre los presupuestos y sobre el programa de tiempos.
- Mejorar la calidad y fiabilidad del software.

El proceso de validación y verificación trata de comprobar que el producto de software alcanza su propósito y aunque esto no signifique que esté libre de defecto, debería ser lo suficientemente bueno para su uso y satisfacer las necesidades del usuario.

Capítulo 2. Essence

Essence nace como respuesta al llamado a la acción de SEMAT[3] (*Software Engineering Method and Theory*) y OMG (*Object Management Group)*[4],el cual tiene como objetivo la redefinición de la ingeniería de software. Y como primer paso tenía el identificar una serie de elementos esenciales que fueran universales en el desarrollo de software y un desarrollar un lenguaje sencillo para describir estos métodos y prácticas.

El núcleo de *Essence*[6] contiene un número de "cosas con las que trabajamos" (Figura 1) también llamadas *alphas* y "cosas que hacemos" (Figura 2) cuando se desarrollan sistemas de software. El núcleo está organizado en tres áreas de interés, las cuales se centran en una dimensión específica del desarrollo de software:

- Cliente: El desarrollo de software siempre implica al menos un cliente para el software que produce y su perspectiva debe estar integrada en el trabajo del día a día para garantizar que se desarrolla una solución adecuada.
- Solución: El objetivo de desarrollo de software es el desarrollo de un sistema de software que trabaje para resolver algún problema. El área de solución contiene todo lo relacionado con la especificación y desarrollo del sistema de software.
- Esfuerzo: El desarrollo de software no es tarea fácil y generalmente implica mucho tiempo y esfuerzo, afecta a muchas personas e involucra un equipo de desarrollo. El área de esfuerzo contiene todo lo relacionado con el equipo de desarrollo y la forma en que se hace el trabajo.

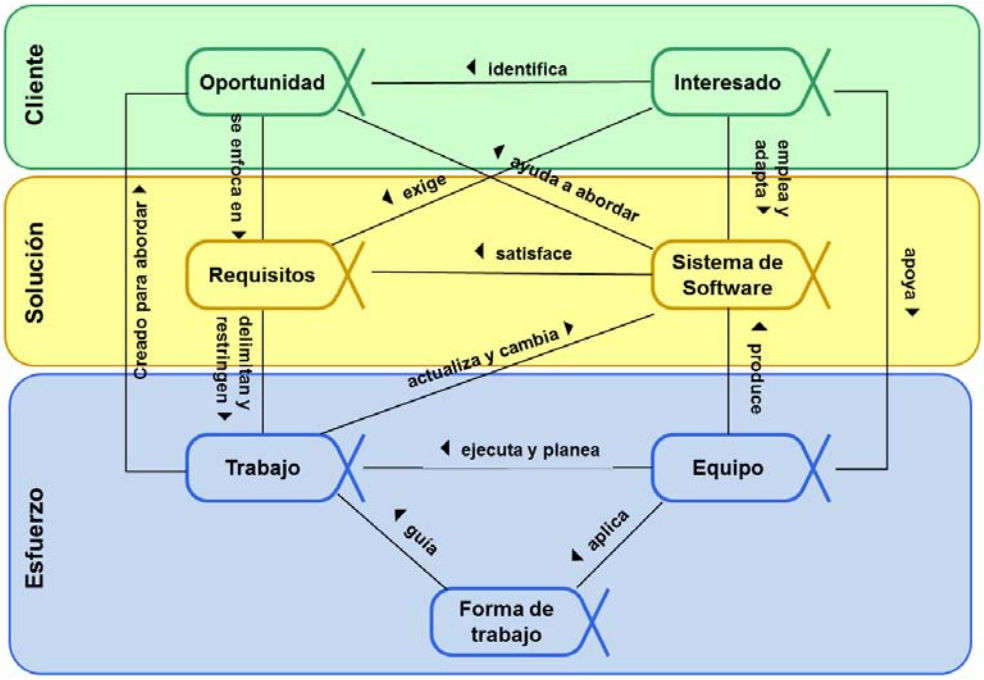

Figura 1. Cosas con las que trabajamos (*alphas*) [19]

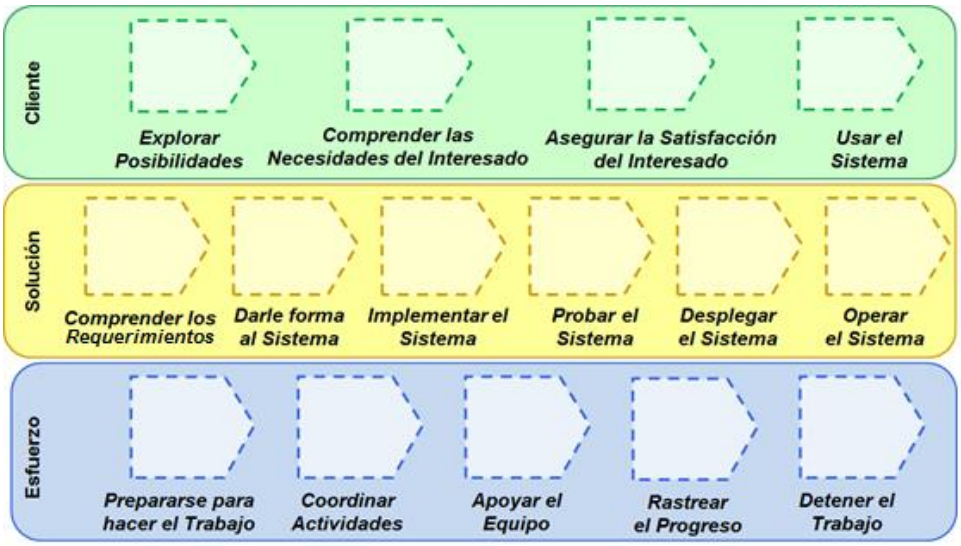

Figura 2. Cosas que hacemos[19]

Los elementos llamados *alphas* tienen un conjunto de estados que permiten medir su progreso independientemente de las prácticas aplicadas durante el desarrollo de software. Estos estados hacen que el núcleo sea operable y le permiten guiar el comportamiento de los equipos de desarrollo de software. Para alcanzar un estado en particular, se proporcionan listas de chequeo que especifican los criterios para alcanzar algún estado.

Dentro de la especificación se proveen tarjetas con el detalle y descripción de cada una de las *alphas* además de sus puntos de verificación (figura 3).

Figura 3. Tarjetas de *Alphas* de *Essence* [19]

2.1 Alpha Requirements

Las a*lphas* representan las cosas esenciales con las que trabajamos durante el desarrollo de software que deben ser monitoreadas para medir su progreso. Los profesionales de software pueden actualizar rápidamente lo que necesitan saber acerca de los requisitos esenciales a través de la *Alpha Requirements* y sus estados.

Alpha Requirements[11] se define como: "Lo que el sistema de software debe hacer para hacer frente a la oportunidad y satisfacer a los interesados".

Mediante tarjetas de definición del *Alpha Requirements* (Figura 4) se podrá mantener la información esencial sobre el núcleo visible para los miembros del equipo.

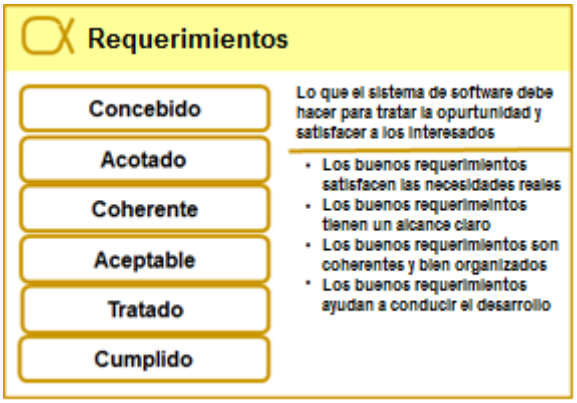

Figura 4. Tarjeta de definición del *Alpha Requirements*

2.2.1 Estados

El *Alpha Requirements* tiene seis estados y los objetivos de cada uno son los siguientes [11]:

- **Concebido**
  Los requisitos comienzan en el estado concebido cuando la necesidad de un nuevo sistema de software es acordado. Los interesados pueden presentar diferentes vistas del significado completo de los requisitos. Sin embargo, todos ellos están de acuerdo en que hay una necesidad de un nuevo sistema de software y una clara oportunidad a perseguir.
- **Acotado**
  Se establecen el alcance total del nuevo sistema y los mecanismos para gestionar y aceptar los requisitos nuevos o cambiados. En el estado acotado aún puede haber inconsistencias entre ítems individuales de requisitos. Sin embargo, los interesados ahora tienen una comprensión compartida del propósito del nuevo sistema y pueden decir si una solicitud califica o no como un requisito válido.
- **Coherente**
  Las características esenciales del nuevo sistema se definen claramente. Los ítems de requisitos continúan evolucionando a medida que se aprende más del nuevo sistema y su impacto sobre los interesados y el entorno. Sin importar cuánto cambien los ítems de requisitos, es esencial que se mantengan dentro de los límites del concepto original y que permanezcan coherentes en todo momento.
- **Aceptable**
  Los requisitos definen un sistema que será aceptable para los interesados al menos como una solución inicial. Los requisitos pueden describir sólo una solución parcial. Sin embargo, la solución descrita tiene el valor suficiente para que los interesados puedan aceptar su uso operacional.
- **Tratado**
  Se implementaron suficientes ítems de requisitos en el nuevo sistema como para que sea importante su liberación y uso. En el estado tratado, el monto de ítems de requisitos que se trata es suficiente para que el sistema resultante provea un claro valor a los interesados. Si el sistema resultante provee una solución completa, entonces los requisitos pueden avanzar de inmediato al estado cumplido.
- **Cumplido**
  Suficientes ítems de requisitos se implementaron para que los interesados estén de acuerdo en que el sistema resultante satisface completamente las necesidades de un

nuevo sistema y no hay ítems excepcionales de requisitos que eviten que el sistema se considere completo.

Durante el desarrollo de un sistema de software los requerimientos deberán ir progresando a través de varios cambios de estado, para que el equipo pueda saber en qué estado se encuentra se tienen tarjetas de cada estado las cuales contienen el nombre del *alpha*, el nombre de estado y una lista de criterios generales que se deben cumplir en ese estado (Figura 5).

Figura 5. Tarjetas de los estados del *Alpha Requirements*

Los requerimientos pueden estar en diferentes estados al mismo tiempo o estar en el mismo estado varias veces durante un ciclo.

Para ayudar a evaluar el estado y progreso de los requerimientos hasta su conclusión con éxito, se proporciona siguiente lista de puntos:

Estado	Puntos de verificación
Concebido	○ El conjunto inicial de interesados está de acuerdo en que se debe desarrollar el sistema ○ Se identifican los interesados que va a utilizar el nuevo sistema ○ Se identifican los interesados que financiarán los trabajos iniciales del nuevo sistema ○ Hay una clara oportunidad para el nuevo sistema.
Acotado	○ Se identifican los interesados involucrados en el desarrollo del nuevo sistema ○ Los interesados están de acuerdo con el objetivo del nuevo sistema ○ Es claro cuál es el éxito para el nuevo sistema ○ Las interesados tienen un entendimiento compartido del alcance de la

Estado	Puntos de verificación
	solución propuesta o La forma en que se describirán los requerimientos está acordada o Los mecanismos para el manejo de los requerimientos se establecen o El esquema de priorización es claro o Las restricciones están identificadas y consideradas o Los supuestos están claramente establecidos
Coherente	o Los requerimientos están capturados y compartidos con el equipo de trabajo e interesados o La procedencia de los requerimientos es clara o La razón detrás de los requerimientos es clara o Los requerimientos en conflicto están identificados y atendidos o Los requerimientos comunican las características esenciales del sistema que será liberado o Los escenarios de uso más importantes para el sistema se pueden explicar o La prioridad de los requerimientos es clara o Se entiende el impacto de implementar los requerimientos o El equipo entiende lo que tiene que ser entregado y se compromete a entregarlo.
Aceptable	o Los interesados aceptan que los requerimientos describen una solución aceptable o La velocidad de cambio de los requerimientos acordados es relativamente baja y está bajo control o El valor proporcionado al implementar los requerimientos es claro o Las áreas de oportunidad satisfechas por los requerimientos son claras o Los requerimientos son verificables
Tratado	o Hay suficientes requerimientos tratados para que el sistema resultante sea aceptado por los interesados o Los interesados aceptan que los requerimientos reflejan con mayor precisión lo que hace y no el sistema o El conjunto de requerimientos implementados proporcionar un claro valor a los interesados o El sistema que implementa los requerimientos del sistema es aceptado por los interesados y puede ser operativo
Cumplido	o Los interesados aceptan que los requerimientos capturan con precisión lo necesario para satisfacer plenamente la necesidad de un nuevo sistema o No hay requerimientos pendientes que eviten que el sistema sea aceptado para cumplir completamente con las necesidades o El sistema es aceptado por los interesados por cumplir completamente las necesidades

Capítulo 3. Técnicas para la especificación, validación y verificación de requerimientos

Las técnicas para la especificación, validación y verificación de requerimientos brindarán al ingeniero de software un procedimiento con el cual trabajar cualquiera de estas etapas tan importantes para el desarrollo de software.

Existen actualmente diversas técnicas para estas etapas pero en este trabajo se compararán las siguientes:

1. Casos de Uso
2. Historias de Usuario
3. Casos de Uso 2.0
4. Theme
5. Features

3.1 Casos de Uso

3.1.1 Origen

Los casos de uso fueron propuestos por Ivar Jacboson en el año de 1992, en este año publica el libro *Object-Oriented Software Engineering - A Use Case Driven Approach,* en el presenta por primera vez el concepto de orientada a objetos como una manera de diseño y desarrollo de sistemas y presenta a los casos de uso como la base del diseño de software.

3.1.2 Motivación

Los casos de uso nacen como resultado de 20 años de experiencia y como respuesta a la crisis de software que había en los años 90.

Con el este modelo se pretende describir la funcionalidad completa de un sistema y de todas las interacciones que se tienen, todo se representa de actores que son quienes inician y reciben la respuesta a una funcionalidad.

3.1.3 Proceso de obtención de requerimientos

El proceso de obtención incluye el desarrollo de diagramas de casos de uso, los que pertenecen al estándar de UML y son utilizados para modelar el comportamiento de un sistema, mostrando mediante una notación gráfica de la relación de un conjunto de casos de uso y los actores de un sistema.

El conjunto de casos de uso debe especificar las diferentes formas de utilizar el sistema, y por lo tanto el comportamiento requerido del sistema y que delimita el alcance del sistema.

Estos son los elementos del diagrama caso de uso:

- **Caso de uso**

  "Un caso de uso es una secuencia de operaciones realizadas por un sistema, lo cual produce valores medibles para un actor en particular", son propuestos originalmente por Jacobson[12]y es importante aclarar que no describe el procesamiento interno del sistema, solo presenta la funcionalidad general de éste.

  Para que un caso de uso logre su objetivo debe tener las siguientes características:

  1. Se deben expresar desde el punto de vista del actor.
  2. Son documentados utilizando Lenguaje Natural

3. Describen tanto lo que hace el actor como lo que hace el sistema cuando interactúa con él.
4. Se inician por un actor.
5. Se acotan a una determinada funcionalidad.

- **Actor**

Un actor es una entidad externa al sistema que interactúa con él demandado alguna funcionalidad, puede ser una persona u otro sistema.

Un actor puede representar a personas, roles o sistemas externos que interactúan con el sistema en un momento determinado. Y solo se deben mostrar cuando sea necesario para el caso de uso.

- **Alcance**

El límite del sistema define la separación entre el sistema que se va a construir y su entorno.

- **Relaciones entre casos de uso**

Las relaciones entre los casos de uso son un elemento muy importante ya que permiten representar las posibles interacciones entre los casos y uso y los actores de un sistema. Existen cuatro tipos de relaciones y son las siguientes:

  - **Asociación**

    Esta relación es la más básica y significa la participación del actor en el caso de uso.

  - ***<include>***

    Esta relación indica que un caso de uso incluye a otros casos de uso (puede ser uno o más), es decir, los casos incluidos forman parte esencial del que los incluye y sin éstos no podría cumplir su objetivo. Esta relación se dibuja como en la Figura 6.

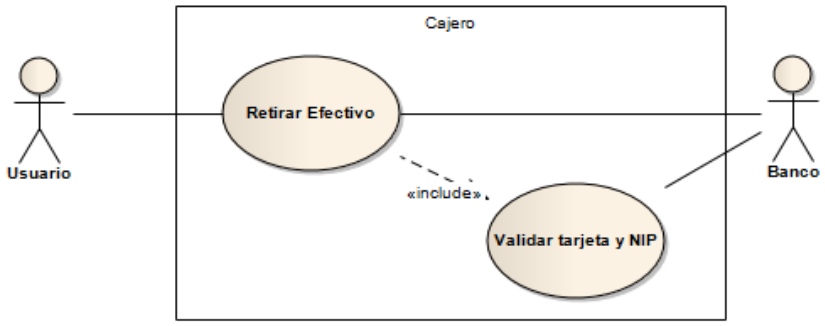

Figura 6. Ejemplo de relación *include*

  - ***<extends>***

    Esta relación indica que cuando un caso de uso extiende a otros, incorpora su funcionalidad como parte integral del que lo extiende sin alterar a éste. Hay una condición en el caso de uso que da paso al caso de uso de extensión. Esta relación se dibuja como en la Figura 7.

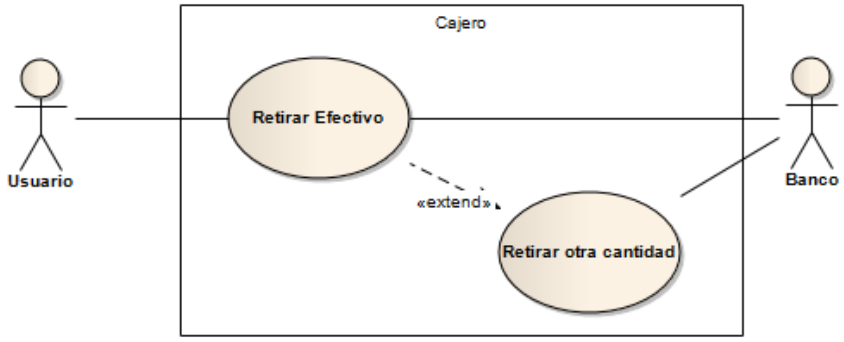

Figura 7. Ejemplo de relación *extends*

- ***Generalización***

  Esta relación se da cuando un actor o un caso de uso hereda todos los atributos, secuencias de comportamiento, puntos de extensión y relaciones definidos en el caso de uso o actor padre.

Para entender mejor estas definiciones en la Figura 8 se tiene un diagrama de casos de uso de un cajero automático, en el cual sus usuarios pueden realizar las operaciones de Retirar Efectivo, Retirar otra cantidad, Depositar Efectivo, Consultar Saldo y Validar tarjeta y NIP al contar con una cuenta y tarjeta del Banco.

Se puede notar que las operaciones se representan en forma de casos de uso y tanto el banco como los usuarios son los actores para el sistema. Por medio de relaciones se puede ver las interacciones que estos actores y casos de uso tienen en el sistema.

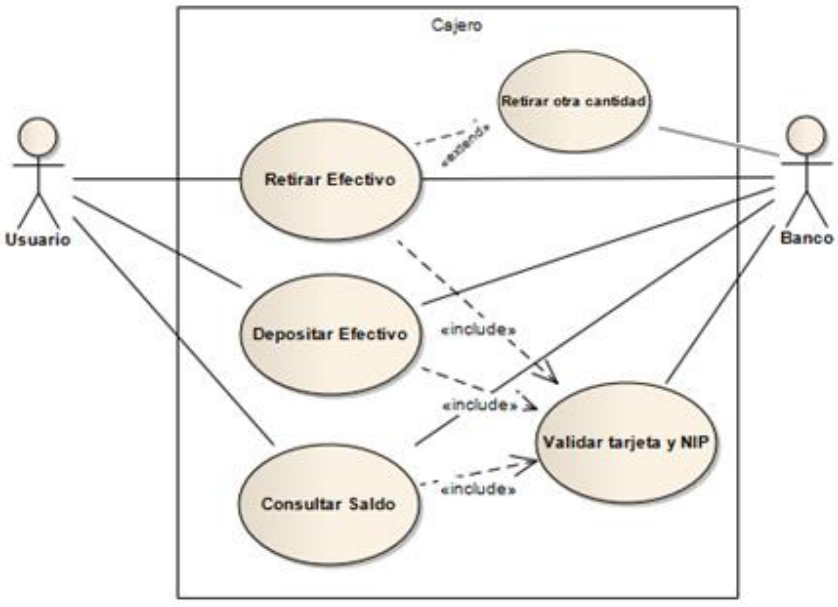

Figura 8. Diagrama general del sistema

3.1.4 Especificación de requerimientos utilizando casos de uso

Para documentar un caso de uso en este trabajo se propone la plantilla [13] de la Figura 9:

Id_CU:*[identificador del caso de uso]*			**Prioridad:***[Alta / Media / Baja]*	
Nombre:*[Nombre con el cual se identificará el caso de uso]*				
Descripción: *[En la descripción se debe expresar de manera breve el objetivo del caso de uso, así como el actor involucrado en él.]*				
Flujo de eventos				
Flujo básico				
Actor		**Sistema**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1		**2**		
3		**…**		
Flujo alternativo				
Actor		**Sistema**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
A1		**A2**		
A3		**A…**		
Pre-condiciones *[Conjunto de eventos que deben ser cubiertos para que inicie un caso de uso]*			**Post-condiciones** *[Estado del sistema si el flujo del caso de uso se ha ejecutado correctamente]*	
<include> *[Identificador del o los casos de uso con los cuales tienen una relación tipo include.]*			**<extends>** *[Identificador del o los casos de uso con los cuales tienen una relación tipo extends.]*	
Excepciones				
Id	**Tipo**		**Acción**	
	[Tipo genérico de la excepción]		*[Acción efectuada al ocurrir la excepción]*	
Datos Requeridos				
Dato	**Descripción**			**Requerido**
	[Descripción del dato requerido en el flujo del caso de uso]			*[Si / No]*

Figura 9. Plantilla para documentar el detallado de caso de uso[13]

En la Figura 10 se muestra el detalle del caso de uso Retirar Efectivo del de un cajero automático de la Figura 8.

Id_CU: CU_002			**Prioridad:** Media	
Nombre: Retirar Efectivo				
Descripción: El usuario requiere sacar efectivo de su cuenta bancaria.				
Flujo de eventos				
Flujo básico				
Actor		**Sistema**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1		**2**	El cajero le muestra las opciones de retiro de efectivo.	
3	El usuario selecciona una opción.	**4**	El cajero valida que la cantidad seleccionada esté disponible en su cuenta y de ser así la entrega al usuario.	EX01
5	El usuario toma el efectivo del cajero	**6**	El cajero le entrega la tarjeta al usuario.	
7	El usuario toma su tarjeta.			
Flujo alternativo				
Actor		**Sistema**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
A1		**..**		
Pre-condiciones Se ha validado la tarjeta y el NIP de la cuenta. Se ha seleccionado la opción retiro de efectivo			**Post-condiciones** El usuario retiro efectivo de su cuenta.	
<include> Validar tarjeta y NIP			**<extends>** Retirar otra cantidad	
Excepciones				
Id	**Tipo**		**Acción**	
EX01	Cantidad no disponible		El cajero le indica que su cuenta no dispone de la cantidad seleccionada y lo manda a la pantalla anterior.	
Datos Requeridos				
Dato	**Descripción**			**Requerido**

Figura 10. Detallado del caso de uso Retirar efectivo.

3.1.5 Validación

Para la validación de requerimientos especificados mediante casos de uso, se propone la utilización de los casos de prueba, donde cada caso de prueba contiene una serie de datos de entrada con valores validos e inválidos en donde se deberán comparar que los resultados esperados y resultados obtenidos para validar que el funcionamiento sea el que el usuario espera.

En la figura 11 se propone una plantilla para probar los casos de uso que se documentaron anteriormente.

Id_CU:*[identificador del caso de uso a probar]*			
Casos de prueba con datos válidos			
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**
[Descripción general del caso de prueba]	*[Datos necesarios para ejecutar el caso de prueba]*	*[Detalle del resultado esperado]*	*[Detalle del resultado obtenido al ejecutar el caso de prueba]*
Casos de prueba con datos inválidos			
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**
Casos de prueba para las reglas del negocio			
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**

Figura 11. Plantilla para validación de casos de uso [13]

3.1.6 Verificación

Para la verificación de requerimientos especificados mediante casos de prueba se propone agregar a la plantilla para la validación una columna donde se registe el resultado de la ejecución de caso de prueba con el que se validarán los requerimientos, Figura 12:

Id_CU:*[identificador del caso de uso a probar]*				
Casos de prueba con datos válidos				
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**	**Resultado el caso de prueba**
[Descripción general del caso de prueba]	*[Datos necesarios para ejecutar el caso de prueba]*	*[Detalle del resultado esperado]*	*[Detalle del resultado obtenido al ejecutar el caso de prueba]*	*[Estatus de la ejecución del caso de prueba: Pasó - La ejecución fue exitosa, se obtuvo el resultado esperado Falló - La ejecución dio un resultado diferente al resultado esperado]*
Casos de prueba con datos inválidos				
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**	
Casos de prueba para las reglas del negocio				
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**	

Figura 12. Plantilla con verificación de casos de uso

3.2 Historias de Usuario

3.2.1 Origen

Las Historias de usuarios tienen su origen en *Extreme Programming*[14](XP) un método ágil y metodologías ágiles como XP, Scrum, Crystal y FDD las adoptaron debido a su ligereza y su enfoque en la colaboración con los clientes.

3.2.2 Motivación

Debido a que las historias de usuario tienen su origen en las metodologías ágiles, con estas se trata de dar ligereza al proceso de especificación de requerimientos para poder responder de una manera rápida a los requerimientos cambiantes y hacer entregas en tiempos cortos.

3.2.3 Proceso de obtención de requerimientos

En el proceso de obtención de requerimientos con historias de usuario en una historia se describe una funcionalidad que es de valor para el usuario [14]. Usualmente está escrita a mano en una nota adhesiva en una o dos frases que representan la intención del usuario.

Las características principales de las historias de usuario son las siguientes:

- Independientes unas de otras
- Negociables
- Valoradas por los clientes o usuarios
- Estimables
- Pequeñas
- Verificables

Las Historias de Usuario presentan un medio de comunicación entre el usuario y el equipo de desarrollo, lo cual la hace una técnica muy útil pues ayuda a la generación de un producto de gran valor, centrándose en las ideas principales y dejando de lado los detalles superfluos.

Existen algunas técnicas como entrevistas con usuarios, cuestionarios, observación y talleres de escritura de historias, que ayudan a la comunicación entre el usuario y el equipo de trabajo a escribir las historias.

En la Figura 13 se muestra un ejemplo de una tarjeta para una historia de usuario.

Id_HU: HU_002	**Nombre:** Retiro de Efectivo
Como: Dueño de una cuenta	
Quiero: Deseo retirar efectivo de mi cuenta bancaria seleccionando una cantidad del cajero o ingresándola en él.	
Para: Disponer de dinero cuando el banco esté cerrado	

Figura 13. Ejemplo de Historia de usuario

En la Figura 14 se muestra el reverso de la tarjeta de la historia de usuario, en donde se escribirán las pruebas de aceptación para la historia descrita.

Pruebas de aceptación
Se debe validar que la cantidad a retirar debe estar disponible en la cuenta.

Figura 14. Reverso de la tarjeta

3.2.4 Especificación de requerimientos utilizando historias de usuario

Como se dijo anteriormente las historias de usuario se escriben en tarjetas, en este trabajo se propone una plantilla para documentar requerimientos, Figura 15.

Id_HU:*[identificador de la historia]*	**Nombre:***[Nombre con el cual se identificará a la historia]*
Como:*[usuario]*	
Quiero: *[servicio]*	
Para:*[razón]*	

Figura 15. Tarjeta para documentar una historia de usuario

3.2.5 Validación

Para validar que una historia ha sido correcta y completamente codificada, en la parte posterior de la tarjeta se escribe la condición descrita por el usuario donde la historia cumple con su funcionalidad. En la Figura 16 se propone una plantilla de la parte posterior de la tarjeta donde se documentó la historia de usuario.

Pruebas de aceptación
[Lista de condiciones a cumplirse para dar por completa una historia]

Figura 16. Reverso de la tarjeta para documentar una historia de usuario

3.2.6 Verificación

Para realizar la verificación de las historias de usuario a la plantilla para la validación se le agrega un cuadro de check donde se debe indicar si el criterio ha sido cumplido o no, Figura 17

Pruebas de aceptación	
Cumplió	**Prueba**
☐ ☐	*[Lista de condiciones a cumplirse para dar por completa una historia]*

Figura 17. Tarjeta con verificación para una historia de usuario

3.3 Casos de Uso 2.0

3.3.1 Origen

En el año 2012 Ivar Jacobson, Ian Spence y Kurt Bittner, publican el libro Use Case 2.0 donde presentan los casos de uso 2.0 como práctica escalable y ágil para capturar requerimientos.

3.3.2 Motivación

Combinar los casos de uso e historias de usuario para crear casos de uso muy livianos que capturen lo estrictamente necesario de las historias.

3.3.3 Proceso de obtención de requerimientos

En su proceso mezcla a los casos de uso y las historias del caso de uso para obtener un conjunto de requerimientos[16].

Se caracterizan por ser:

1. Livianos
2. Escalables
3. Versátiles
4. Fáciles de usar

Como un caso de uso se enfoca en el logro de una meta particular y encierra muchas formas de usar el sistema, se narran historias, las cuales ayudan a entender el caso de uso ya que cubren tanto la manera exitosa de alcanzar las metas como la manera alternativa que pueden ocurrir en el camino.

En la narrativa del caso de uso el flujo básico describe la forma más simple de lograr el objetivo y en el flujo alternativo describe las diferentes maneras en que se puede logar el mismo objetivo. Estos flujos se convertirán en las historias necesarias para describir el caso de uso, estas pueden tener relación con flujos básicos y alternativos.

En la figura 18 se muestra la relación de flujos entre las historias de necesarias para la descripción de 4 casos de uso, así como la relación con sus flujos básicos y alternativos.

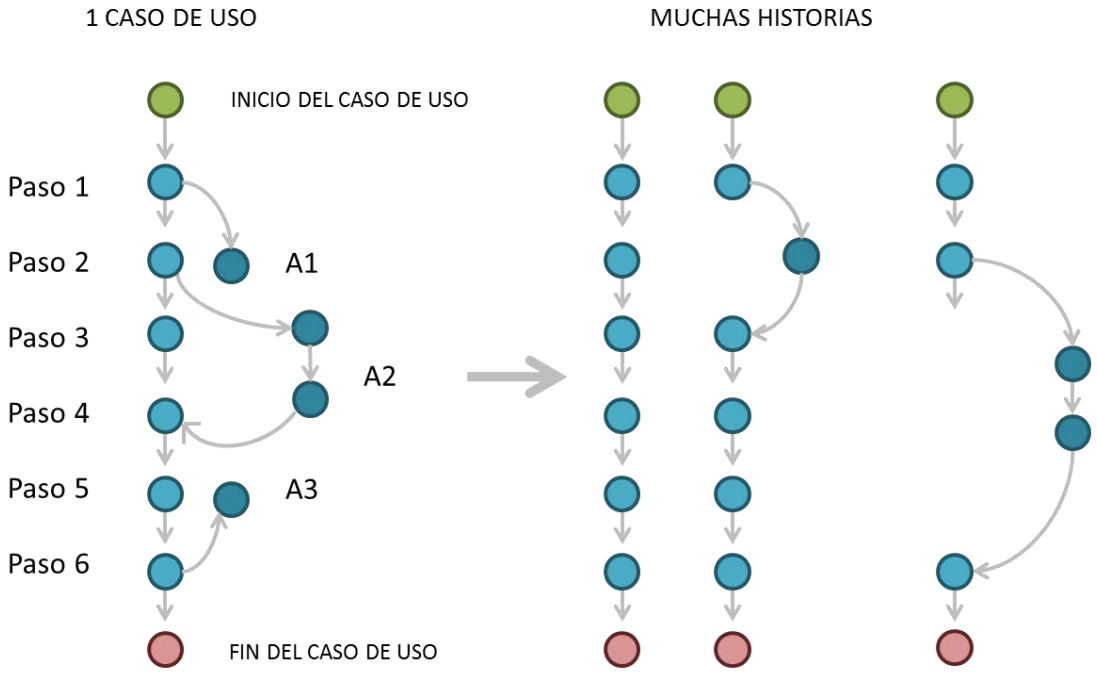

Figura 18. Relación entre flujos e historias.

En la Figura 19 se muestra un ejemplo de la narrativa del caso de uso retiro de efectivo de un cajero automático.

Un caso de uso completo puede resultar difícil de completar en un período corto de tiempo, ya que contiene diversas historias cada una con importancia y prioridad diferente.

Por lo que se necesita una forma de dividir los casos de uso en partes más pequeñas que permitan:

- Seleccionar las piezas del caso de uso a entregar
- Proporcionar una unidad adecuada para el desarrollo y las pruebas del equipo de desarrollo
- Tener piezas de trabajo pequeñas y de tamaño similar que fluyan rápidamente a lo largo del desarrollo.

La división en porciones de casos de uso permite el fácil manejo para el equipo de trabajo, además de que dan la posibilidad de ordenar, priorizar y trabajar en paralelo los requerimientos.

Narrativa de Caso de Uso

Id_CU: CU_002	**Actor:** Usuario de cajero
Nombre: Retiro de efectivo	
Estado: Definido	
Prioridad: Requerido	
Complejidad: Alta	

Flujos Básicos		**Flujos Alternativos**	
1	El usuario selección de un monto sugerido.	**A1**	Cantidad ingresada correcta.
2	El cajero valida que la cantidad esté disponible.	**A2**	Cantidad seleccionada no disponible.
3	El cajero entrega el efectivo.	**A3**	Cantidad superior a lo permitido.
4	El cajero regresa la tarjeta al usuario.		

Figura 19. Narrativa del Caso de uso Retiro de efectivo

La narrativa del ejemplo anterior puede ser dividida en porciones, en la Figura 20 se muestra el ejemplo para la porción del flujo alternativo A1.

Porción

Id_Porción:002_P1	**Nombre:** Cantidad ingresada correcta.
Id_CU:CU_002	**Flujo:**FA
Historia: El usuario ingresa una cantidad que no es múltiplo de 100	
Prueba: Validar que solo se acepten múltiplos de 100	

Figura 20. Porción del caso de uso retirar efectivo

3.3.4 Especificación de requerimientos utilizando caso de uso 2.0

Como ya se mencionó anteriormente hay tres tipos de elementos en el caso de uso 2.0, en la Figura 21 se muestra una propuesta de plantilla para la narrativa del caso de uso.

Narrativa Caso de Uso

Id_CU:*[identificador de la narrativa de caso de uso]*	**Actor:***[Nombre del actor que ejecuta el caso de uso]*
Nombre:*[Nombre con el cual se identificará el caso de uso]*	
Estado:*[Entendido, Conflicto, Atendido]*	
Prioridad:*[Requerido, Importante, Opcional]*	
Complejidad:*[Alta, Media, Baja]*	

Flujos Básicos		**Flujos Alternativos**	
1	*[Secuencia de pasos para logar el objetivo de la forma más simple]*	**A1**	*[Secuencia de pasos alternativos para logar el objetivo]*
…		**…**	

Figura 21. Plantilla para documentar la narrativa del caso de uso

En la Figura 22 se muestra una plantilla para documentar una porción de un caso de uso:

Porción

Id_Porción:*[identificador de la porción]*	**Nombre:***[Nombre con el cual se identificará la porción]*
Id_CU:*[identificador del caso de uso al que pertenece la porción]*	**Flujo:***[Flujo de la narrativa al cual pertenece la porción]*
Historia: *[En la descripción se debe expresar de manera breve el objetivo de la historia]*	
Prueba: *[Condiciones a cumplirse para dar por completa una historia]*	

Figura 22. Plantilla para documentar una porción

3.3.5 Validación

Para la validación de requerimientos especificados, cada porción del caso de uso se tiene que probar antes de que se de por validada, lo cual se hace ejecutando con éxito los casos de prueba de la porción (ver figura 22). Estos se deben elaborar para cada porción, antes de que la porción se entregue a los desarrolladores para su implementación.

Para cada incremento del sistema se debe validar que se implementaron correctamente todas las porciones nuevas del caso de uso, sin dañar ninguna otra parte del sistema.

El equipo debe probar el sistema como un todo para asegurarse de que todas las porciones implementadas son compatibles y que los cambios en el sistema no se tradujeron en fallas de las porciones previamente validadas.

3.3.6 Verificación

Para realizar la verificación de los casos de uso 2.0 se agrega a la plantilla de la porción el estatus de la prueba en donde se registrará si el requerimiento ha sido cubierto o no, Figura 23.

Porción	
Id_Porción:*[identificador de la porción]*	**Nombre:***[Nombre con el cual se identificará la porción]*
Id_CU:*[identificador del caso de uso al que pertenece la porción]*	**Flujo:***[Flujo de la narrativa al cual pertenece la porción]*
Historia: *[En la descripción se debe expresar de manera breve el objetivo de la historia]*	
Prueba: *[Condiciones a cumplirse para dar por completa una historia]*	
Estatus: *[La condición al ejecutarse la prueba se cumplió o no, Pasó o Falló]*	

Figura 23. Plantilla con verificación de una porción

3.4 *Features*

3.4.1 Origen

La utilización de características como técnica para la especificación de requerimientos toma como base a la FeDRE[17] (*Feature-DrivenRequirementsEngineering*) con enfoque en Líneas de Productos de Software (LPS), donde las características (*features*)deben ser identificadas de manera sistemática.

3.4.2 Motivación

Adoptar un modelo sistemático para la representación de requerimientos.

3.4.3 Proceso de obtención de requerimientos

Para obtener los requerimientos con esta técnica se debe identificar cada *Característica* que represente una función del sistema, esta puede estar o no, relacionada con una o varias características del sistema. El sistema se puede representar con una red de características relacionadas entre sí que darán la funcionalidad que espera el usuario.

Características (*Features*[17]) no es propiamente una técnica para la especificación de requerimientos, que están a medio entre ésta y la definición de la arquitectura. Existen propuestas que la complementan usando casos de usos.

3.4.3.1 Relaciones

Ya que la especificación de requerimientos funcionales se hará con apoyo de la técnica de casos de uso, se utilizarán las relaciones de <*include*> y <*exclude*> para describir relaciones entre los casos de uso.

3.4.4 Especificación de requerimientos utilizando *Features*

El analista es responsable de especificar las *Características* en la Figura 24 se muestra una plantilla propuesta en este trabajo para la especificación de características:

Id_C:*[identificador de la Característica]*	**Padre:***[identificador de la cual tiene una relación hijo/padre]*
Nombre:*[Nombre con el cual se identificará a la característica]*	
Descripción: *[En la descripción se debe expresar de manera breve el objetivo del caso de uso, así como el actor involucrado en é.]*	
Prioridad:*[Alta / Media / Baja]*	
<include> *[Identificador de las características con las cuales tienen una relación tipo include]*	***<extends>*** *[Identificador de las características con las cuales tienen una relación tipo extends]*

Figura 24. Plantilla para documentar Características

De acuerdo a esta la plantilla, cada *Característica* debe tener un Id_Característica el cual es identificador único y un nombre. La prioridad de la función debe ser Alta, Media o Baja. Si la función incluye o excluye otra característica(s) se indica el identificador de cada característica necesaria. Si la función tiene un elemento principal, identificador de la característica padre debe ser especificado. Además de una breve descripción de la característica.

En la Figura 25 se muestra un ejemplo de la característica Retirar de un cajero automático:

Id_C: C_02	**Padre:**
Nombre: Retirar	
Descripción: El usuario debe poder hacer retiros de efectivo de su cuenta.	
Prioridad: Media	
<include>	***<extends>***

Figura 25. Característica Retirar

En la Figura 26 se muestra una plantilla para hacer la especificación de los requerimientos funcionales de una *Característica*.

Id_CU:*[identificador del caso de uso]*	**Característica asociada:***[Identificador de las características con las que tiene relación]*			
Nombre:*[Nombre con el cual se identificará el caso de uso]*				
Descripción: *[En la descripción se debe expresar de manera breve el objetivo del caso de uso, así como el o los actores involucrados en él]*				
Flujo de eventos				
Flujo básico				
Actor		**Sistema**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1		**2**		
3		**…**		
Flujo alternativo				
Característica asociada:*[Identificador de la característica con la que se relaciona el flujo alternativo]*				
Condición:*[Acción que activa el flujo alternativo]*				
Actor		**Sistema**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
A1		**A2**		
A3		**A…**		
Pre-condiciones *[Conjunto de eventos que deben ser cubiertos para que inicie un caso de uso]*		**Post-condiciones** *[Estado del sistema si el flujo del caso de uso se ha ejecutado correctamente]*		
<include> *[Identificador del o los casos de uso con los cuales tienen una relación tipo include]*		***<extends>*** *[Identificador del o los casos de uso con los cuales tienen una relación tipo extends]*		

Figura 26. Plantilla para documentar requerimientos funcionales de una característica

En la Figura 27 se ejemplifica la documentación de los requerimientos funcionales de la característica Retirar.

Id_CU: CU_02_001			**Característica asociada**: C_02	
Nombre: Retirar Efectivo				
Descripción: El usuario requiere sacar efectivo de su cuenta bancaria.				
Flujo de eventos				
Flujo básico				
Actor		**Sistema**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1		**2**	El cajero le muestra las opciones de retiro de efectivo.	
3	El usuario selecciona una opción.	**4**	El cajero valida que la cantidad seleccionada esté disponible en su cuenta y de ser así la entrega al usuario.	EX01
5	El usuario toma el efectivo del cajero	**6**	El cajero le entrega la tarjeta al usuario.	
7	El usuario toma su tarjeta.			
Flujo alternativo				
Característica asociada:				
Condición:				
Actor		**Sistema**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
A1		**…**		
Pre-condiciones Se ha validado la tarjeta y el NIP de la cuenta. Se ha seleccionado la opción retiro de efectivo			**Post-condiciones** El usuario retiro efectivo de su cuenta.	
<include> Validar tarjeta y NIP			**<extends>** Retirar otra cantidad	
Excepciones				
Id	**Tipo**		**Acción**	
EX01	Cantidad no disponible		El cajero le indica que su cuenta no dispone de la cantidad seleccionada y lo manda a la pantalla anterior.	
Datos Requeridos				
Dato	**Descripción**			**Requerido**

Figura 27. Requerimientos funcionales la Característica Retirar

3.4.5 Validación

Ya que la especificación de requerimientos funcionales se hizo utilizando la técnica de casos de uso, se pueden validar con casos de prueba.

En la Figura 28 se muestra una plantilla para validar requerimientos funcionales de las *Características* que se documentaron.

Id_CU:*[identificador del caso de uso a probar]*			
Casos de prueba con datos válidos			
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**
[Descripción general del caso de prueba]	*[Datos necesarios para ejecutar el caso de prueba]*	*[Detalle del resultado esperado]*	*[Detalle del resultado obtenido al ejecutar el caso de prueba]*
Casos de prueba con datos inválidos			
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**
Casos de prueba para las reglas del negocio			
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**

Figura 28. Plantilla para validar requerimientos funcionales de las Características

3.1.6 Verificación

En la figura 29 se muestra la plantilla propuesta para realizar la verificación de *características*.

Id_CU:*[identificador del caso de uso a probar]*				
Casos de prueba con datos válidos				
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**	**Resultado el caso de prueba**
[Descripción general del caso de prueba]	*[Datos necesarios para ejecutar el caso de prueba]*	*[Detalle del resultado esperado]*	*[Detalle del resultado obtenido al ejecutar el caso de prueba]*	*[Estatus de la ejecución del caso de prueba:* *Pasó - La ejecución fue exitosa, se obtuvo el resultado esperado* *Falló - La ejecución dio un resultado diferente al resultado esperado]*
Casos de prueba con datos inválidos				
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**	
Casos de prueba para las reglas del negocio				
Caso de Uso	**Entradas**	**Resultados Esperados**	**Resultados Obtenidos**	

Figura 29. Plantilla con verificación de *características*

3.5 *Theme*

3.5.1 Origen

La especificación de requerimientos utilizando *temas* tiene su origen en el *Theme/Doc* es la primera fase del enfoque *Theme*[18] que describe un proceso de desarrollo orientado por temas. El concepto de *tema* que representa una pieza de funcionalidad que se quiere modelar de forma separada del sistema.

3.5.2 Motivación

Para el desarrollo ágil es de suma importancia poder dividir los requerimientos de un sistema en diversas piezas, los temas brindan esta facilidad ya que un sistema puede ser representado a través de las relaciones de varios temas.

3.5.3 Proceso de obtención de requerimientos

Los *temas*, que son una representación de una función del sistema, se obtienen a partir de una lista de requerimientos.

Tema/Doc ofrece un método sistemático para analizar una especificación de requisitos basados en texto con el fin de identificar la base y las preocupaciones transversales, y las relaciones entre ellos.

Los enfoques orientados por aspectos facilitan la identificación, separación y clasificación de funcionalidades y la composición de funcionalidades transversales.

En la Figura 30 se muestra como ejemplo, una lista de requerimientos para un cajero automático:

Id_Req	Descripción del Requerimiento
R_001	Los usuarios pueden retirar efectivo de su cuenta bancaria
R_002	Los usuarios pueden depositar efectivo en su cuenta bancaria
R_003	El cajero deberá validar que la tarjeta pertenezca al banco
R_004	El cajero debe validar que el NIP dado por el usuario sea correcto
R_005	El usuario no puede retirar más de lo permitido por el banco
R_006	El usuario solo puede retirar cantidades múltiplos de 100
R_007	El usuario no puede retirar más de lo disponible en su cuenta
R_008	El cajero debe regresar la tarjeta que el usuario introdujo
R_009	El cajero debe entregar el efectivo al usuario
R_010	El usuario puede consultar su saldo
R_011	Para hacer una operación se debe validar la tarjera y el NIP

Figura 30. Lista de requerimientos para un cajero automático

Existen dos tipos de relaciones entre temas:

- Relación Base: Pueden compartir un poco de estructura y el comportamiento con otros temas.
- Relación transversal (Aspecto): Su comportamiento que se superpone a la funcionalidad de los temas base.

Para el cajero automático se pueden observar en la figura 31 que las relaciones base son las que tienen los temas Retirar, Consultar, Depositar y Validar con los requerimientos R_001,R_002, R_003, R_004, R_005, R_006, R_007, R_009, R_010 y R_011 respectivamente ya que no tienen relación con más de un tema.

Las relaciones transversales se dan entre el tema Validar y los temas Retirar, Depositar y Consultar, así como, el R_008 con los temas Retirar, Depositar y Consultar. Esto se debe a que tanto el R_008 y el tema Validar son referidos por más de un tema.

Es en la representación gráfica (ver Figura 32) en donde se vincularán los requerimientos con los temas que se han identificado y sus relaciones.

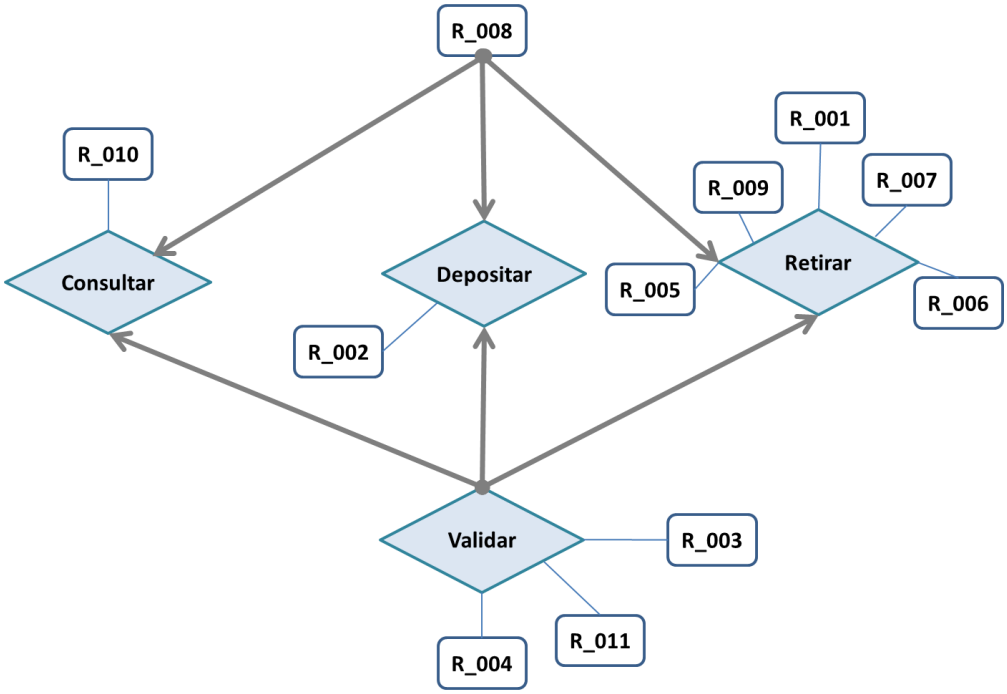

Figura 31. Representación gráfica de los temas y requerimientos de un cajero automático

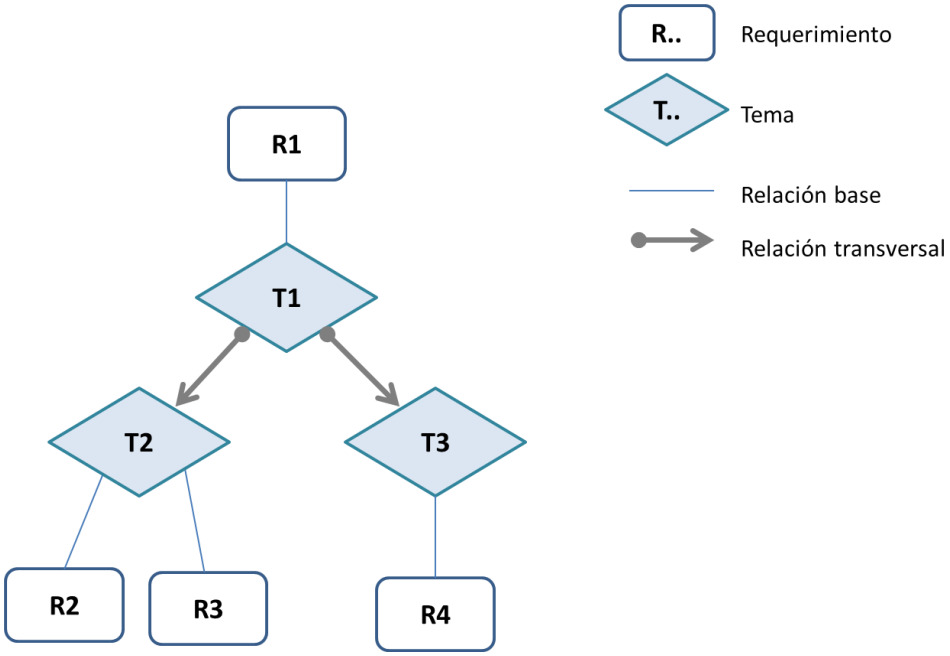

Figura 32. Ejemplo de representación gráfica de requisitos y temas

3.5.4 Especificación de requerimientos utilizando *temas*

Para documentar temas se necesita una tabla con todos los requerimientos del sistema junto con su identificador que debe ser corto para mejor manejo (ver Figura 33).

Id_Req	Descripción del Requerimiento

Figura 33. Plantilla de tabla para requerimientos los requerimientos del sistema

Posteriormente se tiene que hacer el diagrama de relaciones de los requerimientos con los temas que se identificaron, como se muestra en la Figura 34.

Adicionalmente se puede hacer por tema una tabla que contenga los temas y requerimientos relacionados (ver Figura 34):

Id_Tema:	**Nombre:**
Requerimientos	
Id_Req	**Descripción**
Temas	
Relación Base con..	
Id_Tema	**Nombre**
Relación transversal con..	
Id_Tema	**Nombre**

Figura 34. Plantilla para documentar temas

En la Figura 35 se muestra un ejemplo del tema Retirar:

Id_Tema: T_002	**Nombre:** Retirar
Requerimientos	
Id_Req	Descripción
R_001	Los usuarios pueden retirar efectivo de su cuenta bancaria
R_005	El usuario no puede retirar más de lo permitido por el banco
R_006	El usuario solo puede retirar cantidades múltiplos de 100
R_007	El usuario no puede retirar más de lo disponible en su cuenta
R_009	El cajero debe entregar el efectivo al usuario
Temas	
Relación Base con..	
Id_Tema	Nombre
Relación transversal con..	
Id_Tema	**Nombre**
T_001	Validar

Figura 35. Ejemplo plantilla para documentar temas

3.1.5 Validación

La validación los *temas* se pueden hacer mediante pruebas de aceptación donde se determine por cada requerimiento cual será la condición que indique que un requerimiento

esta completado, en la Figura 36 se muestra una plantilla para realizar las pruebas de los requerimientos del sistema.

Id_Tema:*[identificador del tema]*	**Nombre:***[Nombre con el cual se identificará el tema]*
Requerimientos	
Id_Req	**Prueba de Aceptación**
[identificador del requerimiento]	*[Lista de condiciones a cumplirse para decir que un requerimiento está cubierto]*

Figura 36. Plantilla para prueba de requerimientos

3.1.6 Verificación

Para verificar los *temas* se agregó a la plantilla para la validación una columna donde se deberá indicar si la prueba de aceptación, Figura 37.

Id_Tema: *[identificador del tema]*		**Nombre:** *[Nombre con el cual se identificará el tema]*
Requerimientos		
Id_Req	**Estatus**	**Prueba de Aceptación**
[identificador del requerimiento]	*[Cumplió/No cumplió]*	*[Lista de condiciones a cumplirse para decir que un requerimiento está cubierto]*

Figura 37. Plantilla con verificación de requerimientos de un *tema*

Capítulo 4. Comparación de técnicas para la especificación, validación y verificación de requerimientos

Para realizar la comparación de técnicas para la especificación, validación y verificación de requerimientos se tomará como marco el *Alpha Requirements* de *Essence* pues cada uno de sus estados cuentan con un conjunto de puntos de verificación que expresan lo que se debe alcanzar en el estado. Por lo que es necesario identificar que estados serán de utilidad para realizar la comparación, en la Tabla 1 se muestra un análisis breve de los puntos de verificación de cada estado.

De lo concluido en el análisis de los puntos de verificación de cada estado del *Alpha Requirements* de *Essence* se ha decidido que para la comparación solo se utilizarán los estados *Coherente* y *Aceptable* pues las técnicas para la especificación, validación y verificación de requerimientos no cubren con los demás estados.

4.1 Puntos de verificación para la comparación

La lista completa con los puntos de verificación de los estados Coherente y Aceptable donde se compararán las técnicas para la especificación, validación y verificación de requerimientos es la siguiente:

Coherente

1. Los requerimientos están capturados y compartidos con el equipo de trabajo e interesados
2. La procedencia de los requerimientos es clara
3. La razón detrás de los requerimientos es clara
4. Los requerimientos en conflicto están identificados y atendidos
5. Los requerimientos comunican las características esenciales del sistema que será liberado
6. Los escenarios de uso más importantes para el sistema se pueden explicar
7. La prioridad de los requerimientos es clara
8. Se entiende el impacto de implementar los requerimientos
9. El equipo entiende lo que tiene que ser entregado y se compromete a entregarlo.

Aceptable

1. Los interesados aceptan que los requerimientos describen una solución aceptable
2. La velocidad de cambio de los requerimientos acordados es relativamente baja y está bajo control
3. El valor proporcionado al implementar los requerimientos es claro
4. Las áreas de oportunidad satisfechas por los requerimientos son claras
5. Los requerimientos son verificables

4.2 Procedimiento de Comparación

Para realizar la comparación primero se hará una evaluación individual de cada técnica seleccionada y las plantillas propuestas en el trabajo, indicando si éstas cumplen o no con cada uno de los puntos de verificación de los estados Coherente y Aceptable del *Alpha Requirements* de *Essence*.

Después se procederá a unir los resultados de las evaluaciones individuales para medir el porcentaje en el que cumple cada técnica los estados Coherente y Aceptable tanto individualmente como en conjunto.

Estado	Puntos de verificación	Descripción
Concebido	○ El conjunto inicial de interesados está de acuerdo en que se debe desarrollar el sistema ○ Se identifican los interesados que va a utilizar el nuevo sistema ○ Se identifican los interesados que financiarán los trabajos iniciales del nuevo sistema ○ Hay una clara oportunidad para el nuevo sistema.	Con estos puntos se verificará que exista una oportunidad para un nuevo sistema, además de identificar quienes serán los interesados y quien a portará el financiamiento del nuevo sistema. Lo cual pertenece a etapas previas a la especificación de requerimientos
Acotado	○ Se identifican los interesados involucrados en el desarrollo del nuevo sistema ○ Los interesados están de acuerdo con el objetivo del nuevo sistema ○ Es claro cuál es el éxito para el nuevo sistema ○ Las interesados tienen un entendimiento compartido del alcance de la solución propuesta ○ La forma en que se describirán los requerimientos está acordada ○ Los mecanismos para el manejo de los requerimientos se establecen ○ El esquema de priorización es claro ○ Las restricciones están identificadas y consideradas ○ Los supuestos están claramente establecidos	Con estos puntos se verificará que se haya determinado el alcance del sistema y si se ha establecido la planeación del trabajo. Lo cual pertenece a etapas previas a la especificación de requerimientos.
Coherente	○ Los requerimientos están capturados y compartidos con el equipo de trabajo e interesados ○ La procedencia de los requerimientos es clara ○ La razón detrás de los requerimientos es clara ○ Los requerimientos en conflicto están identificados y atendidos ○ Los requerimientos comunican las características esenciales del sistema que será liberado ○ Los escenarios de uso más importantes para el sistema se pueden explicar	Con estos puntos se verificará que los requerimientos del sistema sean claros y que el equipo los entienda.

Estado	Puntos de verificación	Descripción
	○ La prioridad de los requerimientos es clara ○ Se entiende el impacto de implementar los requerimientos ○ El equipo entiende lo que tiene que ser entregado y se compromete a entregarlo.	
Aceptable	○ Los interesados aceptan que los requerimientos describen una solución aceptable ○ La velocidad de cambio de los requerimientos acordados es relativamente baja y está bajo control ○ El valor proporcionado al implementar los requerimientos es claro ○ Las áreas de oportunidad satisfechas por los requerimientos son claras ○ Los requerimientos son verificables	Con estos puntos de verificación se encargarán de verificar que de los requerimientos descritos para el sistema sea han implementado correctamente y dan valor al sistema.
Tratado	○ Hay suficientes requerimientos tratados para que el sistema resultante sea aceptado por los interesados ○ Los interesados aceptan que los requerimientos reflejan con mayor precisión lo que hace y no el sistema ○ El conjunto de requerimientos implementados proporcionar un claro valor a los interesados ○ El sistema que implementa los requerimientos del sistema es aceptado por los interesados y puede ser operativo	Con estos puntos de verificación se determinará si se han implementado suficientes requerimientos para satisfacer las necesidades de un nuevo sistema y si los interesados está de acuerdo con lo que se desarrollo
Cumplido	○ Los interesados aceptan que los requerimientos capturan con precisión lo necesario para satisfacer plenamente la necesidad de un nuevo sistema ○ No hay requerimientos pendientes que eviten que el sistema sea aceptado para cumplir completamente con las necesidades ○ El sistema es aceptado por los interesados por cumplir completamente las necesidades	Con estos puntos de verificación se comprobará que no existan requerimientos pendientes y que el sistema sea aceptado por que cumple plenamente con los requerimientos de los interesados.

Tabla 1. Análisis de los puntos de verificación de cada estado del *Alpha Requirements* de *Essence*

4.3 Evaluación individual

En este apartado se hará la evaluación individual de cada técnica, en este trabajo se han propuesto plantillas con la cual apoyarla la utilización cada técnica, en cada evaluación se mostrara una tabla con cuatro columnas en donde se indicará el Estado evaluado (Coherente y Aceptable), el número del punto de verificación evaluado, Si cumple ✓ o No ✗ y una breve explicación del por qué.

4.3.1 Casos de Uso y Casos de prueba

Los casos de uso permiten modelar el comportamiento de un sistema, mediante relación de un conjunto de casos de uso y los actores del sistema y con la ayuda de los casos de prueba se verifican los requerimientos descritos en un caso de prueba. En la Tabla 2 se muestra la comparación de los elementos que permiten identificar estas técnicas y los puntos de verificación de los estados Coherente y Aceptable.

Estado	Punto de Verificación	Cumple Si/No	Explicación
Coherente	1	✓	La técnica permite capturar los requerimientos del sistema pero es tarea del equipo que estos sean compartidos adecuadamente.
	2	✗	La técnica indica que se debe registrar quien fue la fuente de información con la que se describió el requerimiento.
	3	✗	La técnica no permite identificar fácilmente cual es la razón detrás del requerimiento.
	4	✗	Es tarea del equipo de trabajo identificar y atender los requerimientos en conflicto.
	5	✓	La técnica permite describir las características esenciales de los requerimientos.
	6	✓	Con la técnica se pueden explicar los escenarios de uso más importantes del sistema.
	7	✓	La plantilla propuesta permite indicar la prioridad del requerimiento descrito.
	8	✗	Es tarea del equipo entender el impacto de implementar los requerimientos y la técnica debe apoyar para que este entendimiento se dé.
	9	✗	El equipo debe entender lo que tiene que ser entregado y comprometerse a entregarlo.
Aceptable	1	✗	La técnica no abarca la aceptación de los requerimientos.
	2	✗	El equipo debe controlar la velocidad de cambio de los requerimientos acordados.
	3	✗	El equipo de trabajo y los interesados deben entender del valor que proporcionan los requerimientos al implementarlos
	4	✗	El equipo de trabajo y los interesados deben identificar las áreas de oportunidad satisfechas por los requerimientos.
	5	✓	La técnica permite verificar los requerimientos que se describieron con ella por medio de los casos de prueba.

Tabla 2. Evaluación de técnicas de Casos de Uso y Casos de prueba con estados Coherente y Aceptable del *Alpha Requirements* de *Essence*

4.3.2 Historias de usuario y Pruebas de aceptación

Las Historias de usuario deben de describir las funcionalidades de mayor valor para los usuarios de una manera ligera y con ayuda de las pruebas de aceptación se podrán verificar.

En la Tabla 3 se muestra la comparación de los elementos que permiten identificar estas técnicas y los puntos de verificación de los estados Coherente y Aceptable.

Estado	Punto de Verificación	Cumple Si/No	Explicación
Coherente	1	✓	La técnica permite capturar los requerimientos del sistema pero es tarea del equipo que estos sean compartidos adecuadamente.
	2	✓	La técnica indica que se debe registrar el dueño de la historia.
	3	✓	La técnica permite identificar cual es la razón detrás del requerimiento.
	4	✗	Es tarea del equipo de trabajo identificar y atender los requerimientos en conflicto.
	5	✓	La técnica permite describir las características esenciales de los requerimientos.
	6	✓	Con la técnica se pueden explicar los escenarios de uso más importantes del sistema.
	7	✗	La plantilla propuesta no permite indicar la prioridad del requerimiento descrito.
	8	✗	Es tarea del equipo entender el impacto de implementar los requerimientos y la técnica debe apoyar para que éste entendimiento se de.
	9	✗	El equipo debe entender lo que tiene que ser entregado y comprometerse a entregarlo.
Aceptable	1	✗	La técnica no abarca la aceptación de los requerimientos.
	2	✗	El equipo debe controlar la velocidad de cambio de los requerimientos acordados.
	3	✗	El equipo de trabajo y los interesados deben entender del valor que proporcionan los requerimientos al implementarlos
	4	✗	El equipo de trabajo y los interesados deben identificar las áreas de oportunidad satisfechas por los requerimientos.
	5	✓	La técnica permite verificar los requerimientos que se describieron con ella por medio de los criterios de aceptación.

Tabla 3. Evaluación de técnicas de Historias de usuario y Pruebas de aceptación con estados Coherente y Aceptable del *Alpha Requirements* de *Essence*

4.3.3 Casos de Uso 2.0 y Pruebas de aceptación

Los Casos de uso 2.0 al mezclar elementos de los Casos de uso y las Historias de Usuario permitirán modelar requerimientos livianos, escalables y fáciles de manejar por su tamaño y con ayuda de las Pruebas de aceptación se debe verificar cada caso de Uso 2.0.

En la Tabla 4 se muestra la comparación de los elementos que permiten identificar estas técnicas y los puntos de verificación de los estados Coherente y Aceptable.

Estado	Punto de Verificación	Cumple Si/No	Explicación
Coherente	1	✓	La técnica permite capturar los requerimientos del sistema pero es tarea del equipo que estos sean compartidos adecuadamente.
	2	✓	La técnica indica que se debe registrar el dueño de la historia.
	3	✗	No es tan fácil identificar cual es la razón detrás del requerimiento.
	4	✓	La plantilla permite identificar su estado pero es tarea del equipo atender los requerimientos en conflicto.
	5	✓	La técnica permite describir las características esenciales de los requerimientos.
	6	✓	Con la técnica se pueden explicar los escenarios de uso más importantes del sistema.
	7	✓	La plantilla propuesta permite indicar la prioridad del requerimiento descrito.
	8	✓	La técnica requiere indicar la complejidad del requerimiento lo cual ayudara a que el equipo comprenda el impacto de su implementación.
	9	✗	El equipo debe entender lo que tiene que ser entregado y comprometerse a entregarlo.
Aceptable	1	✗	La técnica no abarca la aceptación de los requerimientos.
	2	✗	El equipo debe controlar la velocidad de cambio de los requerimientos acordados.
	3	✗	El equipo de trabajo y los interesados deben entender del valor que proporcionan los requerimientos al implementarlos
	4	✗	El equipo de trabajo y los interesados deben identificar las áreas de oportunidad satisfechas por los requerimientos.
	5	✓	La técnica permite verificar los requerimientos que se describieron con ella por medio de los criterios de aceptación.

Tabla 4. Evaluación de técnicas de Casos de Uso 2.0 y Pruebas de aceptación con estados Coherente y Aceptable del *Alpha Requirements* de *Essence*

4.3.4 Features y Casos de prueba

Features representa al sistema como una relación de características y apoyado de los casos de prueba se verificarán las características de este.

En la Tabla 5 se muestra la comparación de los elementos que permite identificar esta técnica y los puntos de verificación de los estados Coherente y Aceptable.

Estado	Punto de Verificación	Cumple Si/No	Explicación
Coherente	1	✓	La técnica permite capturar los requerimientos del sistema pero es tarea del equipo que estos sean compartidos adecuadamente.
	2	✗	La técnica indica que se deba registrar quien fue la fuente de información con la que se describió el requerimiento.
	3	✗	La técnica no permite identificar fácilmente cual es la razón detrás del requerimiento.
	4	✗	Es tarea del equipo de trabajo identificar y atender los requerimientos en conflicto.
	5	✓	La técnica permite describir las características esenciales de los requerimientos.
	6	✓	Con la técnica se pueden explicar los escenarios de uso más importantes del sistema.
	7	✓	La plantilla propuesta permite indicar la prioridad del requerimiento descrito.
	8	✗	Es tarea del equipo entender el impacto de implementar los requerimientos y la técnica debe apoyar para que este entendimiento sé de.
	9	✗	El equipo debe entender lo que tiene que ser entregado y comprometerse a entregarlo.
Aceptable	1	✗	La técnica no abarca la aceptación de los requerimientos.
	2	✗	El equipo debe controlar la velocidad de cambio de los requerimientos acordados.
	3	✗	El equipo de trabajo y los interesados deben entender del valor que proporcionan los requerimientos al implementarlos
	4	✗	El equipo de trabajo y los interesados deben identificar las áreas de oportunidad satisfechas por los requerimientos.
	5	✓	La técnica permite verificar los requerimientos que se describieron con ella por medio de los casos de prueba.

Tabla 5. Evaluación de técnicas de *Features* y Casos de prueba con estados Coherente y Aceptable del *Alpha Requirements* de *Essence*

4.3.05 Theme y Pruebas de aceptación

Theme ve los requerimientos como temas los cuales representan funcionalidades del sistema y con apoyo de las pruebas de aceptación se probaran los temas.

En la Tabla 6 se muestra la comparación de los elementos que permite identificar esta técnica y los puntos de verificación de los estados Coherente y Aceptable.

Theme y Pruebas de aceptación			
Estado	**Punto de Verificación**	**Cumple Si/No**	**Explicación**
Coherente	1	✓	La técnica permite capturar los requerimientos del sistema pero es tarea del equipo que estos sean compartidos adecuadamente.
	2	✗	La técnica indica que se deba registrar quien fue la fuente de información con la que se describió el requerimiento.
	3	✗	La técnica no permite identificar fácilmente cual es la razón detrás del requerimiento.
	4	✗	Es tarea del equipo de trabajo identificar y atender los requerimientos en conflicto.
	5	✓	La técnica permite describir las características esenciales de los requerimientos.
	6	✗	La técnica no se enfoca en escenarios.
	7	✗	La técnica no requiere de indicar la prioridad del requerimiento descrito.
	8	✗	Es tarea del equipo entender el impacto de implementar los requerimientos y la técnica debe apoyar para que este entendimiento sé de.
	9	✗	El equipo debe entender lo que tiene que ser entregado y comprometerse a entregarlo.
Aceptable	1	✗	La técnica no abarca la aceptación de los requerimientos.
	2	✗	El equipo debe controlar la velocidad de cambio de los requerimientos acordados.
	3	✗	El equipo de trabajo y los interesados deben entender del valor que proporcionan los requerimientos al implementarlos
	4	✗	El equipo de trabajo y los interesados deben identificar las áreas de oportunidad satisfechas por los requerimientos.
	5	✓	La técnica permite verificar los requerimientos que se describieron con ella por medio de los casos de prueba.

Tabla 6. Evaluación de técnicas de Theme y Pruebas de aceptación con estados Coherente y Aceptable del *Alpha Requirements* de *Essence*

4.4 Discusión de los resultados

Primero que nada es importante aclarar que las evaluaciones se realizaron en función de la información que se podía o no registrar en las plantillas propuestas para cada técnica.

En la Tabla 7 se muestra un concentrado de evaluaciones individuales de las técnicas para la especificación, verificación y validación de requerimientos: Casos de Uso, Historias de Usuario, Casos de Uso 2.0, *Features, Theme*, Casos de Prueba y Pruebas de Aceptación con los estados Coherente y Aceptable del *AlphaRequirements*de*Essence*, de lo cual se puede decir lo siguiente:

1. Las técnicas Casos de Uso, Historias de Usuario, Casos de Uso 2.0, *Features* y*Theme* permiten capturar los requerimientos de un sistema y estos comunican las características esenciales que deberá tener éste.
2. Casos de uso y *Features* cubren el estado Coherente de la misma manera ya que las dos técnicas además de capturar requerimientos y comunicar las características esenciales del sistema, permiten explicar los escenarios más importantes del sistema e indican la prioridad de cada requerimiento.
3. Solo la técnica Casos de uso 2.0 permite identificar los requerimientos en conflicto y si éstos están siendo atendidos.
4. La técnica que cubre más puntos del estado Coherente con un 78%, es Casos de uso 2.0 (ver Figura 38).
5. La técnica que cubre menos puntos del estado Coherente con un 22%, es *Theme* (ver Figura 38).
6. Las técnicas Casos de Prueba y Criterios de aceptación que están enfocadas a la validación y verificación de requerimientos solo cubren un 20% del estado Aceptable ya que permiten verificar los requerimientos que se trabajaron con las técnicas de especificación de requerimientos.
7. Los puntos que no fueron cubiertos por ninguna de las técnicas para la especificación, verificación y validación de requerimientos fueron:

   Estado Coherente:

   a. El equipo entiende lo que tiene que ser entregado y se compromete a entregarlo.

   Estado Aceptable:

   a. Los interesados aceptan que los requerimientos describen una solución aceptable.
   b. La velocidad de cambio de los requerimientos acordados es relativamente baja y está bajo control.
   c. El valor proporcionado al implementar los requerimientos es claro.
   d. Las áreas de oportunidad satisfechas por los requerimientos son claras.

   Estos puntos no son cubiertos por las técnicas para la especificación, verificación y validación de requerimientos debido a que estos están relacionados con las habilidades que debe tener el equipo de trabajo. La forma que el equipo interpreta los requerimientos y los implementa; y si los interesados están de acuerdo con ellos. Las técnicas solo ayudarán al equipo de trabajo a identificar que requerimientos y si estos satisfacen las necesidades del usuario.

Estado	Punto de Verificación	Técnica				
		Casos de uso y Casos de prueba	**Historias de usuario y Criterios de aceptación**	**Casos de uso 2.0 y Criterios de aceptación**	**Feature y Criterios de aceptación**	**Theme y Casos de prueba**
Coherente	1	✓	✓	✓	✓	✓
	2	✗	✓	✓	✗	✗
	3	✗	✓	✗	✗	✗
	4	✗	✗	✓	✗	✗
	5	✓	✓	✓	✓	✓
	6	✓	✓	✓	✓	✗
	7	✓	✗	✓	✓	✗
	8	✗	✗	✓	✗	✗
	9	✗	✗	✗	✗	✗
Porcentaje de cumplimiento estado Coherente		44.44	55.56	77.78	44.44	22.22
Aceptable	1	✗	✗	✗	✗	✗
	2	✗	✗	✗	✗	✗
	3	✗	✗	✗	✗	✗
	4	✗	✗	✗	✗	✗
	5	✓	✓	✓	✓	✓
Porcentaje de cumplimiento estado Aceptable		10	10	10	10	10
Porcentaje de cumplimiento de los dos estado		35.71	42.86	57.14	35.71	21.43

Tabla 7. Concentrado de evaluaciones individuales de técnicas de Casos de Uso, historias de Usuario, Casos de Uso 2.0, *Features, Theme,* Casos de Prueba y Pruebas de Aceptación con los estados Coherente y Aceptable del *Alpha Requirements* de *Essence*

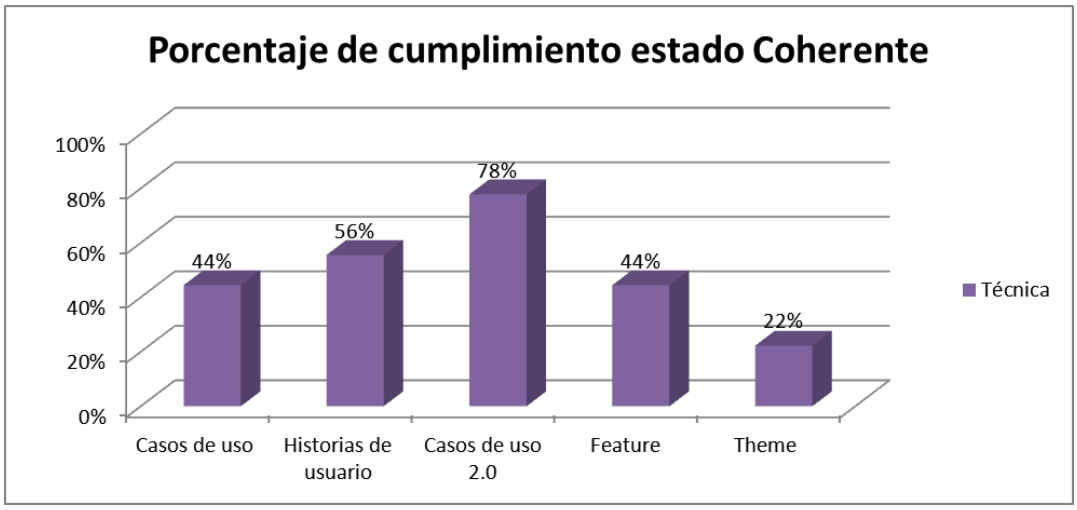

Figura 38. Cumplimiento del Estado Coherente

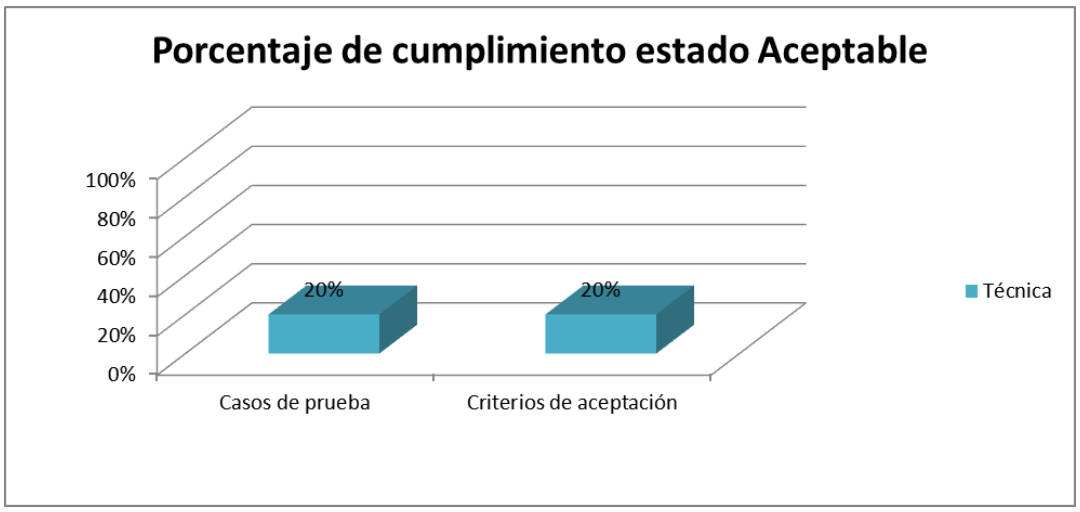

Figura 39. Cumplimiento del Estado Aceptable

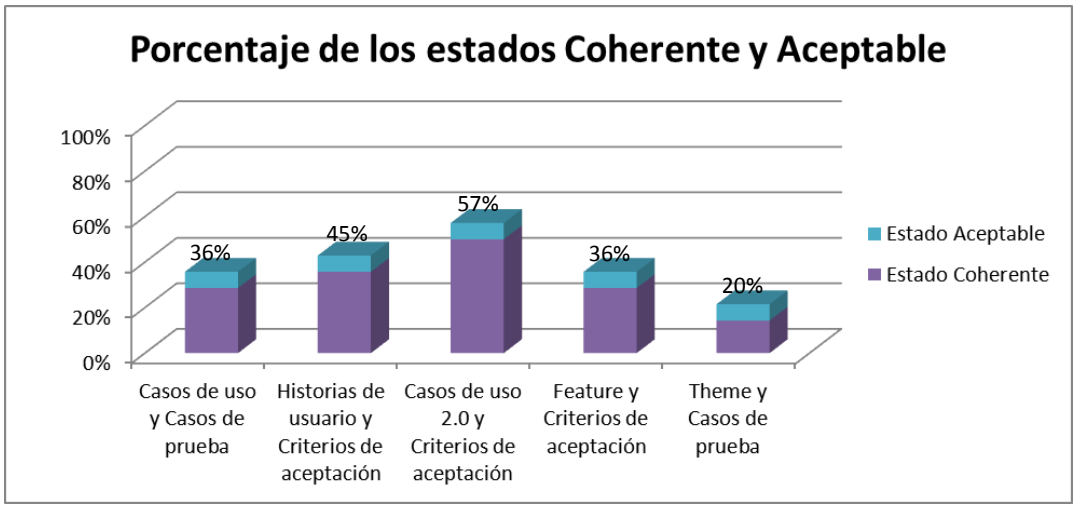

Figura 40. Cumplimiento de los estados Coherente y Aceptable

Conclusiones

Según el objetivo de esta tesis:

Realizar la comparación de las técnicas de Casos de Uso, Historias de usuario, Caso de Uso 2.0, *Features* y *Theme* para la especificación, verificación y validación de requerimientos; y estructurarlas utilizando como marco común el *Alpha Requirements* de *Essence*. Con lo cual se aporten elementos que faciliten la selección de una técnica para la especificación, verificación y validación de requerimientos.

Para cumplir este objetivo fue necesario primero entender que es lo que se iba a comparar y con qué criterios se realizaría esta comparación por lo que se realizaron estas actividades:

- Investigación a fondo del *Alpha Requirements* de *Essence*, así como de cada uno de sus estados y sus puntos de verificación; esto para entender perfectamente el marco y realizar correctamente la comparación.
- Investigación de las técnicas para la especificación, verificación y validación de requerimientos: Casos de Uso, Historias de Usuario, Casos de Uso 2.0, *Features, Theme*, Casos de Prueba y Pruebas de Aceptación, para entender que permite modelar cada técnica y como se trabaja con ella. Como resultado de la investigación de las técnicas se proponen en este trabajo diferentes plantillas para cada una de las técnicas, las cuales facilitaron la comparación y sirven de apoyo para la utilización de las técnicas.

Posteriormente se procedió a realizar la comparación de las técnicas de Casos de Uso, Historias de usuario, Caso de Uso 2.0, *Features* y *Theme* para la especificación, verificación y validación de requerimientos; utilizando los estados Coherente y Aceptable del *Alpha Requirements* de *Essence*.

Con esta comparación se identificaron los puntos débiles y fuertes de cada técnica con los cuales se aportan elementos que facilitan la selección consciente de alguna de ellas para trabajar en especificación, verificación y validación de requerimientos de un sistema.

Trabajo a futuro

Como trabajo a futuro se identificaron los siguientes puntos:

- Integrar otras técnicas de ingeniería de requerimientos que permitan cubrir puntos que no fueron cubiertos con las técnicas comparadas en este trabajo.
- Identificar que técnicas permiten cumplir con los puntos de verificación de otros estados del *Alpha Requirements* de *Essence* que no se evaluaron y en general para todas las *Alphas* de la propuesta *Essence*.

Referencias

[1] IEEE. SWEBOK V3.0, Guide to the Software Engineering Body of Knowledge. IEEE Computer Society, 2004.

[2] Jiang, L., Eberlein A., Far B. H. y Mousavi M. (2008). A methodology for the selection of requirements engineering techniques.

[3] Software Engineering Method and Theory, consultado el día 21 de julio del 2013 del sitio http://www.semat.org/

[4] Object Management Group, consultado el día 23 de agosto del 2013 del sitio http://www.omg.org/

[5] Jacobson I., Meyer B. y Soley R. (2009). Call for action: The SEMAT initiative. Dr. Dobb's Journal.

[6] Jacobson, I., Pan-Wei Ng, McMahon, P., Spence, I. y Lidman S. (2012). The Essence of software engineering: the SEMAT kernel. ACM Queue.

[7] Sommerville I. Software Engineering, 9th ed. Addison-Wesley, 2011.

[8] Boehm, B.W. Gestión de Riesgos Software. IEEE Computer Society Press, 1989.

[9] IEEE-STD-610.121990. IEEE Standard Glossary of Software Engineering Terminology, 1990.

[10] ISO 9000.

[11] Jacobson, I., Pan-Wei Ng, McMahon, P., Spence, I. and Lidman S. (2013). The Essence of Software Engineering—Applying the SEMAT Kernel. Addison-Wesley.

[12] Jacobson, I., Christerson, M., Jonsson P., and OvergaardG. Object-Oriented Software Engineering—A Use Case Driven Approach. Addison-Wesley, 1992.

[13] Morales, M. El proceso de desarrollo y mantenimiento de software propuesto por COMPETISOFT de acuerdo al proceso unificado.2010.

[14] Leffingwell, D., Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley, 2011.

[15] Cohn, M., User Stories Applied for Agile Software Development. Addison-Wesley, 2004.

[16] Jacobson, I., Spence, I. and Bittner, K. Use-Case 2.0. The Guide to Succeeding with Use.Ivar Jacobson International SA, 2005.

[17] Pereira, R., Insfran, E., Abrahão, S., Gonzalez-Huerta, J., Blanes, D., Cohen, S., Santana de Almeida, E. A Feature-Driven Requirements Engineering Approach for Software Product Lines.VII Brazilian Symposium on Software Components, Architectures and Reuse, 2013.

[18] Clarke, S. and Baniassad, E. Aspected-Oriented Analysis and Design. The Theme Aproach. Addison Wesley, 2005
Traducción: Jacobson, I., Pan-Wei Ng, McMahon, P., Spence, I. y Lidman S. (2012). The Essence of software engineering: the SEMAT kernel. De: Zapata, C. La Esencia de la Ingeniería de Software: El Núcleo de Semat, 2013.