Desarrollo y gestión de requisitos: Resultados de una revisión de la literatura

Development and management requirements: Results of a literature review

Edgar Serna M. **Alexei Serna A.**
edgar.serna@uniremington.edu.co *alexei.serna@uniremington.edu.co*
Corporación Universitaria Remington
Medellín, Antioquia. Colombia

RESUMEN
En la literatura se proponen diferentes modelos para gestionar la Ingeniería de Requisitos, que presentan ventajas y desventajas al momento de especificar adecuadamente las necesidades del cliente. Con el objetivo de determinar las buenas prácticas de estos modelos, se realizó una revisión a la literatura para valorar las características y potencialidades de los más populares. En este trabajo se describen los resultados, y se concluye que un modelo para gestionar la Ingeniería de Requisitos debería reunir las buenas prácticas encontradas, y potencializarlas con aportes desde otras áreas del conocimiento, como los métodos formales, la matemática discreta, la lógica, la abstracción, y otras relacionadas. Esto se debe a que los problemas actuales son más complejos y complicados que antes, la seguridad se ha convertido en un asunto clave, las modificaciones son cotidianas, y los tiempos de entrega de los productos software se han acortado drásticamente, a la vez que los presupuestos se deben ajustar a su vida útil.

Palabras clave
Ingeniería de Requisitos, modelos de gestión, calidad del software, Ingeniería del Software.

ABSTRACT
In the literature, different models are proposed to manage the Requirements Engineering, which present advantages and disadvantages when properly specified customer requirements. In order to identify good practices of these models, a literature review was conducted to assess the characteristics and potential of the most popular. The results are described in this work, and concludes that a model to manage the requirements engineering should gather good practices found, and potentiate them with contributions from other areas of knowledge, as formal methods, discrete mathematics, logic, abstraction, and other related. This is because the current problems are more complex and complicated than before, security has become a key issue, the changes are every day, and delivery times of the software products have been drastically shortened, while budgets must be adjusted to their service life.

Keywords
Requirements Engineering, management models, software quality, Software Engineering.

1. INTRODUCCIÓN
La Ingeniería de Requisitos (RE, por sus siglas en inglés) es la fase más importante del proceso de desarrollo de software [1, 2]. Además, típicamente los requisitos son volátiles, y es difícil hacerles seguimiento a los cambios, lo que repercute en las demás fases del ciclo de vida, porque incrementa el costo de los proyectos y afecta el calendario establecido [3]. Si la volatilidad de los requisitos se extiende en el tiempo, la probabilidad de éxito del proyecto se reduce drásticamente, y se vuelve complicado y propenso a errores [4]. Por esto es importante desarrollar y gestionar adecuadamente los requisitos, para que las otras fases no sufran contratiempos inesperados. Berry and Lawrence [5] sugieren que el objetivo de la RE es introducir los principios ingenieriles en la práctica del análisis de los sistemas de información, y para lograrlo se debe implantar bajo un proceso sistemático y disciplinado [6].

Debido a que en las diferentes comunidades de investigación se comprende que es importante para el éxito de los proyectos software atender adecuadamente estos procesos, desde hace algún tiempo han propuesto modelos de gestión de requisitos conformados por actividades estructuradas y repetibles. Esto se hace evidente cuando la mayor parte de las causas de los fracasos se relaciona con esta fase de la Ingeniería de Software (SE, por sus siglas en inglés) [7]. Por otro lado, las fases subsiguientes del desarrollo del producto dependen en gran medida de la RE [8, 9], entonces en el proceso se deben cubrir eficientemente todas las actividades involucradas, como la elicitación, la gestión y la documentación del conjunto de necesidades del sistema y del usuario. Si esta fase no desarrolla adecuadamente, el producto puede sufrir contratiempos, como que no se entregue a tiempo, que desborde el presupuesto, que no satisfaga las necesidades de clientes y usuarios, que sea poco fiable, o que presente demasiados errores. Entre el 40% y el 60% de los defectos de un sistema se relacionan con errores cometidos durante la RE, el 13% de los proyectos fallan debido a requisitos incompletos, y el 9% por el rápido cambio en los mismos. Una forma de evitar estos problemas es aplicar buenas prácticas, procesos, herramientas, tecnologías, metodologías y métodos, cuyo objetivo sea entregar un documento de especificación con suficientes criterios de calidad [10].

En consecuencia, poder identificar a tiempo los problemas y sugerir mejoras en los procesos de la RE, es un principio que puede incrementar el éxito de los proyectos. Desde que Bell y Taylor [11] publicaron su trabajo, se reconoció que la Ingeniería de Requisitos era un proceso ingenieril, y desde entonces se han presentado diversos enfoques con el objetivo de hacer precisamente eso: *aplicar ingeniería*. Aunque la mayoría se ha centrado en la funcionalidad que se espera del producto, y no en el desarrollo y la gestión, algunos métodos y técnicas se orientan a comprenderlos y a modelarlos como un paso importante hacia la perfección del proceso [12], y otros se centran en describir y en proponer mejoras para la práctica, concluyendo cuestiones clave de naturaleza organizacional y no de carácter técnico, como la gestión de documentos y de la incertidumbre [13, 14]. Sin embargo, algunos intentan construir y proponer modelos orientados directamente al desarrollo y la gestión de requisitos, y otros a proponer modelos prescriptivos, en lugar de examinar los descriptivos en la práctica actual [12], e inclusive, otros, se han centrado en los requisitos no-funcionales, y aceptan que los mismos influyen transversalmente en los demás.

El objetivo de esta investigación es realizar una revisión a la literatura, con el objetivo de presentar un estudio comparativo a las prácticas y procesos que proponen los modelos de RE divulgados, mediante un análisis a los indicadores experimentales y prácticos que las comunidades relacionadas reportan. Con esta información se estructura un trabajo futuro, con la idea de integrar esas prácticas en un modelo mejorado, complementado con principios de los métodos formales, la matemática discreta y la lógica, para enriquecer los hallazgos de esta investigación. Esta necesidad se descubre a raíz de que los problemas actuales son más complejos y complicados que antes, la seguridad se ha convertido en un asunto clave, las modificaciones son más cotidianas, y los tiempos de entrega de los productos software se han acortado drásticamente, a la vez que los presupuestos se ajustan a su vida útil.

2. METODOLOGÍA

Las metodologías tradicionales para el análisis de la información tienen en cuenta a la documentación, al mismo tiempo que al entrenamiento necesario para procesarla, e implican otras áreas de mayor nivel, como el descubrimiento de la información requerida, el análisis, la identificación de posibles soluciones, la adecuada producción y la entrega de resultados [15-17]. Por otro lado, los métodos tradicionales de recolección y análisis de datos incluyen al menos una de estas fases, y frecuentemente utilizan entrevistas estructuradas o técnicas grupales para descubrir los datos necesarios. Posteriormente, los analistas pueden modificar la información estructurada para satisfacer las necesidades de clientes y usuarios, y en este proceso deben tener en cuenta los principios de gestión de la calidad, abarcando los diferentes sistemas, estructuras y actores relacionados [18, 19].

Con este objetivo, diferentes empresas e investigadores han propuesto técnicas relacionadas con la documentación computacional, que se utilizan en diferentes escenarios [20]. Es el caso de la encuesta contextual, utilizada para analizar los requisitos de la información, y que se desarrolló como una técnica de investigación interpretativa para recoger y analizar en profundidad los requisitos de usuario, en lo que tiene que ver con el diseño de productos y servicios. El enfoque se basa en un proceso cuidadoso de observación y discusión en el que se involucra a los participantes del proyecto, y a posibles miembros del público, y su aplicación varía en función de los resultados y datos deseados. Los requisitos de usuario se recogen mediante observación y conversación con especialistas y administradores, pero se obtienen mejores resultados cuando el cliente se involucra en los procesos de la organización, lo que implica el trabajo de varias personas o de un equipo.

El mapeo es otra forma de análisis de información, e involucra estrategias y técnicas para crearla, incluso cuando el público es variado y difícil de identificar. Es un enfoque fácil de aprender, y tiene por objeto ayudarles a los diseñadores a analizar, organizar y presentar la información y los materiales de capacitación en un formato modular. Antes de poder usar un método particular para elicitar requisitos los desarrolladores necesitan comprender los principios básicos, los tipos y los bloques de información y los mapas. Weiss [21] propone un enfoque estructurado para el análisis de las necesidades de información que una organización requiere para un producto o servicio, que si se aplica adecuadamente, lo elicitado será público, respetará las reglas y será explícito, debido a que se inicia con una imagen lo más amplia posible, que posteriormente le permitirá al diseñador agregar superposiciones en etapas sucesivas. El modelo incluye actividades como las habilidades y destrezas del equipo, una lista de características y temas, y el análisis de la audiencia, de la cual crea una matriz y determina los parámetros de los formatos a aplicar.

La Ingeniería de Requisitos, como la primera fase del proceso de la Ingeniería de Software, es la tarea más importante para el desarrollo de productos [20, 22], y los requisitos ambiguos son una de las principales razones por las que fracasan los proyectos [23] y se generan defectos en el producto [24, 25]. Debido a esto, se reconoce a la RE como crítica para el desarrollo de software, y actualmente existen diversas metodologías y técnicas para gestionarla. Sin embargo, algunas, que han demostrado éxito para un sistema, parecen no funcionar bien para otro, lo que demuestra que la selección de alguna de ellas puede ser difícil, e incluso llevar al fracaso del sistema. Los cambios continuos en los entornos empresariales también afectan el proceso de esta fase, además, cuando el tiempo para la liberación de las versiones es crítico, una inadecuada selección afectará la calidad del producto, por lo que los modelos y metodología deben proporcionar directrices apropiadas para que los ingenieros tengan una buena base para tomar decisiones.

3. RESULTADOS

Los modelos para la Ingeniería de Requisitos deben describir claramente no sólo actividades que identifican los requisitos del software y del sistema, sino también el entorno de desarrollo, es decir, metas, puntos de vista, necesidades de las partes interesadas, entre otras. A continuación se describen y analizan los diferentes modelos para gestionar requisitos que se encuentran en la literatura, y dado que cada uno tiene un enfoque particular, es necesario analizarlos y comprenderlos con la finalidad de reunir la información necesaria para alcanzar el objetivo de la presente investigación.

NATURE Framework (26, 27]. Este modelo describe al proceso de la RE como un espacio ortogonal de tres dimensiones, como se observa en la Figura 1, y parte del principio de que al comenzar las etapas de la RE el conocimiento del sistema es impreciso, por lo tanto, la especificación es *opaca*, basada en puntos de vista personales, y expresada principalmente mediante representaciones informales. Para que la salida deseada del proceso de especificación de requisitos sea precisa, debe ser *completa*, expresada en un *lenguaje formal*, y *comúnmente aceptada* por todas las partes interesadas, lo que se constituye en los principales objetivos del modelo.

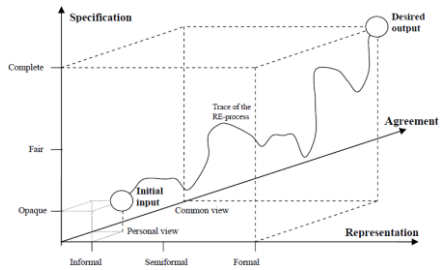

Figura 1. NATURE framework

1. La dimensión de *representación* de requisitos aborda el grado de formalidad de la representación, y tiene el objetivo de transformar al modelo informal en un modelo formal de requisitos.
2. La dimensión de *contrato* de requisitos trata el grado del acuerdo de requisitos, y su objetivo es obtener un acuerdo común acerca del modelo.
3. La dimensión de *especificación* de requisitos se encarga del grado de comprensión de los mismos, en un momento determinado de tiempo. Su objetivo es mejorar la comprensión, hasta entonces opaca, del sistema mediante una especificación

completa de requisitos, en la que la integridad se mide mediante estándares y directrices.

Iterative Requirements Engineering Process Model [28]. Es un modelo no-lineal conformado por tres actividades: elicitación, especificación y validación, que representan el proceso de la RE como iterativo y cíclico por naturaleza, como se observa en la Figura 2. Además, también demuestra las interacciones entre la elicitación, la especificación, la Validación, el usuario y el dominio del problema. Aunque contiene actividades similares a las que proponen los lineales, el orden en el que aparecen no lo es, y sugiere una relación de causa y efecto entre ellas.

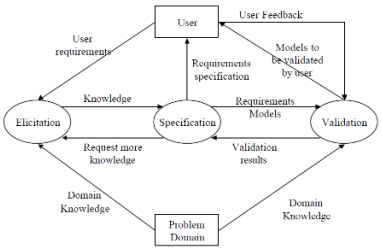

Figura 2. Iterative Requirements Engineering Process Model

1. En la *elicitación* se adquiere el conocimiento necesario para producir un modelo formal de requisitos.
2. En la *especificación* se reciben como entrada los entregables de la elicitación, y el objetivo es crear un modelo formal de requisitos. La especificación de requisitos se puede ver aquí como un documento que define la funcionalidad deseada, sin mostrar cómo se va a alcanzar [29].
3. La *Validación* certifica que el modelo formal de requisitos producido satisface las necesidades de los usuarios.

Este modelo se centra en el principio del desarrollo *inhouse*, porque el sistema se desarrolla al interior de la misma empresa que lo requiere, y en los proyectos de desarrollo por contrato, cuando un proveedor desarrolla un sistema para una empresa cliente [30]. En este tipo de proyectos la especificación y sus salidas se definen como un contrato entre clientes y desarrolladores, sin embargo, el modelo no se ocupa de proyectos en los que no está definido el cliente, por ejemplo, en el desarrollo comercial *off-the-shelf*.

Purely linear process model [31]. Se trata de un modelo puramente lineal, como se observa en la Figura 3, y que no tiene en cuenta la superposición o iteración de actividades, como el conceptual lineal, sino que las clasifica en diferentes grados, aunque la progresión lineal resultante en la documentación es común a ambos modelos. Por otro lado, reconoce que los procesos de RE dependen de la situación, y analiza siete diferentes relaciones entre el cliente y el proveedor, y sus correspondientes procesos en esta fase. El modelo está conformado por cinco actividades organizadas secuencialmente, y, debido a su simpleza, se utiliza principalmente en pequeños proyectos con un nivel de complejidad bajo, pero no es adecuado ni recomendable para proyectos grandes y complejos.

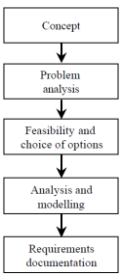

Figura 3. Purely linear process model

The PREview process [32]. PREview es un enfoque orientado a puntos de vista (VOA, por sus siglas en inglés), que considera diferentes perspectivas a la información relacionada con el problema, llamados puntos de vista [33], que surgen debido a las diferentes responsabilidades, funciones, objetivos,… de las fuentes de información (Ver Figura 4). Este enfoque complementa la noción estándar de los puntos de vista con el de las cuestiones organizacionales, como la generalización de la noción de objetivo, que incluye tanto los objetivos como las limitaciones organizacionales que restringen el sistema o proceso a analizar. La fortaleza de PREview se sustenta en su énfasis en las necesidades de las partes interesadas, y la facilidad de integración con los enfoques de requisitos. Además, proporciona un adecuado punto de partida para que los usuarios identifiquen los puntos de vista.

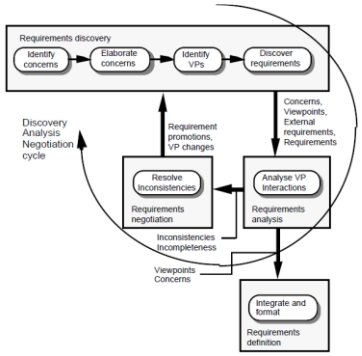

Figura 4. The PREview process

Conceptual linear process model [34]. A diferencia de los modelos lineales e incrementales, en los que las actividades de elicitación y análisis se combinan bajo diferentes premisas, pero siguiendo una transición lineal similar, los autores proponen un modelo conceptual lineal, en el que indican iteraciones entre actividades que se solapan, y que a menudo se realizan iterativamente, como se observa en la Figura 5.

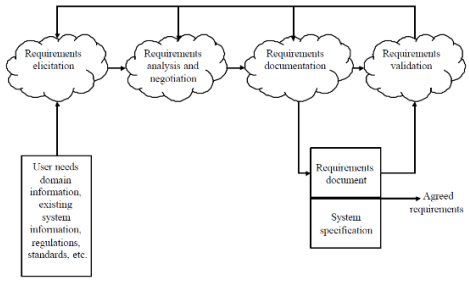

Figura 5. Conceptual Linear Requirements Engineering Process Model

En este tipo de modelos, las actividades se repiten en iteraciones, formando una espiral, y al final de cada iteración se toma una decisión de si se debe aceptar el documento construido o realizar iteraciones adicionales. Para lograrlo, el modelo separa los procesos de la RE en dos etapas: 1) actividades propiamente dichas y 2) gestión de requisitos, que comprenden los principios de formalidad, concordancia y comprensibilidad de un modelo de requisitos. La primera está conformada por la elicitación, el análisis y negociación, la documentación, y la Validación de requisitos. En paralelo con estas actividades se desarrolla el proceso de gestión de requisitos, el cual se encarga de la gestión de los cambios que se presentan en la evolución de los mismos. Estos cambios son necesarios porque, a medida que aparecen nuevas necesidades, permiten el descubrimiento de errores en los requisitos. Esta etapa debe hacerle un seguimiento constante a los cambios, asegurándose que se realicen de manera controlada.

Extreme Programming (XP) [35]. Al igual que RUP, XP tiene un carácter iterativo e incremental (Figura 6), y a primera vista, XP y un análisis de requisitos fundamental parecen contradecirse entre sí, porque el primero está orientado a conseguir un sistema rápidamente implementable en el mercado [35, 36]. Sin embargo, en este modelo también hay enfoques que se complementan bien en el análisis de requisitos, como las historias de usuarios, el juego de planificación y la metáfora del sistema. Por ejemplo, las historias de usuario son informes breves que realizan los usuarios, los cuales se reúnen inicialmente de manera informal; posteriormente se añaden los detalles poco a poco, y a continuación se evalúan.

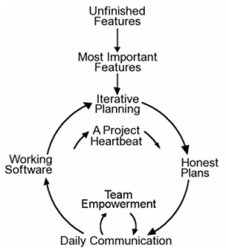

Figura 6. Xtreme Programming model

Problem Frames [37]. Este modelo propone descomponer los problemas complejos en conjuntos estructurados de clases simples y sub-problemas comunes familiares, como se observa en la Figura 7. Las clases de problemas comunes se identifican a partir del análisis del problema, de forma similar al diseño de patrones [38]. Las descripciones/soluciones combinadas para los sub-problemas sirven como descripción/solución para el problema original.

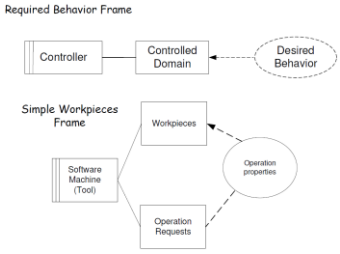

Figura 7. Jackson's problem frames

El modelo sugiere que, una vez que los desarrolladores conocen el contexto del problema, es más fácil reconocer y abordar las variaciones de sus clases, y de anticipar las dificultades, así como producir soluciones eficientes. Los tipos de problemas tratados a menudo son llamados requisitos funcionales, sin embargo, el método reconoce la importancia de los aspectos no-funcionales, y se centra en la composición de las necesidades, para lo que combina contextos de problemas sencillos en marcos compuestos que representan el análisis realista de los problemas complejos.

Development/Management Model [39]. Esta propuesta separa RE en dos grupos de actividades (ver Figura 8): 1) el proceso o desarrollo de requisitos, 2) la gestión de requisitos.

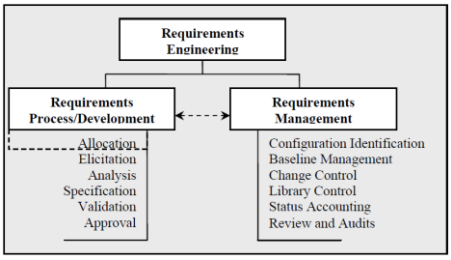

Figura 8. Requirements Development/Management Model

Las actividades del primero grupo tienen como objetivo asignar responsabilidades, involucrar a las partes interesadas en el proceso, y dividir los requisitos en grupos lógicos. Mientras que las del segundo se orientan a gestionar los requisitos mediante una serie de etapas. La aprobación de requisitos se utiliza para determinar si es rentable continuar con el modelo, o si lo mejor es reconceptualizarlo y aplicar sólo una parte del mismo.

AORE model [40]. Se puede considerar como un modelo general de procesos para la RE, cuyo objetivo es separar los requisitos aspectuales de los no-aspectuales, así como de sus reglas de composición, como se observa en la Figura 9. También describe una instanciación concreta utilizando puntos de vista y un lenguaje de composición basado en XML, junto con una herramienta de soporte llamada Arcade. En el enfoque Arcade, los requisitos aspectuales son similares a los propuestos por PREview, en el que los puntos de vista son transversales y se utilizan para elicitarlos. Ambos modelos están representados mediante un *framework* semi-estructurado basado en XML, que también se utiliza para definir las reglas de composición que emplean acciones informales y las operaciones que reflejan cómo los requisitos aspectuales afectan a los grupos de requisitos transversales.

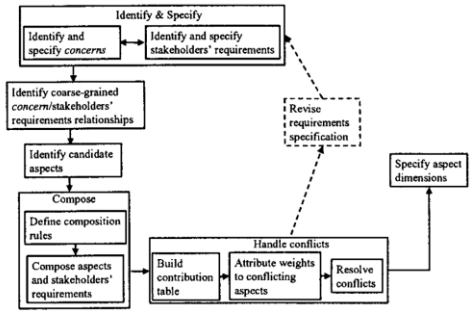

Figura 9. AORE Model

Las contribuciones más valiosas y novedosas de este enfoque son su capacidad para separar y luego componer requisitos transversales y no transversales, un conjunto flexible de operadores de composición, y una presentación original de requisitos en XML. También proporciona un procedimiento exhaustivo para identificar y resolver conflictos. El apoyo de Arcade para el mapeo de requisitos, aunque muy útil, carece del rigor y la minuciosidad de PREview, o el análisis de NFR.

Linear Iterative Requirements Engineering Process Model [41]. El modelo consta de tres fases principales: 1) elicitación, 2) especificación y 3) validación de requisitos, pero debido a su naturaleza iterativa, generalmente se realizan varias veces hasta la resolución, como se muestra en la Figura 10.

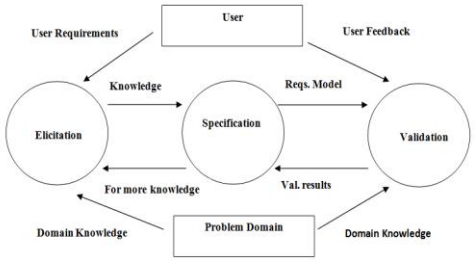

Figura 10. Linear Iterative Requirements Engineering Process Model

Este tipo de modelo de procesos es típico para el desarrollo de software, donde continuamente se producen y liberan nuevas revisiones del producto, lo que requiere procesos y periodos más

cortos. Además, el modelo incorpora comentarios e información de los usuarios, y por tanto, los problemas tienen una mejor posibilidad de ser evaluados antes de la validación. La principal diferencia entre esta propuesta y el modelo completamente lineal es que la fase de validación es iterativa, lo que significa que se puede realizar tantas veces como sea necesario, hasta que los requisitos sean validados por las partes interesadas y acuerden las especificaciones finales del sistema. Este proceso asegura una mejor validación de requisitos.

Requirements Abstraction Model (RAM) [42]. Este modelo (Figura 11) se estructura en tres pasos básicos: 1) *Especificar*, que implica especificar el requisito inicial y elicitar información suficiente al respecto para definir un número de atributos; 2) *Ubicar*, que se centra en averiguar el nivel de abstracción en el que se encuentran los requisitos especificados, y para esto propone los siguientes: nivel del producto, nivel de las características, nivel de las funciones, y nivel de los componentes; 3) *Abstraer*, en el que se hace un trabajo de depuración de cada requisito, e implica abstraer y/o desglosar los requisitos dependiendo de su ubicación en el paso anterior. La depuración consiste en crear nuevos requisitos, llamados requisitos depurados, sobre los niveles de abstracción adyacentes, o de vinculación en función de la situación existente.

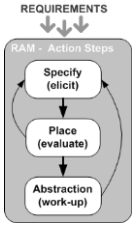

Figura 11. Requirements Abstraction Model (RAM)

V Model [43]. El Modelo V es una adaptación del Modelo Cascada utilizado en la ingeniería de sistema y de software, pero se diferencia principalmente en la gestión de las actividades de validación, pruebas, calidad y riesgos, que son manejadas mediante la adición de varias fases iterativas a la estructura del proceso. La intención es llevar a cabo procedimientos de prueba y verificación de forma continua para mejorar la calidad y minimizar los riesgos, como se observa en la Figura 12.

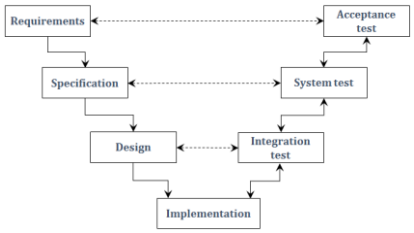

Figura 12. Model V

Las fases de desarrollo pueden ser vistas como diferentes capas, que progresan desde la etapa inicial de alto nivel bajando en el proceso hasta que se alcancen los niveles más bajos, donde se puede iniciar el trabajo de ejecución. En cada fase se re-validan los requisitos especificados; las fases del modelo pueden variar en función de las intenciones y el proyecto, pero el patrón básico permanece igual.

Rational Unified Process (RUP) [44]. RUP es un modelo de proceso de desarrollo de software que consiste de dos dimensiones de procesos (Figura 13): 1) de tiempo, que indica una subdivisión en una estructura en bruto (fases) y una estructura refinada (iteraciones), y 2) se refiere a la parte técnica, y la divide en disciplinas, de las cuales hay seis de proceso primario (incluyendo requisitos) y tres de infraestructura; además, cada disciplina tiene su propio flujo de trabajo definido.

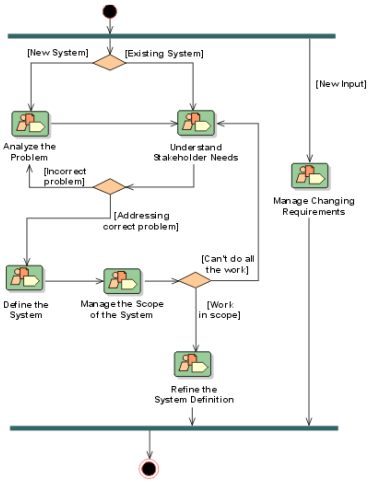

Figura 13. RUP Requirements Discipline

La disciplina de la Ingeniería de Requisitos tiene como objetivo la fiabilidad de las especificaciones y el desarrollo, así como las modificaciones al sistema. En esencia, la RE en RUP se compone de las siguientes actividades principales: 1) análisis del problema, 2) comprensión de las necesidades de las partes interesadas, 3) definición del sistema, 4) gestión del alcance del sistema, 5) cambios en los requisitos, y 6) refinamiento de la definición del sistema. Estas actividades están lógicamente conectadas entre sí, como se observa en la Figura 13, y no se deben considerar como puramente secuenciales.

Requirements Methodology for Agent Oriented Paradigm [45]. La Ingeniería de Requisitos implica los procesos de adquisición y de establecimiento de los requisitos finales de un sistema, y cuando se orienta a agentes modela los requisitos de un sistema en términos de las tareas y metas que varios agentes pueden exhibir. El enfoque orientado a agentes que propone este modelo (Figura 14) se basa en un modelo en espiral, para dar cabida a la RE dinámica que sugiere una versión limitada de requisitos para satisfacer las demandas competitivas o de negocios, pero los detalles de los requisitos exigen un modelo de proceso para dar cabida a un requisito de producto que evoluciona con el tiempo.

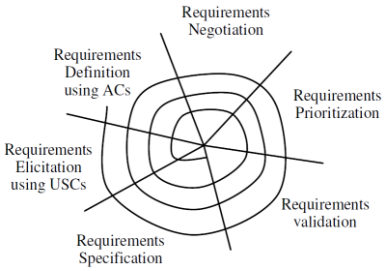

Figura 14. Requirements Methodology for Agent Oriented Paradigm

Cada ronda del ciclo captura requisitos adicionales y le facilita al desarrollador negociar, priorizar y validar los requisitos encapsulados. El proceso comienza con la elicitación de los requisitos en forma de historias de usuario. La información acumulada en User Story Card (USC) se elabora mejor en forma de Agent Card (AC), para promover una mejor comprensión del desarrollador de la viabilidad de los requisitos orientados a agentes. Los requisitos ampliados almacenados en las AC se priorizan con base en negociaciones, que se llevan a cabo entre las partes interesadas.

Goal Elicitation Method [46]. Como se observa en la Figura 15, este método básicamente comprende dos fases consecutivas: 1) elicitación preliminar, para recoger una primera versión del modelo, y 2) elicitación con catálogos, para complementar y refinar los modelos previamente derivados por medio de requisitos no-funcionales (NFR por sus siglas en inglés).

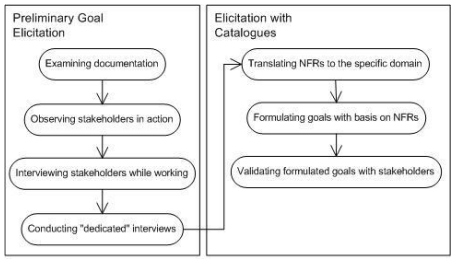

Figura 15. Goal Elicitation Method

De acuerdo con la fuente de información y las técnicas utilizadas para interactuar con los actores del proceso, la primera fase se divide en cuatro etapas: 1) se examina la documentación acerca de los procesos organizacionales, 2) se obtiene un modelo preliminar de objetivos junto con un modelo de procesos de negocio preliminar, 3) se centra en la elicitación de requisitos mediante entrevistas a los actores de la organización, mientras se observan sus acciones, 4) se concentra en entrevistas dedicadas, no sólo a los actores de procesos de negocio, sino también con los jefes de departamento. En la segunda fase se emplea el marco NFR [47-49], con el objetivo de abstraer y extrapolar una serie de requisitos no-funcionales para identificar objetivos que tienen relevancia estratégica para los modelos de procesos de negocio, y que no habían sido identificados previamente.

Volere Model [50]. El modelo Volere fue desarrollado especialmente para la Ingeniería de Requisitos y, además de técnicas para determinar requisitos, proporciona plantillas para estructurar la especificación de requisitos. Está organizado como se observa en la Figura 16. Este modelo les proporciona a los usuarios una plantilla de Ingeniería de Requisitos sistemática, estructurada y completa. Toda la información en ella se mantiene en un documento (monolítico), a la inversa de RUP. Además, con el fin de desarrollar los requisitos utiliza otra plantilla de requisitos. El aseguramiento de calidad es un paso intermedio, denominado, que se utiliza entre la especificación y el análisis de requisitos. El modelo también se debe considerar iterativo.

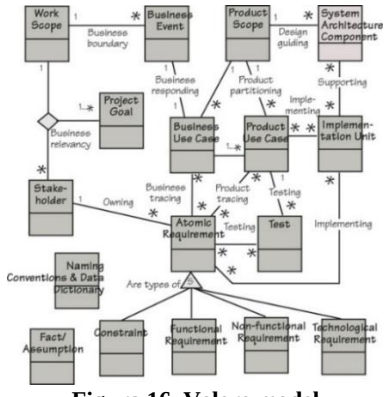

Figura 16. Volere model

4. ANÁLISIS DE RESULTADOS

La Ingeniería de Requisitos abarca varias actividades importantes del ciclo de vida de la Ingeniería de Software, como la elicitación, el análisis, la especificación, la resolución de conflictos y la validación. El objetivo principal de esta fase es especificar claramente las necesidades del cliente, de tal forma que los ingenieros de software obtengan un amplio conocimiento de las funciones, restricciones y propiedades del sistema a desarrollar, así como del entorno en el que funcionará. La comunidad de investigadores ha desarrollado y propuesto diversos modelos y enfoques para lograr este objetivo, y a pesar de que han sido parcialmente aplicados y validados, esos trabajos presentan inconvenientes, como que se basan en ejemplos relativamente simples, o que se llevan a cabo de forma aislada. De hecho, no reflejan un estudio empírico realizado en el campo que estudie de manera sistemática cómo evaluar y comparar la eficacia y eficiencia de los modelos. Además, tampoco ofrecen evidencias empíricas de las ventajas e inconvenientes de la práctica actual de la Ingeniería de Requisitos, por lo que no se basan en un marco de evaluación estructurado, lo que impide su desarrollo, ejecución y reproducción en otros casos de estudio.

En este trabajo se presenta un análisis a varios de los modelos divulgados, y la conclusión es que son diferentes, y a veces de naturaleza conflictiva, que van desde estructuras lineales e incrementales a cíclicas y reiterativas. Los resultados de estudios previos indican que en la práctica algunos difieren de lo comúnmente aceptado en la literatura [51, 52]. Además, esta complicación es situacional, y se ve afectada por la relación cliente-proveedor [31], por el producto, por la madurez técnica, y por la participación disciplinar y la cultura de la organización [34].

Aunque los procedimientos y la documentación de la mayoría de los modelos propuestos son clara y bien presentada, las validaciones experimentales relacionadas demuestran que el seguimiento paso a paso no es suficiente para gestionar adecuadamente la Ingeniería de Requisitos. Algunos alcanzan la especificación suficiente, pero no necesaria para comprender adecuadamente el problema; otros no documentan el proceso, y generan incomprensión de los requisitos; otros no estructuran el diálogo entre las partes, por lo que se debe reformular continuamente el proceso de elicitación. En general se puede concluir que ninguno de los modelos analizados es suficiente para gestionar adecuadamente y de forma estructurada la RE, y se necesita integrar las mejores prácticas de varios de ellos, complementadas con principios de otras áreas del conocimiento, para proponer un modelo de gestión y administración de requisitos.

Un nuevo modelo debería reunir las buenas prácticas encontradas, y potencializarlas con aportes desde otras áreas del conocimiento, como los métodos formales, la matemática discreta, la lógica, la abstracción, y otras relacionadas. Esto se debe a que los problemas actuales son más complejos y complicados que antes, la seguridad se ha convertido en un asunto clave, las modificaciones son más cotidianas, y los tiempos de entrega de los productos software se han acortado drásticamente, a la vez que los presupuestos se ajustan a su vida útil.

5. CONCLUSIONES

Las buenas prácticas encontradas en esta revisión se pueden utilizar en muchos aspectos en beneficio de un modelo integral para desarrollar y gestionar la Ingeniería de Requisitos. Valdría la pena realizar otras investigaciones para identificar un mayor número de buenas prácticas en otros modelos que no fueron cubiertos en esta revisión, o que no han sido publicados desde la industria, no obstante su uso es cotidiano. Aunque muchas buenas prácticas han sido identificadas en trabajos anteriores, todavía hay que determinar si podrían ser desplegadas en un modelo integral, porque se debe cumplir una serie de condiciones necesarias antes que puedan

aplicarse con éxito y ayudarles a los ingenieros de requisitos a realizar de mejor manera su labor.

Mientras se pueda identificar un mayor número de buenas prácticas se podrán descubrir o estructurar modelos más inteligentes, y pensar en la implementación de herramientas de automatización a través de la integración de niveles más altos de formalidad. Porque, actualmente, y como se puede observar en la presente revisión, las propuestas contienen muchas reglas que se deben ejecutar para lograr un especificación adecuada, y, aunque la mayoría proporciona soporte inteligente, su rendimiento decae significativamente al momento de aplicarlos. En tal caso, se requiere un mayor nivel de integración y de apropiación de las reglas inteligentes, se necesita más análisis de gestión, un mejor conocimiento del dominio de los requisitos, mejorar el diseño y la documentación de los requisitos elicitados, y conocer adecuadamente los hábitos de los usuarios y demás partes interesadas.

Un modelo para desarrollar y gestionar la Ingeniería de Requisitos debería reunir las buenas prácticas encontradas en esta revisión, y potencializarlas con aportes desde otras áreas del conocimiento, como los métodos formales, la matemática discreta, la lógica, la abstracción, y otras relacionadas. Esto se debe a que los problemas actuales son más complejos y complicados que antes, la seguridad se ha convertido en un asunto clave, las modificaciones son más cotidianas, y los tiempos de entrega de los productos software se han acortado drásticamente, a la vez que los presupuestos se deben ajustar a su vida útil. Además, el modelo se debe estructurar para hacerles frente a las debilidades encontradas en los propuestos hasta el momento, porque todavía no brindan un soporte inteligente para la Ingeniería de Requisitos, y aún no cubren todas las cuestiones en el desarrollo y gestión de mismos, sin embargo, los resultados iniciales son prometedores. En términos generales:

- Un modelo de Ingeniería de requisitos debe definir actividades que tengan como objetivo identificar requisitos, proporcionando actividades de soporte que contribuyan al éxito y a la calidad de los mismos.
- Para los ingenieros, la flexibilidad en la implementación y la integración de los métodos existentes influye en el éxito del enfoque que adoptan. La ausencia de asociación para notaciones y métodos particulares es un factor clave en la aceptación del enfoque.
- Al principio la introducción de nuevos conceptos puede causar problemas, pero una sesión introductoria, para que los usuarios se familiaricen con los conceptos, puede contribuir al éxito de la implementación del concepto.
- Se necesita un enfoque iterativo en Ingeniería de Requisitos para que los requisitos evolucionen. Sin embargo, se requiere una guía para determinar cuándo comienza o termina cada etapa de esta fase.
- Los ingenieros de requisitos deben buscar activamente los requisitos, en lugar de enfatizar en la elicitación desde fuentes conocidas. La elicitación puede ocurrir cuando es posible identificar la fuente.

6. REFERENCIAS

[1] Serna, M.E. (2012). Analysis and selection to requirements elicitation techniques. *Proceedings 7th Colombian Computing Congress*, pp. 1-7. Medellín, Colombia.

[2] Silva, R. and Dasilva, A. (2012). Elements of a requirements management tool to improve software development. *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS)*, 2(1), pp. 37-42

[3] Boehm, B.W. and Papaccio, P.N. (1988). Understanding and controlling software costs. *IEEE Transactions on Software Engineering*, 14 (10), pp. 1462-1477.

[4] Brooks, F. (1995). *The mythical man month*. Addison Wesley.

[5] Berry, D. and Lawrence, B. (1998). Requirements Engineering. *IEEE Software*, 15(2), pp.26-29.

[6] Leite, J. (1987). *A survey on requirements analysis*. Technical Report RTP-071, University of California at Irvine.

[7] Pfleeger, S. (1998). *Software Engineering – Theory and practice*. Prentice Hall.

[8] Stevens, R., Brook, P., Jackson, K. and Arnold, S. (1998). *Systems Engineering: Coping with Complexity*. Prentice Hall.

[9] Saiediana, H. and Daleb, R. (2000). Requirements Engineering: Making the connection between the software, developer and customer. *Information and Software Technology*, 42(6), pp. 419-428.

[10] Drehmer, D. and Dekleva, S. (2001). A note on the evolution of Software Engineering practices. *Journal of Systems and Software*, 57(1), pp. 1-7.

[11] Bell, T. and Taylor, T. (1976). Software Requirements: Are they really a problem? *Proceedings International Conference on Software Engineering*, San Francisco, USA.

[12] Madhavji, N., Holtje, D., Hong, W. and Bruckhaus, T. (1994). Elicit: A method for eliciting process models. *Proceedings 1994 CAS Conference*, Toronto, Canada.

[13] Lubars, M., Potts, C. and Richter, C. (1993). A review of the state of the practice in requirements modeling. *Proceedings IEEE International Symposium on Requirements Engineering*, pp. 2-14. San Diego, USA.

[14] El Emam, K. and Madhavji, N. (1995). A Field study of Requirements Engineering practices in information systems development. *Proceedings Second International Symposium on Requirements Engineering*, pp.68-80. York, England.

[15] Siddiqi, J. (1996). Requirement Engineering: The emerging wisdom. *IEEE Software*, 13(2), pp.15-19.

[16] McQuilten, G. (2007). *MIS-design: Art in a consumer landscape*. PhD Thesis, Melbourne University.

[17] Caldwell, J. (2009). Approach to Management Information System Design. Online [Dec 2014].

[18] Day, K. and Devlin, R. (1998). The payoff to work without pay: Volunteer work as an investment in human capital. *The Canadian Journal of Economics*, 31(5), pp. 1179-1191.

[19] Jalote, P. (2005). *An integrated approach to software engineering*. Narosa Publishing house, India.

[20] Asghar, S. and Umar, M. (2010). Requirement Engineering challenges in development of software applications and selection of customer-off-the-shelf (COTS) Components. *International Journal of Software Engineering*, 1(2), pp. 32-50.

[21] Weiss, R. (1997). *Strategic management and information systems: An integrated approach*. Financial Times.

[22] Wahono, R. (2003). Analyzing Requirements Engineering problems. *Proceedings IECI Japan Workshop*, pp. 55-58. Chofu Bunka.

[23] Hofmann, H. and Lehner, F. (2001). Requirements Engineering as a success factor in software projects. *IEEE Software*, 18(4), pp. 58-66.

[24] Young, R. (2001). *Effective requirements practices*. Addison-Wesley.

[25] Fernandez, D., Lochmann, K., Penzenstadler, B. and Wagner, S. (2011). A case study on the application of an artefact-based requirements engineering approach. *Proceedings 15th International Conference on Evaluation and Assessment in Software Engineering.* Durham University, UK.

[26] Pohl, K. (1994). The three dimensions of Requirements Engineering a framework and its applications. *Information systems*, 19(3), pp. 243-258.
[27] Pohl, K. (1996). *Process centred Requirements Engineering*. John Wiley & Sons Inc.
[28] Loucopoulos, P. and Karakostas, V. (1995). *System requirements engineering*, McGraw-Hill Book Company Europe.
[29] Davis, A. (1993). *Software requirements - Objects, functions and states*. Prentice Hall.
[30] Lauesen, S. (2002). *Software Requirements - Styles and techniques*. Addison-Wesley.
[31] Macaulay, L. (1996). *Requirements Engineering*. Springer-Verlag.
[32] Sawyer, P., Sommerville, I. and Viller, S (1996). *PREview: Tackling the real concerns of Requirements Engineerin*g. Technical Report: CSEG/5/1996. Lancaster University.
[33] Sommerville, I., Sawyer, P. and Viller, S. (1998). Viewpoints for requirements elicitation: A practical approach. *Proceedings 3rd International Conference on Requirements Engineering: Putting Requirements Engineering to Practice*, pp.74-81. Colorado Springs.
[34] Kotonya, G. and Sommerville, I. (1998). *Requirements Engineering – Processes and techniques*. John Wiley & Sons.
[35] Beck, H. (2000). *Extreme Programming Explained*. Addison-Wesley Reading.
[36] Balzert, H. (2008). *Lehrbuch der softwaretechnik: Softwaremanagement*. Spektrum Akademischer Verlag.
[37] Jackson, M. (2001). *Problem frames: Analyzing and structuring software development problems*. ACM Press.
[38] Gamma, E., Helms, R., Johnson, R. and Vlissdes, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.
[39] Ferdinandi, P. (2002). *A requirements pattern - Succeeding in the internet economy*. Addison-Wesley.
[40] Rashid, A., Moreira, A. and Araujo, J. (2003). Modularisation and composition of aspectual requirements. *Proceedings 2nd International Conference on Aspect Oriented Software Development*. Boston, USA.
[41] Kotonya, G. and Sommervile, I. (2003). *Requirements Engineering process and techniques*. Wiley.
[42] Gorschek, T. and Wohlin, C. (2006). Requirements abstraction model. *Requirements Engineering Journal*, 11(1), pp. 79-101.
[43] Rupp, C. (2006). *Requirements-Engineering und – management*. Hanser Fachbuchverlag.
[44] Passing, J. (2007). *Requirements Engineering in the Rational Unified Process*. Hasso Plattner Insitute for Software Systems Engineering. Germany.
[45] Gaur, V., Soni, A. and Bedi, P. (2010). An agent-oriented approach to Requirements Engineering. *Proceedings 2nd International Advance Computing Conference*, pp. 449-454.
[46] Cardoso, E., Almeida, J., Guizzardi, R. and Guizzardi, G. (2011). A method for eliciting goals for business process models based on non-functional requirements catalogues. *Inter. Jour. of Information System Modeling and Design*, 2(2), pp. 1-18.
[47] Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. (2000). *Non-Functional requirements in Software Engineering.* Kluwer Academic Publishers.
[48] Cysneiros, L. (2007). Evaluating the effectiveness of using catalogues to elicit non-functional Requirements. *Proceedings 10th Workshop in Requirements Engineering*, pp. 107-115. Toronto, Canada.
[49] Lamsweerde, A. (2000). Requirements Engineering in the year 00: A research. *Proceedings 22nd International Conference on Software Engineering*. Limerick, Ireland.
[50] Robertson, S. and Robertson, J. (2012). *Mastering the requirements process: Getting requirements right*. Addison-Wesley.
[51] Nguyen, L. and Swatmann, P. (2000). *Managing the Requirements Engineering process*. School of Management Information Systems, Deakin University. Geelong, Australia.
[52] Houdek, F. and Pohl, K. (2000). Analyzing requirements engineering processes: A case study. *Proceedings 11th International Workshop on Database and Expert Systems Applications*, pp. 983-987. Greenwich, UK.