UNIVERSIDAD AUTONOMA DE SINALOA

POSGRADO EN CIENCIAS DE LA INFORMACIÓN

FACULTAD DE INFORMÁTICA CULIACÁN

Tesis para obtener el grado de:

Maestría en Ciencias de la Información

Título de Tesis:

Aproximación Metodológica para la práctica de la Ingeniería de Requisitos en las pequeñas y medianas empresas (PyMES) de desarrollo de software en el estado de Sinaloa

Que presenta:

Elda Lizbeth Zamudio Vizcarra

Directores:

Dr. José Alfonso Aguilar Calderón

Dra. Carolina Tripp Barba

Culiacán de Rosales, Sinaloa.

INDICE GENERAL

INDICE DE FIGURAS

Página

INDICE DE TABLAS

Página

Resumen

La correcta aplicación de la ingeniería de software para el proceso de desarrollo de un sistema da como resultado un producto de calidad que garantiza la información producida, en esta investigación se muestra que una de las etapas más importantes de este proceso es la Ingeniería de Requisitos (RE), en donde se describen las características y restricciones con las que debe contar el sistema.

En este trabajo se realizó una revisión bibliográfica sobre métodos, técnicas, metodologías y herramientas para la aplicación de las actividades de la RE, incluyendo aquellas desarrolladas ad-hoc para su implementación. También se describe la situación actual de las fábricas de software obtenida gracias a la aplicación de un cuestionario (ver anexo A) a 25 empresas del estado de Sinaloa para conocer las técnicas de RE que aplican en su proceso de desarrollo de software.

Finalmente, se presenta el desarrollo de una aproximación metodológica ad-hoc para el proceso de RE en las fábricas de software del estado de Sinaloa clasificadas como PyMES.

Palabras Claves

Ingeniería de Requisitos, Ingeniería de Software, Técnicas de Ingeniería de Requisitos, Entrevista, Fábricas desarrolladoras de Software, PyMES, Metodologías Aplicadas en el uso de la Ingeniería de Requisitos, Métodos, Técnicas y Herramientas aplicadas al desarrollo de software, Metodologías Ágiles, Metodologías Tradicionales.

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por el apoyo económico otorgado mediante una beca, la cual me permitió culminar de forma satisfactoria mis estudios de maestría.

A mi familia, por su apoyo y comprensión, por la paciencia y el tiempo y por estar ahí motivándome para cumplir mis objetivos.

A mis directores de tesis el Dr. José Alfonso Aguilar Calderón y la Dra. Carolina Tripp Barba gracias por su tiempo, por la dedicación, por su apoyo y confianza en mí, por sus consejos, por la motivación para seguir adelante y sobre todo el compromiso que junto a mi realizó para sacar adelante este trabajo, les aprendí mucho.

Gracias a mis maestros, por motivarme a seguir adelante y no rendirme, por todos los conocimientos otorgados, por su ayuda y apoyo en este camino.

Dedicatoria

Le dedico esta tesis a 3 de las personas más importantes en mi vida por los que día a día me levanto y doy lo mejor de mí para ser una mejor persona y madre para ellos a mis hijos (Ximena, Pedro Pablo y Alan) que son mi motor para salir adelante, lo más bello y hermoso que la vida me dio, así como también a mi esposo (Pedro) por ser mi otra mitad, mi complemento, mi cómplice y mi apoyo, por junto a mis hijos ser más importante en mi vida, los quiero.

Capítulo I. Introducción

Actualmente, los sistemas computacionales se encuentran presentes en todas las áreas del conocimiento, esto ha permitido optimizar los procesos de trabajo de diferentes sectores productivos automatizando tareas y asistiendo en actividades particulares antes realizadas por el ser humano, por ejemplo, el control de un inventario, por lo tanto, deben de garantizar la confiabilidad de los datos y la información producida. En este sentido, ha surgido un escenario complejo para las organizaciones caracterizado principalmente por mercados dinámicos y rígidos basados en el conocimiento que exigen productos con alto valor. Esto se debe principalmente a que, partiendo de los datos que el sistema genera, los usuarios toman decisiones y acciones que afectan directamente los beneficios de su negocio.

1.1. Motivación

La evolución constante de las tecnologías de la información y de la comunicación (TIC's) ha ocasionado que los sistemas de información (SI) se encuentren presentes en distintas actividades humanas en diferentes sectores productivos como el industrial, gubernamental, comercial, educación y entretenimiento. Los SI son una combinación de datos, hardware y software que, a través de un proceso de tres etapas: entrada, procesamiento y salida de datos, generan información de utilidad para una empresa o negocio. Dada su transversalidad, constituye la base del crecimiento de todas las economías modernas y genera una mayor competitividad de la misma. La conceptualización, desarrollo, implementación y mantenimiento de los SI se realiza a través de una rama de la economía global que se conoce como la industria del software, la cual es un sector impulsor del desarrollo económico fundamentado en la competitividad y la generación de empleos bien remunerados, con infraestructura y costos de capital relativamente bajos. El desarrollo de esta tiene un impacto significativo en otras actividades al estimular la absorción de tecnologías de la información con el consecuente incremento en la productividad global. En México, la industria del software se considera como la producción de mayor crecimiento en los últimos

años. Esto se debe en gran medida a las políticas públicas de apoyo, entre las que destaca la Ley de Promoción de Software (No. 25.922) la cual otorga beneficios fiscales a las empresas exportadoras [1]. Para acceder a estos beneficios, entre otros requisitos, las empresas deben certificar la calidad, es decir, una mejora en los procesos de desarrollo de software y con ello mejorar el posicionamiento de nuestro país a nivel internacional, pues actualmente ocupa el número 11 a nivel mundial en el ranking de empresas de software certificadas en calidad. La Ley de Promoción del Software fortifica el hecho de que el desarrollo de software debe ser un proceso controlado y planeado que considere las necesidades de los usuarios y clientes que solicitan lo que el SI debe y no debe hacer, debido a que los errores que se comentan durante la elaboración del producto pueden causar daño y/o pérdida de la información, lo cual trasciende en las decisiones que tomen los usuarios del sistema.

1.2. Problemática

Debido a la importancia de los SI en la economía global, hoy en día se ha producido un interés creciente por el desarrollo de propuestas metodológicas que ofrezcan un marco de referencia adecuado cuando se desarrolla un SI en una fábrica de software (empresa dedicada al desarrollo de software). En este sentido, han sido varias propuestas que han surgido enfocándose en métodos que ofrecen procesos, modelos y técnicas adecuadas para trabajar con este tipo de desarrollos [2]. Lamentablemente, las propuestas, en su mayoría, van dirigidas a la etapa de diseño del ciclo de vida, dejando de lado cuestiones como el reflejo de las necesidades del cliente en el producto final, los objetivos organizacionales de la empresa, entendimiento de las necesidades del usuario final, los procesos que se quieren mejorar a través del SI, la clasificación de la fábrica de software respecto a su capacidad de producción y la capacidad técnica-administrativa del equipo de desarrollo. Formalmente, ésta problemática se atiende a través de una rama de la Ingeniería de Software (*Software Engineering*, SE, por sus siglas en inglés) conocida como Ingeniería de Requisitos (*Requirements Engineering*, RE, por sus siglas en inglés). La RE, se encuentra en las etapas iniciales del proceso de desarrollo del software, su correcta realización es esencial a razón de que los errores más comunes, costosos de reparar y que más tiempo consumen, como el re-trabajo (acción

tomada sobre un producto no conforme de modo que satisfaga los requisitos especificados[]), se deben a un inadecuado cumplimiento de los requisitos, es decir, a no utilizar un método, técnica, metodología o procedimiento ad-hoc a la fábrica de software que guíe al equipo de desarrollo durante la obtención de las necesidades del cliente.

Los requisitos son las descripciones de los servicios que el sistema debe proporcionar, así como las restricciones operativas que este debe de tener, por su parte, se enfocan en las necesidades operacionales del producto software a desarrollar, de infraestructura, hardware o de usuarios. Estos reflejan las condiciones de los clientes para poder ser implementadas en un producto software. Éste proceso utiliza una combinación de técnicas, métodos y herramientas para obtener como resultado final el documento de especificación de los requisitos.

El documento sirve para negociar y validar las peticiones que se hayan hecho para el sistema, también como un contrato formal entre el cliente y el equipo de desarrollo que permite garantizar las características y funcionalidades del producto software a desarrollar. Esta situación se refleja al finalizar el proyecto de desarrollo debido a que cualquier modificación a realizar en el producto final representa una pérdida económica, de tiempo y esfuerzo para la empresa.

En la actualidad, la necesidad de mejorar la calidad de software es alta, al mismo tiempo las tareas para el desarrollo de sistemas han llegado a ser más complejas debido a la constante evolución de las tecnologías de implementación (lenguajes de programación, marcos de trabajo, hardware más potente). En éste sentido, para obtener un proceso de producción sin errores críticos, que permita obtener un producto software de calidad adecuado a las necesidades del cliente, de la empresa, del usuario y que sea entregado en el tiempo estipulado, es importante que el desarrollo de software se lleve a cabo a través de un procedimiento definido ad-hoc, que sea disciplinado y aceptado por todos los integrantes de la fábrica de software de acuerdo a las capacidades del equipo de desarrollo y de las dimensiones de la empresa, lo cual permitirá la correcta aplicación de cada una de las actividades que éste señale.

Para lograrlo, es indispensable comprender la situación actual de la práctica de la RE en la industria del software con el objetivo de mejorar el proceso de desarrollo llevado a cabo en el ámbito profesional [3]. A pesar de la importancia

que tiene la RE, existen cuestiones que deben mejorarse tales como la formalización de la especialización de los requisitos de software (incluyendo el estándar de la ISO), entre otras. Es importante mencionar que la RE, como es de común conocimiento, involucra a clientes, usuarios, equipo de desarrollo, administradores de proyectos, etc., por lo tanto, el proceso no depende solamente de la forma en cómo se perciba el problema, sino también del nivel de experiencia que tengan los involucrados en el sentido de comunicación y trabajo en equipo que debe existir antes y durante el desarrollo del producto. También, es necesario hacer énfasis en el cierre de las brechas que todavía existen, ejemplo, la carencia en herramientas para la administración de requisitos, el dominio de estándares disponibles y la cuestión del diseño de interfaces de usuarios considerando los atributos de calidad del software como usabilidad y accesibilidad y la administración de cambios en los requisitos. Finalmente, a pesar de los esfuerzos derivados de investigaciones en universidades e iniciativa privada, para mejorar la práctica de la RE en pequeñas y medianas empresas (PyMES) de desarrollo de software (fábricas de software) aún persisten cuestiones importantes a resolver relacionadas con la calidad del producto, principalmente la carencia de una aproximación metodológica ad-hoc que permita la optimización de ésta fase considerando la capacidad del equipo de desarrollo y de las fábricas integradas con herramientas para la administración de requisitos.

1.3. Objetivo de la tesis

El trabajo de investigación que presenta esta tesis tuvo como meta realizar una contribución al proceso de desarrollo de las fábricas de software en Sinaloa catalogadas como PyMES, a través de una aproximación metodológica ad-hoc para el proceso de la RE. Esto se logró en base a un análisis del proceso de desarrollo de software, particularmente enfocado en la aplicación de las actividades de la RE (elicitación, análisis, especificación, validación y administración). Para esto, en la tesis se identifican y se presentan las definiciones de las actividades que conforman la RE, así como las técnicas y métodos para la obtención, validación y administración de los mismos. También, se describen los diferentes tipos de requisitos que se encuentra en un proyecto de software. Así mismo, se presenta una propuesta metodológica y las fases

contenidas en esta para una buena elicitación de requisitos para ser aplicada en las PyMES.

1.4. Estructura del documento

El **Capítulo siguiente**, titulado “Marco Teórico-Conceptual”, presenta los fundamentos teóricos y conceptuales que propiciaron la consecución de los objetivos de la tesis. Definiciones de Ingeniería de Software, Ingeniería de Requisitos, proceso de desarrollo del Software, metodologías ágiles, así como las actividades de la Ingeniería de Requisitos las cuales se analizaron y comprendieron a profundidad. Además, se describe la situación actual de las fábricas de software de Sinaloa con respecto a la aplicación de las actividades de la RE, la problemática y la situación actual.

El **capítulo tercero** habla del estado de arte, la situación actual de RE, así como también los trabajos realizados sobre métodos, metodologías, técnicas y herramientas que brinden apoyo a la RE en la etapa de desarrollo.

El **capítulo cuarto** se muestra un estudio realizado a 25 empresas del estado de Sinaloa para de obtener información sobre métodos, técnicas y herramientas de la RE que utilizan para el desarrollo de un proyecto.

El **capítulo número cinco** presenta el núcleo de la tesis, una aproximación metodológica ad-hoc para el proceso de la Ingeniería de Requisitos en las fábricas de software de Sinaloa clasificadas como PyMES. Inicia con la conceptualización de la idea y los fundamentos que se tomaron en cuenta para realizarla. La conclusión y consideraciones finales de la tesis se presentan en el **capítulo número seis**. Finalmente, la tesis se complementa con un glosario de términos, referencias bibliográficas y una serie de anexos como son el A incluye el cuestionario que fue aplicado a las empresas que fueron seleccionadas para el estudio, en el B se observa el Documento de Requisitos de Usuario (DRU), en el que el cliente captura las necesidades del sistema, el anexo C contiene el Documento de Especificación de Requisitos de Software (DERS) en el que se hacen las especificaciones y objetivos a cumplir para el proyecto y el D incluye un catálogo de carencias que se obtuvo de la investigación en las fábricas, además se presentan los resultados del estudio aplicado a una muestra representativa de las PyMES que desarrollan software en Sinaloa.

Capítulo II. Marco Teórico-Conceptual

En éste capítulo se presentan los conceptos teóricos que dieron origen a la investigación realizada en ésta tesis. Se abordan aspectos acerca de en qué consiste la Ingeniería de Software (SE), la Ingeniería de Requisitos (RE), la definición de Requisito Funcional (RF) y de Requisito No Funcional (RNF), las actividades de la RE y las metodologías para el desarrollo de software tradicionales y ágiles enfocados en la etapa de requisitos, así mismo, se presenta una reseña respecto a la problemática que se ha tenido en México, particularmente en Sinaloa en lo que concierne al desarrollo de software, temas indispensables para la generación de conocimiento y la finalización de la investigación efectuada en la tesis.

2.1. Industria del Software

Los cambios de la época actual crean un escenario complejo para las organizaciones caracterizado por mercados dinámicos y exigentes respecto a productos con alto valor agregado y empresas basadas en el conocimiento. La dinámica de este cambio ha originado la aparición de tecnologías genéricas en diversos sectores entre los que destacan las Tecnologías de la Información y la Comunicación (TIC), su importancia se debe a la expansión acelerada y los cambios revolucionarios en el sistema de telecomunicaciones, los procesos vinculados al desarrollo de Internet, la introducción y crecimiento exponencial de las computadoras personales, así como la demanda de una tecnología multifuncional. En este sentido, el registro de la industria nacional del software en México se encuentra en el Directorio Estadístico Nacional de Unidades Económicas (DENUE) [4] establecido en la clasificación 541510, como las unidades económicas dedicadas a proporcionar servicios de diseño de sistemas de cómputo y servicios relacionados entre los que se encuentran integración de hardware, software y tecnologías de comunicación, asesoría de equipo y redes informáticas, administración de centros de cómputo, servicios de instalación de programas y diseño y desarrollo de software a la medida. En el presente año, DENUE tiene un registro de 2853 empresas integradoras de la industria de software registradas en el país. La concentración más importante es en el Distrito

Federal, con 809 empresas, lo que representa un 28.4%, Nuevo León con 323 empresas equivalente a un 11.3% y Jalisco con 227 empresas con una representación del 8%. A Sinaloa le corresponde el 2.2% del total de las empresas nacionales con 63 firmas registradas. En México, la industria del software está integrada por tres segmentos principales: 1) Industria nacional de software y servicios informáticos, 2) Producción interna en las empresas y departamentos de gobierno, 3) Filiales de las grandes.

El software, elemento dual, porque es servicio y producto intangible que no se degrada con el uso, desempeña un papel clave en la reconfiguración actual de nuevas industrias debido a que es indispensable para el procesamiento de datos. El dinamismo de esta tecnología ha generado una industria del sector de servicios y de alta tecnología con una estructura compleja y de gran capacidad de innovación que requiere estrategias competitivas integrales para su desarrollo. La industria del software es un sector impulsor del crecimiento económico que fomenta la competitividad y la generación de empleos bien remunerados, con infraestructura y costos de capital relativamente bajos. Su aumento impacta significativamente en otras actividades económicas, al estimular la absorción de tecnologías de la información, con el incremento en la productividad global [4]. En este sentido, la Ingeniería del Software (SE) es el área que se encarga de la definición y adopción de métodos, técnicas, herramientas y metodologías para la conceptualización, adopción y desarrollo de software.

Dentro de esta industria, salen a relucir las empresas dedicadas al desarrollo del software conocidas con el acrónimo “PyMES” (Pequeñas y Medianas Empresas), el cual es utilizado al clasificar las empresas de acuerdo al número de trabajadores, sin embargo hay que tener en cuenta que este número varía de acuerdo a la región o país que se establezcan [5], por ejemplo, en Irlanda casi el 99% son pequeñas, por lo que emplean menos de 50 personas, en Australia, el 98% de ellas tiene menos de 20 empleados y en México existen aproximadamente 4 millones 15 mil empresas de las cuales 99.8% son PyMES con un rango de entre 1 y 130 empleados, de acuerdo con la Secretaría de Economía [6]. Las empresas que se dedican a la fabricación de software han cobrado importancia en el mundo debido a que favorecen el crecimiento de las economías nacionales [7] a razón de que representan el 99%, en la tabla 1 se

puede observar la distribución que tienen las empresas, de las cuales las micros (menos de 10 empleados) son el 78%, las pequeñas (11 a menos de 50 empleados) con el 16% y las medianas (de 51 a menos de 250 empleados) con el 5% [8].

Microempresas	Pequeñas	Medianas	Grandes	Total	PyMES
78%	16%	5%	1%	100%	99%

Tabla 1: Porcentaje de empresas PyMES que fabrican software a nivel mundial.

La eficiencia de las empresas puede verse opacada por sus problemas particulares, tales como el tiempo de entrega de los productos de software, la calidad esperada y el presupuesto asignado. Para fortalecer este tipo de organizaciones se necesitan prácticas eficientes de SE adaptadas a su tamaño y tipo de negocio. Hoy en día la comunidad vinculada a esta disciplina ha expresado en la última década especial interés en la mejora de procesos de software con el fin de aumentar la calidad y productividad del mismo. En México, el mercado de software es reducido en comparación con grandes países industrializados como Brasil, sin embargo, el mercado mexicano es el segundo en importancia en América Latina [9], lo conforman en su mayoría PyMES dedicadas al desarrollo de software a la medida. En menor cantidad, están algunas empresas líderes de la industria de la tercerización de software latinoamericana como es el caso de *Softtek y Neoris*, firmas con amplio reconocimiento internacional. Además, se cuenta con la presencia de significativas transnacionales: por un lado las orientadas a los de servicios de tecnologías de la información como IBM, *Accenture*, EDS, y por otro lado las orientadas a la venta de productos de software y líderes mundiales de la industria como Microsoft, SAP y Oracle [10].

De acuerdo con [10], la problemática de la industria en México se resume en:

a) Indefinición de la política industrial.
b) Política fiscal no promotora del desarrollo.
c) Financiamiento y tasas de interés no competitivas.
d) Mercado deprimido no propicio para tomar ventajas económicas de escala.
e) Falta de apoyo e incentivos para la pequeña y mediana industria.

f) Servicios públicos no competitivos y de calidad, precio e infraestructura.
g) Prácticas comerciales desleales de empresas de los países signatarios del Tratado de Libre Comercio de América del Norte (TLCAN).
h) Regulaciones ambientales y ecológicas más estrictas y costosas que las de nuestros socios comerciales.

De este modo, de acuerdo con lo señalado en [11], la problemática de las PyMES industriales es muy compleja y tiene que ver tanto con aspectos de políticas públicas que no han sido diseñadas de acuerdo con sus capacidades y limitaciones específicas, como con sus restricciones internas que van desde aspectos financieros, organizacionales hasta aquellos con problemas de acceso a tecnologías de punta y las necesidades que tienen con respecto al diseño de estrategias para su desarrollo como lo establece el trabajo presentado en [12], el cual cita que las empresas y en especial las PyMES tienen grandes carencias financieras para ampliar capacidades, adquirir competencias, desarrollar estructuras productivas y gerenciales e implementar estrategias. Así, el financiamiento adquiere un carácter instrumental para el logro de los principales objetivos que se persiguen: crear condiciones para reducir la brecha de productividad e incrementar la competitividad. Por lo tanto, para avanzar en el desarrollo de un sistema que contemple en forma integrada el financiamiento se requiere una visión de conjunto de las exigencias financieras y de las restricciones que enfrentan para la obtención de créditos. En este sentido, entre otras medidas, se recomienda desarrollar programas vinculados con lo siguiente:

- Garantías de respaldo para créditos a PyMES.
- Capacitación y fortalecimiento de capacidades empresariales para el acceso a diversas formas de financiamientos.
- Productos financieros diferenciados para el segmento de las PyMES.
- Mecanismos de financiamiento no bancario.
- Flexibilización de mecanismos de regulación bancaria y establecimiento de incentivos para el otorgamiento de créditos a las PyMES.
- Simplificación de trámites que ayuden a superar las limitaciones legales que obstaculizan el acceso al crédito para las PyMES.

Existe otro problema de organización en las empresas el cual se asocia a la relación con la administración pública para la canalización de recursos económicos, por medio de programas. Las instituciones no se han preocupado por las PyMES, así como tampoco por las micro empresas que no se encuentran registradas en la Secretaría de Hacienda y Crédito Público, por la excesiva burocracia que existe en la administración pública. Finalmente, dentro de los problemas de las empresas se encuentra la informalidad de los integrantes de la organización para el desarrollo de sus actividades, debido a que hay muchas de ellas que no tienen un organigrama y actúan dentro de la misma como informales, pero complementan al sistema formal que deberían de tener, además de que no tienen bien establecidos sus objetivo, misión, visión y algunos otros elementos administrativos que se requieren para el desarrollo de las fábricas.

2.2. Ingeniería de Software

La Ingeniería de Software es una disciplina que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema hasta su mantenimiento, además proporciona a los desarrolladores un conjunto de procedimientos y técnicas para llevar a cabo la especificación, análisis, diseño, implementación, validación e incluso el mantenimiento del software [13]. Existen diversas definiciones sobre lo que es SE, en la literatura actual destaca la propuesta de [14], en la que el autor la explica como el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica un producto que sea confiable y que trabaje con eficiencia en máquinas reales. Con la aplicación de la SE pueden reducir riesgos de fallos en un sistema en desarrollo e incrementar la posibilidad de la entrega del producto en el tiempo estimado, con la calidad y dentro de los costos presupuestados. Generalmente las etapas utilizadas en el desarrollo de software son: Análisis, Diseño, Implementación, Validación y Mantenimiento.

Un punto prioritario en el desarrollo de software consiste en definir lo que se quiere producir, gracias a que se inicia en el preciso momento en el que se tienen establecidas las características del producto que se va a construir, situación que es más crítica cuando se trata de sistemas complejos, por lo cual es necesario conocer las distintas técnicas de recopilación y definición de requisitos y aplicar la que ofrezca mejores resultados [15]. En este sentido, la obtención de un

software con calidad implica, entre otras cosas, la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba, que permita uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad. La política establecida debe estar sustentada sobre tres principios básicos: i) tecnológico, el cual define las técnicas a utilizar en el proceso de desarrollo de software, ii) administrativo, contempla las funciones de planificación y control del desarrollo del software, y iii) ergonómico, que se encarga de definir la interfaz de usuario. La adopción de una buena política contribuye en gran medida a lograr la calidad del software, pero no la asegura para ello es indispensable su control y evaluación. La tesis se ubica en el principio tecnológico, dentro de la SE [13].

2.3. Ingeniería de Requisitos

Una etapa muy importante dentro de la SE es la Ingeniería de Requisitos (RE, *Requirements Engeneering*), porque define el software que se desea producir y sus especificaciones. Este proceso se realiza mediante las actividades de la RE, las cuales son la obtención, el análisis, la especificación, la validación y la administración de los requisitos del software. Como se mencionó en el punto 1.2 de ésta tesis, un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste [18]. En latinoamérica e hispanoamérica, existe una discusión respecto a si es correcto traducir la palabra "*requirements*" del concepto "*Requirements Engineering*" como "requisitos" o "requerimientos", semánticamente correcto es "requisitos", debido a que los conceptos de "requerimiento" y "requisito" no tienen relación directa entre ellos. Según el diccionario de la Lengua Española de la Real Academia Española (RAE) [16], la palabra "requerimiento", acción y efecto de requerir, ocupar, es un sinónimo de necesidad, por lo que es un concepto dirigido hacia la carencia o falta de algo. La palabra "requisito", es una circunstancia o condición importante para algo, de tal forma que, cuando se desarrolla un software, lo correcto es analizar las condiciones que debe de cumplir para poder obtener una guía para su creación (documento de especificación de requisitos del software). La palabra "requerimiento" utilizada en el contexto de las necesidades del usuario es válida, pero es intrínsecamente

ajena a la especificación de requisitos de software, por lo que es incorrecto hablar de la RE o del análisis de requisitos.

La RE es el área de investigación que propone métodos, técnicas y herramientas que facilitan el trabajo de definición de lo que se quiere de un producto de software [15]. Su objetivo es aumentar el conocimiento del dominio del problema para así representar las necesidades de un cliente en función de una aplicación de software de una manera adecuada y entendible tanto para los usuarios finales como para el equipo de desarrollo. En el autor define a la RE como una disciplina en la que se establece un conjunto de actividades que son utilizadas para obtener, analizar, especificar, validar y administrar las carencias del usuario, conocidas como requisitos del sistema [13]. En este sentido, son las exigencias de los clientes, los servicios que los usuarios desean que proporcione el sistema y las restricciones en las que debe operar. El resultado del proceso de requisitos es la base para el diseño, la implementación y la evaluación del software. De esta forma, si no se descubren y especifican todos podría conducirnos a la entrega de un producto de software incompleto, con alto índice de riesgos, y poco funcional, por lo tanto, es de suma importancia asegurar una buena comunicación con los clientes y/o usuarios porque de ello depende el éxito o fracaso del producto, debido a que a partir de que se obtienen y se definen los requisitos expresados, se comienza con las demás etapas en el proceso, basadas en una lista de ellos, los cuales deben ser concisos, completos, consistentes, no ambiguos y verificables. El resultado del proceso de especificación es la base para el diseño, la implementación y la evaluación del software.

En la literatura actual se mencionan una serie de características que deben de tener los requisitos dentro de un proceso de desarrollo de software, se considera que debe ser [17]:

- Necesario: si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.
- Especificado por escrito: como un contrato o acuerdo entre dos partes.
- Posible de probar o verificar: si no se puede comprobar, entonces ¿cómo se sabe si se cumplió con él o no?

- Conciso: si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- Completo: si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
- Consistente: si no es contradictorio con otro.
- No ambiguo: cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

Existen una serie de dificultades para definir los requisitos que se pueden presentar durante las etapas iniciales del proceso de desarrollo las cuales son importantes de identificar y prevenir, a continuación se presenta un listado con los problemas más comunes en este proceso [17]:

- Los requisitos no son obvios y vienen de muchas fuentes.
- Son difíciles de expresar en palabras (el lenguaje es ambiguo).
- Existen muchos tipos y diferentes niveles de detalle.
- La cantidad de estos en un proyecto puede ser difícil de manejar.
- Nunca son iguales. Algunos son más difíciles, más riesgosos, más importantes o más estables que otros.
- Están relacionados unos con otros, y a su vez se relacionan con otras partes del proceso.
- Tiene propiedades únicas y abarcan áreas funcionales específicas.
- Puede cambiar a lo largo del ciclo de desarrollo.
- Son difíciles de cuantificar, debido a que cada conjunto es particular para cada proyecto.
- El usuario no puede explicar lo que hace.
- Tiende a recordar lo excepcional y olvidar lo rutinario.
- Hablan de lo que no funciona.
- Los usuarios tienen distinto vocabulario que los desarrolladores.
- Usan el mismo término con distinto significado.

2.3.1. Requisitos Funcionales

Los Requisitos Funcionales (RF) describen una interacción entre el sistema y su ambiente, explican cómo deben comportarse ante determinado estímulo. Son declaraciones de los servicios que debe proporcionar, de la manera en que éste

debe reaccionar y de cómo se debe comportar en situaciones particulares. En algunos casos, también pueden declarar explícitamente lo que el sistema no debe hacer. Los RF de un sistema describen lo que el sistema debe hacer. Es importante que se describa el ¿Qué? y no el ¿Cómo? se deben hacer esas transformaciones. Estos requisitos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema [17].

2.3.2. Requisitos No Funcionales

Los Requisitos No Funcionales (RNF), por su parte, se conocen como atributos de calidad del software y están directamente relacionados al rendimiento, fiabilidad, exactitud, seguridad y usabilidad [18]. Comúnmente se acepta que su manejo y equilibrio es una parte importante y difícil del proceso de la RE y que desempeñan un papel fundamental en el desarrollo de un producto. Una de las características de estos requisitos es la especificación de ciertos niveles de calidad y, por consiguiente, en muchos casos, es posible cuantificarlos. Esto es importante no solo para su comprensión, sino también para su planificación. Tratarlos ineficazmente o no tratarlos puede dar lugar a un producto más costoso y posiblemente que se demore su salida al mercado o en el peor de los casos a errores en el desarrollo. Varios estudios, como los presentados en [18] y [19] han demostrado que elicitar es costoso y difícil de manejar, y a menudo son mal comprendidos en comparación con aspectos menos críticos del desarrollo del software [20]. Generalmente, se reconoce que las decisiones acerca de cuáles criterios de calidad deben precisarse en un producto, tienen grandes efectos en su desarrollo y en la elección de la arquitectura. Esto significa que el área de los RNF es importante para comprender con más detalle qué dependencia existe entre su calidad y otros componentes del sistema que se va a desarrollar.

2.3.3. Actividades de la Ingeniería de Requisitos

En este apartado es importante mencionar que se han puntualizado distintos enfoques para definir las actividades de la RE, como las propuestas en [5], y las actividades difieren ampliamente entre sí por varias razones, por ejemplo, dependiendo del dominio de la aplicación, las personas involucradas y la organización de los requisitos. Sin embargo, hay una serie de tareas genéricas

que comúnmente aparecen en todos ellos, tales como elicitación, análisis, especificación, validación y administración. Éstos se detallan como sigue:

- **Elicitación**, cuyo objetivo es descubrir qué problemas deben ser resueltos e identificar a las partes interesadas, y los objetivos que un sistema de software debe alcanzar [19]. Se lleva a cabo a través de la aplicación de diversas técnicas, tales como cuestionarios, lluvia de ideas, prototipos y técnicas de modelado, por ejemplo, los métodos orientados a objetivos [20], [21].
- **Análisis**, que incluye la creación de modelos conceptuales o prototipos con los que lograr la integridad de los requisitos y trata de comprender la estructura de una organización, sus reglas de negocio, metas y tareas y los datos que se necesitan.
- **Especificación**, que es una descripción integral del comportamiento del sistema que se va a desarrollar. Las técnicas más utilizadas son las plantillas, los escenarios, el modelado de casos de uso y el lenguaje natural [22].
- **Validación**, el objetivo de esta fase es establecer si los requisitos obtenidos proporcionan una representación exacta de las necesidades reales de las partes interesadas. Algunas de las técnicas empleadas son las revisiones y la trazabilidad [23].
- **Administración**, que consiste en reconocer los cambios a través de la utilización de requisitos continuos elicitación, e incluye técnicas para la administración de la configuración y el control de versiones [24].

2.4. Técnicas para la Obtención de Requisitos

A continuación, se presentan una serie de técnicas utilizadas para realizar las actividades de captura, definición y validación de requisitos se consideran la facilidad de aprendizaje y de uso, la estabilidad, el costo, la calidad y completitud de los resultados y el tiempo requerido para aplicar las técnicas.

2.4.1. Captura

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema [25]. El proceso de captura puede

resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la RE ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa. En este apartado se presentan un grupo de estas que de forma clásica han sido utilizadas para esta actividad en el proceso de desarrollo de todo tipo de software [25].

- **Entrevistas**: resultan una técnica muy aceptada dentro de la RE y su uso está ampliamente extendido. Permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Existen muchos tipos de entrevistas y son muchos los autores que han trabajado en definir su estructura y dar guías para su correcta realización. Básicamente, la estructura abarca tres pasos: identificación de los entrevistados, preparación, realización y documentación de los resultados. La entrevista, sin embargo, no es una técnica sencilla de aplicar requiere que el entrevistador sea experimentado y tenga capacidad para elegir bien a las personas y obtener de ellos toda la información posible en un período de tiempo siempre limitado. Bajo este aspecto la preparación de esta representa un papel esencial.
- **JAD (*Joint Application Development*/Desarrollo conjunto de aplicaciones)**: esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Está basada en cuatro principios fundamentales dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación WYSIWYG (*What You See Is What You Get*, lo que ve es lo que obtiene). Tras una fase de preparación del JAD al caso concreto, el equipo de trabajo se reúne en varias sesiones. En cada una de ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación. Durante la sesión se discute en grupo sobre estos temas llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema. Esta técnica presenta una serie de

ventajas frente a las entrevistas tradicionales, la cual ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado. Pero requiere un grupo de participantes bien integrado y organizado.

- ***Brainstorming*** **(Tormenta de ideas)**: es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la acumulación de información sin evaluar la misma. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo 10 personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador. Como técnica de captura de requisitos es sencilla de usar y de aplicar, contrariamente al JAD puesto que no requiere tanto trabajo en grupo como éste. Además, suele ofrecer una visión general de las necesidades del sistema, pero no sirve para obtener detalles concretos, por lo que suele aplicarse en los primeros encuentros.
- **Puntos de Vista**: en cualquier sistema existen varios tipos de usuarios que tienen diferentes intereses en los requisitos del sistema. Los enfoques orientados a puntos de vista se utilizan para organizar y estructurar tanto el proceso de obtención como los requisitos [11]. Se considera la existencia de diferentes perspectivas y proporciona un esquema para descubrir problemas propuestos por diferentes usuarios. Los puntos de vista pueden ser considerados como:
    - Una fuente o consumidor de datos, son responsables de producir o consumir datos.
    - Un marco de trabajo de la representación, se considera un tipo particular del modelo del sistema.
    - Un receptor de servicios, son externos al sistema y reciben servicios de él.
- **Casos de Uso**: aunque inicialmente se desarrollaron como técnica para la definición de requisitos, algunos autores los proponen para la captura de estos. Los casos de uso permiten mostrar el contorno (actores) y el alcance (RF expresados como casos de uso) de un sistema. Describen la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Las personas son

elementos externos que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores. La ventaja esencial de esta técnica es que resultan muy fáciles de entender para el usuario o cliente, sin embargo, carecen de la precisión necesaria si no se acompañan con una información textual o detallada con otro método como pueden ser diagramas de actividades.

- **Cuestionarios y *Checklists***: esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (*Checklist*). Este será complementado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista.
- **Observación**: es una técnica en donde el desarrollador se sumerge en el ambiente de trabajo de los usuarios, para observar el trabajo diario que éstos realizan anotando los aspectos importantes, observando factores sociales y organizacionales que afecten el sistema. Existen dos enfoques de este método y son la etnografía y grabar lo observado [11]. Uno de los problemas que surge durante la elicitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada en forma complementaria a otras para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos (contraste).

Existen más técnicas para la captura de requisitos (análisis de otros sistemas, estudio de la documentación, etc.) incluso también es común encontrar alternativas que combinen varias de estas [26]. Sin embargo, las presentadas ofrecen un conjunto representativo de las más utilizadas y resultan suficientes para el objetivo de este trabajo.

2.4.2. Definición

En la actualidad existen un gran número de técnicas propuestas para la definición de los requisitos, en éste apartado se describen las más relevantes para la tesis.

- **Lenguaje natural**: resulta muy ambigua para la definición de los requisitos. Consiste en definir las necesidades en lenguaje natural sin usar reglas para ello. Pero, a pesar de que son muchos los trabajos que critican su uso, es cierto que a nivel práctico se sigue utilizando.
- **Glosario y ontologías**: la diversidad de personas que forman parte de un proyecto software hace que sea importante establecer un marco de terminología común. Esta necesidad se vuelve más patente en los sistemas de información web puesto que el equipo de desarrollo en ellas suele ser más interdisciplinario. Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recogen y definen los conceptos más relevantes y críticos para el sistema. En esta línea se encuentra también el uso de ontologías, en las que no sólo aparecen los términos, sino también las relaciones entre ellos.
- **Plantillas o patrones**: recomendada por varios autores, tiene por objetivo el describir los requisitos mediante el lenguaje natural, pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado explícito, el trabajo de rellenar las plantillas y mantenerlas puede ser demasiado tedioso.
- **Escenarios**: consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textuales o ir encaminada hacia representaciones gráficas en forma de diagramas de flujo [27]. El análisis de los hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales de sistema.

- **Lenguajes Formales**: es utilizada para describir los requisitos de un sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción formal, han sido aplicadas en el mundo de la RE desde hace años [28]. Sin embargo, resultan muy complejas en su utilización y para ser entendidas por los usuarios. El mayor inconveniente es que no favorecen la comunicación entre cliente y analista. Por el contrario, es la representación menos ambigua y la que más se presta a técnicas de verificación automatizadas.

2.4.3. Especificación

La especificación de requisitos de software (ERS) es la actividad en la cual se genera el documento, con el mismo nombre, que contiene una descripción completa de las necesidades y funcionalidades del sistema que será desarrollado; describe el alcance del producto y la forma en como hará sus funciones, definiendo los RF y RNF.

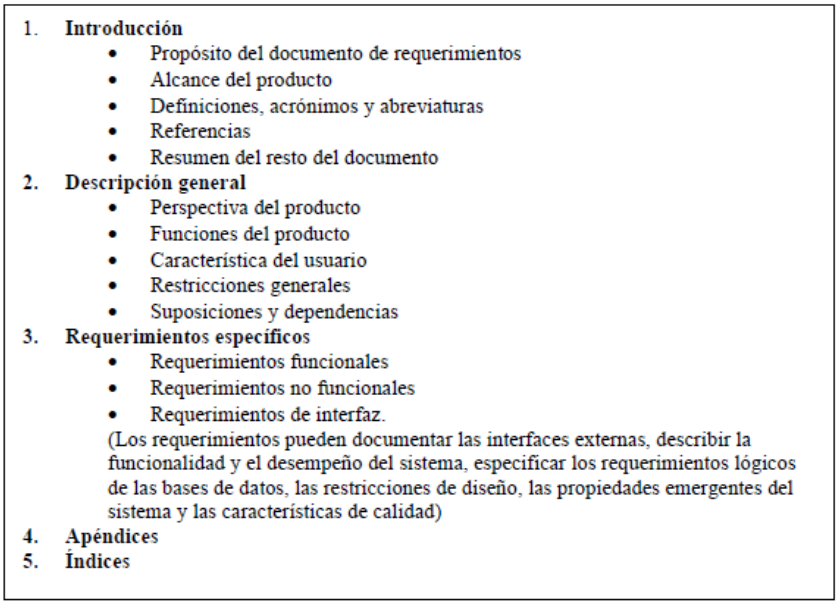
Los requisitos son el punto de acuerdo entre el cliente y el proyecto de desarrollo de software, este entendimiento es necesario para poder construir un producto que satisfaga las necesidades de nuestro usuario. Por lo cual es lógico que para recabarlos haya que obtener la información de primera mano. Esto es, mediante entrevistas o recabando documentación que describa la manera que se desea que funcione el sistema de software. Las necesidades del cliente evolucionan con el tiempo y cada cambio involucra un costo. Por eso, es necesario tener archivada una copia de la documentación original, así como cada revisión o cambio que se haga a esta documentación. Como cada requisito es tratado de diferente forma, es necesario clasificarlos para saber cuáles de ellos serán satisfechos por el software y cuales por algún otro producto del sistema. En la ERS se definen el hardware y software, así como diagramas, modelos de sistemas y cualquier otra información que sirva de soporte y guía para fases posteriores. Es importante destacar que la especificación de requisitos es el resultado final de las actividades de análisis y evaluación; este documento resultante será utilizado como fuente básica de comunicación entre los clientes, usuarios finales, analistas de sistema, personal de pruebas, y todo aquel involucrado en la implementación del sistema [27]. La ERS es utilizada para comparar si lo que se está proponiendo, coincide con las necesidades de la

empresa. Los analistas y programadores la usan para determinar el producto que debe desarrollarse. El documento de ERS debe de tener las siguientes características:

- Correcta: todo requisito de la ERS contribuye a satisfacer una necesidad real.
- Verificable: si para cada requisito expresado en la ERS existe un procedimiento de prueba finito y no costoso para demostrar que el futuro sistema lo satisface.
- No redundante: cada requisito se expresa en un solo lugar de la ERS. La redundancia, de todas formas, no es del todo mala si aumenta la legibilidad.

2.4.4. Documentación

El propósito de la documentación es comunicar los requisitos entre los interesados y los desarrolladores. El documento es la base para evaluar los productos y procesos posteriores (diseño, pruebas, verificación y actividades de validación) y para el control de cambios. Un buen documento de requisitos es inequívoco, completo, correcto, comprensible, coherente, conciso y factible. Dependiendo de la relación cliente-proveedor la especificación de los mismos puede ser parte del contrato. Existen diferentes formas de definir y especificar las necesidades en un documento formal, un gran número de organizaciones se han preocupado por definir estándares para los documentos de especificación. En la figura 1 se muestra el propuesto por la IEEE y la ANSI conocido como el estándar IEEE/ANSI 830-1993, también brevemente llamado como el estándar 830-1993, que sugiere la siguiente estructura para los documentos:



1. **Introducción**
   - Propósito del documento de requerimientos
   - Alcance del producto
   - Definiciones, acrónimos y abreviaturas
   - Referencias
   - Resumen del resto del documento
2. **Descripción general**
   - Perspectiva del producto
   - Funciones del producto
   - Característica del usuario
   - Restricciones generales
   - Suposiciones y dependencias
3. **Requerimientos específicos**
   - Requerimientos funcionales
   - Requerimientos no funcionales
   - Requerimientos de interfaz.

   (Los requerimientos pueden documentar las interfaces externas, describir la funcionalidad y el desempeño del sistema, especificar los requerimientos lógicos de las bases de datos, las restricciones de diseño, las propiedades emergentes del sistema y las características de calidad)
4. **Apéndices**
5. **Índices**

Figura 1. Plantilla de especificación de requisitos IEEE/ANSI 830-1993 [29].

Existen otras plantillas para especificar requisitos como es el caso de Volere, en la figura 2 se observa el formato el cual provee de secciones por cada tipo de los requisitos apropiados para los actuales sistemas de software. Se puede adaptar a su proceso de recolección y a su herramienta [30]. La plantilla puede ser usada con Requisite, DOORS, Caliber RM, IRqA y otras herramientas populares.

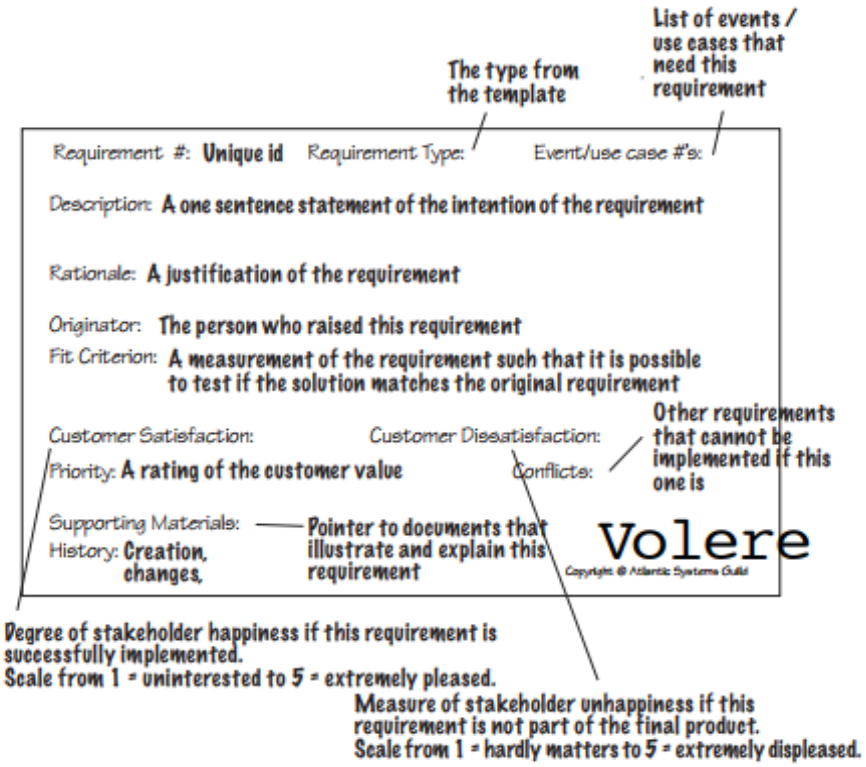


Figura 2. Plantilla de especificación de requisitos de Volere [30].

2.4.5. Validación

Los requisitos una vez definidos necesitan ser aprobados. La validación de estos tiene como misión demostrar que la definición explica realmente el sistema que el usuario necesita o el cliente desea [23]. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de obtención de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos con el usuario para detectar errores o inconsistencias. Aun así, existen algunas técnicas que pueden aplicarse para ello:

- **Auditorías**: consiste en un chequeo de los resultados contra una checklist que, predefinida o definida a comienzos del proceso, es decir que sólo una muestra es revisada.
- **Matrices de trazabilidad**: se utiliza para marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada necesidad, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.

- **Prototipos**: algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema que debe entenderse de que lo que se está viendo es un prototipo y no el sistema final.

2.5. Metodologías para el Desarrollo de Software

El desarrollo de software no es una tarea fácil, prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Al respecto, algunos autores definen una metodología como una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información [31]. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo.

En algunas fábricas de software, el proceso de desarrollo se lleva a cabo por medio de actividades manuales y/o metodologías robustas que pueden llegar a ser en muchos casos pesados e ineficientes. Esta situación trae consigo algunos problemas relacionados con la dificultad para producir software de manera oportuna, ágil, a bajo costo y con un alto nivel de calidad. Una manera de mejorar esta situación está en añadir al proceso de desarrollo de software el formalismo y la abstracción necesaria que permita automatizar y optimizar las tareas más críticas definidas, a partir de las metodologías utilizadas en las fábricas de software y desde una perspectiva ágil. Esto añadiría valor agregado a los negocios y mejoraría el proceso de software considerablemente [32]. En este sentido, diversos autores coinciden en señalar algunos requisitos que deben tener las metodologías de desarrollo son:

- Visión del producto.
- Vinculación con el cliente.
- Establecer un modelo de ciclo de vida.
- Administración de los requisitos.
- Plan de desarrollo.

- Integración del proyecto.
- Medidas de progreso del proyecto.
- Métricas para evaluar la calidad.
- Maneras de medir el riesgo.
- Como administrar los cambios.
- Establecer una línea de meta.

2.5.1. Metodologías tradicionales

Las metodologías tradicionales (formales) son denominadas, a veces, de forma peyorativa, como metodologías pesadas. Centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial. Otra de las características importantes dentro de este enfoque, son los altos costos al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. Además se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, etc.) [12].

2.5.2. Metodologías Ágiles

Los métodos ágiles constituyen una reacción a los métodos tradicionales planificados que enfatizan un enfoque excesivamente ingenieril y racional y que asumen que todos los problemas se pueden especificar completamente. El enfoque “tradicional” propugna una planificación intensa, procesos estrictos y rigurosa reutilización, de tal modo que la actividad de desarrollo se asemeja a una ingeniería estricta. Por el contrario el enfoque “ágil” promueve una respuesta rápida a entornos cambiantes, cambios en los requisitos y modificaciones en los plazos de entrega [33]. Los principales beneficios de los modelos ágiles son el rápido desarrollo y la reducción de costos [34]. Los métodos ágiles obligan a los miembros del equipo a colaborar, esto no significa solo un aviso, donde la comunicación es simplemente enviar y recibir mensajes, la colaboración describe como trabajar activamente en la creación de productos y la toma de decisiones [35]. La comunicación sin embargo, es esencial cuando se trata de colaborar eficazmente. Las metodologías ágiles promueven la idea de equipos auto-organizados, los cuales consisten en miembros de todas las disciplinas como desarrolladores, arquitectos, diseñadores, gerentes de equipo, tal vez un

representante del cliente de preferencia, todo el equipo debería estar en la misma habitación. Lo que esto crea es un entorno donde cara a cara fluye libremente y esto permite identificar rápidamente los problemas y las decisiones sobre cómo hacer frente a estas situaciones. Para mejorar aún más la capacidad de respuesta de los miembros del equipo, se anima a desarrollarse lo más ampliamente posible y ampliar sus competencias. Si uno miembro del equipo no puede terminar sus tareas otro miembro debería ser capaz de realizarlas. La auto-organización también significa que el control desde fuera del equipo debe ser limitado. El equipo debe ser confiado para manejar el proyecto ellos mismos, otro punto clave es la responsabilidad compartida. Todo el equipo debe sentirse responsable del software que se está entregando, esto anima a los miembros de diferentes disciplinas a comentar del proyecto con el fin de identificar los problemas antes de que se conviertan en problemas mayores). El equipo debe estar actualizado sobre el progreso del proyecto que se está realizando y adaptarse a los cambios que se realicen.

A continuación, se describen las metodologías ágiles más importantes y se presentan aspectos como su forma de aplicarse y su eficiencia enfocadas en la parte de requisitos para conocer como es que realizan su administración durante el proceso de desarrollo [36] y [37].

2.5.2.1. SCRUM

Es una metodología para la administración y control de proyectos, centrada en la construcción de software que satisface las necesidades del cliente, cumple con los objetivos del negocio y el equipo de desarrollo que construye el producto. Al no establecer prácticas de SE, se combina fácilmente con otras metodologías de desarrollo. El autor define a este método ágil como una colección de procesos para la administración de proyectos, que permite centrarse en la entrega de valor para el cliente y la potenciación del equipo para lograr su máxima eficiencia, dentro de un esquema de mejora continua [38]. En la figura 3 se puede ver el proceso, un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4, límite máximo de *feedback* y reflexión) [39]. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final

que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

Figura 3. Proceso ágil de SCRUM [40].

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. Las actividades que se llevan a cabo en SCRUM son las siguientes:

- Planificación de la iteración. El primer día de la iteración se realiza la reunión de planificación de la iteración.
- Ejecución de la iteración. Cada día el equipo realiza una reunión de sincronización (15 minutos máximo). Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido.
- Inspección y adaptación. El último día de la iteración se realiza la reunión de revisión de la iteración.

En la metodología SCRUM las principales técnicas para la especificación de los requisitos son los *backlogs*, los *sprints* y las reuniones diarias. En lo que respecta en esta metodología, los *backlogs* del producto son la manera de recolectar las necesidades mediante una lista de *backlog*, tal vez esta técnica pueda compararse con un documento de requisitos que contiene información necesaria

para el desarrollo, esta es plasmada desde el inicio del proyecto, dicho catalogo va a sufrir un proceso de priorización de cada una de las tareas plasmadas allí y que posteriormente serán distribuidas en diferentes *sprints* (entendiéndose un como un ciclo en el cual se desarrollaran las tareas seleccionadas). Las reuniones diarias son juntas de 15 minutos en las cuales cada desarrollador responde a las siguientes preguntas: ¿Qué hizo desde la última reunión?, ¿Qué dificultades concretas tiene en el desarrollo de la tarea?, ¿Qué va a hacer hasta la próxima reunión diaria? Una de las políticas que tiene esta metodología es que durante la ejecución del *sprint* no se podrán realizar modificaciones a los elementos allí involucrados, sino que esto podrá hacerse luego de finalizado el mismo para ser tenidas en cuenta a partir del próximo *sprint* [36].

2.5.2.2. eXtreme Programming (XP)

XP es una metodología de desarrollo ágil diseñada para proyectos adaptativos donde los requisitos son vagos y no claros. Uno de las características de XP es la rapidez de respuesta a los cambios de los requisitos del cliente [38]. Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo. El ciclo de desarrollo consiste en los siguientes pasos [41]:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al desarrollador a realizar más trabajo que el estimado, porque se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración. El ciclo de vida ideal de XP consiste de seis fases

[42]: Exploración, Planificación de la Entrega (*Release*), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Fase I.- Exploración: en esta fase, los clientes plantean a grandes rasgos las historias de usuario (es una descripción de una característica que proporciona valor comercial al cliente, son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar, deben ser programadas en un tiempo entre una y tres semanas que son de interés para la primera entrega del producto [43], [44]. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Fase II.- Planificación de la Entrega: en esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Fase III.- Iteraciones: Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible porque es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

En la figura 4, se describe el proceso de desarrollo ágil de XP, en dónde se puede observar que la idea principal de ésta metodología es el bajo costo derivado del cambio en los requisitos del software. XP no habla explícitamente acerca de las técnicas de requisitos en detalle, sino sobre el proceso general de desarrollo de

software y sobre lo que se debe hacer durante el proceso. Varias prácticas de XP (o técnicas utilizadas en estas prácticas) pueden compararse con técnicas ligeramente modificadas de RE.

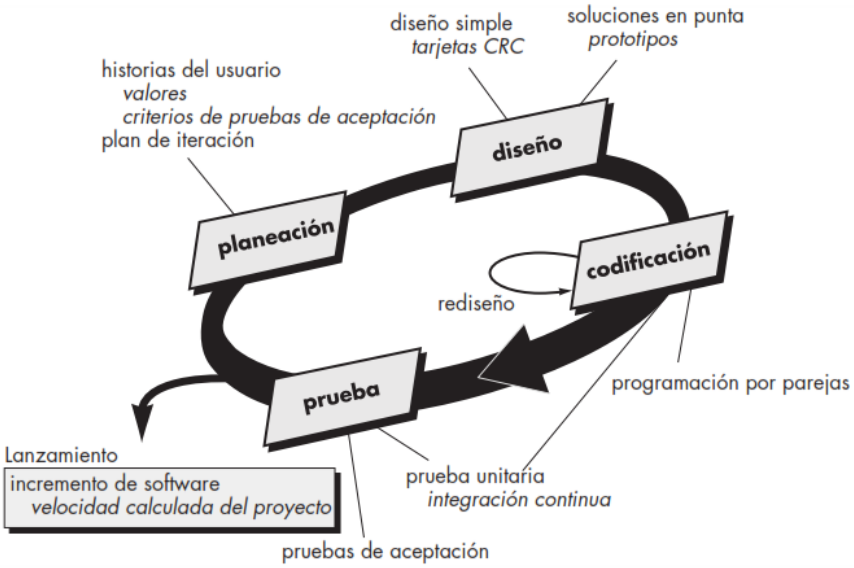


Figura 4. Proceso de desarrollo ágil de XP [45].

Las técnicas que esta metodología utilizada para el proceso de obtención de requisitos son: entrevistas, historias de usuarios, lluvia de ideas. XP utiliza tarjetas de historias para la elicitación.

2.5.2.3. Método de Desarrollo de Sistemas Dinámicos (DSDM)

El método de desarrollo [46] de sistemas dinámicos (DSDM) se origina en 1994 en Gran Bretaña con los trabajos de Jennifer Stapleton directora del DSDM Consortium. DSDM, además, proporciona un marco de trabajo completo de controles para Desarrollo Rápido de Aplicaciones (RAD) y lineamientos para su utilización y se puede complementar con otras metodologías. El objetivo principal de esta metodología es definir primero tiempo y costo, una vez fijados, se establecen las funcionalidades que se pueden implementar en el producto. Esto es expresado en las reglas de *Must*, *Should*, *Could* y *Want* que se conocen con el nombre de MoSCoW, a saber:

- *Must have* (debe tener). Son los requisitos fundamentales del sistema. El subconjunto mínimo debe ser satisfecho por completo.
- *Should have* (debería tener). Son requisitos importantes para los que habrá una solución a corto plazo.

- *Could have* (podría tener). Son requisitos que podrían quedar fuera del sistema si no hay más remedio.
- *Want to have but won't have this time around* (se desea tener, pero no lo tendrá en este momento). Son requisitos valorados, pero pueden esperar.

En la figura 5 se muestran las cinco fases por las que DSDM está compuesto.

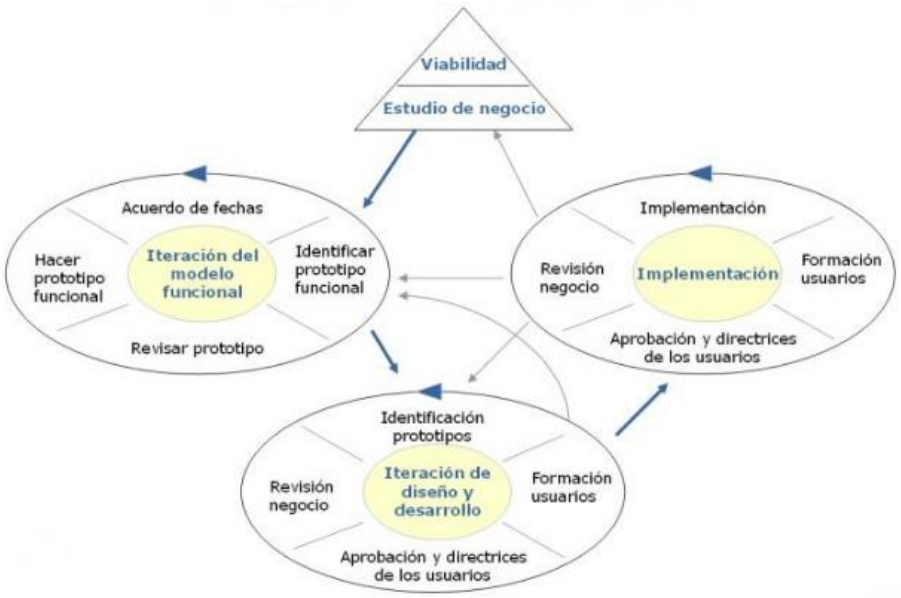


Figura 5. Proceso de desarrollo de DSDM [47].

A continuación, se describen cada una de las fases del proceso de desarrollo de DSDM:

1. Estudio de viabilidad: se evalúa si se utiliza DSDM u otra metodología. Si se opta por DSDM se analizan las posibilidades técnicas y riesgos. Los artefactos son el Reporte de Viabilidad y Plan Sumario para el Desarrollo. Si la tecnología no se conoce se hace un pequeño prototipo para probar. Se espera una duración de pocas semanas.

2. Estudio de negocio: se analizan las características del negocio y la tecnología. Se desarrollan talleres donde los expertos del cliente consideran las facetas del sistema y acuerdan prioridades. Se genera una definición del área de negocios con los procesos y las clases de usuario, utilizando descripciones de alto nivel como el Documento de Especificación de Requisitos (DER) o modelos de objetos de negocio. Otros artefactos son definición de arquitectura del sistema (es un primer borrador y se admiten cambios) y el plan de bosquejo de prototipado

(establece un plan de las siguientes etapas y administración de configuración).

3. Iteración de modelo funcional: en cada iteración se planea el contenido y la estrategia, se realiza la iteración y se analizan los resultados pensando en las siguientes Se produce un modelo funcional conteniendo el código del prototipo y los modelos de análisis, se realizan pruebas constantemente. Hay cuatro artefactos que resultan de esta fase: listado de funciones priorizadas, documentos de revisión del prototipado funcional, listado de requisitos funcionales y análisis de riesgos de desarrollo anterior.

4. Iteraciones de diseño y construcción: el artefacto es un sistema probado que complementa las reglas de MoSCoW. El diseño y la construcción son iterativos y los prototipos funcionales revisados por los usuarios.

5. Implementación: el sistema se transfiere al ambiente de producción. Se capacita a los usuarios. Otros artefactos son el: manual del usuario y reporte de revisión del sistema. Hay cuatro cursos de acción posibles:

a. Si el desarrollo satisface a los usuarios, se ha terminado.

b. Si quedan requisitos por resolver, se comienza nuevamente desde las primeras fases.

c. Si se ha dejado de lado alguna prestación no critica, se puede comenzar desde la fase de iteración del modo funcional.

d. Si algunas cuestiones técnicas no pudieron resolverse, se puede comenzar desde la fase de iteraciones de diseño y construcción.

En lo que respecta a la RE, DSDM proporciona un marco para el desarrollo rápido de aplicaciones [48]. Las dos primeras fases del DSDM son el estudio de factibilidad y el estudio de negocios. Durante estas dos fases se obtienen los requisitos básicos. Otros requisitos se obtienen durante el proceso de desarrollo. DSDM no insiste en ciertas técnicas. Por lo tanto, cualquier técnica RE puede ser utilizada durante el proceso de desarrollo. En esta metodología, las pruebas se integran a lo largo del ciclo de vida.

2.5.2.4. Desarrollo de Software Adaptativo (ASD)

El Desarrollo de Software Adaptativo (ASD) lo propuso Jim Highsmith 1998 como una técnica para construir software y sistemas complejos. Los apoyos filosóficos del ASD se enfocan en la colaboración humana y la organización propia del equipo. Consiste en un cambio de filosofía en las organizaciones pasando de la transición del modelo Comando-Control el cual ha derivado en burocracias mecanizadas, cuyas patologías más frecuentes es la de disimular la ausencia de liderazgo, al modelo Liderazgo-Colaboración, el enfoque requiere que los administradores de recursos, es decir, los gerentes que pueden trabajar con la gente, permitirán humanos-intervenciones, y crear un ambiente amistoso [49]. Basado en los conceptos de los sistemas adaptativos complejos relacionada con la inteligencia artificial, Highsmith lleva los mismos al campo de la SE en particular. Dada la complejidad inherente al software concluye que la aplicación de esta teoría es esencial para el nuevo escenario que plantea la economía global. En la figura 6 se puede ver el ciclo de vida Especular-Colaborar-Aprender que utiliza ASD. El proyecto comienza con una fase de especulación en que en que se lleva a cabo la planificación tentativa del proyecto en función de las entregas que se irán realizando. En esta etapa se fija un rumbo determinado a ser seguido en el desarrollo, sabiendo a partir de ese momento que no será el lugar en que finalizará el proyecto. En cada iteración, se aprenderán nuevas funcionalidades, se entenderán viejas cuestiones, y cambiarán los requisitos.

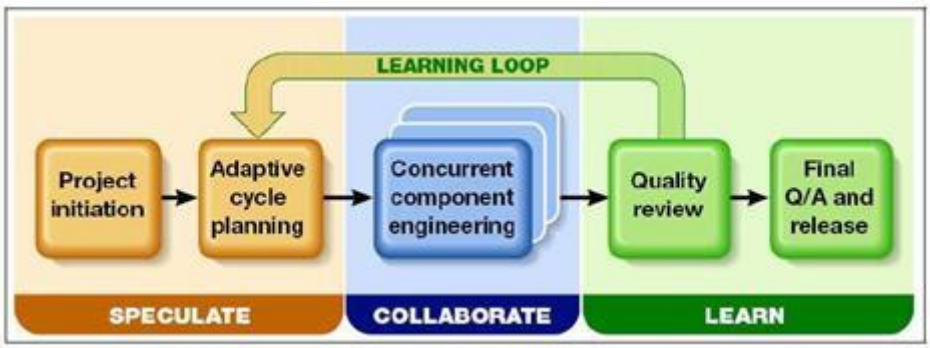


Figura 6. Proceso de desarrollo de ASD [50].

Gracias a centrarse en la especulación, ASD permite administrar estos proyectos de alto cambio y rápido desarrollo que se encuentran en el borde del caos. La

siguiente fase del ciclo de vida, Colaborar, es aquella en la que se construye la funcionalidad definida durante la especulación. La metodología define un Componente como un grupo de funcionalidades o entregables a ser desarrollados durante un ciclo iterativo [49]. Durante cada iteración el equipo colabora intensamente para liberar la funcionalidad planificada. También, existe la posibilidad de explorar nuevas alternativas, realizar pruebas de concepto, pudiendo eventualmente alterar el rumbo del proyecto profundamente. ASD no propone técnicas ni prescribe tareas al momento de llevar a cabo la construcción simplemente mencionando que todas las prácticas que sirvan para reforzar la colaboración serán preferidas, siguiendo de esta forma la línea de las metodologías ágiles respecto a la orientación a componentes. La fase final es la de Aprender, consiste en la revisión de calidad que se realiza al final de cada ciclo. En la misma se analizan cuatro categorías de cosas para aprender:

- Calidad del resultado de la desde la perspectiva del cliente.
- Calidad del resultado de la desde la perspectiva técnica.
- El funcionamiento del equipo de desarrollo y las prácticas que este utiliza.
- El status del proyecto.

ASD proporciona un marco para el desarrollo iterativo de sistemas grandes y complejos. El método fomenta el desarrollo incremental, iterativo con prototipado constante [9]. Las técnicas utilizadas de este método para la obtención de requisitos son las sesiones JAD, durante las sesiones, desarrolladores y clientes discuten las características del producto deseado. Este tipo de discusión puede ser muy productivo si el líder de la sesión impide que los participantes "se queden sin curso". Los participantes incluyen ejecutivos, directores de proyectos, usuarios, expertos en sistemas y personal técnico externo [51].

2.5.2.5. Crystal

La familia de metodologías Crystal se basa en los conceptos de *Rational Unified Process* (RUP), en la figura 7 se muestra que la metodología está compuesta por *Crystal Clear*, *Crystal Yellow*, *Crystal* Orange y *Crystal* Red; el nivel de opacidad del color en el nombre indica un mayor número de personas implicadas en el desarrollo, un mayor tamaño del proyecto y, por lo tanto, la necesidad de mayor control en el proceso [10].

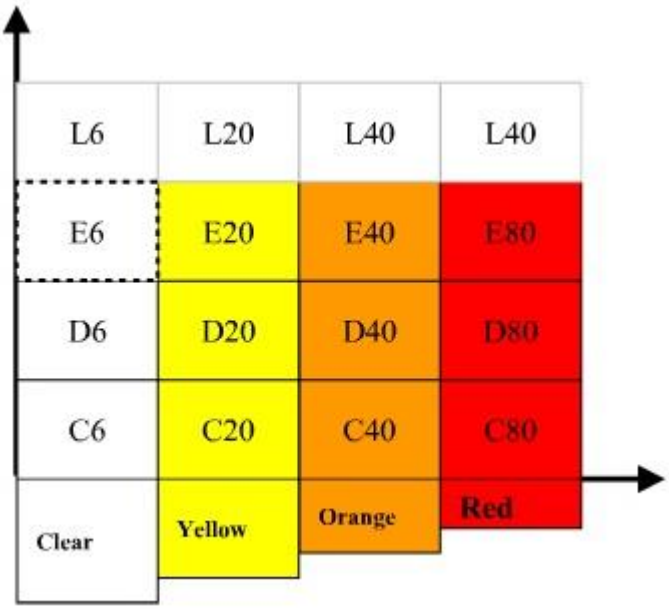


Figura 7. Proceso de desarrollo de Crystal [52].

Los principios de Crystal son:

Normas de política: son prácticas que necesitan ser aplicadas durante el proceso de desarrollo [46].

- Entrega incremental sobre una base regular. *Crystal Clear* (CC) las entregas entre dos a tres meses y *Crystal Orange* (CO) puede ser extendido a cuatro meses como máximo.
- Seguimiento del progreso por hitos basados en entregas de software y decisiones importantes en lugar de documentos escritos.
- Involucramiento directo del usuario.
- Pruebas de funcionalidad de regresión automatizadas.
- Talleres de producto (y metodología) ajustados a principios y mediados de cada incremento.

Artefactos: algunos artefactos difieren entre ambas metodologías, pero los comunes son: secuencias de entregas, modelos de objetos, manuales de usuario, casos de prueba y código. Ahora las diferencias:

- CC incluye descripciones de características del producto y casos de uso anotados y CO necesita un documento de requisitos.

- La planificación en CC se debe hacer teniendo en cuenta las diferentes entregas y las reuniones con los clientes. CO exige algo más exhaustivo, por ejemplo, documentos de diseño de la interfaz de usuario, reportes y especificaciones de los diferentes equipos.

Asuntos locales: son procedimientos que CC especifica que deben realizarse pero no especifica cómo.

Herramientas: CC necesita un compilador, herramienta de versionado y una herramienta de administración, así como pizarras y reuniones. En el caso de CO las herramientas relacionadas con control de versiones, programar, pruebas, comunicación, seguimiento de proyecto, dibujo y medición de rendimiento.

Los valores compartidos por los miembros de la familia Crystal están centrados en las personas y en la comunicación. Sus principios indican que: el equipo puede reducir trabajo intermedio en la medida que produce código con mayor frecuencia y utiliza mejores canales de comunicación entre las personas; los proyectos evolucionan distinto con el tiempo por lo que las convenciones que se adopten tienen que adecuarse y evolucionar; los cambios en el cuello de botella del sistema determinan el uso de trabajo repetido; y el afinamiento se realiza sobre la marcha. Las técnicas usadas en esta metodología para la obtención de requisitos son los prototipos o revisiones.

2.5.3. Diferencias entre Metodologías Ágiles y Tradicionales

A continuación, la Tabla 2 enumera las principales diferencias entre Metodologías Ágiles y Metodologías Tradicionales (llamadas peyorativamente “no ágiles” o “pesadas”).

METODOLOGÍAS TRADICIONALES	METODOLOGÍAS ÁGILES
Su base son las normas provenientes de estándares seguidos por el entorno de desarrollo.	Se basan en heurísticas provenientes de prácticas de producción de código.
Cierta resistencia a los cambios.	Preparados para cambios durante el proyecto.
Impuestas externamente.	Impuestas internamente en el equipo.
Proceso muy controlado, numerosas normas.	Proceso menos controlado, con pocos principios.
Contrato prefijado.	Contrato flexible e incluso inexistente.
Cliente interactúa con el equipo de desarrollo mediante reuniones.	El cliente es parte del desarrollo.
Grupos Grandes.	Grupo pequeños, menores de 10 personas.
Más artefactos.	Pocos artefactos.
La arquitectura del software es esencial.	Menor énfasis en la arquitectura del software.

Tabla 2. Diferencias entre Metodologías ágiles y tradicionales.

La Tabla 2 muestra las diferencias que no se refieren sólo al proceso en sí, sino también al contexto de equipo y organización que es más favorable a cada uno de estas filosofías de desarrollo de software [53]. Lo cual implica que las metodologías ágiles nacen con la finalidad de reducir a su máximo el enfoque de las metodologías tradicionales respecto a la documentación del proceso de desarrollo, de tal forma que permite al equipo de trabajo centrar su atención en entregas del producto software a desarrollar sin dejar de lado la elaboración de una documentación del proceso ejecutado para su posterior administración. Lo anterior se refleja, como lo muestra la Tabla 2, que se realizan pocos artefactos durante el proceso, en este sentido, un artefacto se define como un producto tangible obtenido durante el proceso de desarrollo de software. Además, una de las principales ventajas de las ágiles con respecto a las tradicionales es que se diseñaron para grupos de trabajo pequeños (menos de 10 personas), lo que permite tener un flujo de trabajo constante y la situación actual del proyecto eficazmente sin la necesidad de depender directamente de mayor número de personas para obtener resultados o verificar su situación.

2.6. Herramientas para la Ingeniería de Requisitos

En este apartado se describen las herramientas que se utilizan en un proceso de desarrollo de software en la etapa de requisitos.

2.6.1. Herramientas CASE

En el desarrollo de software se cuenta con una ventaja proporcionada por las herramientas *CASE* (Ingeniería del Software Asistida por Computadora) modo en que se conoce a todo aquel software que es usado para ayudar a las actividades del proceso de desarrollo del software, en donde se ubica la RE, que se ha venido tratando en este trabajo de investigación. Estas herramientas se concentran en capturar requisitos, administrarlos y producir una especificación; de estas existen muchas y muy variadas que pueden ser utilizadas por los desarrolladores de software en sus proyectos, y de la forma más conveniente para ellos. Es importante hacer ver que estas herramientas fungen como un medio facilitador para agilizar y mejorar los procesos involucrados en todo el ciclo de vida presentado por la RE, y que en conjunto ayudan a la construcción final de un producto de software terminado. En el apartado siguiente se mencionan las más utilizadas para la especificación de requisitos, entre las que destacan *RequistePro, VORLDTool, Catalyze Enterprise, Caliber RM, Prosareq, RETO, CONTROLA, OSRMT (Open Source Requirements Management Tool), JEREMIA y RAMBUTAN*. Todas ellas se explican a continuación.

2.6.2. RequisitePro

RequisitePro es la herramienta que ofrece *Rational* Software para tener un mayor control sobre los requisitos planteados y de los nuevos que surjan durante el ciclo de vida del proyecto. En *RequisitePro* las especificaciones se encuentran documentadas bajo un esquema organizado de documentos; estos esquemas cumplen completamente con los estándares requeridos por algunas de las instituciones a nivel mundial más reconocidas en el desarrollo de software, tales como: Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), Organización Internacional para la Estandarización (ISO), Modelo de Capacidad de Madurez (CMM) y por el RUP [11].

En la figura 8 se muestra la interfaz de usuario, esta herramienta se integra con aplicaciones para la administración de cambios, herramientas de modelado de

sistemas y de pruebas. Esta integración asegura que los diseñadores conocen los requisitos del usuario, del sistema y del software en el momento de su desarrollo.

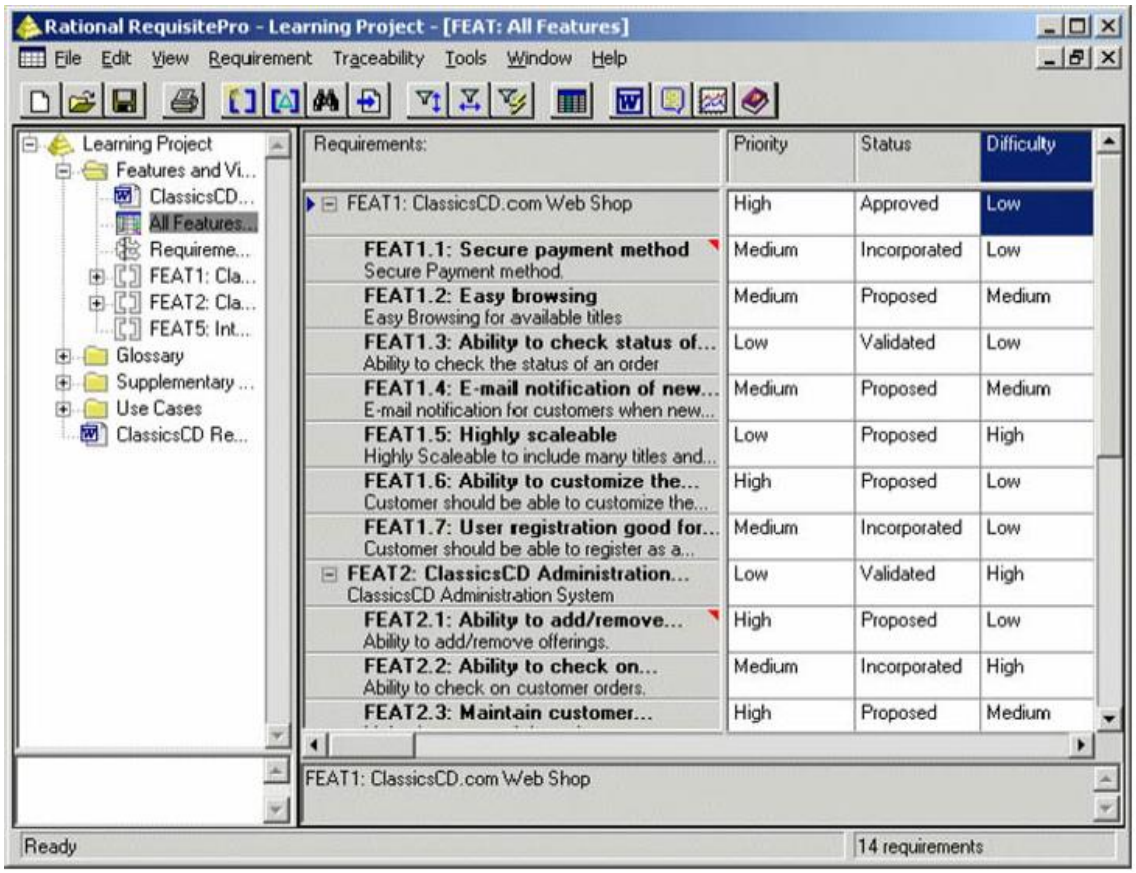


Figura 8. Interfaz de Usuario de RequisitePro [11].

2.6.3. VORDTool

La herramienta VORDTool fue desarrollada por Ian Sommerville y Katonya en el año de 1996, se basa en la especificación de requisitos orientado a puntos de vista. En la figura 9, se puede contemplar el modelo adoptado por VORDTool es orientado a servicios análogos en un sistema cliente-servidor [31]. El sistema capta los servicios y pasa información de control al sistema. Estos mapean clases de usuarios finales de un sistema o las interfaces hacia otros sistemas, los que constituyen la base del modelo son conocidos como puntos de vista directos. Los que respectan a otros sistemas que tienen influencia en el dominio de la aplicación también son considerados y son llamados indirectos. Ambos representan a los RF y RNF respectivamente.

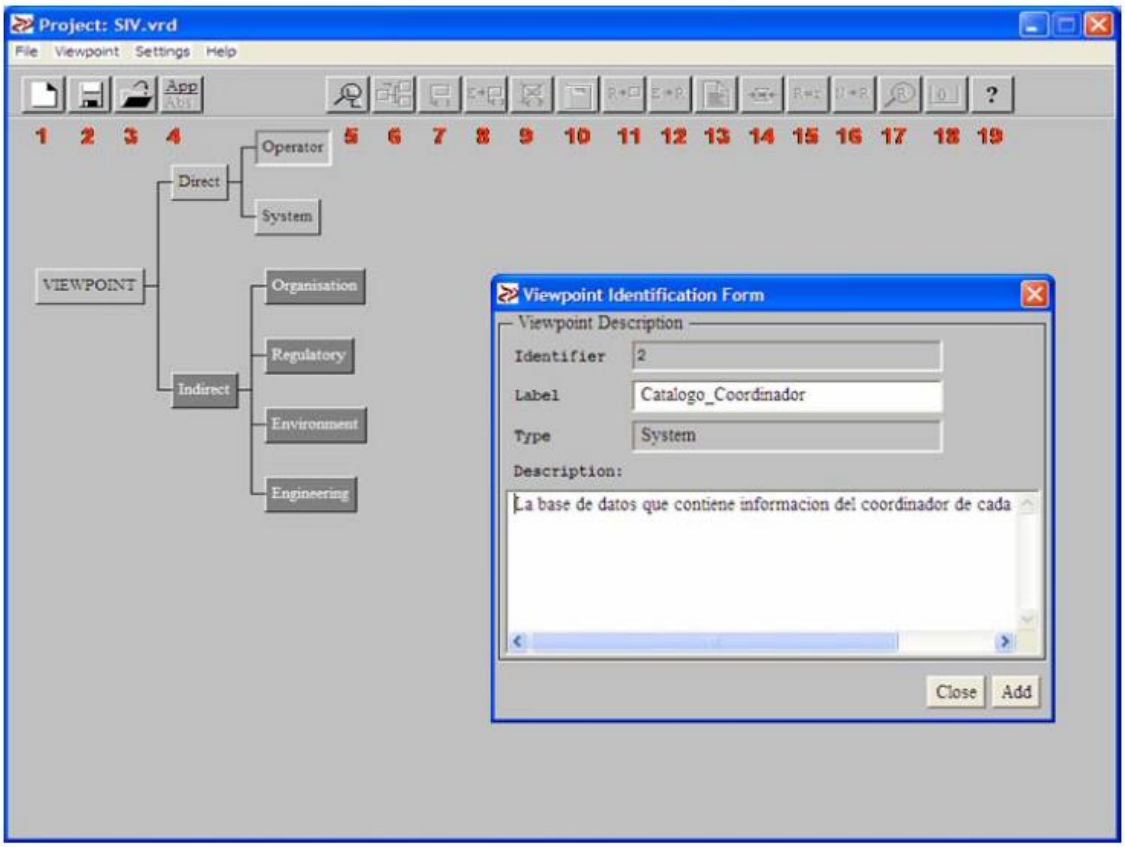


Figura 9. Interfaz de usuario de VORDTool [11].

2.6.4. Catalyze Enterprise

La herramienta ofrecida por Steel Trace, utiliza el paradigma orientado a objetos con un enfoque basado en casos de uso [12]. Estos son un hilo de funcionalidad que el sistema podría realizar, una colección de casos de uso pueden expresar el comportamiento entero del sistema. La figura 10 muestra la herramienta Catalyze la cual proporciona una manera sencilla de definir los términos utilizados en el proyecto, de esta forma reduce la ambigüedad para los desarrolladores y aumenta la comprensibilidad de los revisores y analistas del sistema. También, se puede especificar una lista de actores que van a interactuar con el sistema en desarrollo.

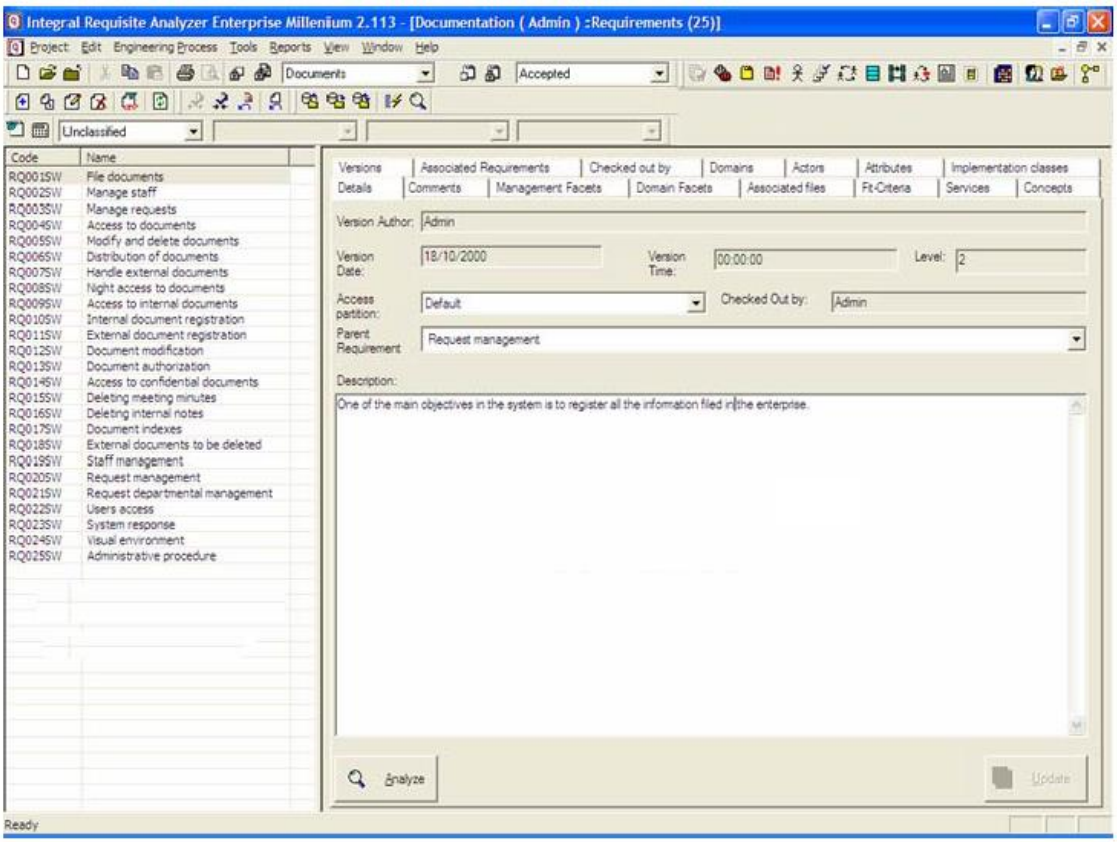


Figura 10. Interfaz de usuario de catalyze Enterprise [11].

Los casos de uso se encuentran estructurados en paquetes, en donde cada paquete contiene un modelo de estos, y cada caso de uso esta descrito mediante un escenario. Un modelo está representado mediante una jerarquía descendente de casos de uso, y están ordenados de acuerdo a su participación en el sistema.

2.6.5. Integral Requisite Analizer (IRqA)

En la figura 11 se puede ver la interfaz de usuario de la herramienta IRqA al cual está desarrollada por TCP Sistemas *e* Ingeniería [53]. Esta herramienta es especialmente diseñada para dar soporte a los analistas y a los ingenieros de software en el proceso de RE. Proporciona la funcionalidad no solo de administrarlos y producir una especificación, sino también la posibilidad de formularlos dentro del contexto del sistema. IRqA proporciona dos modos de abrir los proyectos, una consiste en abrir un proyecto existente y la otra es crear un nuevo proyecto.

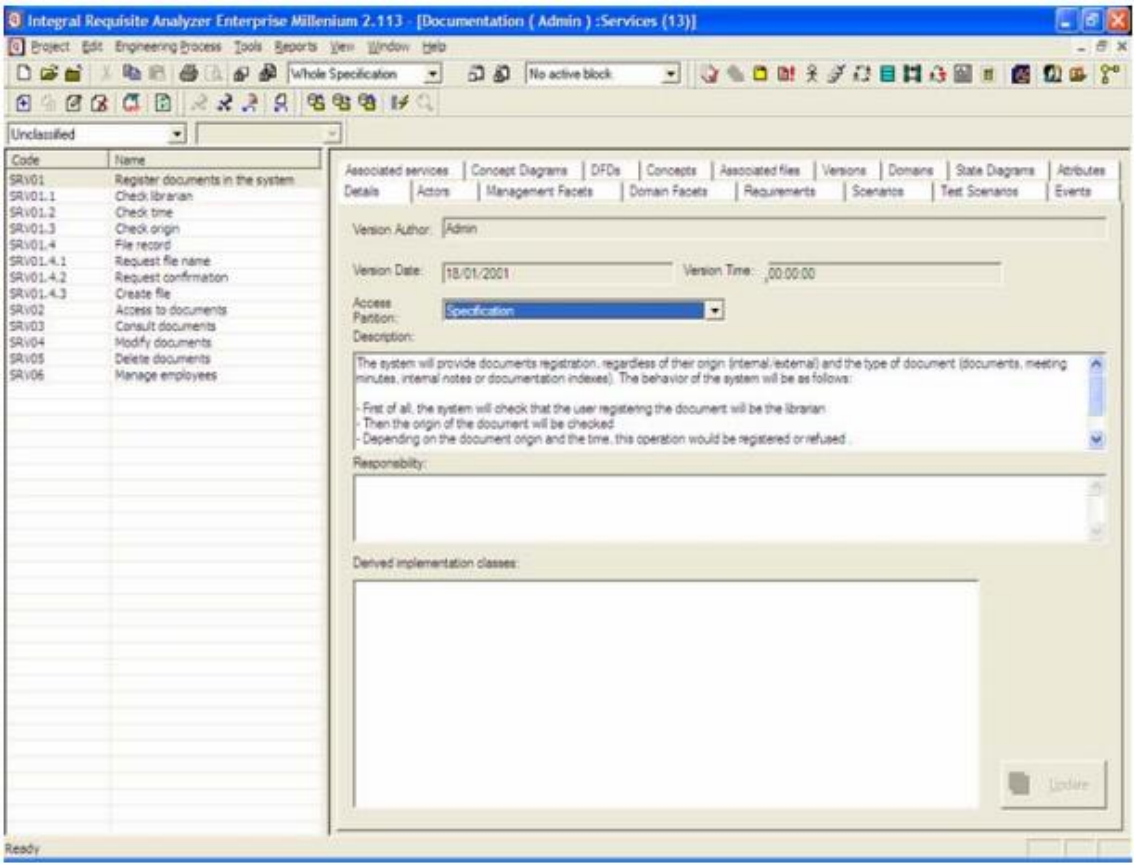


Figura 11. Interfaz de usuario de la herramienta IRqA [11].

2.6.6. Caliber RM

En la figura 12 observamos la interfaz de usuario de Caliber RM que es un sistema de administración de requisitos que permite a los equipos de proyecto ofrecer aplicaciones de mayor calidad que cumplan las especificaciones del usuario final. Ayuda a las aplicaciones a satisfacer las necesidades, al permitir que todos los interesados del proyecto, equipos de marketing, analistas, desarrolladores, probadores y administradores, colaboren y comuniquen la voz del cliente a lo largo del ciclo de vida del software.

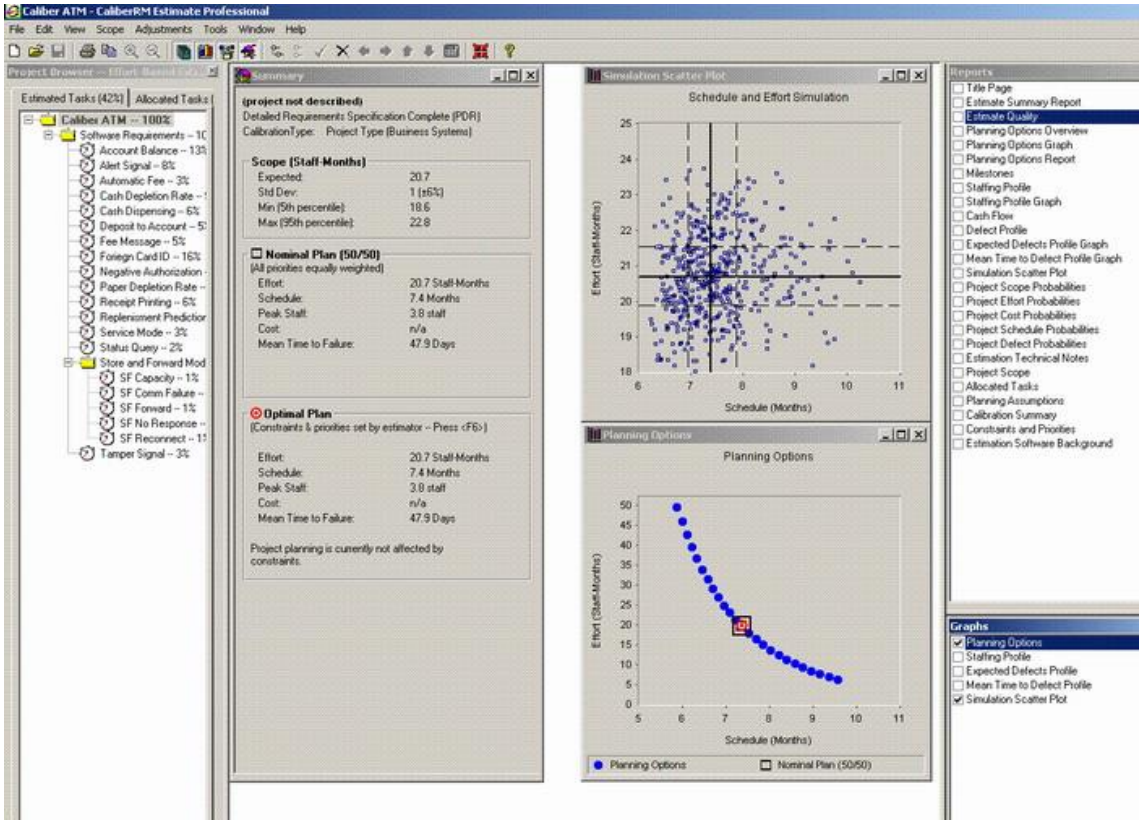


Figura 12. Interfaz de usuario de Caliber RM [54].

2.6.7. RETO

Esta herramienta propone un modelo de requisitos para capturar los aspectos funcionales del sistema; básicamente, mediante tres técnicas complementarias entre sí: la definición de la misión del sistema, la construcción del árbol de refinamiento de funciones y el desarrollo del modelo de casos de uso [55]. Además, se introduce un proceso de análisis que permite traducir el modelo de requisitos en el modelo conceptual, manteniendo la trazabilidad entre ambos y propiciando una representación de la información en el segundo prototipo.

2.6.8. CONTROLA

Herramienta de apoyo al proceso de SE en pequeñas empresas. Se creó gracias a la expansión que tuvo el mercado y a la generación de grandes y pequeñas empresas, las cuales requieren un instrumento para el desarrollo de sus proyectos [55]. En la figura 13 se muestra la interfaz de usuario, en la cual se ofrecen recursos importantes tales como: administración de requisitos, administración de casos de uso, administración de casos de prueba y error, planeamiento de liberaciones, administración de implementaciones, control de

dependencia entre implementaciones, matriz de rastreabilidad y rastreabilidad de los requisitos.

Figura 13. Interfaz de usuario de Controla [56].

2.6.9. OSRMT (Open Source Requirements Management Tool)

La figura 14 presenta la interfaz de usuario de una herramienta libre para la administración de requisitos, cuyas principales características son: trabajar en arquitectura cliente/servidor, estar desarrollada bajo Java; la versión 1.3 trae un módulo para manejar la trazabilidad y lo introduce para el control de cambios; así mismo, genera la documentación de los requisitos tratados [55].

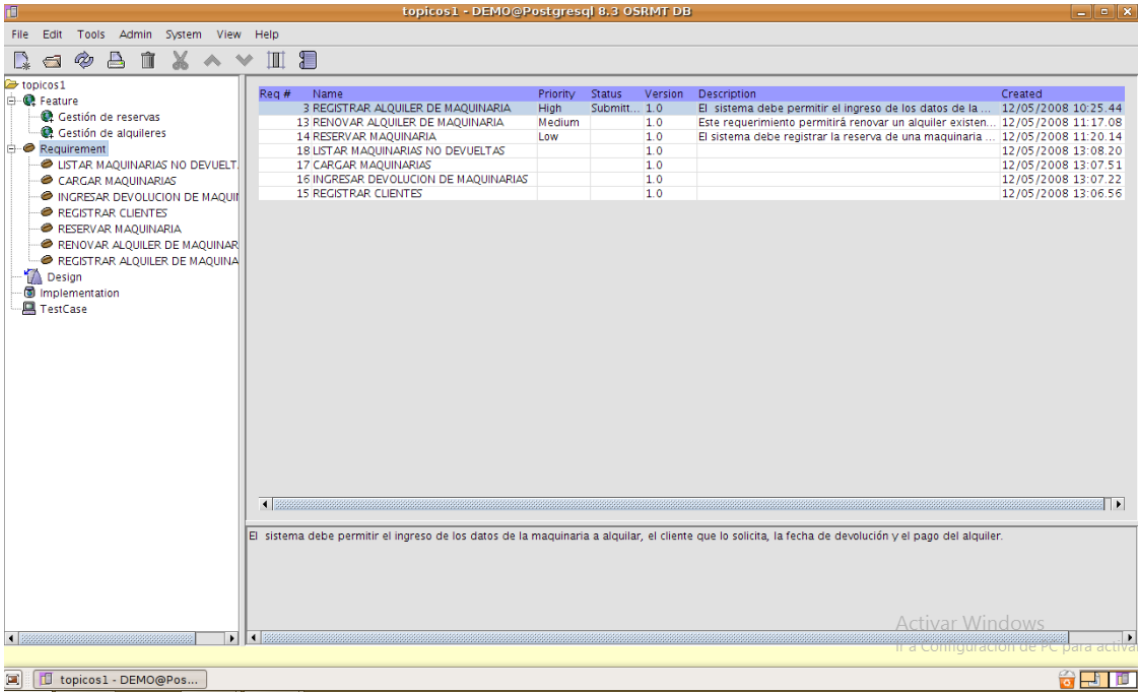


Figura 14. Interfaz de usuario de OSRMT [57].

2.6.10. JEREMIA

Se trata exclusivamente de una aplicación cliente, lo cual no permite la posibilidad de trabajar en equipo [55]. En la figura 15 se aprecia la interfaz de usuario la cual ayuda durante el desarrollo del sistema, especialmente en el seguimiento de cambios de los requisitos a lo largo del ciclo de vida. Con JEREMIA es posible captar las necesidades, analizarlas y clasificarlas. Implementa un módulo orientado a la generación de la documentación posible de exportar en formato DocBook XML, la cual junto con los requisitos, se almacena en una base de datos en MySQL.

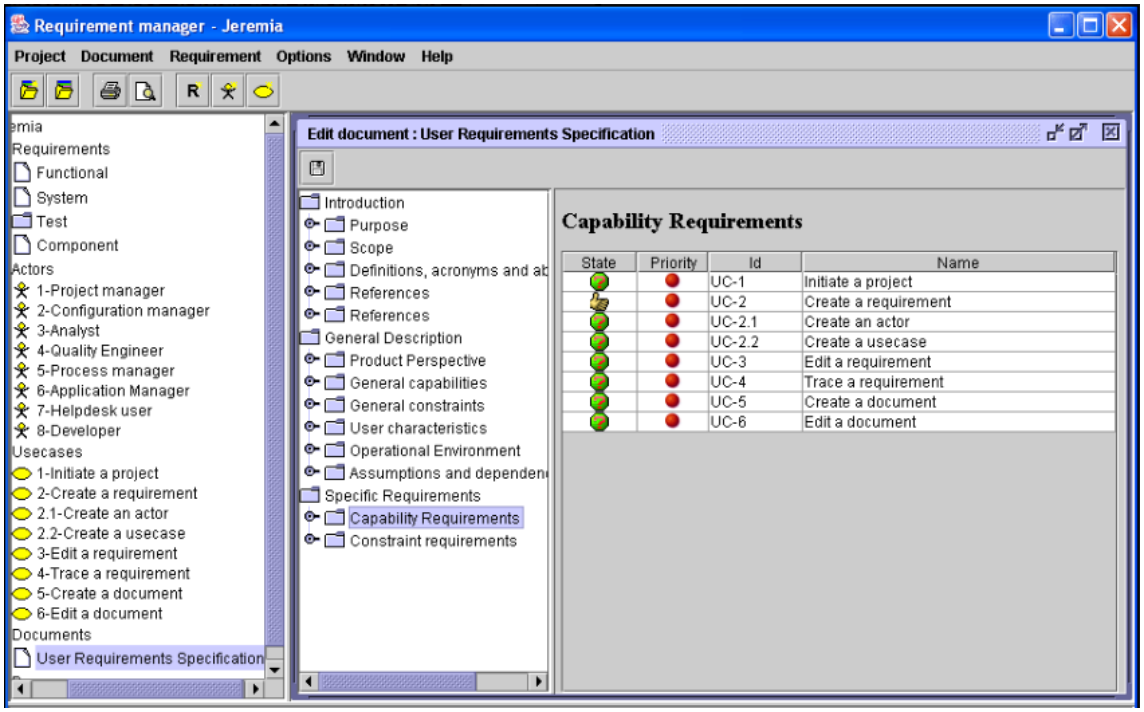


Figura 15. Interfaz de usuario de Jeremia [58].

2.6.11. RAMBUTAN

Esta herramienta está basada en XML, realmente consta de un conjunto de aplicaciones para el usuario final, ayudando a los analistas de sistemas en la recopilación y categorización de hechos en un documento de especificación de requisitos [55]. En la figura 16 se muestra la interfaz de usuario en la cual destaca que tiene un cliente para palm (PDA), el cual se utiliza para recopilar los hechos en el lugar donde está ubicado el cliente mientras que la aplicación de escritorio recibe la información, edita y perfecciona. Ambas aplicaciones permiten al usuario introducir, modificar y visualizar los datos que componen un documento de especificación de requisitos. Comparada con otras herramientas de administración de requisitos, Rambutan ofrece las siguientes ventajas competitivas: aplicación cliente para *palm* (*PDAclass*), portabilidad entre plataformas, es independiente de cualquier metodología de especificación de requisitos y permite distribución libre.

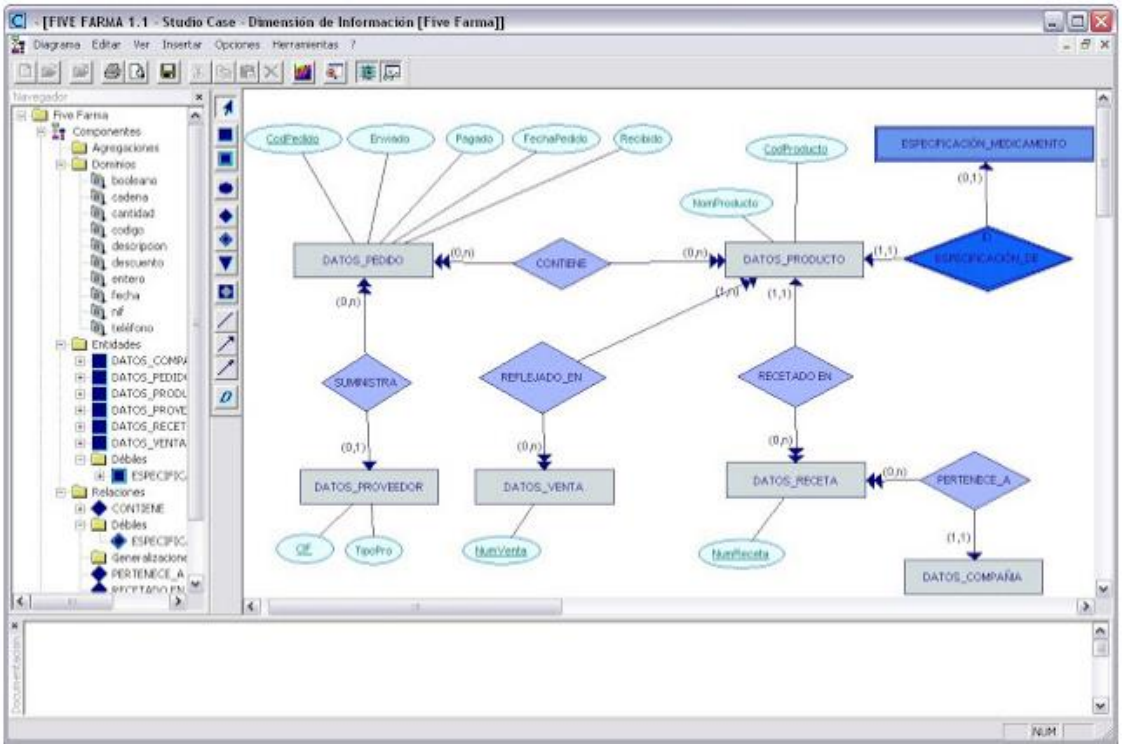


Figura 16. Interfaz de usuario de Rambutan [59].

2.7. Industria del software en Sinaloa

En Sinaloa, la industria de software tiene sus antecedentes a principios de los años ochenta, sin embargo en el año 2002 durante el gobierno de Juan S. Millán 1998-2004, con la apertura del clúster denominado Fondo Impulsor de la Industria del Software en Sinaloa (FIDSOFTWARE), promovido por el sector empresarial y la Secretaría de Desarrollo Económico del estado (SEDECO), se crea un organismo rector en la integración y de una cadena de valor sólida conformada por academia, empresas, gobierno y mercado para el incremento de la facturación del sector mediante la incorporación de la innovación tecnológica como elemento diferenciador [13]. En la tabla 3 se pueden ver los avances de la industria que se visualizan en el informe 2005-2009 presentado por FIDSOFTWARE.

Año	Total Empresas	Empresas Certificadas	Incubadoras	Proyectos PROSOFT	Inversión en la Industria (mdd)	Facturación en la Industria (mdd)	Generación de Empleos
2005	12	0	1	5	25.4	3	200
2006	20	1	2	24	90.1	5	500
2007	30	10	5	55	137.5	10	1500
2008	50	20	7	43	200	20	2200
2009	64	30	12	27	250	30	2500

Tabla 3. Industria del software en Sinaloa en el periodo 2005-2009.

El reconocimiento de los resultados y logros de las acciones implementadas en la industria del software en la entidad se concretó en el año 2007, cuando el Sistema Nacional de Indicadores de Tecnologías de Información (SNITI), ubica a Sinaloa en el segundo lugar, empatado con Jalisco, en el número de certificaciones en procesos de calidad internacional, lo que representaba el 14% del total de las empresas certificadas en algún modelos de calidad en el país: CMMI y Moprosoft [4]. No obstante al evidente crecimiento de la Industria del software en Sinaloa, en el año 2010, se cierra el clúster, y con ello desaparece un organismo estratégico que vinculaba al sector productivo con las universidades mediante proyectos de capacitación y certificación tecnológica de incubadoras y centros de investigación e innovación tecnológica de apoyo a la industria. A raíz de la desaparición de este organismo impulsor de las TIC, la industria del software en la región de Sinaloa, que empezaba a figurar en los mercados nacionales, enfrenta un considerable descenso en el desarrollo de sus productos y servicios.

Es un hecho que la industria del software está evolucionando constantemente, Sinaloa ha sido participe de esta evolución, como muestra de ello encontramos que se han establecido alrededor de 84 PyMES (registradas hasta 2016), algunas de ellas se observan en la tabla 4 [60]. Para lograr que no fracasen y tengan que dejar de existir, deberán continuamente revisar la validez de los objetivos del negocio, sus estrategias y su modo de operación, tratando siempre de anticiparse a los cambios y adaptando sus planes de acuerdo a dicho cambio.

Nombre de la Empresa	Municipio
ARA CONSULTORÍA Y SOLUCIONES	Mazatlán
A B C CONSULTING	Mazatlán
ADSI SOLUCIONES INFORMÁTICAS	Mazatlán
ADSUM SOFTWARE EXPERIENCE	Culiacán
ALGORIA SOFTWARE	Culiacán
API E PRINT, S.A DE C.V.	Culiacán
ASECOM	Culiacán
BISOFT	Culiacán
BM SISTEMAS	Culiacán
CEGA SOFTWARE	Culiacán
CITESIC	Culiacán
COMPAÑÍA MEXICANA DE PROCESAMIENTO	Culiacán
DESPACHO DE CONSULTORIA	Culiacán
DIMA SOFTWARE	Culiacán
DVLUM	Culiacán
E SOFT	Culiacán
ECOMP CONSULTING AND SOFTWARE GROUP, S.C	Culiacán
ERGOSISTEMAS INTEGRALES	Culiacán
ESPACIOS VIRTUALES	Culiacán
FIXCOM	Mazatlán
FORBIDEM 403	Culiacán
FORMANDO REDES	Culiacán
GO BUSINESS WERE	Mazatlán
HUNABSYS INVESTIGACIÓN Y DESARROLLO	Culiacán
IN CONCERT	Culiacán
INNOVAWEB, S.A DE C.V	Culiacán
INNOVAR DISEÑO ARQUITECTÓNICO	Culiacán
INTERFACES	Culiacán
INVENTUM	Culiacán
MESICOM	Mazatlán
MICRO SISTEMAS	Mazatlán
NEORIS CONSULTORIAS	Culiacán
NETDEVSOLUCIONES S.A DE C.V	Culiacán
O K COMPUTER SOFTWARE	Mazatlán

PROYECTOS TELÉFONICOS DEL PACIFICO	Culiacán
RED 2000	Mazatlán
RED PACIFICO	Culiacán
RED RABBIT, S.C	Culiacán
REPARO PC	Culiacán
ROKET	Culiacán
SAIL TECHNOLOGIES SOLUCIONES TI	Culiacán
SC/ ARQUITECTOS	Culiacán
SISTEMAS PRIMUS LABORATORIOS DE MÉXICO	Culiacán
SOLEMTI LIDERES EN ESTRÁTEGIAS WEB	Mazatlán
SOLVENCIA INFORMÁTICA	Culiacán
SYS ADVANCE	Culiacán
SYSTEMATICO TECNOLOGÍA EMPRESARIAL, S.C	Culiacán
TRIPLES SOLUCIONES ADMINISTRATIVAS, S.A DE C.V	Mazatlán
TUCSON INTERNACIONAL	Culiacán
WRP, S.A DE C.V	Culiacán
ZAZU MX	Culiacán

Tabla 4. Fábricas de software en el estado de Sinaloa catalogadas como PyMES.

2.7.1. Problemática

La problemática que enfrenta actualmente la industria del software en Sinaloa se identificó a partir de entrevistas y reuniones temáticas (*focus group*) con actores académicos, de gobierno, empresas desarrolladoras de software y responsables de los *clústeres* principalmente de la zona centro, donde se discutió la importancia para el desarrollo económico de la región y los problemas que se presentan en la actualidad que obstaculizan el crecimiento de este sector.

Los aportes permiten identificar los siguientes problemas [4]:

- Número reducido de fábricas de software a la medida con capacidad para competir internacionalmente debido a la falta del cumplimiento de estándares de calidad y certificaciones internacionales.
- Insuficiente formación del capital humano en gerencia, control, emprendimiento, innovación y dominio del idioma inglés.

- Necesidad de estrategias integrales para mejorar sus beneficios financieros y de infraestructura, especialización del talento humano e impulso a la innovación tecnológica.
- Escasa vinculación de proyectos de investigación e innovación tecnológica entre el sector educativo, empresarial y en consecuencia con el desarrollo regional.
- Infraestructura empresarial por debajo del promedio nacional.
- Escases de centros de desarrollo de software como parques tecnológicos e incubadoras de software y el desarrollo de productos o servicios verdaderamente innovadores. Las empresas se limitan a la generación de desarrollos incrementales o adaptativos suficientes sólo para el mercado local y regional.
- La elaboración de estrategias integrales de fomento se enfocan en las capacidades de producir software y no en las capacidades de producir innovación en las que tenga cabida las exigencias vistas desde las particularidades del sector servicios, en áreas como el financiamiento, y administración del talento humano, la creación de redes, la inversión extranjera y la propiedad intelectual.
- El grado de utilización de las TIC en proyectos del sector público, no ha sido suficiente para fortalecer al sector, a pesar de que el gobierno debiera ser el principal cliente de la industria.
- La generalización de la industria del software como sector de las TIC, los servicios que realizan trabajadores independientes o empresas que no están registradas y las actividades que se realizan , generan una base de datos que no presenta una perspectiva longitudinal y homogénea de la evolución de los servicios de software.
- En el año 2012 de acuerdo con la evaluación de infraestructura empresarial, el Foro Consultivo Científico y Tecnológico (FCCyT) posiciona a Sinaloa en la veinteava posición, lo cual indica que se encuentra por debajo del promedio nacional en este componente.
- En el componente de inversión para el desarrollo de capital humano, el estado ocupa la posición 21 con respecto del total de entidades, también este indicador posiciona a Sinaloa por debajo de la media nacional.

La problemática anteriormente descrita provoca que la competitividad del sector aún sea relativamente baja, lo cual queda de manifiesto en el Índice de Potencial de Innovación Estatal 2010 que de ocupar la posición 10, Sinaloa se colocó en la posición 19 debido a tres indicadores: disminución de patentes solicitadas por cada millón de habitantes, reducción del número de empresas certificadas y crecimiento del Producto Interno Bruto (PIB) del sector servicios en menor proporción que el resto de los estados. En consecuencia, la industria estatal del software desaprovecha la ventaja competitiva que tiene con el principal consumidor de servicios de software, EEUU, donde la proximidad física y cultural y un mismo horario facilitan la comunicación directa con los clientes, factor crítico para la eficiencia de ese sector. En este sentido un estudio realizado en [7] muestra que más del 53% de los proyectos de software fracasan por no realizar un análisis previo de los requisitos. Otros factores importantes son la falta de participación del usuario, requisitos incompletos y sus constantes cambio. Ante esta situación, es importante conocer las metodologías que las empresas han usado e identificar las que han funcionado y aquellas que no, esto debido a que las técnicas empleadas no han sido las correctas o no se conoce bien su uso, motivo por el cual se estudiaron las que han sido empleadas en las empresas sinaloenses y en base a esta investigación se determinaron las ventajas y desventajas que permitieron diseñar el núcleo de la tesis.

2.7.2. Situación actual

El Plan Estatal de Desarrollo 2011-2016 reconoce que los esfuerzos para orientar la vida económica hacia las tecnologías de la información y desarrollo de software aún están lejos de impactar la estructura productiva y plantea en una de las líneas de promoción de proyectos detonantes fomentar la creación de desarrollo de software y TI en universidades, empresas y estructuras gubernamentales, así como servicios relacionados, y estimular el fortalecimiento de estas empresas [4]. En la tabla 5 se observa el plan que sienta las bases para la organización colectiva de los actores e instituciones que buscan nuevamente el desarrollo Industria del Software en la entidad. A partir del año 2012, la colaboración entre los empresarios, gobierno estatal e instituciones de ésta industria.

Sinaloa TI Clúster	Formado por 50 empresas distribuidas en la zonas centro, norte y sur de la entidad, con el propósito de impulsar la industria Impulsar y promover el crecimiento de la Industria de las TI en Sinaloa a través de las empresas que la conforman y fomentar el surgimiento de nuevas empresas, para convertirla en un pilar económico, generador de empleos y de mejores prácticas de calidad y negocios que permitan colocar los productos y servicios en el mercado global y en consecuencia, agregar valor económico tanto a los aliados del clúster como a la sociedad en su conjunto.
Clúster ADETIC	Organismo donde se reúnen 10 empresarios del sector, en la zona norte de la entidad, con el fin de promover el desarrollo de la industria, en busca del crecimiento y consolidación de sus productos a nivel regional, nacional e internacional.
Impulse TI	Asociación de empresas de TI de la zona centro, trabajando en colaboración con el Gobierno de Sinaloa, Gobierno Federal, Instituciones Educativas y Organizaciones afines por el crecimiento del sector, con el fin de desarrollar y consolidar capacidades y competencias organizacionales para la oferta de servicios TI tanto en el mercado nacional como en el internacional.
Clúster TI del Sur de Sinaloa	Agrupación de 7 empresas, colaborando bajo el esquema de la triple hélice que incluye a la Industria, la Academia y el Gobierno del estado para atender las necesidades en el área de Tecnologías de la Información de la región, con el desarrollo de software y soluciones informáticas, generando mayor productividad y competitividad en la zona sur de Sinaloa.

Tabla 5: Clúster de TI en Sinaloa.

Por otra parte, el Gobierno, la Universidad Pública y las PyMES que se dedican al desarrollo de software han constituido en el año 2010 [60] un tridente gobierno-universidad-empresa con la idea de impulsarlas a través del trabajo en conjunto, de tal forma que las universidades, produjeron recurso humano capacitado y actualizado, el gobierno apoyó con subsidio para certificaciones y proyectos y se contrataron a los egresados universitarios. La idea en particular resultó muy atractiva, existieron casos de éxito en la ciudad de Monterrey Nuevo León en donde se impulsó el crecimiento de la industria del software y creció significativamente. Debido al crecimiento e importancia que tuvo la industria de software en el desarrollo del mismo en Sinaloa fue necesario realizar proyectos de investigación que permitieron obtener la condición actual de las PyMES que desarrollan software en cuanto a proceso de desarrollo, aplicación de la RE, y el uso de herramientas de este tipo para poder generar soluciones que funcionen en verdad o de acuerdo al perfil de las empresas Sinaloenses.

2.8. Comentarios Finales

Uno de los puntos críticos en el desarrollo de software es la satisfacción de los requisitos. Se ha demostrado que la mayoría de los errores en los productos de software se producen en la fase RE. También debe considerarse que dentro de la SE, las necesidades están dentro de las primeras fases del proceso de desarrollo de software y el costo asociado con la corrección de un error, una vez que el proyecto se entrega, es significativamente mayor. Por estas razones, es necesario que las organizaciones implementen una metodología de desarrollo de software ad-hoc para su equipo de proyectos, para ello es fundamental la adaptación del proceso de especificación de requisitos de acuerdo a la capacidad de su equipo si desean hacer su proceso de desarrollo más eficiente.

Capítulo III. Estado del Arte

Este capítulo se enfoca a la revisión bibliográfica y el estado del arte acerca de métodos, técnicas, metodologías y herramientas de soporte para el proceso de desarrollo de software enfocado en la Ingeniería de Requisitos (RE) La revisión bibliográfica se llevó a cabo a través de un análisis de la literatura (*Systematic Literature Review,* SLR), el cual es un estudio derivado del área médica que fue adaptado a la Ingeniería de Software por Barbara Kitcheham. Una SLR es una forma de evaluar e interpretar toda la investigación relevante disponible acerca de una interrogante de investigación particular, en un área temática o fenómeno. Los estudios individuales que contribuyen a una SLR se denominan estudios primarios, una revisión sistemática se considera un estudio secundario. En particular este método propone tres etapas fundamentales que son (i) planificación, (ii) desarrollo y (iii) publicación de los resultados de la revisión, las que a su vez se encuentran dividas en otras etapas que detallan la forma en que se deben desarrollar.

3.1. Metodologías y métodos utilizados en la Ingeniería de Requisitos

La aplicación de la SE en el proceso de desarrollo de un sistema es un mecanismo que se emplea para garantizar la calidad de un producto de software. Una de las etapas importantes del proceso de desarrollo es la RE, etapa en donde se definen inicialmente las características y restricciones con las que debe contar el sistema en desarrollo. En este sentido, en la literatura actual se han presentado trabajos enfocados a paliar las limitantes encontradas en la RE, entre ellos se encuentra el trabajo presentado en, en el cual realizó un análisis de las técnicas, métodos y herramientas utilizadas en el proceso de RE, en la investigación se seleccionaron algunas de las técnicas expuestas para aplicarlas en el desarrollo del caso de estudio, el cual trataba acerca de un Sistema de Inscripción Virtual (SIV), sin embargo, los resultados obtenidos no fueron satisfactorios, es decir, ninguna de las metodologías que se analizaron comprendían todas las actividades que se definieron en el proceso de la RE, debido a que solo se contemplaron las tradicionales [11]. Por lo que se definió un proceso que cubriera todas las actividades y tareas de la RE, así como técnicas específicas para lograr realizar cada actividad, considerando las

características del sistema, del entorno y de los usuarios. Además, se propuso un esquema del documento de requisitos tomando como base el estándar definido por la IEEE. La conclusión a la que llega el autor es que es difícil proponer una técnica o método para cubrir las especificaciones de los requisitos, por lo que comenta que lo ideal es combinar varios métodos de acuerdo al proyecto a desarrollar.

Hoy en día se reconoce ampliamente la necesidad de un enfoque disciplinado en el desarrollo de software y existe un cierto consenso en cuanto a que la SE ha ganado madurez, sin embargo, la explosión tecnológica que se viene produciendo desde la segunda mitad de los años 90's y que por ejemplo se puede observar en el auge de múltiples tecnologías de componentes distribuidos y en la evolución de las comunicaciones, no ha sido seguida por una evolución de la misma magnitud en la práctica de la SE.

La tesis doctoral de [6], se enfoca en un aspecto concreto en el desarrollo de software, a través de una propuesta de administración de requisitos en líneas de productos de software que integran modelos y requisitos textuales. Con este trabajo se trata de acercar la investigación en la práctica de la RE, en particular en líneas de producto. Por un lado, la situación se ha centrado en modelos de requisitos, y en el marco particular para las líneas de productos, en modelos que representan de un modo u otro la variabilidad en los productos. Por otro lado, en la práctica todavía muchas personas involucradas en algún sentido en el desarrollo de software prefieren la documentación en lenguaje natural de los requisitos. El interés de este trabajo se justifica en la importancia de la RE en el desarrollo de software y en la conveniencia de administrar de forma integrada modelos de requisitos y especificaciones textuales, generando documentación de requisitos en lenguaje natural de forma sincronizada con los modelos desarrollados. Esta propuesta se fundamenta también en la necesidad de mejorar la reutilización del software, como vía necesaria para mejorar la calidad y productividad en el desarrollo de un producto.

Respecto a las metodologías utilizadas, el autor en [80] plantea una basándose en la descomposición de las tres etapas del eje central del proceso de RE (elicitación, especificación y validación), la cual denomina Documentación de Requisitos Centrada en el Usuario (DORCU) caracterizada por su flexibilidad y orientación al usuario. Esta considera los mejores resultados de otros enfoques

estudiados y se apoya en diversos métodos, técnicas y herramientas desarrolladas por otros autores, sin comprometerse con un paradigma particular. Este procedimiento consta de cuatro fases: elicitación de requisitos, análisis de requisitos, especificación de requisitos y validación y certificación de los requisitos. De acuerdo a los autores de la misma, aporta las pautas para entender otras metodologías propuestas, contribuye a un mejor entendimiento de la RE, ofrece libertad para la selección e integración de diferentes herramientas a emplear en cada fase del proceso, contempla aspectos metodológicos destinados a lograr un documento de requisitos de clara interpretación, puede ser aplicada a diferentes paradigmas y cubre errores en el tipo de documentación generados al proponer documentos isomorfos (Documento de Requisito Técnico ($DR_{T)}$, y Documento de Requerimientos del Usuario ($DR_{U)}$).

Otra de las metodologías propuestas, es la de Ancora (Análisis de Requisitos Conducente al Reuso de Artefactos) [61]. Esta al igual que las anteriores, se basa en algunas de las actividades fundamentales del proceso de Ingeniería de Requisitos ya descritas, esto es, la deducción de requisitos, el análisis y la validación. Afirman que esta metodología ha sido utilizada exitosamente en más de cincuenta proyectos de software, con gran satisfacción por parte de los usuarios.

En [62] se presenta un proyecto de investigación que tiene como objetivo proponer una metodología de RE para el análisis de Sistemas Embebidos (SE). El modelo de requisitos propuesto y su forma de utilización se ilustran mediante un caso de aplicación consistente en la obtención de datos para un sistema de censado de movimiento, embebido en un sistema de alarma para hogar. Algunas metodologías orientadas a sistemas embebidos sólo permiten expresar los requisitos que se encuentran en el nivel de sistema, es decir, los que tienen que ver con funcionalidades, pero olvidan aquellos que están en niveles externos, en donde se encuentran las interrelaciones con el súper-sistema y otros sistemas embebidos. Como conclusión comenta que el modelo propuesto en este trabajo explota y especializa las categorías del modelo de la metodología ABC-Besoins, mostrando que las grandes categorías de ésta cubren las especificaciones de los sistemas embebidos, confirmando de esta manera la universalidad y expresividad de dicho modelo de requisitos.

IBRA [64.] (*Inquiry-Based Requirements Analisys* o análisis de requisitos basado en preguntas), es otra metodología de análisis de requisitos. El cual consiste en la propuesta de una estructura cíclica denominada “ciclo de investigación”, la cual sirve de guía en la administración de los requisitos. Este modelo se divide en tres etapas: la especificación de requisitos, análisis y discusión y la evolución de los mismos, finalmente los autores recomiendan tomar en cuenta a la hora de aplicar el modelo, los siguientes aspectos: el modelo debe ser dinámico, no debe estar asociado a un lenguaje de especificación o estilo de expresión, es útil aplicarlo con el respaldo de tecnología de hipertexto, pero esta no es una condición obligatoria.

3.2. Técnicas utilizadas en la Ingeniería de Requisitos

Para el análisis de los requisitos existen varias técnicas, siendo los escenarios una de las que permite describir el comportamiento esperado del software en un lenguaje reconocido por los involucrados con lo que se logra una buena comunicación con el cliente. Por otra parte, los patrones permiten reutilizar elementos previamente comprobados funcionando como un registro de experiencias que ayudan a agilizar y mejorar los procesos. En [15] se presenta una propuesta para la definición de patrones de escenario y un catálogo para dominio de la telemática que permiten una descripción clara y precisa de las especificaciones en esta área. El uso de patrones en la construcción de escenarios permite ahorrar tiempo para la elaboración de los mismos, ofreciendo una visión estructural de las situaciones, y con esto determinar el tipo de escenario que corresponda a cada una de ellas. El catálogo propuesto se basa en el análisis realizado a distintas fuentes de información y problemas de definición de estos requisitos en el dominio de empresas que se dedican al desarrollo de software en el área de telemática, específicamente orientado al área de transporte, y que presentan necesidades donde sus clientes desean obtener información sobre sus unidades en tiempo real. Como conclusión, el autor menciona que una de las ventajas que tienen los escenarios es la posibilidad de ser construidos a partir de patrones, de tal manera que es posible reutilizar las soluciones a situaciones similares que se presentan en diversos proyectos. El contar con una representación formal para un escenario facilita la creación de un catálogo de patrones que ayude a agilizar y hacer más segura la

definición de requisitos. Necesitamos buscar técnicas que hayan sido usadas creadas o existentes de otras áreas para los requisitos (en cualquiera de las actividades de la RE).

El autor en [63] basándose también en algunos de los elementos ya discutidos del proceso de RE, plantea un modelo para el proceso de toma de requisitos. En el cual parte indicando que, a pesar de las múltiples mejoras hechas en varias de las actividades involucradas en el proceso de desarrollo de software, la elicitación, análisis y modelado de requisitos de usuario, son las actividades menos exploradas, y afirma que estas actividades son las más importantes, pues una validación y corrección temprana de los requisitos de usuario, ahorraría muchos de los problemas asociados al desarrollo de software, particularmente en la fase de manutención. Está claro, que es mucho más barato detectar y corregir errores temprano dentro del ciclo de desarrollo de software, que más tarde. También aporta muchos otros comentarios y razones publicados en diversos artículos, que justifican la afirmación de que la RE, es un área muy interesante para investigar. Apoyado en estas razones, plantea su modelo, el cual se basa fundamentalmente en las fases de elicitación, modelado y validación, utilizando algunas técnicas de Adquisición de Conocimiento (KA), particularmente *Ripple Down Rules* (RDR) para la fase de especificación de requisitos y la técnica de modelado conceptual Formal *Concept Analysis* FCA para el análisis de requisitos.

El autor en [64], presenta una propuesta para elicitar requisitos en *Data Mining* para proyectos de *Business Intelligence* (DM-BI). En esta propuesta, se afirma que existe la necesidad de adaptar el proceso de ingeniería de requisitos tradicional, para su aplicación en proyectos de DM-BI. Esta afirmación, se basa en la premisa de que el análisis de requisitos para este tipo de proyectos, difiere sustancialmente del que se realiza en los sistemas de información convencionales. Evidencia de esta afirmación, se encuentra en la observación de los problemas más comúnmente presentados en el desarrollo de una amplia gama de proyectos de DM-BI, tales como: el cliente no comprendía las técnicas ni el léxico utilizado por el grupo que desarrolla el proyecto, no tenía claras las metas del proyecto de DM-BI o lo que se podía lograr con su desarrollo, o los modelos definidos por los desarrolladores eran diferentes a los que él había previsto. Para resolver estos y otros problemas, es necesario deducir información

específica en cada proyecto de DM-BI. Esta información puede ser modelada por un catálogo de conceptos los cuales deben ser identificados; basados en esta lista de conceptos que modelan la información relacionada con los problemas observados, se propuso un método compuesto por cinco pasos que son: 1) Comprender el dominio del proyecto, 2) Conocer el dominio de los datos del proyecto, 3) Comprender el alcance del proyecto, 4) Identificar los recursos humanos y conocimientos requeridos, y 5) Seleccionar las correctas herramientas de DM-BI, la propuesta se focaliza, en la identificación de información que permita comprender el dominio de un proyecto de *Data Mining*, en base a la experiencia en el desarrollo de este tipo de productos. Estos conceptos están relacionados con aspectos tales como: la comprensión del dominio del proyecto de DM-BI (establecimiento de los canales de comunicación necesario). La metodología ER-DM [64] define un proceso iterativo e incremental que, a partir de un conjunto de requisitos organizacionales preliminares y un conjunto de elaboraciones sucesivas de desarrollo de modelos de prueba, deriva finalmente, en la versión final del documento de contrato. También la metodología considera, la definición de un conjunto de plantillas para la descripción de los requisitos organizacionales y del sistema a ser desarrollado.
El autor en [65.] propone una técnica utilizada para la definición de requisitos en proyectos de *Data Warehouse* conocida como DWARF (*Data Warehouse Requirements deFinition*). Esta técnica, está estructurada en cuatro etapas, planificación de la gestión, especificación, validación y control de la administración de requisitos. En la primera etapa de la definición de requisitos, se debe generar un conjunto de pautas orientadas a la correcta aplicación de la RE en sistemas de *Data Warehouse*. La definición de estas pautas, en términos de reglas de negocio, procedimientos o procesos, permitirá la correcta aplicación de la técnica DWARF y clarificar aspectos tales como, cuál es el grado de granularidad de cada *Data Mart* o qué limitaciones restringen el análisis multidimensional de datos.

3.3. Herramientas de soporte

En el trabajo presentado en [66] el autor desarrolló una herramienta para análisis y modelado de requisitos en aplicaciones web aplicando el lenguaje de modelado orientado a objetivos i*, la herramienta se realizó a través del paradigma de

desarrollo de software dirigido por modelos y fue acreedora al reconocimiento a la mejor demostración de software en la *12th International Conference on Web Engineering.* Su funcionamiento es el siguiente: el analista comienza a modelar la aplicación web especificando los requisitos en un modelo, una vez creado el prototipo que representa a la aplicación web, el cliente lo verifica, y si está de acuerdo a través de una serie de transformaciones se generan el modelo de navegación y el modelo de dominio que puede ser enriquecido por el desarrollador mismo y reutilizable en otros proyectos.

El autor en [67] plantea un método y un prototipo que se basa en estructuras gramaticales que sirven para representar y enunciar los objetivos y los problemas en los diagramas causa-efecto y de objetivos de KAOS (*Knowledge Acquisition in Automated Specification)*, además, propone unas reglas de consistencia que permiten establecer al momento de la construcción del diagrama la relación entre los objetivos y los problemas de estos. Aunque es un gran avance en la etapa de educción de requisitos de la SE, es importante resaltar que este trabajo lo realiza utilizando lenguaje controlado, el cual minimiza la expresividad del interesado, además, para el uso de este lenguaje debe mediar una capacitación previa entre interesado y analista.

3.4. Ingeniería de Requisitos en PyMES

En los últimos años el crecimiento de desarrollo de software ofrece la oportunidad de producir productos y servicios a bajo costo, esta situación hace el desarrollo de software atractivo para las fábricas, por lo tanto, alrededor del 99% de las empresas en América Latina están compuestas por PyMES. En este contexto, estas tienen una necesidad continua de mejorar su proceso de desarrollo, así como la, calidad de los productos, con el fin de mantenerse en el mercado y lograr un crecimiento constante. Para cubrir esta necesidad, las empresas han preferido la aplicación de las metodologías ágiles. Desafortunadamente, la mayor parte del tiempo la aplicación de estas metodologías se basa en los beneficios que se producen en otras organizaciones. Esta falta de conocimiento, se ve reflejada además en los productos de baja calidad, así como en ciclos de desarrollo de software que no tienen el rendimiento esperado.

En el trabajo presentado en [49] se expone la situación de los métodos, los marcos y las herramientas desarrolladas para evaluar si una organización ha adoptado y utilizado de una manera correcta los principios ágiles enfocando en las PyMES de Latinoamérica, así como identificar la metodología ágil más usada. Como conclusiones presenta el resultado del análisis de los datos obtenidos de un estudio primario, es importante destacar que la mayoría de las fábricas utilizan metodologías ágiles principalmente SCRUM y sus variantes. Los resultados de la encuesta, validan el incremento en el uso de esta metodología, sin embargo, se mencionaron además las tradicionales o de una combinación de ambas.

Por otra parte, en el trabajo de [68] se presenta una comparativa entre teoría y realidad de la caracterización de necesidades que presentan las PyMES para implementar una mejora de procesos de software exitosa. En específico el estudio presentado en este artículo se centra en la Región de Zacatecas, México y los resultados obtenidos al realizar una revisión de literatura enfocada en la caracterización de estas. Como conclusión, expone qué se ha identificado como acción para apoyar a las fábricas en la utilización de mejora de procesos de acuerdo a las necesidades identificadas y la incorporación de herramientas que soporten y proporcionen apoyo en el funcionamiento de los mismos. Lo anterior a través de talleres enfocados en el desarrollo, con el fin de motivar a los empleados a implementar un avance de procesos de software.

3.5. Ingeniería de Requisitos en aplicaciones móviles

Los dispositivos móviles forman parte de la vida cotidiana y son cada vez más sofisticados, su poder de cómputo genera posibilidades hasta hace años no pensadas [69]. La creciente demanda de software específico para estos mecanismos ha generado nuevos desafíos para los desarrolladores, debido a que este tipo de aplicaciones tiene características propias, restricciones y necesidades únicas, lo que difiere del desarrollo de software tradicional. Las particularidades de este entorno incluyen: alto nivel de competitividad, tiempo de entrega necesariamente corto y la dificultad adicional de identificar los *stakeholders* y sus requisitos. Las aplicaciones se generan en un entorno dinámico e incierto; generalmente, son pequeñas, no críticas, aunque no menos importantes. Están destinadas a un gran número de usuarios finales y son liberadas en versiones rápidas para poder satisfacer las demandas del mercado.

El desarrollo de aplicaciones móviles es, actualmente, un gran desafío, dado las demandas específicas y las restricciones técnicas de un entorno móvil, tales como dispositivos con capacidades limitadas, pero en evolución continua; varios estándares, protocolos y tecnologías de red, necesidad de operar sobre diferentes plataformas, requisitos específicos de los usuarios y las exigencias estrictas en tiempo del mercado.

En el trabajo de [69] se presenta un estudio comparativo de tipos de aplicaciones para dispositivos móviles, a partir de un caso experimental desarrollado para la plataforma educativa Web UNLP que es un entorno virtual de enseñanza y aprendizaje que permite a los docentes mediar sus propuestas educativas. El desarrollo planteado en este trabajo consiste en extenderlo, con la construcción de una aplicación móvil que permita acceder a determinadas funcionalidades del sistema a través de un artefacto móvil. El enfoque propuesto incluye un análisis de la misma solución, comparando el desarrollo nativo, web e híbrido, a fin de establecer cuál de ellos es conveniente. Como ocurre con cualquier desarrollo de software, la construcción de una aplicación móvil implica tener claramente definido su propósito y cuáles requisitos debe cumplir, como conclusión se tienen que las versiones nativas han cumplido todas las necesidades. Aunque la mayor desventaja consiste en la no portabilidad, lo que implicó un desarrollo específico para la plataforma que se desee cubrir. La versión híbrida ha logrado conjugar la simpleza del desarrollo web con el uso de todas las capacidades del dispositivo. Este tipo de enfoque pretende suplir las desventajas de los dos enfoques previos, hecho que lo posiciona como la primera elección condicionada por los requisitos específicos a cumplir.

En [70], el autor presenta una síntesis sobre el desarrollo de aplicaciones, sistemas operativos y metodologías de desarrollo, para lo cual, se han seleccionado las tres (3) metodologías ágiles más referenciadas, con mayor presencia de documentación en Internet y orientadas a desarrollos de tamaño reducido propio de las aplicaciones para dispositivos móviles, como son *Extreme Programing (XP), SCRUM y Test Driven Development* (TDD), dentro de este trabajo se exponen algunas características de las metodologías más usadas; así como algunas generalidades del desarrollo de estas aplicaciones con características de sus sistemas operativos, para concluir con una revisión de las metodologías utilizadas actualmente en el desarrollo de aplicaciones para

dispositivos móviles. Como conclusión aporta que las metodologías ágiles son una excelente alternativa para guiar proyectos de desarrollo de software de tamaño reducido, como es el caso de dispositivos móviles, gracias a la gran facilidad de adaptación que poseen; pero estas deben ser adaptadas a las características especiales de estos dispositivos con el fin de obtener productos de calidad.

3.6. Ingeniería de Requisitos en el desarrollo de aplicaciones Web

La RE se encarga precisamente de establecer un proceso ingenieril para la captura, análisis, comprensión, documentación y representación de los requisitos de un sistema software, tal como un sitio Web [71]. El criterio principal de calidad de un sistema software es el grado de satisfacción de la especificación de las necesidades del sistema. La calidad de los desarrollos Web ha sido severamente cuestionada por las investigaciones específicas. Un estudio del Cutter Consortium sobre desarrollo de aplicaciones Web estableció que sólo el 16% satisfacen plenamente los requisitos de los contratantes entre el 53% que no los satisfacen.

En el trabajo presentado se exponen los resultados de una investigación exploratoria de las prácticas de la RE en el desarrollo de aplicaciones Web [71]. Se identificaron las técnicas utilizadas en el proceso de elicitación clasificándolas en tres grupos de actividades: Captura, Definición y Validación. Para analizar las prácticas en el desarrollo Web, se realizó una investigación experimental mediante una encuesta a desarrolladores que utilizan alguna plataforma Web. Partiendo de la interdependencia entre el problema y la solución, en el caso de utilizar las metodologías tradicionales se requieren ajustes al nuevo dominio, lo que se detectó en este trabajo corresponde a interrogarse si, pese a los ajustes de los enfoques más estabilizados que se utilizan habitualmente no introducen restricciones a las necesidades de los procesos de las organizaciones cuando son utilizados para el desarrollos Web.

En [44] se presenta el concepto de RNF dentro de la literatura existente en la RE y establece como base conceptual que los RNF son requisitos de calidad y son restricciones. En base a esto se realizó un estudio comparativo de seis enfoques de desarrollo de aplicaciones Web. Se estudian los procesos que cada una de ellas utiliza para determinarlos; las técnicas de la RE que proponen para su

elicitación, especificación, validación y administración y se establece en qué fase del ciclo de vida de desarrollo de software se identifican y tratan a los RNF, como conclusión de su trabajo comenta que si bien actualmente la RE proporciona numerosas técnicas y herramientas, no son aplicadas muy a menudo por los profesionales de sistemas, particularmente en el desarrollo Web. La madurez del proceso de RE parece ser insuficiente y demanda que surjan nuevos enfoques o evolucionen los existentes para el tratamiento de los RNF en particular. Por su importancia, merecen contar con técnicas y lineamientos específicos.

El autor de [72] desarrolla una propuesta denominada NDT (*Navigational Development Techniques*), que es un proceso de desarrollo para sistemas Web que se mueve en las primeras fases del ciclo de vida. Su característica principal es que se centra en una exhaustiva etapa de requisitos que sirve como base para todo el proceso de desarrollo. NDT se centra en una detallada fase de RE guiada por objetivos, que contempla tanto la captura, como la definición y la verificación de los mismos. Comienza definiendo los objetivos y en base a éstos se describe un proceso por el que se pueden capturar y definir las diferentes especificaciones del sistema. Éstos son clasificados y tratados dependiendo la tipología a la que pertenezcan. La característica más destacable de NDT es que el paso de especificación de requisitos a estos modelos se hace de una manera sistemática e independiente, porque NDT define algoritmos que indican cómo conseguir cada modelo a partir de la definición. Y es independiente porque, a pesar de que existen relaciones entre los modelos, hecho que es imposible de evitar puesto que todos se refieren a un mismo sistema, no es necesario conseguir el modelo conceptual para conseguir el modelo de navegación o el de interfaz abstracta.

El trabajo del autor [73] presenta el Método Orientado a Objetos para Soluciones Web (OOWS) que es una metodología de desarrollo que permite especificar sistemas de software para ambientes Web, extendiendo un método orientado a objetos (OO) existente. Este tipo de aplicaciones tiene dos puntos en común con las aplicaciones de software tradicionales: la funcionalidad del sistema y la interacción con los usuarios. Sin embargo, introduce características navegacionales para representar de una manera más precisa la interacción de los usuarios y componentes de la aplicación. Esta metodología produce un

desarrollo de software rápido, usa una estrategia de código basada en modelos, importante para desarrollar la aplicación Web en un tiempo planificado y limitado. Es presentado en [74] el uso de la metodología A-OOH (*Adaptive Object Oriented Hypermedia*)**.** El cual es una extensión del método de modelado OOH (*Object Oriented Hypermedia*) dirigido al usuario. Las técnicas utilizadas en esta aproximación para la especificación de requisitos son el framework i* y UML-Profiles. El proceso se lleva a cabo por el diseñador utilizando modelos i*, SR (*Strategic Relationship*) y SD (*Strategic Dependency*). Los requisitos son basados en las metas y objetivos del usuario, de esta manera evita la especificación de requisitos de forma textual (con el esfuerzo que implica realizarlo de esa forma) y la basada en tareas (la cual depende en la mayoría de los casos de la experiencia del analista).

El resultado de un estudio comparativo que analiza las diferentes metodologías que actualmente son aplicables al desarrollo de un Sistema de Información Web (SIW) [75]. Un aspecto importante a considerar en relación con estos sistemas es la necesidad de ofrecer al usuario un sistema simple y navegación con una interfaz de calidad que proporciona recuperación de información eficaz. Esto, junto con el hecho de que en muchos casos el usuario tiene múltiples funciones, cada una de las cuales requiere una interfaz adecuada, significa que la interfaz, la navegación y la información recuperada son todos aspectos vitales a considerar cuando desarrollo de Sistemas de Información basados en la Web (WIS). Como conclusión el comenta que ninguna metodología ofrece un marco totalmente adecuado para el desarrollo del Sistema de Información y Operación (SIO). Una metodología para el SIO son fases del ciclo de vida, indicando las actividades a seguir en cada uno de ellos, las técnicas a aplicar y los productos resultantes de ellos. Pero también debe permitirnos tratamiento de los aspectos clásicos de los sistemas de gestión, requisitos de almacenamiento y funcionalidad, así como las nuevas características que WIS heredan de la multimedia: navegación, la importancia de la interfaz y multimedia.

3.7. Comentarios Finales

En Sinaloa para lograr el crecimiento, en la industria del software es necesario realizar proyectos de investigación que permitan conocer la situación actual de las fábricas de software. En la actualidad se ha hecho evidente el fracaso de

algunas empresas las cuales han cerrado antes de cumplir los 5 años de actividad, esto debido a ciertos factores como: la falta de experiencia, teniendo al mando de la empresa a personas sin el conocimiento de cómo solucionar los problemas que se presentan, así como personal en el área de desarrollo donde no solo es indispensable el conocimiento sino que también es importante la experiencia, otro factor es la falta de análisis, a veces no es claro lo que se desea realizar, tanto por el cliente como por los desarrolladores y junto lleva a desarrollar un producto que a corto plazo necesitará una modificación, el no utilizar alguna metodología o técnica apropiada para la obtención de los requisitos lleva a que en el proceso de desarrollo se necesiten hacer algunos cambios, los cuales retrasan los tiempos e incrementan los costos del mismo.

Con esto se ha podido identificar que se han realizado trabajos en los que se ha visto la falta de la práctica de la RE en el desarrollo de proyectos tal es el caso del trabajo realizado por el autor de [11], en el cual hace una comparación de las técnicas, métodos y herramientas que se utilizan en el desarrollo de software, a lo cual difiere con mi trabajo porque en ésta tesis se presenta una aproximación metodológica para la RE en el proceso de desarrollo de software en las empresas catalogadas como PyMES en el estado de Sinaloa.

Por otra parte, en el análisis realizado a las fábricas de software, de acuerdo a las entrevistas y cuestionarios realizados como complemento a la tesis (dentro del Proyecto PROFAPI 2014/002.) nos arroja que la mayoría de las empresas utiliza metodología interna, es decir, desarrollada por ellos mismos de acuerdo a sus necesidades, utilizando para el desarrollo de la misma técnicas como entrevistas, cuestionarios, lluvias de ideas, casos de uso, apoyadas con herramientas como diagramas de clases, diagramas de flujo de datos, diagramas de secuencias, historias de usuarios, etc. Se vio que no en todos los proyectos siguen la misma metodología u orden, es decir, si anteriormente ya se había trabajo algo similar se toma ese caso como base para el desarrollo del nuevo proyecto, también cabe mencionar que hay incluso algunas fábricas que si toman para su desarrollo metodologías como SCRUM o XP, pero no siguen al 100% el proceso, esto por tiempos o desconocimiento de las fases algunas no son incluidas. A continuación se muestra una gráfica (figura 17) en donde se presentan las técnicas más utilizadas para la RE.

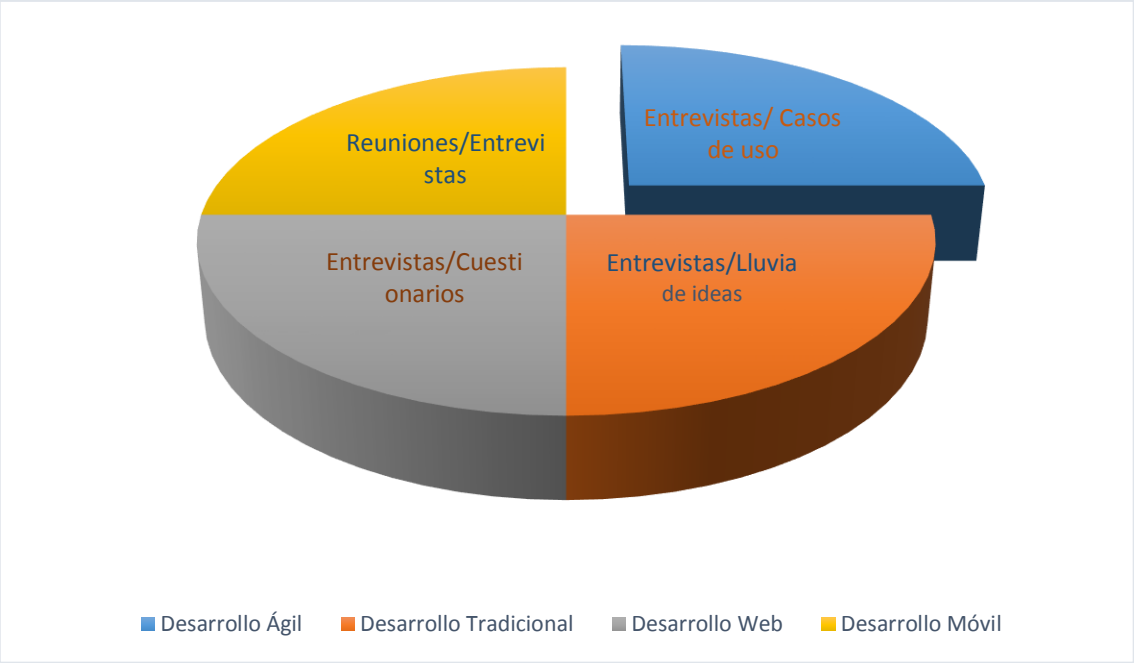


Figura 17. Técnicas más utilizadas para la Obtención de Requisitos.

Además, se ha identificado que actualmente no existe un método formal que apoye a las PyMES en la implementación y uso adecuado de las metodologías ágiles, los modos actuales se centran en los años de uso de la metodología dentro de la organización de forma continua, el nivel de los usuarios de la misma o el tipo de proyectos en los que es usada.

Capítulo IV. Estudio de Técnicas en empresas.

En este capítulo se muestra un estudio de campo que se realizó a 25 fábricas desarrolladoras de software del estado de Sinaloa, con la finalidad de obtener información sobre métodos, técnicas y herramientas de la Ingeniería de Requisitos (RE) que utilizan para el desarrollo de un proyecto.

Se realizaron entrevistas y encuestas las cuales incluían una serie de preguntas abiertas y cerradas, diseñadas para obtener información de tipo prevista e imprevista, las cuales se prepararon usando la herramienta de *Google Forms* para facilitar el proceso de recolección de información. Para la investigación se tomaron empresas de las ciudades de Mazatlán, Culiacán y los Mochis.

Para la selección de las fábricas a las cuales se les aplicó el cuestionario se tomó como base la lista de aquellas dedicadas al desarrollo de software del estado de Sinaloa [60.]. En la figura 18 se puede ver el porcentaje y la ubicación de las mismas.

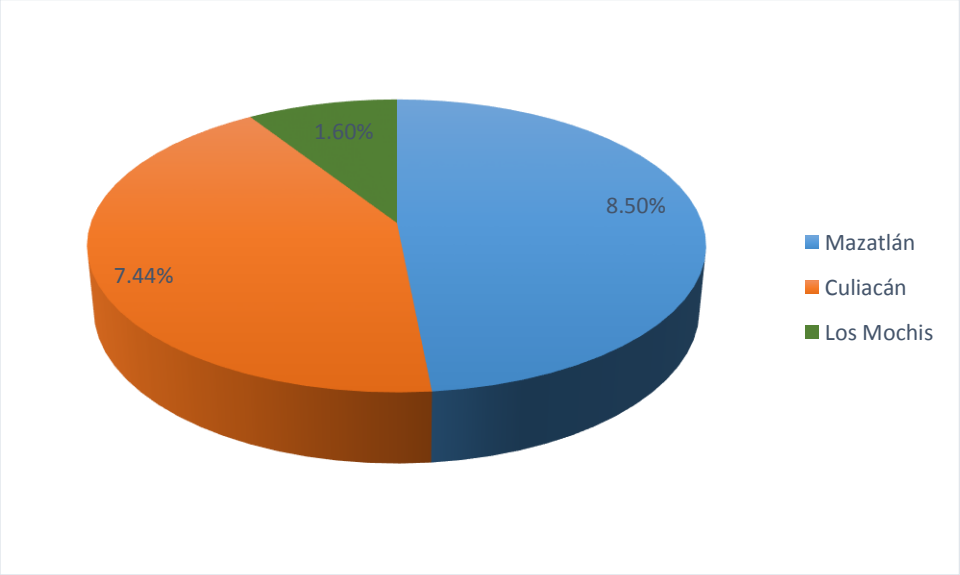


Figura 18. Ubicación de las empresas seleccionadas.

Como se observa en la figura 18 la mayoría de las empresas entrevistadas pertenecen a Mazatlán con un 8.50%, a ello le sigue Culiacán con un 7.44% y Los Mochis con un 1.60%.

Para el desarrollo de la investigación fue de gran importancia conocer la edad de las empresas, debido a esto pudimos concretar que algunas que tienen varios años dedicadas al desarrollo de software no tienen la madurez necesaria en sus procesos durante la elaboración de un proyecto. En la figura 19 se muestran los rangos de edades obtenidas, así como los porcentajes.

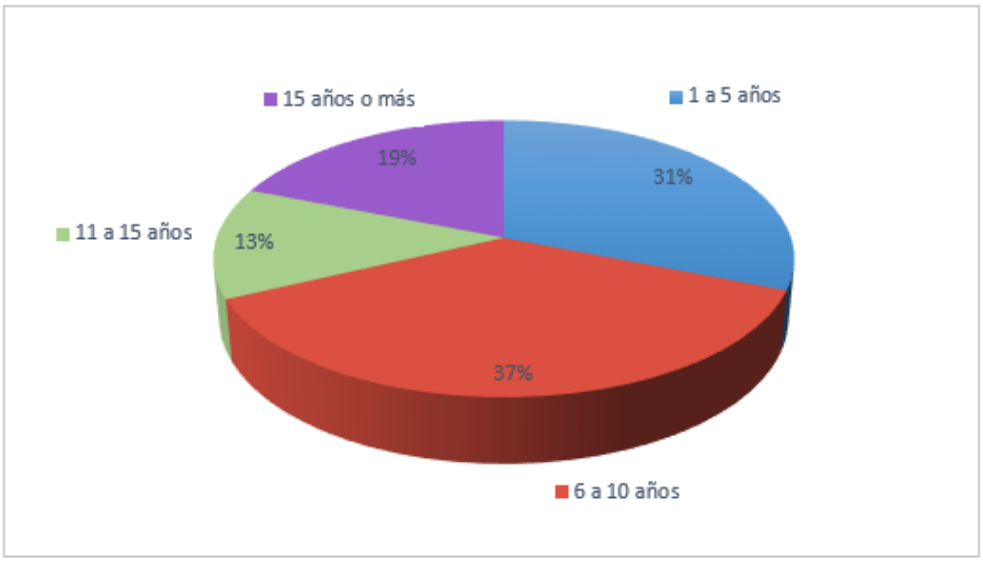


Figura 19. Tiempo de las empresas prestando el servicio.

De las fábricas seleccionadas tenemos que el 37% de las mismas se encuentran entre los 6 a 10 años de actividad y que con el 13% están las de 11 a 15 años, por lo que podemos mencionar que la mayoría son empresas jóvenes de entre 1 a 10 años y que pasado ese lapso de tiempo si no cuentan con una metodología que se adapte al tipo de proyectos que desarrollan pueden desaparecer en los próximos 5 o 10 años.

La RE en el desarrollo de software va en ascenso, por lo cual es necesario que las fábricas se relacionen con el término de requisito y le den a esta etapa del proceso el enfoque y el valor que merece, para evitar costos innecesarios en las etapas posteriores del proyecto.

En la figura 20 podemos observar que el 80% de las empresas encuestadas argumento que sí están familiarizados con la RE, mientras que solo el 20% respondió que no la utilizan.

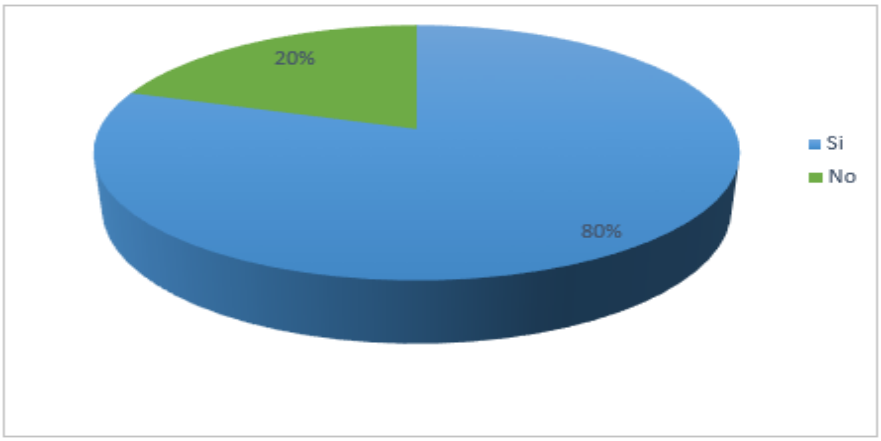


Figura 20. La Ingeniería de Requisitos en las empresas.

En todo desarrollo es de vital importancia saber qué es lo que se quiere construir y que limitaciones debe tener, por lo cual es necesario conocer cuáles son las necesidades que se van a considerar para la elaboración del sistema. En la figura 21 se muestran los conceptos de requisito o requerimiento que son utilizados por las fábricas.

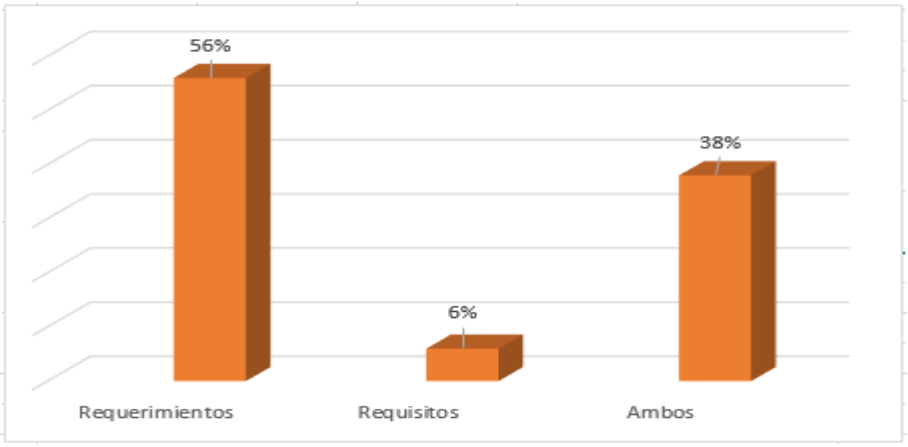


Figura 21. Concepto de requerimiento o requisito.

.En esta gráfica observamos que aún existe la incomprensión por la palabra requerimiento, algunas empresas, el 38% utilizan ambos conceptos, el 56 % emplea solo requerimiento, mientras que solo el 6% usa el término requisito, lo correcto es utilizar solo requisito, debido a que la definición de este es una circunstancia o condición importante para algo, y requerimiento es la acción y efecto de requerir, ocupar, es un sinónimo de necesidad, por lo que va dirigido hacia la carencia o falta de algo.

Como ya lo hemos mencionado anteriormente la relevancia que tiene la RE en el desarrollo de software, es alta, debido a que si no le damos la importancia que tiene se pueden ocasionar problemas en las siguientes etapas del proceso, las cuales den como resultado productos ineficientes y costos elevados. En la figura 22 se muestra que para el 50% de las fábricas entrevistadas es importante, seguida del 31% de las mismas para las que es lo más importante y solo el 6% le da una mínima importancia.

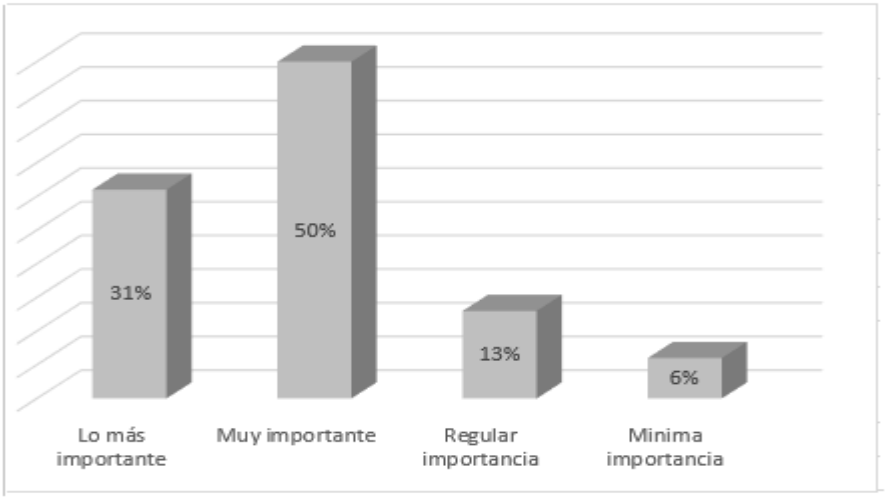


Figura 22. Importancia de la RE en las empresas.

Dentro de la etapa de la RE encontramos algunas actividades que ayudan durante todo el proceso de desarrollo, las cuales son: elicitación, análisis, especificación, validación y administración, por lo cual es necesario conocer el periodo que se le dedica a estas durante la elaboración de un proyecto. En la figura 23 observamos el tiempo (en horas) que se le da a las actividades.

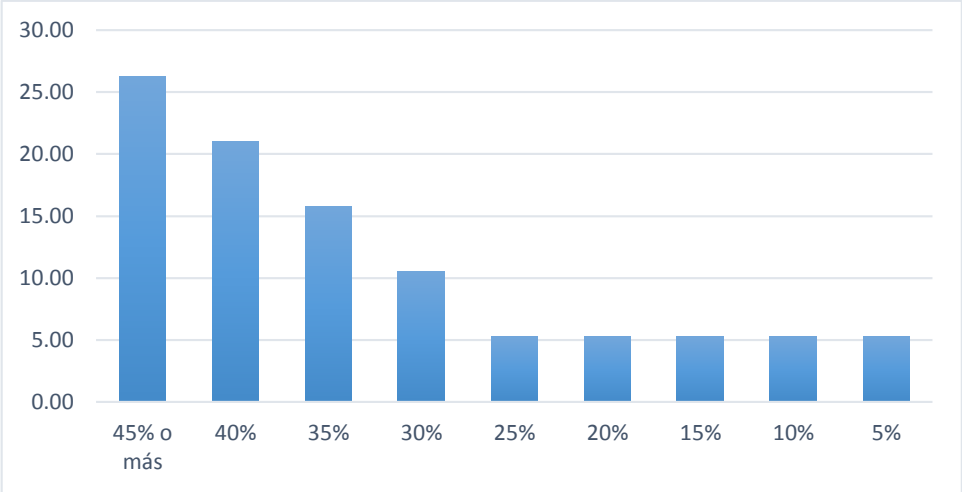


Figura 23. Tiempo (horas) dedicado a las actividades de la RE.

El gráfico anterior muestra que una gran cantidad de tiempo se dedica a las actividades relacionadas con la RE cuando se trabaja en un sistema, el 45% dedican 25 horas y solo el 5% 5 horas durante el proyecto.

En el proceso de desarrollo de software es indispensable poder contar con métodos que nos faciliten el proceso para especificar los requisitos, por lo que en la figura 24 se muestran los más utilizados.

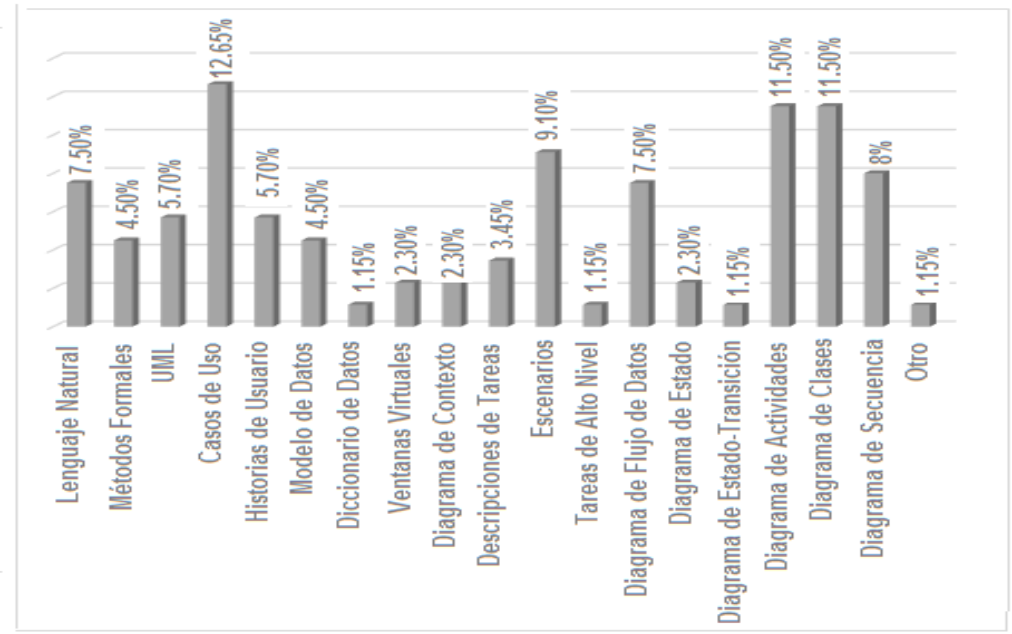


Figura 24. Métodos de especificación de requisitos.

Entre las técnicas para la especificación, concretamente de modelado, encontramos que la mayoría de las empresas, esto es el 12.65% de ellas utilizan casos de usuario y en segundo lugar los diagramas de clases y con un 7.50% el lenguaje natural, lo cual demuestra que el lenguaje natural sigue siendo uno de los factores claves durante las etapas iniciales del proceso de desarrollo.

En la figura 25 se muestran las técnicas de especificación, de las cuales tenemos que el 26.38% utiliza el cuestionario, seguido de la lluvia de ideas con un 20.83% y con un 2.70% la observación, la ventaja del uso de técnicas es que se pueden emplear más de una con lo cual se puede garantizar una mejor especificación de los requisitos que el sistema contendrá.

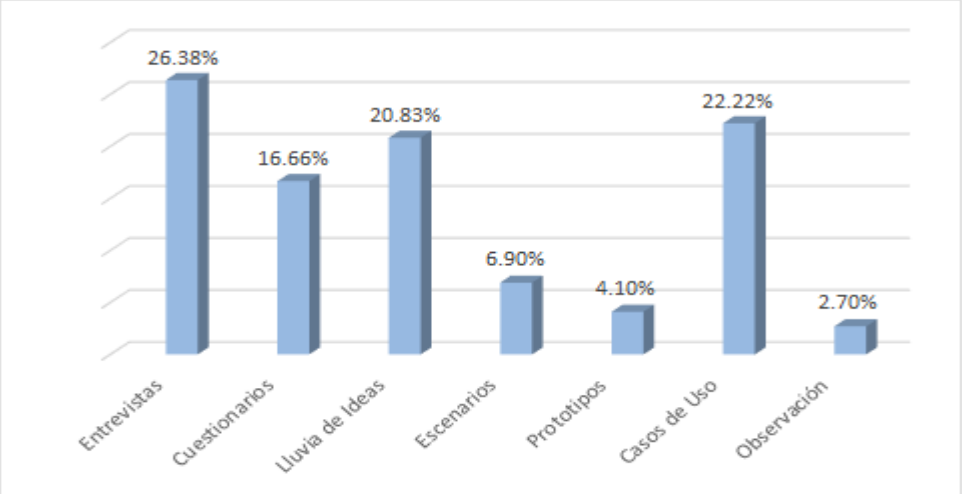


Figura 25. Técnicas de especificación de requisitos.

Una vez definidos los métodos y las técnicas, lo siguiente es ver las herramientas que son utilizadas por las fábricas con más frecuencia al elaborar un proyecto. En la figura 26 podemos ver que de las opciones mostradas *Rational RequisitePro, Tracecloud, Visual Paradigm Requirements Capturing, Objetif Requirements Modeller y Testrack* con 1% fue de las más conocidas por las empresas, sin embargo, la que más se utiliza con un 9% definida en la opción de otros, es una herramienta comercial llamada *Enterprise Arquitech* la cual se ajusta a las necesidades de cada una.

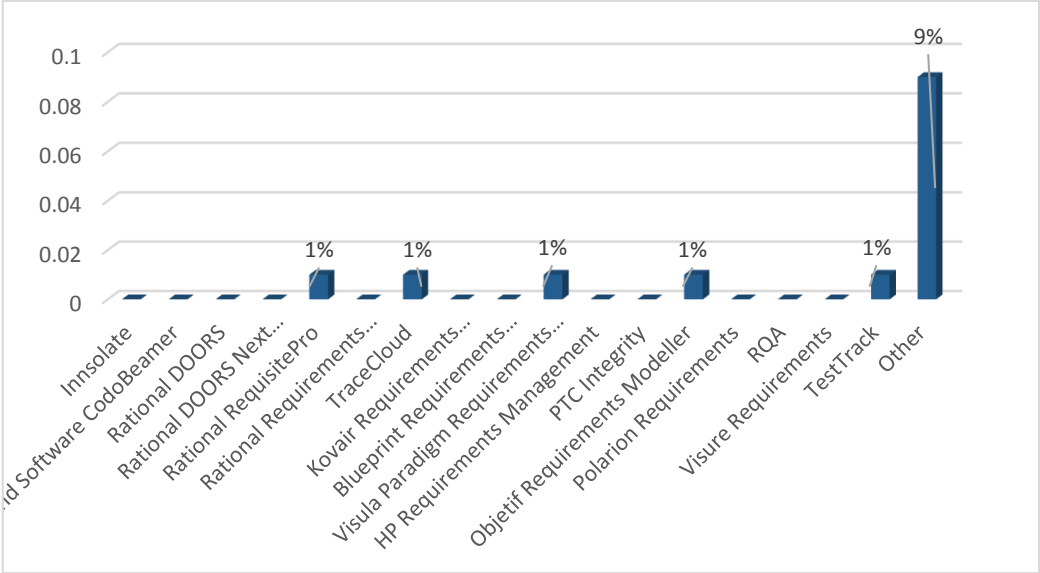


Figura 26. Herramientas de especificación de requisitos.

Durante el desarrollo de la tesis, en el capítulo 2 explicamos la importancia de hacer una buena especificación de los requisitos, las ventajas y desventajas que tenemos de no hacer un buen proceso, ahora es necesario conocer dentro de las empresas a las personas que se encargan de realizar esta tarea durante el proyecto. En la figura 27 se muestran los individuos que especifican la mayoría de estos.

En la siguiente gráfica podemos observamos que el 50% se encuentra entre el Líder del proyecto y el analista, mientras que con un 6.3% tenemos a los desarrolladores y tester.

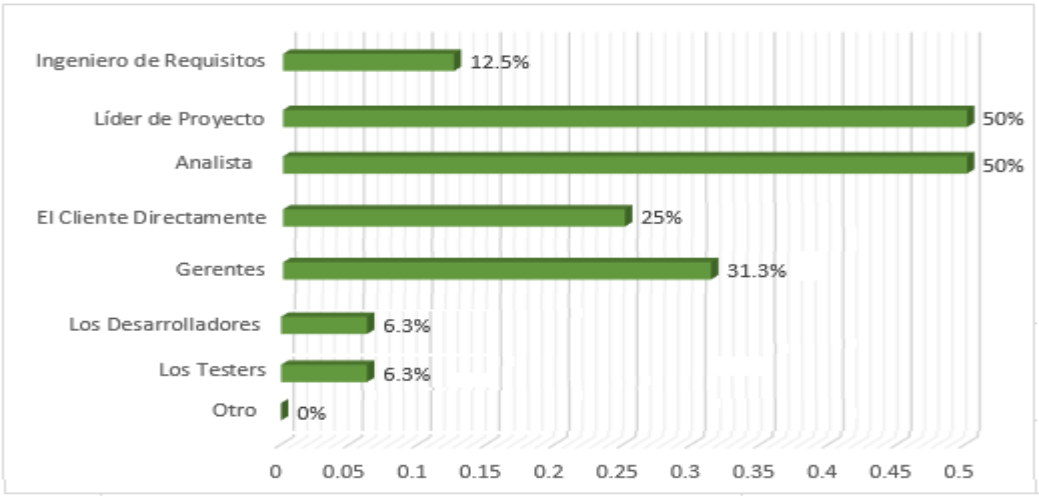


Figura 27. Personas que especifican los requisitos.

Capítulo V. Aproximación Metodológica

En este capítulo se muestra el desarrollo de una aproximación metodológica para la práctica de la Ingeniería de Requisitos (RE) para las PyMES desarrolladoras de software del estado de Sinaloa. En la cual se describe cada una de las fases contenidas en el procedimiento, así como los documentos para la especificación de los requisitos.

Un hecho claro es, sin embargo, que no existe una única metodología que sea adecuada para cualquier proyecto. Es por esto que resulta muy importante conocer las diversas metodologías que puedan ser aplicables a los proyectos, así como manejar las herramientas que permitirán su selección, adaptación y su formulación.

La tendencia dominante en la actualidad para el desarrollo de proyectos de software es reconocer que los mismos requieren ser tratados como un proceso, con un conjunto de prácticas generalmente aceptadas como favorables para aumentar la calidad y repetibilidad de los productos, tales como el desarrollo iterativo e incremental, el manejo consistente de los requisitos y el modelado visual, entre otras. Se ha acuñado incluso el término Ingeniería de Software para poner de manifiesto que se desea lograr la formalidad y precisión propias de las disciplinas tradicionales de ingeniería al desarrollar aplicaciones informáticas [35].

En el capítulo 3 de esta investigación se mostraron algunos trabajos de metodologías realizadas enfocadas a la RE en los cuales se muestra que el uso de estas ha sido de gran ayuda para poder generar sistemas no erróneos y de buena calidad, la diferencia con la que se presenta en este trabajo es que va enfocada a las PyMES dedicadas al desarrollo de software.

La aproximación metodológica que proponemos la cual hemos llamado Documentación para la Especificación de Requisitos en el Desarrollo de Sistemas de Software (DERDSS) cuenta con 3 fases y dos formatos los cuales se llenarán durante la etapa de elicitación. La propuesta se enfoca al desarrollo de proyectos nuevo o aquellos que se vayan a modificar.

Durante el avance de este trabajo hemos explicado la importancia de la RE en la elaboración de proyectos, y también de las actividades que se llevan a cabo durante las fases del desarrollo, las cuales son elicitación, análisis,

especificación, validación y administración. Para el desarrollo de esta aproximación metodológica se utilizaron solo las tres primeras fases: elicitación, análisis y especificación, debido a que el objetivo es descubrir que problema debe ser resuelto, conocer a las partes interesadas y cuáles son los objetivos que el sistema debe alcanzar, el análisis durante la que se trata de comprender a la organización como tal, sus reglas de negocio, sus metas, tareas y todos los datos que se necesiten y por último la especificación que es una descripción integral del comportamiento del sistema a desarrollar. Las actividades de validación y administración no se utilizaron para el desarrollo de esta aproximación debido a que van enfocadas a la representación de las necesidades mediante prototipos, verificación de cambios y control en las versiones, y para esta propuesta el objetivo es poder determinar los requisitos.

El autor [80] en su trabajo de investigación menciona que la Ingeniería de Requisitos, sugiere la existencia de un eje troncal de etapas, dejando abierta la posibilidad de que cada uno de los estudiosos del tema las refine cuanto sea necesario. Por tanto, si bien existen diferentes enfoques, éstos tienen un común denominador, que puede resumirse en las siguientes etapas fundamentales: Elicitación, Análisis y Especificación, que son las que se adoptan en su investigación. La gran mayoría de empresas asegura que emplean metodologías de desarrollo, pero realmente no realizan estas actividades de la forma correcta, o en otros casos no las realizan, lo que ocasiona problemáticas en cuanto a criterios para la aceptación de proveedores de necesidades, criterios para la aceptación y ausencia de administración de la trazabilidad y de los cambios de los requisitos [76]. Para ofrecer una alternativa viable de mejoramiento de procesos se definió y documentó una metodología ágil de RE para las empresas emergentes la cual comprende tres fases: elicitación, especificación y gestión de requisitos.

5.1. Documentación para la Especificación de Requisitos en el Desarrollo de Sistemas de Software (DERDSS)

La aproximación metodológica DERDSS será utilizada por los analistas o líderes de proyectos durante la etapa de especificación en el desarrollo de un proyecto. DERDSS cuenta con 3 fases, las cuales son: elicitación, análisis y especificación de requisitos. Los objetivos de cada una de ellas son los siguientes:

- **Elicitación de requisitos:** esta es la etapa en donde se obtiene el conocimiento del problema que el cliente nos presenta como una necesidad mediante el Documento de Requisitos de Usuario (DRU), así como también conocer a las partes interesadas, el resultado de esta fase es tener claro cuál es el problema a resolver mediante el desarrollo de un sistema.
- **Análisis de requisitos:** durante esta etapa se conocen las reglas de negocio de la empresa, las metas y la utilización de ciertas técnicas para obtener datos necesarios que nos ayuden a conocer mejor el problema iniciar con la recolección de información de lo que el sistema deberá realizar y lo que no, los cuales en la siguiente etapa se definirán como las especificaciones del sistema.
- **Especificación de requisitos:** en esta etapa se continúa con el empleo de técnicas que nos ayuden a definir los requisitos funcionales y no funcionales, objetivos y restricciones que deberá tener para lo que se apoyará en las etapas anteriores las veces que se requiera, los cuales dan como resultado el Documento de Especificación de Requisitos de Software (DERS), este archivo contendrá todas las especificaciones del sistema a desarrollar y será de gran ayuda para las etapas posteriores del proceso.

En la figura 28 se muestra la representación gráfica de DERDSS.

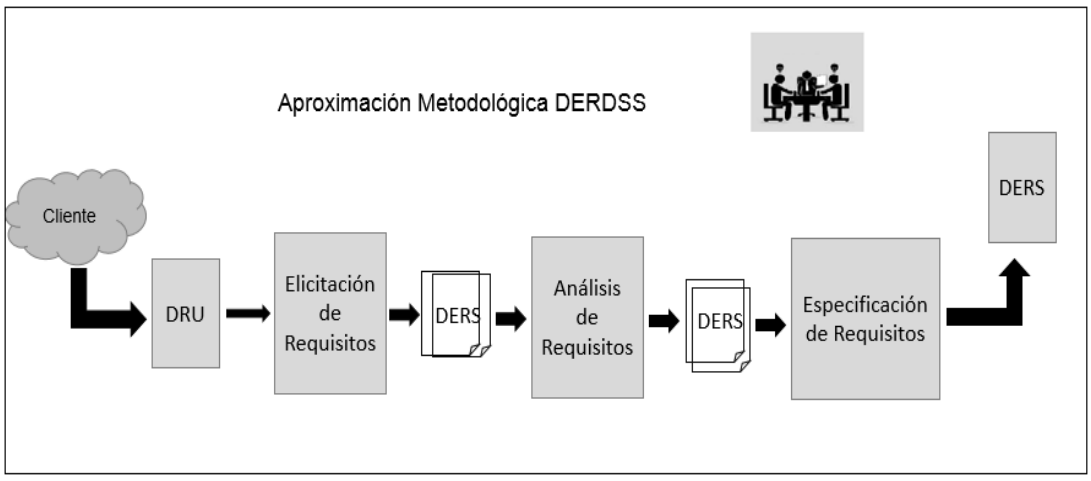


Figura 28. Aproximación Metodológica DERDSS.

La figura anterior nos muestra de forma general las fases que comprende DERDSS, esto es solo para darnos una visión general de como es el proceso. El proceso inicia cuando el cliente solicita el desarrollo de un sistema, esto mediante la solicitud en el DRU, posteriormente el analista toma el documento y empieza a aplicar las etapas de la metodología DERDSS para poder generar el DERS el cual va a contener las especificaciones de su sistema, las cuales se tomarán para el desarrollo de las siguientes fases. A continuación se explica detalladamente en que consiste cada una de las etapas de esta aproximación metodológica.

5.1. Fase 1: Elicitación de Requisitos

La Fase 1, llamada Elicitación de Requisitos, comprende el conocimiento del problema a resolver. Esta se lleva a cabo utilizando 2 técnicas observación y cuestionario, se considera primordial para establecer el punto de partida sobre el cuál se llevará a cabo el proceso de desarrollo del software a construir. Para realizarla, se necesitan tres actividades, estas son i) Llenado del DRU, ii) Revisión de la Solicitud (DRU) y iii) Conocer el problema a Resolver. A continuación se explica en que consiste cada una de ellas.

1.1 Llenado del DRU. Todo desarrollo de un sistema inicia con una necesidad, durante este punto de la primer fase de la aproximación el cliente debe realizar la captura del DRU. En este documento se especificarán las necesidades que el usuario solicite, los cuales se podrán escribir desde el punto de vista del cliente o de la persona interesada. En el DRU se redacta el problema que se tenga de forma que no expresen mayor nivel de detalle, se pone en pocas palabras o de forma muy general cual es la problemática actual, el objetivo que debe cumplir el nuevo sistema para dejar de ser insatisfactorio.

Documento de Requisitos de Usuario (DRU)

Tipo de Proyecto	Fecha	Folio	Versión
☐ Nuevo ☐ Modificación	dd/mm/aaaa		

Nombre del cliente: se va a capturar el nombre de la persona que está realizando la solicitud.
Nombre de la empresa: Se captura el nombre de la empresa a la cual se le va a desarrollar el proyecto.

Descripción general de la solicitud: En esta parte del documento el usuario o cliente podrá plantear que es lo que necesita que el sistema realice, de forma muy general y sencilla sin entrar en mayor detalle (300 caracteres).
Problemática actual: en esta punto el cliente debe describir cual es la problemática actual que se tiene por lo cual el sistema se ha vuelto insatisfactorio, deberá ser muy explícito al momento de indicar la situación que se tiene para que se pueda obtener toda la información que el usuario nos quiera proporcionar (300 caracteres).
Alcance del sistema a desarrollar: se debe de poner el objetivo de la solicitud, que es lo que se quiere lograr con el desarrollo de este sistema.

Figura 29. Documento de Requisitos de Usuario (DRU).

En la figura 29 se observa el DRU que es el documento en donde el cliente nos captura la solicitud del proyecto a desarrollar, el usuario va a captura de forma general las necesidades y la problemática que tiene, mediante este documento y el uso de las fases contenidas en la aproximación metodológica que presentamos se obtendrá el DERS en el cual se van a especificar los requisitos y limitaciones que el sistema a desarrollar contendrá. En el formato de este documento se muestran dos opciones de solicitudes nueva o modificación, si la petición a desarrollar es nueva en el campo folio se va a capturar el siguiente número y como versión se pondrá 1, de lo contrario si es modificación se tiene que capturar el folio de la solicitud que ya fue desarrollada, debido a que será sobre esta que se va a trabajar y como versión se pondrá el siguiente, es decir, si ya hay una versión se va a capturar el número 2.

1.2 Revisión de la solicitud (DRU). Para realizar esta actividad es indispensable leer la solicitud de ser posible dos o más veces, esto para tener muy claro cómo funciona el sistema, cual es el problema que se tiene, que es lo que quiere el cliente y el objetivo que quiere lograr, una vez que el analista tiene claro esto no lo debe de perder de vista ni desviarse.

1.3 Conocer el problema a resolver. En esta etapa se empiezan a emplear las técnicas de especificación que no ayuden a poder comprender de forma más clara y sencilla el problema, la primera técnica a utilizar es la observación en la cual el analista deberá identificar cómo opera el sistema, sin necesidad de hacer preguntas todavía, solo tomando nota. En este caso se deberá de hablar con las personas para convencerlas de que no se les está evaluando la forma en la que manejan el sistema, si no que se quiere ver el

problema que nos plantean en la solicitud, de forma contraría lo harán que trabaje de manera normal y esto nos ayudará a poder proponer la mejor solución. En este punto el analista debe de tener en cuenta que las preguntas que realice deben ir enfocadas a entender el problema que el cliente nos planteó en un inicio. Continuando con el uso de técnicas aquí sería necesario aplicar la técnica del cuestionario aplicando las preguntas (las cuales no deben exceder a 5) que el analista en el proceso de observación escribió, proporcionarlas al usuario y darle un tiempo considerable para que las responda, estas respuestas nos van a ayudar a poder formular la entrevista al cliente tomando solo la información que se necesita.

En la figura 30 se muestra el diagrama en donde se representas los puntos comprendidos de la fase de elicitacion de requisitos.

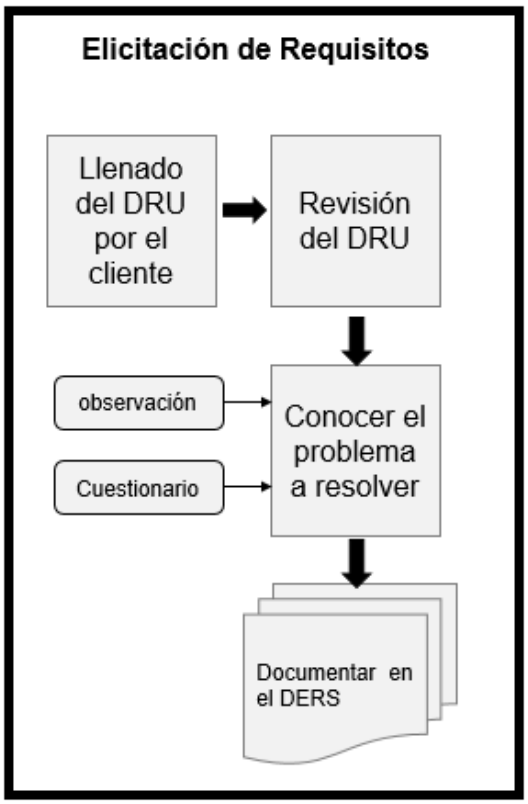


Figura 30. Diagrama de la fase de Elicitación de Requisitos.

5.2. Fase 2: Análisis de Requisitos

En esta fase de la DERRSS se llevan a cabo las actividades necesarias para analizar los requisitos recolectados. Para ello, se recomienda utilizar técnicas como: lluvia de ideas, entrevistas y cuestionarios. La Fase 2 comprende las siguientes actividades:

2.1 Reunión de equipo. En esta actividad el analista se reúne con el equipo que va a desarrollar el proyecto, en esta junta se le mostrará al equipo la solicitud del cliente, los puntos tomados en cuenta durante la observación y las respuestas del cliente sobre el cuestionario que se le proporcionó Esto con el fin de poder obtener dudas y puntos de vista que cada integrante tiene acerca del sistema, para esto utilizaremos también la lluvia de ideas, el analista debe tener claro el objetivo y no permitir que nadie se desvíe del tema, se deben anotar todas las dudas y preguntas que se plantearon para poder generar una entrevista con el cliente.

2.2 Elaboración de entrevista. Una vez que el analista terminó con la reunión del equipo, procede a generar la entrevista que va a tener con el cliente para poder tener más información y responder a las dudas de su equipo, es importante que las preguntas se hagan con tiempo y se lleven escritas (se aclara que ante alguna respuesta puede surgir una pregunta que no se tiene contemplada en la entrevista y se debe anotar), es importante entrevistar a la persona correcta, es decir, que sepa el funcionamiento del sistema y que nos pueda dar respuestas que sirvan para la formulación de los requisitos, de otra forma no obtendremos la información necesaria y nos retrasa en los tiempos hablar con personas no consideradas correctas para la entrevista. Para que este paso se realice con éxito se le debe de dar la confianza y seguridad al cliente de que él es el único que sabe cómo funciona el sistema y el problema que tiene, no poner cosas que no dijo o asumir que quedamos en el mismo entendido, siempre es importante corroborar lo que transmitió, tratar de que sea una plática amena y utilizar los términos adecuados (esto es no utilizar términos muy técnicos si el cliente no los domina y tratar de explicar de forma clara para obtener la respuesta que buscamos), cuidar de no tomar demasiado tiempo para que no sea algo tedioso para él. Al finalizar la entrevista el analista debe comentar al cliente cual será el siguiente paso y hacerle saber que puede haber otra reunión de dudas (en el caso que puedan surgir) con él antes de ver el

contrato. Al momento de aplicar esta técnica debemos tener en cuenta los siguientes puntos:

- Tener presente que información se necesita.
- Formular las preguntas adecuadas para obtener la información necesaria.
- Escuchar con atención y facilitar que el entrevistado sepa que está siendo escuchado.
- Resumir el contenido y significado de las respuestas. Cuando esto se hace periódicamente durante la entrevista resulta de gran ayuda para el entrevistador. El entrevistado puede añadir más detalles y el entrevistador puede detectar errores de interpretación. A la hora de realizar una entrevista debemos de partir de las preguntas más sencillas a las más complicadas. Tratar con respeto a la persona entrevistada.

En la figura 31 se observa el diagrama de la fase de análisis de requisitos.

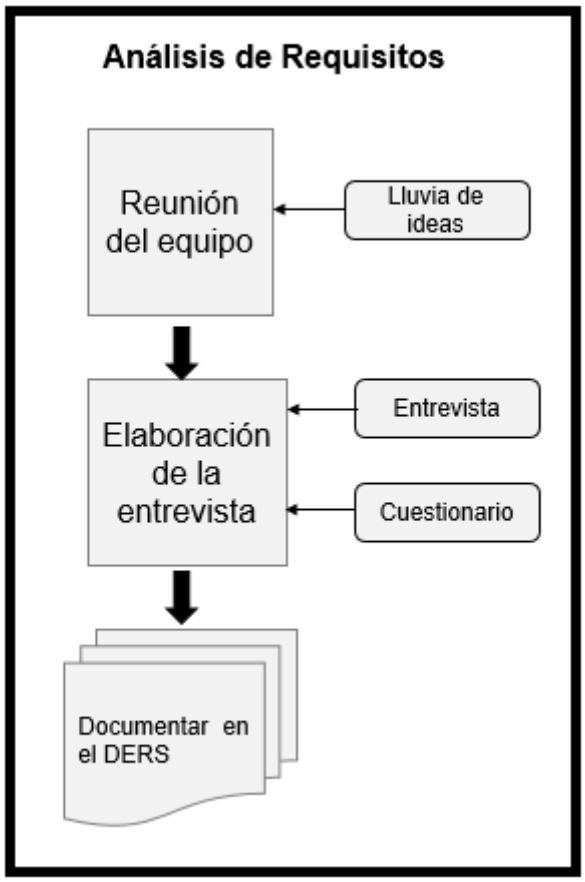


Figura 31. Diagrama de la fase de Análisis de Requisitos.

5.3. Fase 3: Especificación de Requisitos

En la Fase 3, denominada Especificación de Requisitos se llevan a cabo las actividades necesarias para delimitar lo que el sistema deberá realizar y se obtiene como producto el DERS. Para realizarse, es necesario que el analista realice las siguientes tareas:

3.1 Determinar los requisitos: En este paso el analista con la información que tiene de la entrevista especifica los requisitos funcionales y no funcionales que va a tener el sistema, así como también el desarrollo de diagramas de flujo (esta información es capturada en el DERS), es importante obtener todas las especificaciones posibles, porque con esto evitaremos errores o cambios una vez aprobado el documento. Dado a que estos expresan el propósito del sistema sin considerar como se va a llevar a cabo, especifica lo que el sistema debe hacer (sus funciones) propiedades esenciales y deseables, identifican el "qué" y no el "cómo".

Cómo deben ser redactados:

1. Deben ser redactados en tercera persona (Ejemplo: El sistema)
2. Deben estar redactados en tiempo presente, deben ser claros, afirmativos y no ambiguos.
3. Deben ser redactados partiendo de lo que el sistema debe de hacer (Ejemplo: El sistema debe…)
4. No se deben mencionar tecnicismos: tablas, campos, tipos de datos, lenguaje de programación, etc.
5. No deben estar redactados como un flujo de un caso de uso.
6. Cada uno debe expresar solo una funcionalidad.
7. Deberán ser unívocamente identificable mediante algún código o sistema de numeración adecuado.

Se debe de tomar en cuenta que los requisitos aquí especificados van a ser el resultado de lo que el cliente nos pide en la solicitud, incluyendo también el documento (DRU), es decir, lo que el cliente pide convertirlo en condición, para que a su vez se convierta en funcionalidad.

Los requisitos deben de poseer las siguientes características:

- **Necesario:** es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o

factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.

- **Conciso:** es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- **Completo:** está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
- **Consistente:** es consistente si no es contradictorio con otro requerimiento.
- **No ambiguo:** no es ambiguo cuando tiene una sola interpretación.
- **Verificable:** es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección análisis, demostración o pruebas.
- **Trazables:** La ERS es trazable si se conoce el origen de cada requisito y se facilita la referencia década uno a los componentes del diseño y de la implementación. Para poder especificar de forma correcta los clasificamos en funcionales y no funcionales:

  Funcionales: son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. Describen lo que el sistema debe hacer. Es importante que se describa el ¿Qué? y no el ¿Cómo? se deben hacer esas transformaciones.

  - Cómo identificar los requisitos Funcionales:
  - Deben de responder a las preguntas:
  - ¿Qué hará el sistema?
  - ¿Cuándo lo hará?

  No Funcionales: tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

3.2 Reunión de equipo. Una vez hecha la especificación se debe reunir con el equipo de trabajo para mostrar los requisitos del sistema, una vez determinadas se platica como va a ser el desarrollo de las funciones que se

realizarán para el cumplimiento, así como también un escenario de que hará el sistema. Una vez terminada la reunión el analista debe pasar toda la información al formato DERS, el cual contiene las especificaciones del sistema.

3.3 Reunión de revisión de DERS. Una vez realizados los pasos anteriores se realiza de nuevo una reunión de todo el equipo y el cliente en donde se revisa paso a paso el documento de especificación y se le muestra al usuario mediante un escenario como será el funcionamiento que el sistema va a tener, de acuerdo a las especificaciones que se realizaron, ya revisado esto se firma y acepta para pasar a la siguiente fase del proyecto.

En la figura 32 se muestra el diagrama de la fase de especificación de requisitos.

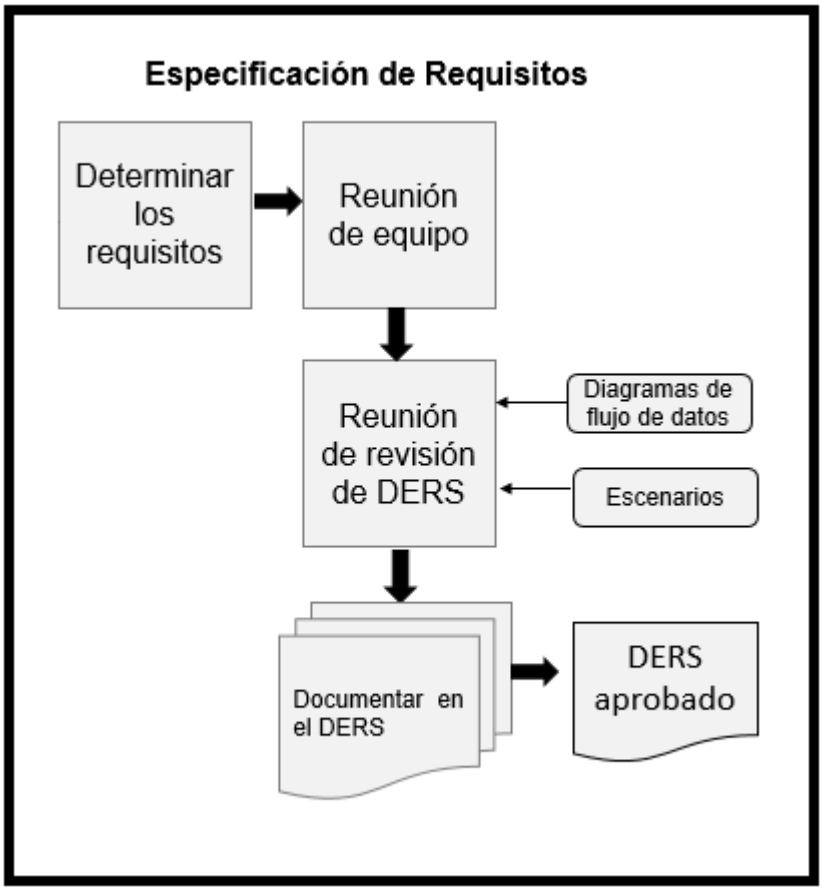


Figura 32. Diagrama de la fase de Especificación de Requisitos.

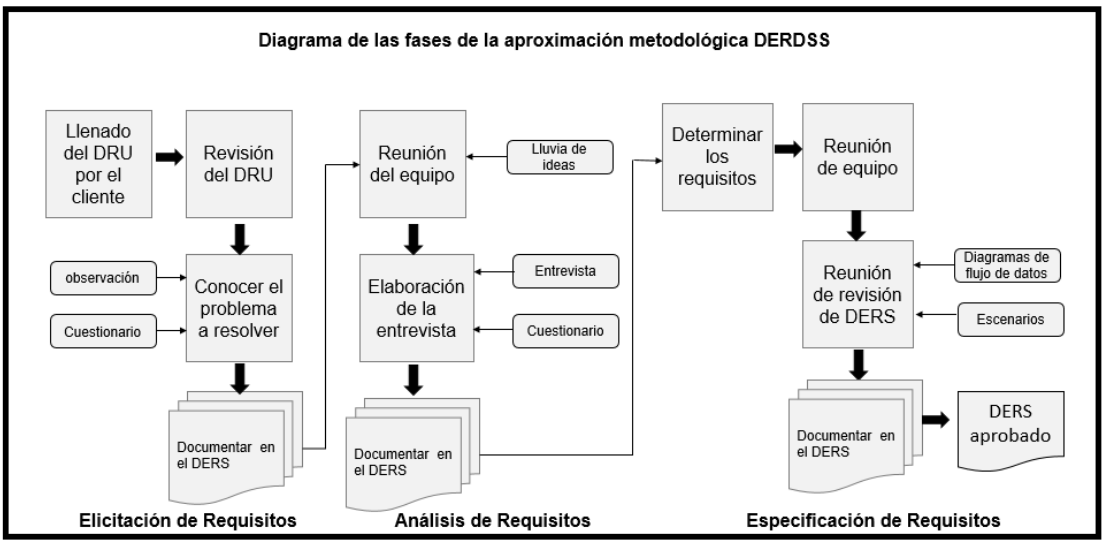


Figura 33. Diagrama de la Aproximación Metodológica DERDSS.

En la figura 33 se muestra el diagrama con las fases que cuenta la aproximación Metodológica DERDSS, como se observa la primera etapa es para conocer el problema a fondo, para lo cual se hace uso del DRU con la información que el cliente proporciono, así como también de las técnicas de observación y cuestionario para obtener todos los datos necesarios, los cuales se deberán capturar en el DERS. La segunda fase de Análisis de Requisitos, en la cual ya se tiene identificado el problema, se conoce al equipo encargado del desarrollo del sistema, en esta etapa se inicia con las reuniones en donde salen las dudas que se tienen sobre el proyecto, para esto se hace uso de la lluvia de ideas, así mismo se llevan a cabo las reuniones con el usuario en donde se aclaran las ideas y las dudas planteadas por el equipo de desarrollo para lo cual utilizamos de la entrevista, posterior a esta el analista comienza a identificas los requisitos que el sistema contendrá, durante esta etapa se documenta en el DERS todas las reuniones que se han hecho, así como la información obtenida de las técnicas aplicadas. Y como última fase tenemos la Especificación durante la cual el analista especifica los requisitos funcionales y no funcionales que el sistema debe tener, así como los diagramas de flujo, los cuales son presentados al equipo para su aprobación o rechazo en caso de que algunos no estén claros, una vez que sean aceptados se actualiza el DERS con toda la información del proyecto y se muestra al cliente, esto para confirmar que todo lo especificado en el documento será lo que el sistema realice.

5.4. Documento de Especificación DERS

Una vez iniciado el proyecto cada una de las etapas antes mencionadas se va a ir documentando en el DERS esto con la finalidad de tener la especificación del sistema y poder llevar un orden y seguimiento del proyecto. En la siguiente imagen se puede ver la estructura, así como cada una de las opciones contenidas en él.

<table>
<tr><th colspan="4">Documento de Especificación de Requisitos de Software (DERS)</th></tr>
<tr><td>Tipo de Proyecto
☐ Nuevo
☐ Modificación</td><td>Fecha
dd/mm/aaaa</td><td>Folio</td><td>Versión</td></tr>
<tr><td colspan="4">Nombre del cliente: se va a capturar el nombre de la persona que está realizando la solicitud.
Nombre de la empresa: Se captura el nombre de la empresa a la cual se le va a desarrollar el proyecto.</td></tr>
<tr><td colspan="4">Descripción general de la solicitud: En esta parte del documento el usuario o cliente podrá plantear que es lo que necesita que el sistema realice, de forma muy general y sencilla sin entrar en mayor detalle (300 caracteres).</td></tr>
<tr><td colspan="4">Problemática actual: en esta punto el cliente debe describir cual es la problemática actual que se tiene por lo cual el sistema se ha vuelto insatisfactorio, deberá ser muy explícito al momento de indicar la situación que se tiene para que se pueda obtener toda la información que el usuario nos quiera proporcionar (300 caracteres).</td></tr>
<tr><td colspan="4">Alcance del sistema a desarrollar: se debe de poner el objetivo de la solicitud, que es lo que se quiere lograr con el desarrollo de este sistema.</td></tr>
<tr><td colspan="4">Nombre del proyecto: se pone el nombre que se le va a asignar al proyecto.</td></tr>
<tr><td colspan="4">Fecha inicio: dd/mm/aaaa Fecha fin: dd/mm/aaaa</td></tr>
<tr><td colspan="4">Datos del equipo de desarrollo: se capturan los nombres de los integrantes del equipo.
Analista: es la persona encargada de especificar los requisitos que el sistema debe de cumplir.
Líder de proyecto: Se encarga de dar seguimiento a los avances, así como a las fechas determinadas para cada actividad.
Arquitecto: es la persona encargada de diseñar la arquitectura del proceso (especifica las actividades que los desarrolladores van a realizar).
Desarrolladores: son los que desarrollan el código en el cual se va a programar el sistema.
Tester: se encarga de realizar las pruebas al sistema.
Auditor de procesos: se encarga de que se lleve a cabo la metodología.</td></tr>
<tr><td colspan="4">Reglas de negocio: en esta parte se pondrán las reglas de negocio de la empresa.</td></tr>
</table>

Requisitos: expresa el propósito del sistema sin considerar como se va a llevar a cabo, especifica lo que el sistema debe hacer. Para facilitar el proceso se clasifican en Funcionales y No funcionales.

Funcionales: Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. Describen lo que el sistema debe hacer. Es importante que se describa el ¿Qué? y no el ¿Cómo? se deben hacer esas transformaciones.

No Funcionales: Tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

Apéndice: en esta parte se especifican las técnicas y modelos empleados durante la aproximación metodológica.

Observación:

El analista deberá observar cómo opera el sistema, sin necesidad de hacer preguntas todavía, solo tomando nota. Esto para asegurarse de la problemática que existe. En esta parte el analista debe describir de forma muy detallada todo lo que vea y con lo que tenga dudas, esto le ayudará a la formulación de la entrevista con el cliente. No hay límite de escritura, y deberá captura la fecha y hora de inicio y fin de esta actividad. Una vez terminado este proceso se debe de actualizar el campo de problemática actual del sistema de acuerdo a lo observado.

Cuestionario:

De acuerdo a las notas que el analista realizó en el paso anterior deberá realizar un pequeño cuestionario de no más de 5 preguntas (esto es opcional si el analista lo cree necesario) y dejárselo al usuario para que lo responda sin presiones, en caso de haber preguntas deben de capturarse aquí junto con su respectiva respuesta.

Reuniones de Trabajo:

En esta parte se deben de especificar todas las reuniones que se hayan tenido durante el proceso de elicitación, y se deben de capturar los siguientes datos:

Fecha: dd-mm-aaaa (día, mes, año) Hora inicio: Hora fin:

EP= Equipo de proyectos

C= Cliente

Reunión con: (Tendrá que poner la opción descrita arriba para ver si es el cliente o equipo de proyecto)

Tema a tratar: Especificar de que se trató la reunión.

Puntos relevantes de la minuta: Aquí se tiene que especificar los acuerdos a los que hayan llegado, o los pendientes que se vayan a revisar.

Entrevista: en la entrevista con el cliente, el analista deberá de capturar las preguntas hechas al cliente con sus respectivas respuestas, así como datos adicionales que el cliente comento y que no vienen en la petición. Fecha: dd-mm-aaaa (día, mes, año) Fecha inicio: Fecha Fin: Nombre del cliente: **Preguntas:** En esta parte se incluirán las preguntas y posteriormente las respuestas a estas mismas.
Diagrama de Flujo de Datos / Escenarios: En esta parte del documento se tendrá que mostrar el diagrama de flujo de datos, el cual se presentará al cliente, de hacer alguna modificación en el mismo se deberá actualizar.

Figura 34. Documento de Especificación de Requisitos de Software (DERS).

Un ejemplo del llenado del DERS se encuentra en el Anexo C de la tesis. Finalmente, una vez que se han realizado las tres fases que comprende la aproximación metodológica que proponemos, llamada Documentación para la Especificación de Requisitos en el Desarrollo de Sistemas de Software (DERDSS), se obtiene como producto final un documento llamado DERS con el listado de requisitos funcionales y no funcionales obtenidos del cliente. Con esto, es posible comenzar el proceso de desarrollo de un software de una forma ágil y con menor esfuerzo para los integrantes del equipo de desarrollo.

5.5. Comentarios del capítulo

Durante el proceso de investigación en las empresas sobre la metodología que emplean en la etapa de elicitación de requisitos para el desarrollo de sus sistemas, se pudo apreciar que en la mayoría utilizan métodos tradicionales y aquellas que utilizan metodologías ágiles no la usan al 100% debido a que hay etapas incluidas en el proceso que son ignoradas en ocasiones por desconocimiento y en otras tantas por los tipos de proyectos que realizan. Dentro de esta tarea realizada se encontraron algunas fallas que se pueden considerar problemas a la hora de hacer un desarrollo:

- Uno de los problemas comunes con los que conviven a menudo es la falta de integración del cliente en el desarrollo, es decir, es importante que se le involucre durante la etapa de elicitación, debido a que él es la persona que más conoce sobre la petición que está solicitando y puede ayudar mucho en las especificaciones.

- Es importante que se tome cada proyecto como algo nuevo, debido a que algunas empresas toman otro desarrollo que se haya hecho de un sistema igual al que se pide y se dan por hecho cosas que no venían especificadas.
- En la utilización de las técnicas es importante tomar en cuenta el uso de más de alguna, porque se observó que en algunos casos la mayoría toma solo la entrevista y hay cuestiones que no se pueden determinar del todo solo con preguntar al cliente, en este mismo punto se debe considerar a la persona que se va a entrevistar porque del conocimiento que ella tenga depende que se puedan obtener todos los requisitos que necesita el sistema.
- En los equipos de proyectos más de un integrante en ocasiones realiza más de un rol, en estos casos se encontró que más de una persona del equipo puede también realizar la especificación al mismo tiempo que codifica el sistema.

Capítulo VI. Conclusiones

A continuación se explican los resultados que se obtuvieron durante la investigación descrita en esta tesis. Posteriormente, se listan y comentan una serie de conclusiones que se presentan dentro del ámbito de la investigación realizada.

Como se mencionó en capítulos anteriores, la Ingeniería de Requisitos (RE) dentro del desarrollo de software tiene un papel muy importante, debido a que la especificación de requisitos en un proyecto propicia la buena ejecución de las etapas posteriores del proceso de desarrollo del software, lo que incrementa la posibilidad de obtener un producto de calidad y adecuado a las necesidades del cliente.

Esta tesis presentó una aproximación metodológica denominada DERDSS, la cual cuenta con una serie de actividades y tareas que ayudan a los analistas durante la etapa de especificación de requisitos dentro de un proceso de desarrollo de software. La propuesta está conformada en tres etapas, cada una de ellas con las técnicas específicas que se usan la fase de elicitación. La etapa 1 llamada Elicitación de Requisitos, consiste en conocer el problema que el cliente nos presenta como una necesidad. La etapa 2, denominada Análisis de Requisitos, permite saber las reglas de negocio, las metas y la utilización de ciertas técnicas para obtener más información y, finalmente, la etapa 3, Especificación de Requisitos, se utiliza para emplear técnicas que nos ayuden a obtener los requisitos funcionales y no funcionales. Una descripción más completa de cada una de las etapas puede consultarse el Capítulo V de esta tesis. El producto final consiste en un documento de especificación de requisitos de software, el cual se encuentra en el Anexo C de ese trabajo. Finalmente, se mostraron los detalles de su definición en el ámbito de la Ingeniería de Software y la aplicación de la RE en el proceso de desarrollo. Es importante mencionar que DERDSS se desarrolló para ser aplicada en PyMES que fabrican software en el estado de Sinaloa, esto con la finalidad de reducir esfuerzo durante la fase inicial del proceso de desarrollo de software.

Se concluye la tesis listando los aportes que se obtuvieron durante la investigación realizada con respecto al área de la Ingeniería de Requisitos:

- Revisión sistemática sobre métodos, técnicas y herramientas utilizadas actualmente por las fábricas de software para la especificación de requisitos durante la etapa de elicitación, en la cual se concluye la problemática que tiene Sinaloa en cuanto al desarrollo de software con respecto a la utilización de técnicas.
- Estado actual de la práctica de la Ingeniería de Requisitos en las PyMES que fabrican software en el estado de Sinaloa, concretamente en las ciudades de Mazatlán, Culiacán y Los Mochis.
- Se sugiere utilizar más de una técnica para la elicitación en el proceso de desarrollo potencializando con esto una buena especificación de requisitos y la obtención de un sistema de mayor calidad. En este sentido, la entrevista y la lluvia de ideas son las dos más utilizadas durante esta etapa, con lo cual se recomienda su combinación.
- Catálogo de carencias encontradas durante la fase de elicitación de requisitos en el proceso de desarrollo aplicado por las fábricas de software en Sinaloa (consultar Anexo D) que, al tomarse en cuenta, permite dos cosas básicamente: i) mejorar la productividad de la fábrica de software y ii) propiciar el crecimiento económico del estado de Sinaloa debido a que permitirá que las empresas inicien un proceso de madurez en su estructuración y operación aumentando la productividad.
- Las universidades con estudios de licenciatura del área de informática y sistemas podrán actualizar sus planes de estudio con los resultados del estudio realizado en la tesis, por ejemplo, agregando a los contenidos temáticos las herramientas que las fábricas de software en Sinaloa utilizan actualmente durante su proceso de desarrollo, lo que permitirá que las PyMES puedan obtener recurso humano capacitado con las tecnologías utilizadas hoy en día.

Por lo que respecta a la aproximación metodológica obtenida como producto principal de la tesis, las ventajas se detallan a continuación:

- Facilitar la construcción de software en MiPyMES de reciente creación que no tengan la solvencia económica suficiente para adoptar un proceso

de desarrollo de software certificado como lo es CMMI *for Dev (Capability Maturity Model Integration).*

- Involucrar al cliente desde la primera fase del proyecto, para que desde un inicio tenga el conocimiento de que es lo que se está desarrollando y al pasar a una segunda etapa no haya dudas de que lo que pidió es lo que va a recibir.
- La aproximación metodológica puede aplicarse en proyectos cortos o largos.
- Reducir los errores críticos por una mala especificación de los requisitos.

Finalmente, es importante mencionar que la revisión sistemática de la literatura sobre la práctica de la RE aplicada a las PyMES desarrolladoras de software realizada en esta tesis, más el estudio a las 25 fábricas en Sinaloa y la aproximación metodológica obtenida constituyen los productos obtenidos derivados de la investigación realizada en la tesis.

6.1. Trabajo Futuro

La validación en campo de la aproximación metodológica queda programada como trabajo futuro, así como el desarrollo del software y el número de registro de propiedad intelectual del producto generado, se propone validar la aproximación metodológica DERDSS a través de un estudio, el cual consistirá en la selección de 3 fábricas de software del estado de Sinaloa que implementen la propuesta presentada en la tesis a través de proyectos piloto. De esta forma, se podrán comparar los resultados obtenidos con las metodologías con las que las fábricas seleccionadas trabajan en la actualidad, y validar su desempeño. Asimismo, este estudio será el punto de partida para continuar con el desarrollo de las etapas posteriores a la especificación de requisitos (análisis, gestión, validación, obtención) y generar una metodología completa.

Anexo A. Cuestionario

Se muestra el cuestionario que fue aplicado a las empresas seleccionadas para la investigación.

Sección 1 Datos de la Empresa

1. Nombre de la Empresa.
2. Ubicación de la Empresa:

   Responder poniendo el inciso ( )

   a) Culiacán

   b) Mazatlán

   c) Los Mochis
3. ¿Cuántos trabajadores tiene la empresa en total?
4. ¿Cuánto tiempo lleva la Empresa prestando Servicio?

Sección 2 Ingeniería de Requisitos

5. ¿Están familiarizados en la empresa con la Ingeniería de Requisitos?
6. ¿Manejan en la empresa el concepto de requisito o requerimiento?

   Responder poniendo el inciso ( )

   a) Requisitos

   b) Requerimientos

   c) Ambos
7. ¿Están familiarizados en la empresa con las etapas de la Ingeniería de Requisitos?

   Análisis, Especificación, Elicitación, Verificación y Administración

8. ¿Qué tan importante cree que es la IR para la empresa?

   Responder poniendo el inciso ( )

a) Lo más importante

b) Muy importante

c) Regular importancia

d) Mínima importancia

e) No es importante en absoluto

9. ¿Por qué considera ese nivel de importancia?

10. ¿La empresa fomenta u ofrece, capacitación o certificación para sus empleados en el área de sistemas? ¿Cómo?

11. ¿Cuánto tiempo (horas) se dedica a actividades relacionadas con IR por proyecto? En términos de porcentaje, siendo 100% todo el proceso. Responder poniendo el inciso ( )

a) 5

b) 10

c) 15

d) 20

e) 25

f) 30

g) 35

h) 40

i) Otro: Especifique

12. ¿Cómo especifican los requisitos de un nuevo proyecto?

Responder poniendo todos los incisos que correspondan ( )

a) Lenguaje natural

b) Métodos formales

c) UML

d) Casos de uso

e) Historias de usuario

f) Modelo de datos

g) Diccionario de datos

h) Ventanas virtuales

i) Diagramas de contexto

j) Descripciones de tarea

k) Escenarios

l) Tareas de alto nivel

m) Diagramas de flujo de datos

n) Diagramas de estado

o) Diagramas de estado transición

p) Diagrama de actividades

q) Diagrama de clases

r) Diagrama de secuencia

s) Otro:

13. ¿Quién especifica la mayoría de los requisitos?

Responder poniendo el inciso ( )

a) Ingeniero de requisitos

b) Líder del proyecto

c) Analista

d) El cliente directamente

e) Gerentes

f) Los desarrolladores

g) Los testers

h) Otro

14. ¿Cómo identifican requisitos incompletos, mal planteados o erróneos?

15. ¿Sus clientes están de acuerdo en participar en actividades relacionadas con la Ingeniería de Requisitos?

16. ¿Cuáles de estas técnicas de elicitación de requisitos ha utilizado?

Responder poniendo todos los incisos que correspondan ( )

a) Entrevistas estructuradas

b) Cuestionarios/Encuestas

c) Emparrillado (Repertory grids)

d) Lluvia de ideas

e) Prototipos de software

f) Escenarios/Casos de uso

g) Método Delphi (Juicio de Expertos)

h) Documentos relacionados

i) Minería de Datos

j) Modelado Semiformal (DFD, UML)

k) Modelado Formal (Z, VDM, SDL)

l) Grupos de concentración

17. ¿Cuáles de estas técnicas de análisis de requisitos ha utilizado?

Responder poniendo todos los incisos que correspondan ( )

a) Checklist

b) Matrices de interacción

c) Entrevistas

d) Lluvia de ideas

e) Diagrama de flujo de datos

f) Diccionario de datos

g) Diagrama estructurado de datos

h) Gráficas estructuradas

i) Tablas de decisión

j) Arboles de decisión

k) Casos de uso

l) Modelado de clases

18. ¿Qué tan a menudo se comunican con el cliente para discutir los requisitos?

Responder poniendo el inciso ( )

a) 1 vez al mes

b) 1 vez por sema

c) 2 veces por semana

d) 3 o más veces por semana

e) No se contacta al cliente

f) Otro:

19. ¿Utilizan plantillas basadas en algún estándar para el documento de requisitos? Ejemplo: IEEE ¿Cuáles?

20. ¿Qué herramientas de IR utiliza?

Selecciona todos los que correspondan.

a) Innoslate Intland

b) Software codoBeamer

c) Rational DOORS

d) Rational DOORS Next Generation

e) Rational RequisitePro

f) Rational Requirements Composer

g) TraceCloud

h) Kovair Requirements Management

i) Blueprint Requirements Definition & Management

j) Visual Paradigm Requirements Capturing

k) HP Requirements Management

l) PTC Integrity

m) ObjectiF Requirements Modeller

n) Polarion Requirements

o) RQA Requirements Quality Analyzer for system engineering projects

p) Visure Requirements TestTrack

q) Otro:

21. ¿Cree que haga falta información o estudios en el Estado de Sinaloa, relacionados con IR? ¿Por qué?

Anexo B. Llenado del documento DRU

En el Capítulo V de la tesis se habla sobre el contenido que debe llevar Documento de Requisitos de Usuario (DRU), en esta parte se muestra un ejemplo de cómo se debe llenar.

<table>
<tr><th colspan="4">Documento de Requisitos de Usuario (DRU)</th></tr>
<tr><td>Tipo de Proyecto
[x] Nuevo
[ ] Modificación</td><td>Fecha
16/07/2017</td><td>Folio
1245</td><td>Versión
1</td></tr>
<tr><td colspan="4">Nombre del cliente: Lizbeth Zamudio
Nombre de la empresa: Restaurant Fontan</td></tr>
<tr><td colspan="4">Descripción general de la solicitud: Quiero un sistema en el cual me permita generar mis facturas en formato xml y que se puedan enviar a mis clientes el mismo día que las generan.</td></tr>
<tr><td colspan="4">Problemática actual: En la actualidad en mi restaurant no tengo en mi sistema del restaurant una opción para poder generar a mis clientes facturas de acuerdo al nuevo formato que se pide por el SAT.</td></tr>
<tr><td colspan="4">Alcance del sistema a desarrollar: Quiero que en mi sistema se cuente con la opción de generar factura y podérsela enviar al cliente a su correo.</td></tr>
</table>

Anexo C. Llenado del documento DERS

Documento de Especificación de Requisitos de Software (DERS)			
Tipo de Proyecto ☑ Nuevo ☐ Modificación	**Fecha** 16/07/2017	**Folio** **1245**	**Versión** **1**
Nombre del cliente: Lizbeth Zamudio **Nombre de la empresa:** Restaurant Fontan.			
Descripción general de la solicitud: Quiero un sistema en el cual me permita generar mis facturas en formato xml y que se puedan enviar a mis clientes el mismo día que las generan.			
Problemática actual: En la actualidad en mi restaurant no tengo en mi sistema del restaurant una opción para poder generar a mis clientes facturas de acuerdo al nuevo formato que se pide por el SAT.			
Alcance del sistema a desarrollar: Quiero que en mi sistema se cuente con la opción de generar factura y podérsela enviar al cliente a su correo.			
Nombre del proyecto: Facturación electrónica xml.			
Fecha inicio: 18/07/2017		**Fecha fin:** 15/09/2017	
Datos del equipo de desarrollo: se Analista: María González. Líder de proyecto: Sebastián Rivera. Arquitecto: Julio Sánchez. Desarrolladores: Rafael López. Tester: Carmen Padilla. Auditor de procesos: Mirna Pasos.			
Reglas de negocio: Se debe de generar una factura por cada ticket no importa el total del mismo y poner todos los datos de la empresa.			
Requisitos: Funcionales: - Capturar la información del ticket fecha, número de nota, sucursal, total). - Capturar los datos fiscales de la persona que solicite la factura (RFC, domicilio, teléfono, etc). - Generar un folio por cada factura generada. No Funcionales: - La factura se podrá reimprimir.			

- Se debe de validar todos los datos y no permitir generarla si van datos en blanco.

Apéndice: en esta parte se especifican las técnicas y modelos empleados durante la aproximación metodológica.

Observación:

Durante el proceso de observación pudimos constatar que el único recibo generado por el restaurant son notas y que en la mayoría de los casos los clientes le solicitaban una factura electrónica.

Cuestionario:

1. ¿Qué es lo que habitualmente haces cuando un cliente que ya consumió pide le des factura para comprobar ese gasto?

Reuniones de Trabajo:

En esta parte se deben de especificar todas las reuniones que se hayan tenido durante el proceso de elicitación, y se deben de capturar los siguientes datos:

Fecha: 20-07-2017 (día, mes, año) Hora inicio: 4:00 Hora fin: 6:00

EP= Equipo de proyectos

C= Cliente

Reunión con: EP

Tema a tratar: Revisión del problema a resolver.

Puntos relevantes de la minuta:

- Que datos le interesa al cliente que vengan en la factura.
- Como va a ser la asignación de folios por parte del SAT.

Entrevista: en la entrevista con el cliente, el analista deberá de capturar las preguntas hechas al cliente con sus respectivas respuestas, así como datos adicionales que el cliente comento y que no vienen en la petición.

Fecha: 22-07-2017 (día, mes, año) Fecha inicio:10:00 Fecha Fin:12:00

Nombre del cliente: Héctor Tejeda.

Preguntas:

1. ¿Qué datos le interesa que vengan en la factura tanto del cliente como de la empresa?
2. ¿Cómo va a ser la asignación de folios por parte del SAT?

Diagrama de Flujo de Datos / Escenarios:

Para este proyecto no fue necesario.

Anexo D. Catálogo de carencias

Durante el proceso de investigación realizado en la tesis se encontraron algunas carencias en la etapa de RE en las PyMES que fueron seleccionadas, la cuales se enlistan a continuación:

1. La mayoría de las empresas cuenta con una metodología o un proceso para el desarrollo de un sistema o proyecto, pero no siempre lo cumplen al 100%, esto debido a que son métodos comprados y ajustados al tipo de proyectos que desarrolla la empresa, por lo que el tipo de proyectos que la empresa maneja no son lo suficientemente robustos para grandes metodologías comerciales.
2. No hay una definición específica de las actividades que debe realizar cada miembro del equipo de desarrollo, esto es, que cada uno puede hacer más de un rol, lo cual implica que no haya un dominio de sus actividades.
3. La falta de capacitación a los empleados en el desarrollo de nuevas técnicas y herramientas, debido, entre otras, a la falta de presupuesto.
4. La poca importancia que se le da a la etapa de RE durante el proyecto hace que se generen productos de baja calidad.
5. La falta de presupuesto en la empresa hace que los proyectos que se generen sean pequeños y la infraestructura con la que cuenta no dé para el desarrollo de proyectos más grandes.
6. Desconocimiento de la teoría con respecto a lo que es un Requisito Funcional y un Requisito No-Funcional, lo cual afecta al momento de llevar a cabo una correcta especificación de requisitos para iniciar el proceso de desarrollo, así como en las etapas finales del proceso, porque no pueden validar los requisitos ni mantenerlos.

Glosario

Término	Significado
Ad-hoc	Se refiere al término hecho a la medida.
Cliente	Persona física que solicita la manufactura de un sistema y se encarga de realizar el contrato con el equipo de desarrollo. Generalmente se le denomina de esta forma porque él es el que realiza el pago del sistema.
Elicitación	Conseguir u obtener información específica.
Herramienta	Conjunto de instrumentos que se utilizan para desempeñar un oficio o un trabajo determinado.
Método	Modo ordenado y sistemático de proceder para llegar a un resultado o fin determinado.
Metodología	Se denomina la serie de métodos y técnicas de rigor científico que se aplican sistemáticamente durante un proceso de investigación para alcanzar un resultado teóricamente válido.
Procedimiento	Es un conjunto de acciones u operaciones que tienen que realizarse de la misma forma, para obtener siempre el mismo resultado bajo las mismas circunstancias.
Proceso	Procesamiento o conjunto de operaciones a que se somete una cosa para elaborarla o transformarla.
PyMES	Es el acrónimo de pequeña y mediana empresa. Se trata de la empresa mercantil, industrial o de otro tipo que tiene un número reducido de trabajadores y que registra ingresos moderados.
Requerimiento	Se enfocan en las necesidades operacionales del producto software a desarrollar, de infraestructura, hardware o de usuarios.

Requisito	Son las descripciones de los servicios que el sistema debe proporcionar, así como las restricciones operativas que este debe de tener.
Revisión Bibliográfica	Es un procedimiento estructurado cuyo objetivo es la localización y recuperación de información relevante para un usuario que quiere dar respuesta a cualquier duda relacionada con su práctica, ya sea ésta clínica, docente, investigadora o de administración.
Sistema	Es un conjunto de elementos relacionados entre sí y que funcionan como un todo.
Técnica	Conjunto de procedimientos o recursos que se usan en un arte, en una ciencia o en una actividad determinada, en especial cuando se adquieren por medio de su práctica y requieren habilidad.
Usuario	Persona que utiliza un servicio, proceso o sistema de cómputo.

Referencias

[1] Estayno, M.; Dapoza, G.; Cuenca, L.; Greiner, C.; Medina, Y.; Ferraro, M.; Acuña, C.: Métodos y Herramientas Orientados a la Calidad del Software (2012).

[2] Escalona, M. J., & Koch, N. (2004). Requirements engineering for web applications-a comparative study. J. Web Eng., 2(3), 193-212.

[3] Saiedian, H.; Dale, R.: Requirements Engineering: Making the Connection between the Software Developer and Customer. Department of Computer Science – University of Nebraska (1999).

[4] Félix, M.A.; Sandoval, L.A., Martínez, R., López, E.: Análisis de la Industria del Software. Caso Sinaloa, Congreso Internacional de Investigación, ISSN 1946-5351.

[5] Ros, J.N.: Una Propuesta de Gestión Integrada de Modelos y Requisitos en Líneas de Productos Software, Tesis Doctoral, 2009.

[6] Muñoz, M.; Gasca, G.; Valtierra, C.: Caracterizando las Necesidades de las PyMES para Implementar Mejoras de Procesos de Software: Una Comparativa entre la Teoría y la Realidad. Risti (Revista Ibérica de Sistemas y Tecnología de la Información), 2014.

[7] Gómez, G.E.; Aguileta A.A.; Ancona G.B.; Gómez, O.S.: Avances en las Mejoras de Procesos Software en las MiPyMES Desarrolladoras de Software: Una Revisión Sistemática.

[8] Fayad, M.E.; Laitinen M.; y Ward, R.P.: Software Engineering in the Small, communications of the ACM, pp 123-132 (2000).

[9] Navarro, A.; Fernández, J.; Morales, J.: Revisión de metodologías ágiles para el desarrollo de software, Universidad Icesi 2013.

[10] Gomis, R. y Hualde A.: “La innovación en la industria de software”. En Villavicencio Daniel López Pedro Luis. Sistemas de Innovación en México: regiones, redes y sectores. P. 191-186, 2009

[11] Medina, J.C.: Análisis Comparativo de Técnicas, Metodologías y Herramientas de Ingeniería de Requerimientos, tesis de maestría, México, D.F 2004.

[12] Sitio Web Oficial de Cepal en URL: www.cepal.org, consultado en 15 de Noviembre de 2016.

[13] Somerville, I.: Ingeniería de Software. Editorial Pearson Educación, Madrid España. (2005).

[14] Pressman, R.: Ingeniería del Software, un Enfoque Práctico. Editorial Mc Graw Hill, México, D.F, Séptima Edición (2010).

[15] Cortes, C.A.; Abud, M.A.; Romero, C.: Propuesta de un Catálogo de Patrones de Escenario para la Definición de Requisitos, Congreso Internacional de Mejora de Procesos Software CIMPS 2015.

[16] Herramienta case caliberRM, consultada el 20 de Abril de 2017 en: https://imgalib.wordpress.com/2009/11/12/caliber-rm-an-excellent-tool-for-requirements-management/

[17] Arias, M.: La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software, Costa Rica 2006

[18] Serna, E.: Estado Actual de la Investigación en Requisitos No Funcionales. Bogotá Colombia, 2012.

[19] Cysneiros, L. M. y Leite, J. C.: Integrating non-functional requirements into data model. Fourth IEEE International Symposium on Requirements Engineering (ISRE 99). Limerick, Ireland, 7-11, 1999, pp. 162-171.

[20] Chung, L.; Nixon, B. A.; Yu, E. y Mylooulos, J. Non-functional requirements in software engineering. Norwell, USA: Kluwer Academic Publishers, 2000.

[21] Nuseibeh, B., Easterbrook, S. M.: ‘Requirements Engineering: a Roadmap’. Proc. Int. Conf. On Software Engineering (ICSE), New York, 2000, pp. 35-46.

[22] Maiden, N., Rugg, G.: 'ACRE: Selecting Methods for Requirements Acquisition’, Software Engineering Journal, 1996, 11, (3), pp. 183-192.

[23] Yu, E.: ‘Modelling Strategic Relationships for Process Reengineering’. PhD thesis, University of Toronto, 1995.

[24] Bass, L., Merson, P., Clements, P., Bergey, J., Ozkaya, I., Sangwan, R.: ´A Comparison of Requirements Specification Methods from a Software Architecture Perspective´. Software Engineering Institute, Carnegie Mellon University, 2006.

[25] Escalona, M.J., Koch, N.: Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.

[26] S. E. Institute: 'CMMI for Development: Guidelines for Process Integration and Product Improvement', (Addison-Wesley Professional, 2011, 3rdedn.)

región del Valle del Évora, Sinaloa, México, consultada en: 10 de Diciembre de 2016 en: http://reaxion.utleon.edu.mx/Art_Impr_Clusters_Empresas_Valle_Del_Evora_Sonora.html

[27] Villalta, A., Carvallo, J.P.: Modelos de Calidad de Software: Una Revisión Sistemática de la Literatura, MASKANA, CEDIA 2015.

[28] Plantilla IEEE, consultada el día 20 de Julio de 2017 en: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

[29] Plantilla de especificación de requisitos volere, consultada el día 20 de julio de 2017 en: http://www.volere.co.uk/pdf%20files/template_es.pdf

[30] Scalone, F.: Estudio comparativo de los modelos y estándares de calidad del software, 2006.

[31] Ingeniería de Software, consultado en: 10 de Enero de 2017 en: http://www.academia.edu/5623280/Curso-de-introduccion-a-la-ingenieria-del-software.

[32] Sitio Web Oficial de Wikipedia. URL: www.wikipedia.org: consultada en: 09 de Febrero de 2017 en: https://es.wikipedia.org/wiki/ISO_8402.

[33] Mansilla, D., Pollo Cattaneo, M. F., Pytel, P., & García Martínez, R. (2012). Modelo de proceso para la elicitación de requerimientos en proyectos de explotación de información. In XIV Workshop de Investigadores en Ciencias de la Computación.

[34] Hossian, A., Sierra, E., García-Martínez, R., Ochoa, M. A., & Britos, P. (2007). Hacia una Metodología Orientada al Conocimiento para la Educción de Requisitos en Ingeniería del Software. In JIISIC (pp. 107-114).

[35] Orjuela, A.; Rojas, M.: The Methodologies of Agile Development like an Opportunity for the Engineering of Educative Software, 2008.

[36] Alfonso, P.L.; Mariño, S.; Godoy, M.V: Propuesta Metodológica para la Gestión del Proyecto de Software Ágil basado en la Web. 2011

[37] Rizwan. M.; Jacob, S.: Proposal of Enhanced Extreme Programming Model, 2015.

[38] Metodologías de Desarrollo Ágil, consultada en: 15 de Diciembre de 2016 en: http://rdsoporteymantenimientodepc.blogspot.mx/2014/03/metodologias-de-desarrollo-agiles-vs.html

[39] Paetsch, F., Eberlein, A.; Maurer, F.: Requirements Engineering and Agile Software Development, consultada en: 13 de enero de 2017 en: http://ase.cpsc.ucalgary.ca/uploads/Publications/PaetschEberleinMaurer.pdf

[40] Jeffries, R.; Anderson, A.; Hendrickson, C.: Extreme Programming Installed. Addison-Wesley. 2001.

[41] Beck, K.: "Extreme Programming Explained. Embrace Change", Pearson Education, 1999. Traducido al español como: Una explicación de la programación extrema. Aceptar el cambio, Addison Wesley, 2000.

[42] Joskowicz, J.: Reglas y Prácticas en Extreme Programming, 2008.

[43] Oliveros, A., Wehbe, R., Rojo, S. D. V., & Rousselot, J. (2011). Requerimientos para aplicaciones web. In XIII Workshop de Investigadores en Ciencias de la Computación.

[44]Metodología XP, consultada el 10 de Enero de 2017 en: (https://www.google.com.mx/search?q=desarrollo+agil+xp&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiC0qL8dzVAhVE5mMKHcBAD4cQ_AUICigB&biw=1366&bih=651)

[45] Rivadeneira, S.G.: Metodologías Ágiles Enfocadas al Modelado de Requerimientos. Mayo, 2012.

[46] Metodología DSDM, consultada el 10 de Enero de 2017 en: (https://www.google.com.mx/search?biw=1034&bih=655&tbm=isch&sa=1&q=DSDM&oq=DSDM&gs_l=psyab.3..0l4.967613.970737.0.971142.45.10.0.0.0.0.539.738.0j1j5 1.2.0....0...1.1.64.psy-ab...43.2.735.CpZ2pwKgFlU)

[47] James A.: Highsmith III: Adaptive Software Development, Dorset House Publishing, 1996.

[48] Schenone, M. H.: Diseño de una Metodología Ágil de Desarrollo de Software, 2004.

[49]Metodología ASD, consultada el 10 de Enero de 2017 en: (https://www.google.com.mx/search?tbm=isch&q=adaptive+software+development&spell=1&sa=X&ved=0ahUKEwjDurnud3VAhXkyFQKHaw1Ce4QvwUIIigA&biw=1366&bih=700&dpr=1)

[50] Bojórquez, F.; Bojórquez A.: Cultura empresarial en la PyMES. Caso Sinaloa.

[51]Metodología Crystal, consultada el 10 de Enero de 2017 en: https://www.google.com.mx/search?q=metodologia+crystal&dcr=0&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjErZOftL_WAhVI3WMKHR-aBDUQ_AUICigB&biw=1034&bih=655#imgdii=IwyYyh9AiBotnM:&imgrc=EnVdKnsmhn2zBM

[52] Díaz, Ramón.: Las metodologías ágiles como garantía de calidad del software. Revista Española de Innovación, Calidad e Ingeniería del software, Vol.5, No 3, 2009.

[53] Herramienta case caliberRM, consultada el 20 de Abril de 2017 en: https://imgalib.wordpress.com/2009/11/12/caliber-rm-an-excellent-tool-for-requirements-management/

[54] Alarcon, A., & Sandoval, E. (2011). Herramientas CASE para ingeniería de Requisitos. Cultura Científica, 6(6), 70-74.

[55]Herramienta case controla, consultada el 10 de Agosto de 2017 en: http://herramientascaseerick.blogspot.mx/2016/10/ingenieria-requisitos-ejemplos-irqa-43.html

[56] Herramienta case Osrmt, consultada el 10 de Agosto en: http://www.ipcorp.com.ar/blog/2008/12/11/osrmt-open-source-requirements-management-tool/

[57] Herramienta case Jeremía, consultada el 10 de Agosto de 2017 en http://herramientascaseerick.blogspot.mx/2016/10/ingenieria-requisitos-ejemplos-irqa-43.html

[58]Báez, M. G., & Brunner, S. I. B. (2001, November). Metodología DoRCU para la Ingeniería de Requerimientos. In WER (pp. 210-222).

[59] Sitio Web Oficial de INEGI. URL: www.inegi.org, Consultada en: 23 de diciembre de 2016 en: http://www.beta.inegi.org.mx/app/mapa/denue/.

[60] Metodología DSDM, consultada el 10 de Enero de 2017 en: (https://www.google.com.mx/search?biw=1034&bih=655&tbm=isch&sa=1&q=DSDM&oq=DSDM&gs_l=psyab.3..0l4.967613.970737.0.971142.45.10.0.0.0.0.539.738.0j1j5 1.2.0....0...1.1.64.psy-ab...43.2.735.CpZ2pwKgFlU)

[61] González Palacio, L., & Urrego Giraldo, G. (2008). Modelo de requisitos para sistemas embebidos: Model of requirements for embedded systems. Revista Ingenierías Universidad de Medellín, 7(13), 111-127.

[62] Richards, D. (2000). A process model for requirements elicitation. In Proceedings of The 11th Australasian Conference on Information Systems, Brisbane, Australia.

[63] Arancibia, J. A. G. (2009). Metodología para la definición de requisitos en proyectos de data mining (er-dm) (Doctoral dissertation, Universidad Politécnica de Madrid).

[64]Rilston, F. P., & Castro, S. J. FB (2003). DWARF: An Approach for Requirements Definition and Management of Data Warehouse Systems. In Proceeding of the 11th IEEE International Requirements Engineering Conference (pp. 08-12).

[65] Aguilar Calderon, J., Garrigós, I., Casteleyn, S., & Mazón, J. N. (2012). WebREd: A model-driven tool for web requirements specification and optimization. Web Engineering, 452-455.

[66] Cita Vargas Agudelo, F. A. (2010). Método para establecer la consistencia de los problemas en el diagrama causa efecto con el diagrama de objetivos de Kaos(Doctoral dissertation, Universidad Nacional de Colombia sede Medellín).

[67] Sistema Económico Latinoamericano y del Caribe - SELA. “Desarrollo de una industria regional de software en América Latina y el Caribe: consideraciones y propuestas”. Venezuela: Sistema Económico Latinoamericano y del Caribe, 2009.

[68] Delía, L., Galdamez, N., Thomas, P. J., & Pesado, P. M. (2013). Un análisis experimental de tipo de aplicaciones para dispositivos móviles. In XVIII Congreso Argentino de Ciencias de la Computación.

[69] Balaguera, Y. D. A. (2015). Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Estado actual. Revista de Tecnología, 12(2).

[70] Oliveros, A., Danyans, F. J., & Mastropietro, M. L. (2014). Prácticas de Ingeniería de Requerimientos en el desarrollo de aplicaciones Web. In Proceedings of the XVII Ibero-American Conference on Software Engineering, Pucón, Chile, 2014, pp. 491-505.

[71] Oliveros, A., Wehbe, R., Rojo, S. D. V., & Rousselot, J. (2011). Requerimientos para aplicaciones web. In XIII Workshop de Investigadores en Ciencias de la Computación.

[72] Escalona, M. J., Torres, J., Mejías, M., Jurado, M. C., & Fillerat, L. L. NDT: Navigational Development Techniques. 2006.

[73] Ocampo Vásquez, E. A., Naranjo, R., & Bladimir, G. (2009). Desarrollo del sistema de control de proyectos para Santos CMI CONSTRUCTION INC utilizando la metodología OOWS (Bachelor's thesis, SANGOLQUÍ/ESPE/2009).

[74] Garrigos, I., & Mazón, J. N. Aproximaciones en ingeniería web para el análisis de requisitos: una revisión sistemática de la literatura, Universidad Alicante España, 2014.

[75] Escalona Cuaresma, M. J., Mejías Risoto, M., & Torres Valderrama, J. (2002). Methodologies to develop web information systems and comparative analysis. Upgrade: the European Online Magazine for the Information Technology Professional, 3(3), 25-36.

[76] Merchán, L., Urrea, A., & Rebollar, R. (2008). Definición de una metodología ágil de ingeniería de requerimientos para empresas emergentes de desarrollo de software del sur-occidente colombiano. Revista Guillermo de Ockham, *6*(1).