Integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados

Autor: Ing. Mauricio Ortiz

Director: Doctor Leandro Antonelli

Co-Directora: Doctora Roxana Giandini

Tesis presentada para obtener el grado de Magister en Ingeniería de Software

Facultad de Informática

Universidad Nacional de La Plata

Abril 2015

Resumen

El presente trabajo de tesis, tiene como objetivo permitir la integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados. Esta integración, se logra mediante la formulación y planteamiento de un método que consta de dos etapas denominadas: "*identificación de sistemas legados*" e "*integración de BPM en el proceso de requerimientos*"; si bien la interrelación entre las dos etapas es la que permite la integración, se puede inclusive trabajar con cada una de ellas por separado.

La etapa de identificación de sistemas legados utiliza el cuadrante de análisis propuesto por Andrew Sage en lo referente a la especificación de prioridades de modernización y lo complementa mediante la formalización de las experiencias del autor en identificación y catalogación de aplicaciones. Por otro lado, la etapa de integración de BPM en el proceso de requerimientos presenta un enfoque innovador al concentrarse únicamente en la primera fase del proceso de software y plantear cuatro tipos de tareas BPMN2 relacionadas específicamente con el análisis del sistema legado en producción y no en el diseño de la solución como plantean la mayoría de autores investigados.

Como sustento teórico para la presentación del método, se recopila el estado del arte referente a tres líneas de investigación: sistemas legados, gestión por procesos de negocio y requerimientos de software. El método de integración propuesto en esta tesis, se encuentra fundamentado en dos pilares: en primer lugar, se utiliza el proceso de requerimientos de software sugerido en el Cuerpo del Conocimiento de la Ingeniería de Software y en segundo lugar, se utiliza la generación de modelos conceptuales a partir de notaciones BPMN2; estos modelos conceptuales se convierten en la base para el análisis de requerimientos de software de un sistema legado.

La aplicación del método propuesto se evalúa mediante un caso de estudio; el mismo que implementa la propuesta en integral, con la expectativa de identificar los sistemas legados y documentar de una manera formal los diferentes procesos de negocio que se encuentran asociados a las funcionalidades presentes en los sistemas legados.

Abstract

The aim of this Project is to integrate BPM with software requirements that will upgrade legacy systems. The integration is possible through the formulation and planning of a method made up of two stages called: "*identification of legacy systems*" and "*the integration of BPM in the requirements process*"; although the interrelation between these two stages is what allows the integration, they can be worked on separately.

The stage of identification of legacy systems using the quadrant analysis proposed by Andrew Sage regarding the specification modernization priorities and complemented by formalizing the author's experiences in identifying and cataloging applications. On the other hand, the stage of integration of BPM in the requirements process presents an innovative approach by focusing only on the first phase of the software process and propose four types of BPMN2 tasks, specifically related to the analysis of legacy system into production and not the solution design as posed most authors investigated.

As theoretical basis for the presentation of the method, there are three research lines: legacy systems, business process management and software requirements. The integration method proposed in this thesis is based in two mainstays: firstly, the use of software requirements process suggested in Software Engineering Body of Knowledge, and secondly, the creation of conceptual models from BPM2 notations; these conceptual models are the basis for the analysis of software requirements of a legacy system.

Another point worth highlighting is the application of the proposed method in a case study that implements the proposal completely in order to identify legacy systems and document the different business processes associated with the current functions in legacy systems.

Agradecimiento

A mi familia, quienes siempre confiaron en mí y que me dieron el apoyo y empuje necesario para que esta tesis se realizara de la mejor manera.

Al equipo de desarrollo de la Universidad Politécnica Salesiana, por su colaboración en el caso de estudio y por sus acertados comentarios y sugerencias a esta propuesta.

A toda la planta docente de la Maestría en Ingeniería de Software de la UNLP, en especial para Alejandra, Roxana, Silvia, Zulma, Alejandro, Federico, Gustavo, Javier y Leandro; quienes aparte de transmitir sus vastos conocimientos en cada una de sus áreas, supieron demostrar ese amor por lo que se hace, manifestando en cada una de sus intervenciones y evaluaciones una pasión, entusiasmo y responsabilidad envidiables.

A la Universidad Politécnica Salesiana, por su apoyo financiero en la realización de este magíster y sobre todo por la oportunidad de compartir mis conocimientos tanto en el área profesional así como en el área académica.

A Lourdes Mongou, por su eficaz, atenta y dedicada ayuda a la distancia.

Finalmente, un agradecimiento especial al Doctor Leandro Antonelli y a la Doctora Roxana Giandini por confiar en este tema y por guiarme con sus oportunos comentarios, correcciones y sugerencias a lo largo de este año de trabajo.

Control de Versiones

Fecha	Descripción	Ver.
2013-10-30	-Aceptación de la propuesta v 2.2	
2013-11-14	-1. Estructura Inicial de la tesis con información de la propuesta v 2.2	0.1
2014-01-27	-1. Introducción primera parte (Motivación y Objetivos) -2. Estado del arte de sistemas legados	0.2
2014-02-27	-Versión 0.2 revisada, corregida y aprobada -2. Estado del arte de gestión por procesos de negocio hasta BPMS 2.2.5	0.3
2014-05-09	-Versión 0.3 revisada y aprobada -2. Estado del arte de gestión por procesos de negocio	0.4
2014-05-17	-Versión 0.4 revisada y aprobada -2. Estado del arte de Requerimientos hasta Especificación 2.3.4	0.5
2014-06-04	-Versión 0.4 revisada y aprobada -2. Estado del arte finalizado	0.6
2014-07-02	- Versión 0.6 revisada y aprobada - Modificación de la presentación (viñetas, sangrías, espaciado) - 2. Estado del Arte finalizado - 3. Descripción del Problema finalizado	0.7
2014-08-12	- Versión 0.7 revisada y acogiendo sugerencias del Director - 3. Descripción del Problema finalizado - 4. Propuesta de la solución (esquema de secciones)	0.8
2014-08-27	- Versión 0.8 revisada y aprobada - 4. Propuesta de la solución (Introducción finalizada)	0.9
2014-10-05	- Versión 0.9 revisada y aprobada - 4. Propuesta de la solución (Etapa de identificación de sistemas legados finalizado)	0.10
2014-10-15	-Versión 0.10 revisada y acogiendo sugerencias del Director - 4. Propuesta de la solución finalizada.	0.11
2014-11-18	-Versión 0.11 revisada y acogiendo sugerencias del Director - 1. Resumen e introducción revisada y finalizada - 2. Estado del Arte revisado y finalizado - 3. Descripción del Problema revisado y finalizado - 4. Propuesta de la solución revisada y finalizada. - 5. Caso de estudio primera parte (hasta 5.4. Soluciones planteadas)	0.12
2014-12-02	-Versión 0.12 revisada y acogiendo sugerencias del Director - 5. Caso de estudio (hasta 5.6 Etapa de integración de BPM en el proceso de requerimientos)	0.14
2014-12-15	-Versión 0.12 revisada y acogiendo sugerencias del Director. - 5. Caso de estudio finalizado.	0.15
2014-12-30	-Versión 0.12 revisada y acogiendo sugerencias del Director. - 6. Conclusiones finalizadas.	1.0
2015-03-04	-Versión 1.0 revisada y acogiendo sugerencias de los directores	1.1
2015-03-09	-Versión 1.1 aumentando el resumen en inglés	1.2
2015-03-25	-Versión 1.2 modificando redacción del capítulo 1 y acogiendo sugerencias de los directores	1.3
2015-04-07	-Versión 1.3 aprobada por los directores	2.0

Tabla de contenidos

Índice de figuras

Índice de tablas

Glosario

- **BPM**: Gestión por procesos de Negocio.
- **RAE**: Real Academia Española.
- **SOA**: Arquitectura Orientada a Servicios.
- **MDA**: Arquitectura Dirigida por Modelos.
- **SoaML**: Lenguaje de Modelado para Arquitecturas Orientadas a Servicios.
- **UML**: Lenguaje de Modelado Unificado.
- **WSDL**: Lenguaje de Descripción de Servicios Web.
- **CIM**: Modelo Independiente de la Computación.
- **PIM**: Modelo Independiente de la Plataforma.
- **PSM**: Modelo Específico para la Plataforma.
- **BPMN**: Notación para Modelamiento de Procesos de Negocio.
- **OMG**: Grupo de Gestión de Objetos.
- **AOP**: Programación Orientada a Aspectos.
- **ISO**: Organización Internacional de Normalización.
- **IES**: Institución de Educación Superior.
- **XPDL**: Lenguaje de Descripción de Procesos en XML.
- **TI**: Tecnologías de la Información.
- **ERP:** Sistemas de Planificación de Recursos Empresariales.
- **CRM:** Gestión de Relaciones con Clientes.
- **SCM:** Gestión de Cadena de Suministros.
- **EPC**: Cadenas de Procesos controladas por Eventos.
- **BPEL**: Lenguaje de Ejecución de Procesos de Negocio.
- **SWEBOK**: Cuerpo del Conocimiento de la Ingeniería de Software.
- **SRS**: Especificación de Requerimientos de Software.
- **ITIL**: Biblioteca de infraestructura de tecnologías de información.
- **CMMI:** Integración de modelos de madurez de capacidades.

1. Introducción

Esta sección presenta al lector la motivación del autor para la elaboración de la tesis, además detalla el contexto, metodología y estructura utilizada para su realización.

1.1. Motivación

En la actualidad, existen en funcionamiento sistemas de información que fueron construidos hace décadas y por lo tanto, fueron desarrollados con tecnología que en estos tiempos se puede considerar como obsoleta. En este contexto, los nuevos requerimientos, cambios o mejoras se vuelven cada vez más costosas, ya que dichos sistemas se encuentran todavía en funcionamiento y juegan un rol crítico dentro del dominio del problema.

Con respecto a las organizaciones, muchas de ellas, por el desconocimiento o malas experiencias al tratar de reutilizar los sistemas legados con arquitecturas como SOA, prefieren modernizar sus aplicaciones ya sea reconstruyendo el sistema con una nueva tecnología de desarrollo o buscando implementaciones en el mercado que permitan solucionar sus requerimientos. Este tipo de decisiones por parte de las organizaciones, son reales y pueden devenir en la pérdida de todo el conocimiento de los procesos que se realizan en los sistemas legados y del conjunto de reglas de negocio que manejan.

Con la intención de mantener la información de los sistemas legados de una organización, antes que desaparezcan o sean reemplazados, se debe oportunamente adquirir todo el conocimiento e información posible de los mismos. Esta información debe servir para crear una especificación de requerimientos software; la misma que debería ser utilizada ya sea para desarrollar un nuevo sistema, modernizarlo o como base contractual para la compra de uno nuevo.

Basándose en la captura de procesos de negocio, se debe crear una especificación de requerimientos, para que la misma permita no solo reducir el gap existente entre los analistas de software y los analistas de negocios, sino también servirá para verificar los procesos actuales o en su defecto refinarlos para que a mediano plazo sirvan como base fundamental de conocimiento.

1.2. Objetivos

Objetivo General

Proponer un método que integre BPM[1] en el proceso de requerimientos de software para la modernización de sistemas legados.

Objetivos Específicos

- Estudiar el estado del arte relativo a los sistemas de información legados, gestión por procesos de negocio e ingeniería de requerimientos.
- Identificar sistemas con plataformas tecnológicas de implementación obsoletas y que necesiten una modernización.
- Proponer un método que permita especificar los requerimientos de sistemas legados como procesos de negocio; mediante la especificación se analizan los

[1] BPM: **B**usiness **P**rocess **M**anagement.

procesos, se podría decidir mejorar los mismos y por consiguiente cambiar la implementación para que se ajuste a los nuevos procesos.

- Utilizar el método propuesto mediante su aplicación en un caso de estudio.

1.3. Contribuciones

Como primera contribución de esta tesis, está la presentación de una manera objetiva e imparcial en la identificación de los sistemas legados de una organización, mediante la definición de un conjunto de parámetros que se convierten en la base para la creación de un catálogo que permitirá identificar cuáles sistemas se consideran legados y si dichos sistemas necesitan además entrar en un plan de modernización.

La identificación de requerimientos de software mediante procesos de negocio en sistemas legados conduce una mejor lectura de los analistas de negocio en torno a las funcionalidades actuales del sistema, quedando a su criterio, optimizar o no los procesos de negocio antes de proceder a una modernización.

Si bien SOA[2] permite evitar gastos en recursos y la reutilización de los sistemas legados, también es cierto que cuando la organización opta por una modernización completa, entonces no existe una metodología clara que permita obtener toda la información de los sistemas legados en notaciones de procesos de negocio (do Nascimento, et al., 2012).

Como se aprecia en la bibliografía, se ha recolectado muchos trabajos relacionados con la integración de SOA (Khadka, et al., 2012) y procesos de negocio (Bazan, et al., 2012) (Cruz, et al., 2013) (Hertis & Juric, 2013) (do Nascimento, et al., 2012), pero en dichos trabajos no se profundiza en la concepción de los requerimientos como tal, ya que lo que pretenden es resolver problemas a nivel de diseño, enfocándose en el *Cómo*. Mientras que el aporte creativo de la propuesta de esta tesis se centra en el *Qué*, es decir, mediante la utilización de BPM se puede comprender o analizar el funcionamiento actual de los sistemas legados y sus respectivas reglas de negocio. Esta integración de BPM en el proceso de requerimientos de software, se realiza específicamente en el nivel de elicitación y análisis, introduciendo modelos conceptuales representados en BPMN2.

1.4. Metodología

Un paso importante en la realización de este trabajo, es la identificación de los diferentes tipos de investigación que se realizarán. Razón por la cual, se ha planteado a la investigación exploratoria y a la investigación cualitativa como los tipos seleccionados en esta tesis.

Como primer punto, se plantea utilizar *investigación exploratoria* para determinar el estado del arte de los tres principales temas de investigación denominados: sistemas legados, gestión por procesos de negocio y proceso de requerimientos de software.

Por otro lado, se realiza una *investigación cualitativa*; la misma que permitirá inferir un método necesario para la integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados. La investigación cualitativa será utilizada por su carácter de flexibilidad, ya que da la pauta para recurrir a grandes líneas

[2] SOA: **A**rquitectura **O**rientada a **S**ervicios

de acción mediante la modificación del camino y los métodos según sea necesario (Cataldi & Lage, 2004).

Para la definición de la metodología propiamente dicha, se presenta a continuación el conjunto de actividades, técnicas y herramientas que se utilizarán para la presente investigación:

- Delimitar los temas de investigación exploratoria. En este caso, se plantean a los sistemas legados, gestión por procesos de negocio y proceso de requerimientos de software.
- Definir fuentes de investigación válidas: libros revisados por un equipo editorial, artículos científicos de revistas, artículos de congresos indexados y publicaciones académicas certificadas.
- Recolectar información adecuada y con un alto factor de impacto[3], sobre todo en lo relacionado a la fecha de publicación.
- Clasificar la información recolectada en los diferentes temas de investigación previamente definidos.
- Lectura crítica de las diferentes publicaciones y generación de resúmenes de lectura que permitan la sistematización de la información en el estado del arte.
- Relacionar los temas del estado del arte utilizando mapas conceptuales que permitan identificar los problemas existentes en una posible integración.
- Proponer el método y sus etapas utilizando diagramas de procesos que permitan relacionar sus actividades y que a su vez integren BPM en el proceso de requerimientos de software para la modernización de sistemas legados.
- Aplicar la integración inferida en un caso de estudio y así evidenciar la utilidad de la propuesta.

En lo concerniente al formato de citas, se ha identificado tres tipos que podrían servir en el presente trabajo. El formato APA (autor, año), el IEEE [numérico] y el Harvard (autor, año). De estos formatos, se ha escogido como estilo por defecto al Harvard, considerando que admite una citación más resumida cuando se trata de artículos científicos de conferencias.

1.5. Estructura de la tesis

El presente trabajo de tesis se divide en seis capítulos detallados a continuación:

- **Introducción**: Este capítulo presenta al lector la motivación del autor para la elaboración de la tesis, además detalla el contexto, metodología y estructura utilizada para su realización.
- **Estado del Arte**: En el contexto de esta tesis, el estado del arte es visto como la base teórica de los diferentes componentes de investigación que se utilizarán en el planteamiento del método de integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados. Esta investigación detallada, consiste en la recolección, categorización, análisis y citación de libros, tesis y artículos científicos relacionados con las diferentes secciones propuestas en el tema

[3] **Factor de impacto**: Es un valor cuantitativo que representa o mide la importancia de una publicación.

de tesis. Por tal motivo, este capítulo se ha dividido en tres secciones: sistemas legados, gestión por procesos de negocio y requerimientos de software.

- **Descripción del problema***:* Este capítulo establece la justificación para la realización del presente trabajo de tesis. Es decir, es aquí en donde se presenta una serie de argumentos relacionados con la necesidad de integrar BPM en el proceso de requerimientos de software para la modernización de sistemas legados. La descripción del problema se ha dividido en cuatro partes: introducción, identificación de sistemas legados, necesidades de modernización en sistemas legados y el rol de los requerimientos en la modernización.
- **Propuesta de la solución**: En este capítulo, se presenta la propuesta del autor para solventar ciertos problemas referentes a los sistemas legados, especificar sus requerimientos y responder a las limitaciones de ciertos modelos conceptuales para representar procesos dentro del análisis. Para plantear dicha solución, se presenta un método dividido en dos etapas: la primera permite la identificación de un sistema legado y la segunda presenta las acciones propuestas para analizar y especificar requerimientos utilizando modelos conceptuales con BPMN2.
- **Caso de estudio**: Este capítulo es un aporte adicional a la propuesta del capítulo anterior, ya que presenta una experiencia concreta de la aplicación del método propuesto tanto en la etapa de identificación de sistemas legados, así como en la etapa de integración de BPM en el proceso de requerimientos. El contexto de aplicación de este método, gira en torno a los sistemas informáticos de la Universidad Politécnica Salesiana del Ecuador[4].
- **Conclusiones:** Son enunciados o descripciones breves que pretenden resumir y explicar en ciertos casos los puntos principales de la tesis. Además, en este capítulo se plantea futuras líneas de investigación que podrían ser derivadas de este trabajo.

[4] **Universidad Politécnica Salesiana**: Es una institución de educación superior humanística y politécnica, de inspiración cristiana con carácter católico e índole salesiana.

2. Estado del Arte

En el contexto de esta tesis, el estado del arte es visto como la base teórica de los diferentes componentes de investigación que se utilizarán en el planteamiento del método de integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados. Esta investigación detallada, consiste en la recolección, categorización, análisis y citación de diferentes libros, tesis y artículos científicos relacionados con las diferentes secciones propuestas en el tema de tesis. Por tal motivo, este capítulo se ha dividido en tres secciones: sistemas legados, gestión por procesos de negocio y requerimientos de software.

Cabe destacar, que aunque las tres primeras secciones se tratarán por separado; las mismas mantendrán un lineamiento general que permita al lector entender la propuesta de integración que el autor plantea en capítulos posteriores.

2.1. Sistemas Legados

En muchos casos, los trabajos relacionados con sistemas legados se pueden presentar de una manera sesgada al dominio de la aplicación, (Méndez, et al., 2012) lo que significa que las propuestas de modernización del mismo se adaptan a un software y en algunas ocasiones a hardware específico (Sneed & Erdoes, 2013). Esta situación, se da porque en un inicio, y a falta de proceso definido de software, los diferentes analistas, diseñadores, programadores, gerentes de proyecto, etc. creaban software de forma inadecuada, ya que poseían entre otras deficiencias, poca o ninguna documentación, carecían de estándares para el proceso de software y se implementaban mediante paradigmas o lenguajes de programación que en la actualidad se consideran obsoletos. En esta primera sección, se irá presentando un conjunto de perjuicios asociados a los sistemas legados, aunque, no todo es desventaja en este tipo de sistemas, ya que de hecho, al mantenerse tanto tiempo en funcionamiento, significa que cumplen o se han adaptado al dominio del problema, resistiéndose en algunos casos a desaparecer por razones de índole técnicas, procedimentales o sociales (Dedeke, 2012).

En ciertos casos, la falta de capacitación o la resistencia al cambio de los diferentes actores, hace que los sistemas no se actualicen. (Petter & Ward, 2013) En este contexto, los mismos jefes de proyecto o del departamento pueden mantener dudas respecto a una modernización del sistema, ya que tienen mucha experiencia en el sistema legado y especulan que también sus capacidades de dirección, gestión o mantenimiento queden obsoletas en conjunto con el mismo.

2.1.1. Definiciones

Sistemas Legados es una palabra compuesta en donde Sistema hace referencia al *"Conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto."* (Real-Academia, 2011) Esta definición se ha extrapolado en la informática para definir un conjunto de componentes de hardware y aplicaciones de software que interrelacionados procesan información. De esta forma, quedaría definido satisfactoriamente el término sistemas, pero la traducción de legado en español dista mucho de lo que se quiere representar, ya que la RAE[5] define a legado como una herencia o *"Aquello que se deja o transmite a los sucesores, sea cosa material o inmaterial."* (Real-Academia, 2011). Este concepto es el más cercano, pero en la realidad no brinda la característica de obsolescencia que se pretende conceptualizar. Lo más probable es que el uso de la palabra "legado" se haya generado de una versión españolizada del término *legacy*, que en ingles tiene una definición de adjetivo *"que denota o se relaciona con el software o hardware que debe ser sustituido, pero es difícil de reemplazar debido a su amplio uso"* (Oxford-English, 2012).

Con la aparición del modelo cascada para el desarrollo de software propuesto por Winston W. Royce en 1970, los investigadores comienzan a tomar conciencia de los problemas que ha generado un proceso de desarrollo inadecuado, y es por tal motivo que a inicios de los ochenta (Lehman, 1980) (Canning, 1984) comenzaron trabajos importantes para contrarrestar las implicaciones de contar dentro de la organización con sistemas legados.

Con el antecedente expuesto en el párrafo anterior y teniendo en cuenta que en cualquier contexto, un sistema actual podría terminar convirtiéndose en legado, hace que la

[5] RAE: **R**eal **A**cademia **E**spañola

investigación sobre el tema continúe y que en los noventa surjan definiciones ampliamente utilizadas hasta la actualidad considerando a un sistema legado como *"cualquier sistema de información que de manera significativa se resiste a la modificación y la evolución"*. (Brodie & Stonebraker, 1995)

Otra definición bastante difundida en los diferentes artículos relacionados al tema, especifica que un sistema legado "*Es un software fundamental que no puede ser modificado de manera eficiente*" (Gold, 1998)

Trabajos con definiciones actuales no han sido lo suficientemente difundidos ya que el término sistema legado se ha convertido en un estándar, al punto de que el diccionario Inglés de Oxford (Oxford-English, 2012) tiene una definición bastante clara; la misma que fue transcrita en párrafos anteriores.

Por otro lado, el autor Ian Sommerville, más que una definición emite una reflexión que se puede resumir como la o las aplicaciones de diez o veinte años de antigüedad que se han convertido en esenciales, ya que cualquier tipo de falla en estos sistemas tendría un serio efecto en la gestión diaria del negocio y en el mantenimiento de los servicios de la organización (Sommerville, 2011).

La mayoría de autores citados, coinciden en un conjunto de características inherentes a la obsolescencia que un sistema debería poseer para considerarse como legado. Dentro de estas características se encuentran:

- Aplicaciones complejas difíciles de mantener por su extensión en líneas de código.
- La documentación no mantiene estándares que permitan realizar el seguimiento de la evolución funcional del sistema, situación que conduce a mantenimientos excesivos que terminan por corromper la estructura inicial de la aplicación (Blunden, 2012).
- Sistemas escritos en lenguajes legados con paradigmas lineales o estructurados como Cobol, Ensamblador, Fortran, RPG, etc.
- Usualmente aplicaciones de más de diez años de existencia y que mantienen su funcionamiento porque cumplen una función crítica dentro de la organización.
- Aplicaciones monolíticas con demasiada autonomía y su consecuente falta de interacción con otros sistemas.
- En algunos casos, las aplicaciones dependen del hardware en donde fueron instaladas y por lo tanto no se pueden escalar o no se adaptan a nuevos equipos.

2.1.2. Clasificaciones de sistemas legados

La clasificación es una manera eficiente de organizar el conocimiento sobre un tema específico, sobre todo cuando se trata de temas extensos y dispersos. Este es el caso de los sistemas legados, por lo que es pertinente identificar las clasificaciones que han venido realizando explícita o implícitamente algunos autores en sus diferentes publicaciones.

En la revisión bibliográfica realizada, se ha podido clasificar a los sistemas legados de tres formas: clasificación mediante generaciones de lenguajes (Langer, 2012), mediante nivel de acoplamiento (Massari & Mecella, 2002) (Gomez, 2010) y clasificación por tipos de migración (Matos & Heckel, 2009) (Salvatierra, et al., 2012).

Clasificación por generaciones de lenguajes

Esta clasificación (Langer, 2012) relaciona directamente al sistema legado con la generación del lenguaje de programación utilizada para su desarrollo.

- ***Primera Generación:*** En esta generación se encuentran las aplicaciones desarrolladas en lenguaje máquina; el mismo que no utilizaba código binario para enviar instrucciones al hardware, sino que un comando representaba directamente una acción.

  Es sumamente raro encontrar software legado de este tipo, ya que eran aplicaciones más en el ámbito de la investigación que aplicaciones críticas de negocio.

- ***Segunda Generación:*** Al ser el lenguaje máquina muy complejo para los primeros desarrolladores, esta segunda generación es caracterizada porque los sistemas comienzan a utilizar lenguajes ensambladores; los mismos que permiten trabajar con comandos de un nivel más alto y que se podían traducir a lenguaje máquina.

  Estos sistemas todavía se pueden encontrar embebidos en antiguas mainframes que se utilizan para realizar complicados algoritmos de cálculo.

- ***Tercera Generación:*** Con esta generación aparecen los primeros lenguajes de alto nivel como Cobol, Fortran, C, RPG[6], etc. Estos lenguajes tienen comandos muy parecidos al lenguaje natural, en el sentido de que se aplicó como un estándar de facto al idioma inglés para identificar los comandos de alto nivel utilizados en estos lenguajes de programación.

  Con esta generación, los diferentes lenguajes se comenzaron a especializar en dominios específicos, ya sea para la parte de gestión de datos como Cobol o en el manejo de cálculos matemáticos y científicos en Fortran. En el caso de este último, la complejidad de la lógica de los cálculos ha hecho que grandes aplicaciones legadas se mantengan en la actualidad, sobre todo en el ámbito de predicciones climáticas. (Méndez & Tinetti, 2011) (Méndez, et al., 2012)

  En lo que concierne a la modernización de este tipo de aplicaciones, los esfuerzos deberían enfocarse al análisis del proceso de negocio y en los datos, ya que lo más seguro es que no se disponga de una documentación inicial o de un control de cambios.

- ***Cuarta Generación:*** La mayoría de los sistemas legados que se mantienen funcionando en la actualidad, han sido desarrollados con lenguajes pertenecientes a esta generación: Visual Basic, C++, PowerBuilder, Delphi, etc. Son lenguajes de programación que permiten al desarrollador utilizar interfaces de usuario más amigables y en algunos casos herramientas de generación de código.

  El impacto que tuvieron estos lenguajes se debe en gran medida a la utilización de las bases de datos relacionales, que si bien comenzaron en la tercera generación; es en esta, que los diferentes lenguajes de consulta a bases de datos comienzan a converger en un estándar que perdura hasta la actualidad y que se denomina Lenguaje Estructurado de Consulta o SQL por su acrónimo en inglés.

- ***Quinta Generación***: Esta clasificación de lenguajes: Prolog, Haskell, OPS5[7], Mercury, etc. Es utilizada para sistemas expertos o con inteligencia artificial. Estas técnicas

[6] **RPG**: **R**eport **P**rogram **G**enerator
[7] **OPS5**: **O**fficial **P**roduction **S**ystem v5

permiten al usuario plantear el problema para que el sistema elija la solución adecuada. Este procedimiento de resolución no se adecúa al control del procesamiento de la información que están acostumbradas las organizaciones, razón por la cual estos lenguajes no han sido populares en el ámbito de los negocios y de las soluciones empresariales.

Clasificación por nivel de acoplamiento

Cuando Massari & Mecella (Massari & Mecella, 2002) definieron esta clasificación, lo hicieron tomando en cuenta el nivel de acoplamiento de la información de los sistemas. El nivel de acoplamiento puede resultar relativo en muchas situaciones, por lo que esta clasificación identifica tres capas diferentes dentro de un sistema: presentación, lógica del negocio y acceso a datos.

En resumen, mientras mayor acoplamiento tenga cada una de las capas, más compleja será la modernización. Es decir, mientras más dependientes sean las partes que constituyen un sistema informático, entonces será mayor la complejidad en la metodología de modernización.

- ***Monolíticos**:* Estos sistemas no poseen ningún tipo de estructura y fusionan las tres capas como un solo bloque.

  Contrastando con la clasificación por generaciones, se podría decir que este tipo de sistemas son desarrollados con lenguajes de primera y de segunda generación, razón por la cual es difícil encontrar todavía en producción sistemas heredados de este tipo.

- ***Desacoplamiento de programas**:* Los sistemas legados agrupados en este tipo fusionan en un solo bloque la capa de acceso a datos y la capa de lógica de negocio, quedando libre la capa de presentación. Estos sistemas, no acceden directamente a los datos, por lo que necesitan funciones específicas que permitan realizar las transacciones. En esta categoría se puede catalogar a la mayoría de los sistemas legados.

- ***Desacoplamiento de datos**:* Esta categoría identifica a sistemas legados que fusionan en un solo bloque la capa de lógica y la capa de presentación, mientras que el acceso a los datos se lo hace de forma independiente. En estos sistemas legados se puede acceder directamente a la lógica del negocio, pero no a los datos.

- ***Altamente desacoplados**:* Los componentes de las aplicaciones son independientes gracias a que poseen interfaces bien definidas para la interacción de cada una de las capas.

  En la actualidad, comienzan aparecer los sistemas legados categorizados de esta manera, porque generalmente las versiones del framework con los que fueron desarrollados cambian de versión constantemente. Esta situación, puede generar que la omisión de una actualización haga que ciertas librerías desaparezcan en la versión siguiente y por lo tanto el sistema comience a fallar por los cambios o bugs detectados y corregidos entre una y otra versión.

Clasificación por tipos de migración

Esta categoría se ha definido mediante la migración que necesitan o que se planea para modernizar los sistemas legados. En este caso, hay dos puntos de vista, el de Matos &

Heckhel (Matos & Heckel, 2009) que hablan de una migración funcional cuando han cambiado los requerimientos o de una migración tecnológica cuando la plataforma tecnológica ha quedado obsoleta. Por otro lado, pero en la línea de las migraciones tecnológicas se presenta una clasificación implícita (Salvatierra, et al., 2012) denominada migración directa y su contraparte la migración indirecta.

- ***Migración funcional***: Los sistemas que necesitan de este tipo de migración se caracterizan porque la lógica del negocio ha cambiado por situaciones legales o reglamentarias para nuevos usuarios, pero también se necesita mantener la lógica legada para usuarios anteriores. Esta situación, es la más compleja que puede suceder, ya que se necesita de un sistema multiprocesos que dependiendo de las entradas realice el nuevo proceso o el proceso legado.
- ***Migración tecnológica***: En este caso, los sistemas han quedado obsoletos en la parte tecnológica por varias situaciones como: Omisión de actualizaciones del framework en que ha sido desarrollada la aplicación, incumplimiento de la empresa que realizó el desarrollo inicial, falta de personal capacitado en el software base de desarrollo, incompatibilidad con nuevo hardware más eficiente, etc. Todos estos escenarios hacen que aunque la lógica y funcionalidad no hayan cambiado se tenga que migrar a una nueva plataforma tecnológica por eficiencia y rendimiento.
- ***Migración indirecta***: Este tipo de migración propone el reemplazo de la aplicación legada y la reestructuración de una nueva aplicación con plataformas tecnológicas actuales.

  Realizar este procedimiento tiene un alto costo, tanto desde el punto de vista de recursos como del tiempo necesario para su puesta en marcha y posterior migración de información de un sistema a otro.
- ***Migración directa***: Esta migración propone mantener en funcionamiento los sistemas legados y continuar utilizando sus procedimientos, funciones, datos, etc. pero con interfaces de conexión a nuevos sistemas. Es decir, los sistemas legados continúan funcionando pero utilizan envoltorios, generalmente servicios web para que los nuevos sistemas puedan acceder a la información y lógica de negocio de los sistemas legados.

  Esta forma de migración, hace que la empresa tenga nuevas funcionalidades en menos tiempo y con un costo considerablemente bajo, pero teniendo en cuenta que en algún momento se deberá hacer una migración indirecta.

2.1.3. Metodologías de modernización

Otro punto de vista para el estudio de los sistemas legados es la profundización en las diferentes metodologías utilizadas o propuestas para su modernización.

En lo concerniente a terminología, se podría tender a una ambigüedad entre el término de metodología y técnica; estos términos pueden ser dependientes pero finalmente diferentes, ya que una metodología es un conjunto de pasos, procedimientos y políticas que persiguen un objetivo, mientras que las técnicas de modernización son medios utilizados para la aplicación de la metodología.

Dentro de las técnicas de modernización se puede encontrar: análisis de documentación, migración, reingeniería (Perez-Castillo, et al., 2011), reemplazo, redesarrollo, envoltorios, análisis de código fuente, análisis del dominio, etc.

Esta investigación, ha clasificado cinco grupos de metodologías de modernización: mediante SOA, a través de trasformaciones MDD, mediante procesos, a través de metodologías genéricas y finalmente utilizando metodologías personalizadas para un dominio específico. (Khadka, et al., 2012)

Modernizaciones mediante SOA

Este tipo de modernizaciones (Khadka, et al., 2011) (Baghdadi & Al-Bulushi, 2013) (Zhang, et al., 2010) (Xu, 2010) (Cruz, et al., 2013) tienen como denominador común el utilizar servicios web como envoltorios para sistemas legados.

El procedimiento de envoltura o wrapping propuesto en (Baghdadi & Al-Bulushi, 2013) constituye cuatro pasos resumidos en:

1. *Análisis y comprensión del sistema legado*: Es la parte más importante y más compleja del proceso, ya que se debe llegar a conocer la funcionalidad y determinar si es estrictamente necesario exponer el sistema legado como un servicio web.
2. *Extracción de las reglas del negocio* (do Nascimento, et al., 2012): La extracción de las reglas de negocio, significa localizar declaraciones de código simples y compactas que permitan la definición de los diferentes aspectos del negocio.
3. *Selección de la técnica de envoltura adecuada*: Esta elección dependerá del tipo de información que se necesite de los sistemas legados, ya que se puede acceder directamente a datos mediante transacciones o permitir el acceso a procedimientos y funciones.
4. *La construcción y ensamblaje del servicio web*: El servicio web no realizará ningún proceso de negocio, ya que se debe limitar a recibir entradas para que el sistema legado procese los datos y entregue información de salida.

El método SERVICIFI (Khadka, et al., 2011) propone seleccionar un conjunto de servicios que permitan ensamblarse en uno solo mediante los siguientes pasos:

1. *Iniciación del Proyecto*: Es en esta fase donde se define los objetivos que se persigue, la viabilidad del proyecto y el plan de modernización.
2. *Identificación de los servicios*: Es el análisis de la situación del sistema legado, así como la identificación de los servicios candidatos y su posterior priorización.
3. *Especificación de los servicios*: En esta fase se detallan y se eligen de entre los diferentes servicios candidatos, quienes se implementarán en la siguiente fase.
4. *Construcción y testeo de los servicios*: Mediante diferentes técnicas de programación se puede extraer el código fuente del sistema legado para construir los servicios web. Además, es en esta fase donde se construye los diferentes casos de prueba tanto funcionales como no funcionales.
5. *Despliegue, monitoreo y gestión*: Finalmente, se debe definir la estrategia que se utilizará para el uso tanto de la información técnica, así como la información

necesaria para quienes consumirán los servicios. Esto significa que se debe crear un catálogo de los diferentes servicios, así como definir un control de versiones.

Otro concepto o técnica utilizada en los sistemas legados es la minería de datos. Estudios sobre el tema (Zhang, et al., 2010) manejan la técnica de la minería de secuencias como un caso particular de la minería de datos para obtener conocimiento del sistema legado y así modernizarlo. Dentro de este contexto son utilizados los siguientes enfoques SOA:

- ***Enfoque de caja negra*:** Este enfoque (Canfora, et al., 2008) es el más típico, ya que envuelve la funcionalidad del sistema legado mediante un servicio web. En primer lugar, se debe analizar las funcionalidades heredadas para luego definir cuáles de ellas serán implementadas con servicios web. Este tipo de enfoque hace que los sistemas sean difíciles de evolucionar, ya que satisfacen necesidades a corto plazo pero complican el mantenimiento y la gestión del sistema a largo plazo.
- ***Enfoque de la lógica del negocio***: La recuperación de la lógica del negocio es la base en este enfoque. Esto significa que mediante técnicas como la reingeniería, análisis del dominio, análisis del código fuente o la misma minería de datos se logra recuperar las reglas o lógica del negocio, para que como segundo paso y mediante un sistema de servicios web se permita la implementación de la lógica obtenida. El punto débil de este enfoque se refleja en que a pesar de reducir el mantenimiento posterior, la recuperación de la lógica es un proceso desgastante, que necesita de una gran inversión inicial tanto en tiempo como en recursos y costes.
- ***Enfoque de caja gris*:** Es un enfoque que integra o combina los dos enfoques anteriores, haciendo que cierta lógica sea recuperada, pero que algunas partes del sistema legado sigan funcionando mediante interfaces de servicios web.

  En la actualidad y en la praxis, este tipo de enfoque de caja gris es el más utilizado como parte de las modernizaciones SOA, ya que la complejidad de la obtención de la lógica y reglas del negocio, puede hacer que ciertas partes deban ser envueltas mediante servicios web que provean interfaces al sistema legado mediante un enfoque de caja negra.

La modernización mediante SOA se ha convertido en una de las más populares en el ámbito de los sistemas legados. En esta sección, se hizo referencia a tres estudios, pero en investigaciones sobre el tema, se ha detectado que en la primera década del dos mil se han definido más de cien estudios representativos sobre el tema (Khadka, et al., 2012).

Modernizaciones mediante Transformaciones MDA

La Arquitectura dirigida por modelos relacionada en varios trabajos (Zhang, et al., 2009) (Huang & Chu, 2012) se conceptualiza como una referencia o marco de trabajo, donde la idea principal es separar al sistema en dos partes bien definidas: la primera es las especificaciones, modelos o lógica del negocio y por otro lado se presentan las capacidades de la plataforma tecnológica en donde es implementada la lógica. (Funes, et al., 2012)
Las transformaciones son la base en la arquitectura MDA[8], ya que dependiendo de los tipos de modelamiento, se podrá ir definiendo cada vez mejor los niveles de detalle de un sistema,

[8] MDA: **A**rquitectura **D**irigida por **M**odelos

pudiendo tener transformaciones automáticas a partir de los modelos más abstractos, hasta llegar al código fuente de la aplicación. (Nahuel, et al., 2012)

En este marco, se han presentado algunos trabajos relacionados a la modernización de sistemas legados mediante transformaciones MDA. En uno de ellos, (Zhang, et al., 2009) se ha definido una metodología con los siguientes pasos:

1. *Representación arquitectónica del sistema legado*: Como primer paso se debe reconstruir un modelo arquitectónico de la aplicación legada; este modelo genérico debe estar basado en el código fuente, reglas de negocio, dominio de la aplicación y en lo posible de la documentación existente.
2. *Rediseño del modelo de arquitectura*: El modelo original es transformado en un modelo para servicios web; el mismo que no distingue entre proveedores y consumidores de servicios. Para la creación del modelo se utiliza SoaML[9], que es el lenguaje estándar de arquitectura orientada a servicios y que forma parte de la especificación MDA.
3. *Transformación MDA*: Con las definiciones de modelos MDA se puede transformar fácilmente de modelos SoaML o UML[10] a código fuente para la especificación de servicios web en WSDL[11].
4. *Generación de Servicios Web*: Una vez que se tenga el servicio web en WSDL, se puede generar automáticamente su implementación.
5. *Invocación en servicios web de funcionalidades legadas*: Como último paso, se realiza la invocación y por lo tanto, se mantiene la funcionalidad de los sistemas legados.

En la misma línea, pero utilizando modelos más abstractos Huang & Chu (Huang & Chu, 2012) proponen una metodología ágil e iterativa de modernización basada en SOA y MDA. La metodología implica cuatro pasos:

1. *Análisis del sistema legado y conceptualización de las funcionalidades mediante un modelo CIM*[12]: Este tipo de modelos, son modelos de dominio que están expresados de tal forma que puedan ser entendidos por diferentes usuarios y desde diferentes puntos de vista.
2. *Construir modelos PIM*[13] *acorde al modelo CIM*: Mientras que en los modelos CIM se representa funcionalidades genéricas, con modelos PIM se puede representar estructuras de datos más precisas, pero independientes de las características de la plataforma de implementación.
3. *Construir modelos PSM*[14] *acorde al modelo PIM*: Los modelos PSM amalgaman las especificaciones PIM con los detalles necesarios para describir y utilizar una plataforma tecnológica específica.

[9] SoaML: **L**enguaje de **M**odelado para **A**rquitecturas **O**rientadas a **S**ervicios.
[10] UML: **L**enguaje de **M**odelado **U**nificado.
[11] WSDL: **L**enguaje de **D**escripción de **S**ervicios **W**eb
[12] CIM: **C**omputation **I**ndependent **M**odel
[13] PIM: **P**latform **I**ndependent **M**odel
[14] PSM: **P**latform **S**pecific **M**odel

4. *Generación de código*: A partir de modelos PSM se puede generar el código necesario para la implementación de las funcionalidades legadas que se especificaron en el primer paso.

Las modernizaciones mediante transformaciones MDA no son utilizadas con mucha frecuencia, o mejor dicho no se presentan muchas propuestas formales de este tipo en comparación con la arquitectura SOA.

En principio, al ser MDA una arquitectura abstracta, se puede interpretar que otras metodologías de modernización como SOA utilizan implícitamente el concepto de MDA, ya que siempre representan funcionalidades legadas con modelos UML (Ortiz & Plaza, 2013), BPMN[15], WSDL, etc. antes de implementar los diferentes servicios.

Modernizaciones mediante Procesos de Negocio

Este tipo de modernizaciones (do Nascimento, et al., 2009) (De la Vara, et al., 2009) se enfocan o parten de un análisis del dominio de la aplicación. En este sentido, la técnica común que utilizan para realizar el análisis son los procesos de negocio.

En un primer criterio, (Bazán, 2009) (Bazan, et al., 2012) se hace una aproximación a la integración de SOA y BPM. Esta integración, en realidad no está pensada para sistemas legados, sino más bien para la implementación de sistemas mediante SOA. Esta metodología, se podría extrapolar a la modernización de sistemas legados, ya que los pasos a seguir son similares a las modernizaciones SOA de los apartados anteriores:

1. *Identificación y Especificación de Requerimientos*: En esta metodología, la identificación de requerimientos está pensada como la fase inicial del ciclo de vida del proceso de software.

   Dentro de las técnicas para la identificación del dominio, la autora propone la utilización del diseño participativo de procesos.
2. *Modelado del Negocio*: En esta etapa, se identifican los diferentes procesos, actividades, flujos, actores y restricciones del negocio. Para favorecer la comprensión del negocio en esta fase, se puede utilizar la técnica de los casos de uso, ya sea mediante una descripción textual o definiendo los diferentes diagramas con sus respectivos ámbitos y límites.
3. *Modelado de Procesos*: Mediante procesos de negocio se podría representar los diferentes requerimientos del sistema.

   BPMN es la notación a utilizar para el modelamiento de los diferentes procesos. En el caso de que no sea suficiente la información del diagrama de procesos, se pueden detallar los mismos mediante formularios que registren información de los objetos y de las actividades relacionadas.
4. *Modelado de Servicios*: En esta fase se define, especifica y categoriza los diferentes servicios. Además, es aquí en donde se podría refinar los procesos de negocio especificados anteriormente.
5. *Definición de los Componentes*: Los componentes son un conjunto de servicios empaquetados que poseen una entrada y que pueden generar varias salidas. Definir

[15] BPMN: **B**usiness **P**rocess **M**odeling **N**otation

los componentes implica verificar que mantengan interfaces estándar de interoperabilidad.

6. *Implementación de los Componentes*: Una vez definido los componentes, se debe poner en marcha el sistema utilizando los diferentes estándares tecnológicos del mercado.

Para que esta metodología se pueda utilizar en sistemas legados, se la podría extender para que la primera fase de identificación y especificación de requerimientos se realice mediante un conjunto de técnicas y herramientas utilizadas para obtención del dominio de una aplicación legada.

Otra metodología de modernización para sistemas legados utilizando procesos de negocio (do Nascimento, et al., 2009) involucra un conjunto de pasos englobados en dos grandes fases:

1. *Definición e identificación de procesos de negocios*: Esta fase se subdivide en varios pasos que se enumeran a continuación: 1.- Definir el alcance que tiene el proceso, 2.- identificar claramente los eventos de inicio y fin del proceso, 3.- identificar cuáles son las actividades humanas o manuales que no se pueden automatizar, 4.- identificar las actividades necesarias, pero que no se encuentran como parte del sistema legado, es decir, los nuevos requerimientos previos a la modernización, 5.- identificar los roles para la ejecución de actividades humanas, 6.- identificar actividades automáticas y 7.- validación del modelo de proceso obtenido.

   En resumen, esta fase genera un modelo de proceso que consistirá en un conjunto de diagramas de proceso detallados a partir de las necesidades del negocio y de los diferentes subprocesos, actividades, flujos, actores y restricciones que se vayan identificando.

2. *Implementación de los procesos de negocio*: En primer lugar, se debe escoger una herramienta BPM adecuada para implementar los procesos de negocio. La elección puede depender de varios factores como: tipos de licencia, documentación disponible, facilidad de uso, popularidad, etc.; pero lo que no debe entrar en discusión es que la herramienta soporte la notación en la cual fueron diseñados los procesos. Con respecto al estándar de notación, en la actualidad existe el BPMN de la OMG[16]; el mismo que se ha convertido en la notación más utilizada y que se explica en detalle en lo posterior.

   Una vez identificados los procesos y la herramienta para su implementación, entonces se procede con la ejecución de los mismos. Con respecto a la ejecución, existen herramientas que automatizan directamente (Castillo, 2011) los procesos y que obtienen funcionalidades de sistemas externos o sistemas legados mediante la creación de servicios web.

   Finalmente, las actividades que son parte del proceso y que deben ser ejecutadas de forma manual por parte del usuario, son implementadas como formularios generados automáticamente por la herramienta BPM en la mayoría de los casos. Dentro de las actividades manuales pueden encontrarse el ingreso de datos, configuración de reportes o cambio de estados.

[16] OMG: **O**bject **M**anagement **G**roup

Las modernizaciones mediante procesos de negocio están teniendo una gran popularidad en la actualidad, sobre todo porque existe un conjunto importante de herramientas BPM que transparentan al usuario la parte técnica del sistema, para que éste se concentre en definir, redefinir o refinar los procesos de negocio mediante notaciones estándares.

Modernizaciones Genéricas

Las diferentes metodologías de modernización pueden ser vistas de una forma personalizada al dominio. Pero, desde el punto de vista operativo, se puede definir o integrar cualquier modernización en un proceso genérico: (Langer, 2012)

1. Analizar los sistemas legados existentes.
2. Descomponer al sistema legado, para determinar calendarizaciones de migración o estrategias de vinculación con nuevos sistemas.
3. Diseñar vínculos con otros sistemas.
4. Diseñar mejoras para el sistema legado.
5. Diseñar reemplazos de partes o del sistema legado en integral.
6. Diseño e integración de nuevas bases de datos.
7. Determinar la nueva infraestructura necesaria.
8. Implementar las mejoras del sistema legado.
9. Implementar los vínculos entre sistemas.
10. Migrar las bases de datos legadas.
11. Migrar las aplicaciones legadas que serán reemplazadas.

En la Figura 2.1, se aprecia el proceso de interacción entre las diferentes actividades enumeradas anteriormente:

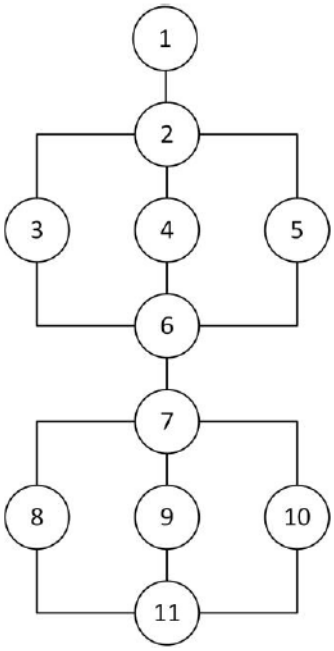


Figura 2.1: Proceso genérico de modernización; *(Langer, 2012)*

El Análisis de Portafolio, (Dedeke, 2012) más que una metodología de modernización es una herramienta que permite determinar el tipo de mantenimiento que debería tener cada uno de los activos de software de la organización.

En la matriz de la Figura 2.2, se clasifica a los diferentes activos de software en cuatro cuadrantes que se determinan mediante dos ejes principales:

Valor Técnico: El valor técnico de un activo de software puede ser definido cualitativamente o cuantitativamente. De cualquier forma, se debe tener en cuenta parámetros como:

- Fiabilidad: Se refiere a la continuidad del servido prestado por el sistema.
- Costos de mantenimiento: Implica las erogaciones financieras necesarias para mantener al sistema en funcionamiento.
- Calidad de características: Un sistema con características de alta calidad podría ser modificable o funcionalmente adecuado a los requerimientos y reglas de negocio.
- Factor de degradación: Este factor dependerá de la escala que se fije la organización. Mientras más alto sea el factor de degradación, menor será el valor del sistema.

Valor de Negocio: Para determinar el valor de negocio de un software se debe medirlo en función de un conjunto de parámetros detallados a continuación:

- Ventaja Competitiva: En qué nivel el activo de software puede ayudar en el aprovechamiento de las oportunidades del mercado.
- Impacto de rentabilidad: Este parámetro indica cómo se comportan los costos de mantenimiento dentro del presupuesto.
- Potencial de crecimiento: La escalabilidad, adaptabilidad son los criterios principales dentro de este parámetro.
- Interdependencia con otros sistemas: Obedece al porcentaje de aplicaciones que dependen del sistema legado en el ámbito de procesamiento funcional o de datos.
- Seguridad: Dependerá de la robustez del sistema con respecto a las diferentes amenazas de seguridad.

Cada activo puede tener un valor alto o bajo en cada eje. De esta forma, si un activo de software tiene un valor de negocio alto y un valor técnico bajo, entonces se puede considerar la modernización del activo. Como otra posibilidad de relación, pero siguiendo la misma matriz, si un activo tiene un valor técnico bajo y un valor de negocio bajo, entonces se debe considerar el reemplazo total del sistema legado.

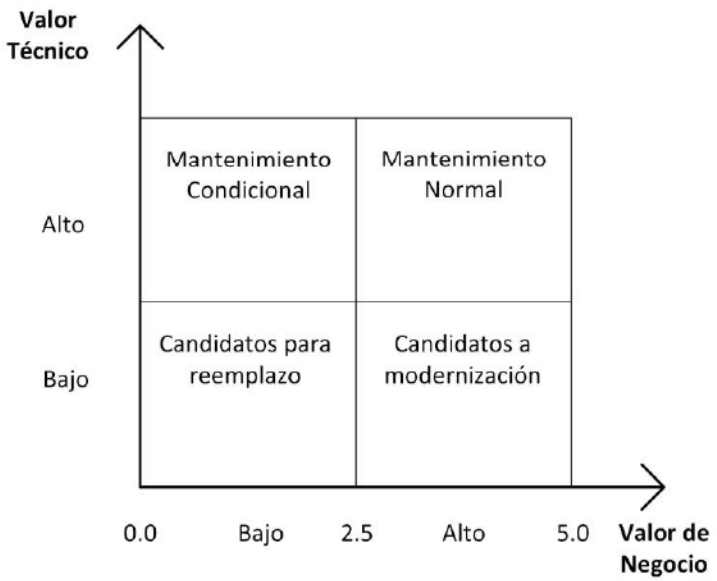


Figura 2.2: Matriz de análisis de portafolio; *(Dedeke, 2012)*

Modernizaciones Personalizadas

Existe un amplio conjunto de modernizaciones personalizadas o dependientes del dominio. Es más, todas las metodologías detalladas anteriormente, se adaptan o mantienen un sesgo con respecto al caso de estudio que realizan. Dentro de este amplio abanico de posibilidades, se ha definido dos personalizaciones representativas:

- ***Aplicaciones del clima***: Las aplicaciones legadas relacionadas al dominio del clima han tenido una gran cobertura, ya que desde un inicio el área de estudio definió exactamente modelos matemáticos probados relacionados a predicciones o lecturas del clima.

  El hecho de que el dominio de la aplicación se haya registrado de una manera eficaz (Méndez & Tinetti, 2011), ha provocado que los científicos no se preocupen en un principio de la actualización de la plataforma tecnológica; esta situación deriva en que cada vez más la plataforma tecnológica vaya quedando obsoleta frente a todas las oportunidades que los avances de las ciencias de la computación están brindando en la actualidad, como nuevas visualizaciones, sistemas distribuidos, interconexiones, etc. (Conover, et al., 2013)

  La mayoría de estas aplicaciones, están definidas en Fortran, razón por la cual autores como Barnes & Jones (Barnes & Jones, 2011) han propuesto una metodología basada en reescritura del código fuente, sin una reestructuración profunda de la definición de lógica de negocio.

  Como pasos necesarios para reestructurar aplicaciones del clima desarrolladas en Fortran se propone:

  1. Correr la aplicación desarrollada en Fortran para entender el algoritmo y dominio del problema.
  2. Almacenar los datos de entrada y salida para hacer pruebas y validación del código reescrito.
  3. Volver a escribir el sistema entero en otro lenguaje.

4. Considerar la posibilidad de corregir el algoritmo analizado.
5. Extender al software con visualizaciones, interfaces web, etc.

- ***Refactorizaciones orientadas a aspectos*:** Una forma interesante de metodologías de modernización de sistemas legados (Mortensen, et al., 2012) es mediante la utilización de la Programación orientada a aspectos (AOP[17]); la misma que se considera como un paradigma relativamente nuevo que trata de ver al sistema de una forma modular, identificando y separando los aspectos funcionales de los aspectos no funcionales como distribución, persistencia, replicación, sincronización, manejo de errores, etc. Para resumir, un Aspecto es la identificación y agrupación de conceptos y funcionalidades comunes que se encuentran repartidos en varias clases, pero que en sí no son parte de ellas. Como ejemplo, se sabe que los objetos deben mantener la persistencia de datos, aunque las anotaciones que utiliza Java no son parte ni de las clases, ni de la lógica del negocio (Ortiz & Plaza, 2013).

  Las funcionalidades que se encuentran diseminadas en distintas partes del sistema y que se pueden implementar como un aspecto se denominan asuntos transversales o "*crosscutting concerns*" en inglés.

  Lo que se pretende con este tipo de metodología es refactorizar los sistemas legados utilizando los siguientes pasos:

  1. Identificar los asuntos transversales del sistema legado que se puedan refactorizar como un aspecto.
  2. Implementar el aspecto.
  3. Eliminar el código correspondiente a los asuntos transversales de la aplicación.
  4. Llevar a cabo las pruebas de integración de la aplicación refactorizada.

Las distintas personalizaciones que se pueden hacer de una modernización dependerán del dominio y sobre todo de los conocimientos a nivel operativo y de las decisiones en tiempo y recursos a nivel gerencial.

[17] AOP: **A**spect **O**riented **P**rogramming

2.2. Gestión por Procesos de Negocio

En esta sección, se presentan los procesos de negocio como una forma de gestión de las actividades administrativas u operativas asignadas a una persona dentro de una organización.

La forma de gestión de una organización, generalmente puede realizarse de dos formas: una gestión funcional o una gestión por procesos de negocio. La administración tradicional se enfoca en una *gestión funcional*. Es decir, al empleado se le asigna varias funciones inherentes a su cargo. Por ejemplo, a un Coordinador de desarrollo se le asigna las funciones de planificar, analizar, diseñar, implementar y poner en marcha un proyecto de desarrollo de software, mientras que a un Administrador de base de datos se le asigna las funciones de respaldar, monitorear y mantener la disponibilidad de la base de datos.

Como se aprecia en el ejemplo anterior, parecen actividades diferentes, pero se sabe que éstas, necesariamente se entrelazarán si el sistema a desarrollar posee una base de datos. En este caso, el principal problema que acarrea la gestión funcional es no poder conocer la interacción entre actividades o cuáles se podrían realizar de forma paralela o secuencial, ya que en el papel, cada responsable se hace cargo de sus funciones por separado.

Los *procesos de negocio*, vienen a integrar la administración de la organización como un conjunto de procesos y subprocesos que interactúan entre sí. De esta forma, en un proceso como "Desarrollo de software" se podrán incluir varias actividades que interactúen entre sí y que sean asignadas a varios actores y en diferentes tiempos.

En el ambiente empresarial y tecnológico, la gestión por procesos de negocio es conocida e identificada popularmente con el acrónimo de ***BPM*** por sus siglas en inglés. Incluso, en el título de la presente tesis se utiliza esta sigla y se da por sentado que se comprenderá el contexto general.

2.2.1. Definiciones

Para una mejor profundización, se desacoplará el tema de gestión por procesos de negocio en diferentes conceptos; los mismos que al ser definidos por separado, permitirán luego precisar un concepto sólido de BPM a partir de premisas particulares de procesos, procesos de negocio y gestión.

Desde el punto de vista del idioma español, los conceptos de procesos que la RAE menciona[18], no abarcan el contexto que se pretende en este apartado. Por tal motivo, para iniciar con las definiciones se utiliza un estándar ISO[19] muy popular como el denominado ISO-9000:2004. Dicho estándar, presenta o está compuesto de un conjunto de normas que tratan sobre la calidad y su gestión. En este marco, se presentan y se ha extraído tres conceptos (ISO, 2004) relacionados con la temática a tratar:

- ***Proceso*:** Conjunto de actividades mutuamente relacionadas o que interactúan; las cuales transforman elementos de entrada en resultados. Las entradas de un proceso pueden ser las salidas de otros procesos.

[18] Proceso: (Del lat. processus) Acción de ir hacia adelante / Transcurso del tiempo / Conjunto de fases
[19] ISO: **I**nternational **O**rganization for **S**tandardization

- ***Procedimiento***: Forma especificada de llevar a cabo un proceso.
- ***Producto / Servicio:*** Resultado de un proceso que utiliza un procedimiento que puede o no estar documentado.

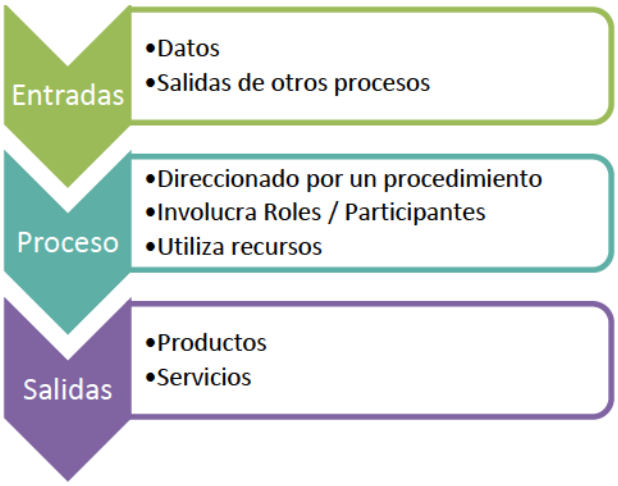


Figura 2.3: Proceso (ISO, 2004)

Una vez que el concepto de proceso está claramente definido, como segundo paso se introducirán los conceptos de Procesos de negocio. Esta vez, desde el punto de vista organizacional, concibiendo a una organización como un conjunto de instalaciones y personas que pueden desenvolverse en el ámbito público o privado y que poseen una disposición de responsabilidades, autoridades y relaciones.

Abordando un poco de historia, la humanidad siempre ha visto la necesidad de crear productos o servicios que se puedan ofrecer a un cliente. Es así, que antes de la revolución industrial, cualquier producto o servicio era considerado único y en ciertos casos hasta irrepetible, ya que las actividades necesarias para su construcción no eran registradas, ni gestionadas. Por otro lado, lo único considerado como fundamental era el producto y la satisfacción del cliente frente a las funcionalidades de éste.

A inicios del siglo XX y con la creciente necesidad de mejorar la gestión de la creación de productos o servicios a gran escala, varios empresarios buscan maneras de aumentar la productividad creando cadenas de montaje como la implementada por Henry Ford en 1908. Es en esta época, que comienzan a aparecer esfuerzos para normalizar los productos y procesos de construcción, pero en realidad es luego de la segunda guerra mundial que aparecen los primeros conceptos y criterios de calidad que no solamente piensan en el producto final sino en su *proceso de construcción*.

Con estos antecedentes, se aprecia que la calidad se encuentra íntimamente ligada con los procesos (Beimborn & Joachim, 2011) y por lo tanto, los principales autores de calidad han extrapolado el control de calidad de procesos como una forma integral de servicios organizacionales denominados *procesos de negocio*. A continuación, se presenta la definición de Davenport:

- ***Proceso de Negocio***: "Un conjunto estructurado y medible de actividades diseñadas para producir un producto especificado para un cliente o mercado específico. Implica un fuerte énfasis en cómo se ejecuta el trabajo dentro de la organización, en contraste

con el énfasis en el qué, característico de la focalización en el producto" (Davenport, 1993).

Desde un punto de vista más integral de proceso de negocio y reflejando una primera aproximación de sus componentes (Bazan, et al., 2011), se puede detallar que un *proceso de negocio* es una unidad persistente de trabajo que se inicia mediante la activación de un hecho o suceso de importancia dentro del negocio denominado evento. Al activarse un evento, el proceso es direccionado mediante reglas de negocio que permiten la ejecución de tareas y subprocesos, a los cuales se les asignan recursos; los mismos que se definen como unidades organizativas capaces y autorizadas para desempeñar papeles específicos en los procesos. (Dayal, et al., 2001)

Para ejemplarizar un proceso de negocio, se presenta el caso de la *Admisión a una Institución de Educación Superior* que se rige al siguiente proceso: En primer lugar, el estudiante se inscribe en la carrera de su elección, luego debe rendir un examen de admisión. Si lo aprueba, el aspirante podrá continuar con otro proceso denominado matriculación, pero de no aprobar el examen tiene la posibilidad de realizar un curso de nivelación; el mismo que de ser reprobado provocaría la finalización del proceso principal, caso contrario al aprobar el curso de nivelación automáticamente el aspirante es admitido y puede proceder con el proceso de matriculación.

A primera vista y examinando el proceso desde un punto de vista textual parece engorroso, pero según la teoría es un claro proceso de negocio ya que posee características como: actividades, interacciones, subprocesos, eventos y participantes.

El mismo proceso del ejemplo descrito textualmente, se puede representar gráficamente para que sea más legible y comprensible o también se puede representar en lenguajes que permitan a ciertas herramientas automatizar o simular la implementación del proceso. En este caso, en la Figura 2.4 se representa el proceso mediante una notación gráfica estándar denominada BPMN2 (De la Vara, 2011) y en la Figura 2.5 se utiliza el lenguaje XPDL[20] para la representación del proceso como un flujo de trabajo en formato XML.

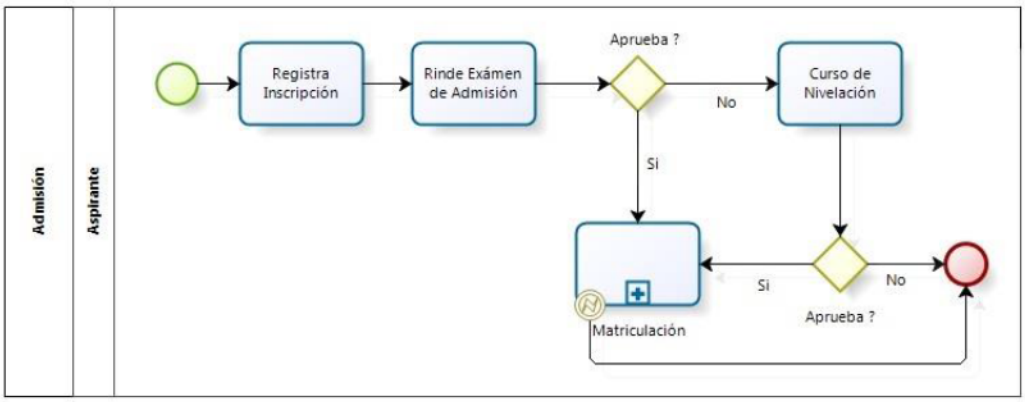


Figura 2.4: Proceso de Admisión a una IES[21] representado en BPMN

[20] XPDL: **X**ML **P**rocess **D**efinition **L**anguage
[21] IES: **I**nstitución de **E**ducación **S**uperior

```
<Pool Id="971f1450-0074-488c-8d84-0e79d1483fdb" Name="Main Process" Process="2344289d-95e9
  <Lanes />
  <NodeGraphicsInfos>
    <NodeGraphicsInfo ToolId="BizAgi_Process_Modeler" Height="0" Width="0" BorderColor="-1
      <Coordinates XCoordinate="0" YCoordinate="0" />
    </NodeGraphicsInfo>
  </NodeGraphicsInfos>
</Pool>
<Pool Id="96260b7f-218d-4083-8eda-e041c1d84f9b" Name="Admisión" Process="a70aa50f-070d-4d1
  <Lanes>
    <Lane Id="141f3895-9740-48f0-bd5c-c2ec6d8fac1e" Name="Aspirante" ParentPool="96260b7f-
      <NodeGraphicsInfos>
        <NodeGraphicsInfo ToolId="BizAgi_Process_Modeler" Height="285" Width="693" BorderC
          <Coordinates XCoordinate="50" YCoordinate="0" />
        </NodeGraphicsInfo>
      </NodeGraphicsInfos>
      <ExtendedAttributes />
    </Lane>
  </Lanes>
  <NodeGraphicsInfos>
    <NodeGraphicsInfo ToolId="BizAgi_Process_Modeler" Height="285" Width="743" BorderColor
      <Coordinates XCoordinate="40" YCoordinate="20" />
    </NodeGraphicsInfo>
  </NodeGraphicsInfos>
</Pool>
```

Figura 2.5: Proceso de Admisión a una IES representado en XPDL

Como se puede apreciar en la Figura 2.4 y Figura 2.5, un proceso de negocio puede representarse de varias formas; las mismas que se convierten en métodos que ayudan a los involucrados en el manejo de los procesos de negocio desde muchos puntos de vista. Partiendo de esta premisa, también existen otros conceptos, métodos, técnicas, herramientas, etc. que permiten ayudar con la representación y análisis de procesos de negocio. Por esta razón, el autor Mathias Weske define que:

- ***Gestión por procesos de negocio (BPM)***: "Es un conjunto de conceptos, métodos y técnicas que dan soporte al diseño, administración, configuración, mejora y análisis de procesos de negocio". (Weske, 2012)

Para finalizar, BPM provee una serie de beneficios tanto desde el punto de vista organizacional, así como desde el punto de vista del analista del negocio. En cuanto a la organización, BPM provee beneficios como: efectividad, eficiencia, consistencia, productividad, ahorro y calidad (Noguera, 2011). Mientras que para el analista la ventaja radica en que al poseer un mayor grado de abstracción, esta característica permitirá trabajar independientemente de la plataforma y por ende permitir agilizar el proceso de desarrollo e identificar posibles errores en fases tempranas.

2.2.2. Ciclo de Vida de los Procesos de Negocio

En la sección de definiciones, se presentó una relación bastante fuerte entre los procesos de negocio y los conceptos de calidad. Es por tal motivo, que en esta sección también se habla de un ciclo de vida de los procesos de software (Pourshahid, et al., 2008) bastante similar al ciclo de la calidad propuesto por la Gestión de la Calidad Total (Yaxiong, et al., 2008). En este marco, se puede presentar un ciclo de vida con cinco etapas principales (Magliano, et al., 2013): diagnóstico, diseño, implementación, ejecución y monitoreo. Este punto de vista es bastante básico, razón por la cual en la Figura 2.6 se presenta un ciclo de vida más detallado y apegado a nuestra propuesta. Este ciclo provee cinco etapas iterativas:

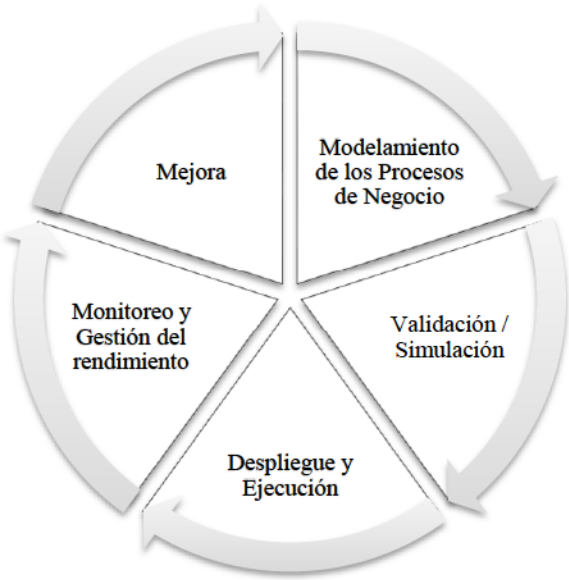


Figura 2.6: Ciclo de vida de los procesos de negocio

- ***Modelamiento de Procesos de Negocio****:* En esta fase, y como primera acción se tiene que definir quienes intervienen o intervendrán en las diferentes actividades del proceso. Además, es importante que se defina el nivel de relevancia de los diferentes involucrados en la toma de requerimientos. Una vez identificados los involucrados, el analista deberá proceder a la toma de requerimientos sobre el proceso y modelarlos de tal suerte que luego puedan ser validados.
- ***Validación / simulación****:* Una vez modelados los procesos de negocio, se necesita validarlos de alguna manera. Para esto se debe tomar en cuenta las diferentes sugerencias de los involucrados en el orden de relevancia, sabiendo que el dueño del proceso será el principal responsable de la validación. Por otro lado, y dependiendo de la herramienta que se utilice para la modelación de los procesos se puede realizar una simulación de los mismos.
- ***Despliegue y Ejecución****:* Los procesos de negocio que han sido correctamente validados, deben ser desplegados y ejecutados en esta fase. Es por tal motivo, que para esta etapa es recomendable utilizar herramientas que permitan la automatización de los procesos de negocio, aunque esto no significa sea la única manera de implementarlos, ya que mediante diferentes tipos de herramientas y arquitecturas se pueden desarrollar aplicaciones que permitan el despliegue y ejecución de los procesos de negocio, sobre todo si estos procesos son demasiado transaccionales y sin muchos cambios de estados.
- ***Monitoreo y gestión del rendimiento****:* En esta fase, se deben crear estrategias a nivel organizacional que permitan obtener datos, reportes o consultas que sirvan para profundizar el monitoreo de los procesos de negocio implementados y sobre todo verificar que se estén cumpliendo los principios de eficiencia y productividad.
- ***Mejora****:* El aumento de la productividad o la mejora del proceso debe ser la respuesta a un problema observado durante la fase de monitoreo y gestión del rendimiento. En la fase de mejora, se pueden realizar las mediciones o cuantificar los resultados del desempeño de la fase de seguimiento. La principal razón para llevar un marco de medición de los resultados, es la definición de un repositorio de patrones de diseño de

procesos de negocio que permita ayudar a la organización en la construcción y mantención de una base de conocimiento de resolución de problemas genéricos que pueden resolverse de forma similar.

2.2.3. Patrones de Procesos de Negocio

En el contexto de esta tesis, la palabra patrón define a un conjunto de situaciones o hechos recurrentes que pueden ser detallados con anterioridad para la resolución de problemas futuros. Luego, los patrones de procesos de negocio permiten a los analistas identificar situaciones análogas con relación a los procesos de negocio y que se resuelven de la misma forma. Por ejemplo, durante una solicitud de movilización es necesario reservar en paralelo: hotel, auto y vuelo, para luego entregar los viáticos al empleado. Las tres primeras tareas se pueden ejecutar en paralelo, pero a su vez todas deben completarse para que se proceda con la siguiente actividad. Para resolver esta situación, se debe utilizar el patrón de *sincronización*; el mismo que da la posibilidad de esperar a que un conjunto de ramas entrantes se completen antes de continuar con la siguiente actividad.

El profesor Van der Aalst (Aalst & Hofstede, 2012) es uno de los principales estudiosos sobre el tema y ha definido 21 patrones de negocio organizados en seis categorías como lo muestra la Tabla 2.1. Explicar todos y cada uno de los patrones resultaría bastante extenso, más que complejo. Por esta razón, se presentará una breve descripción de cada categoría y además se detallará la forma de identificación del patrón de Sincronización en la Tabla 2.2, para dejar a cargo del lector la profundización sobre los demás patrones.

Tabla 2.1: Patrones de Procesos de Negocio *(Dumas, et al., 2005)*

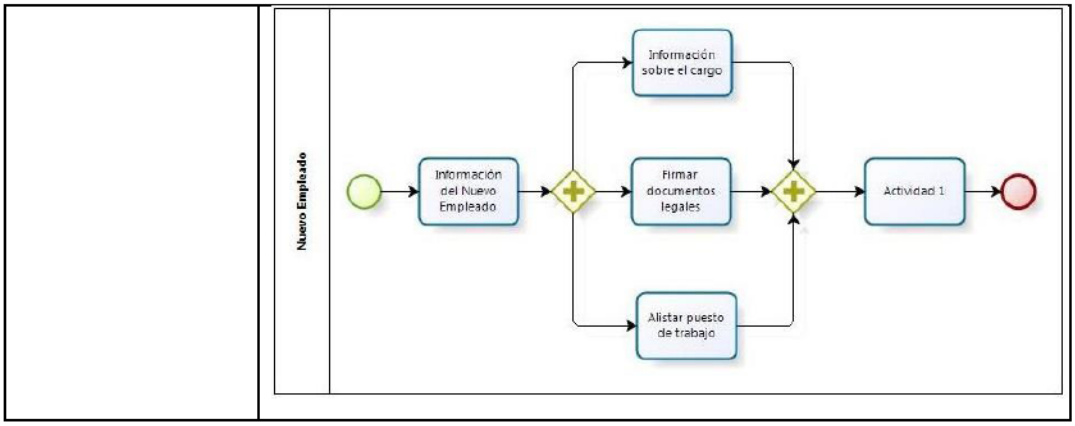
Control básico de flujo: -Secuencia. -División paralela. -Sincronización. -Decisión exclusiva. -Unión simple.	**Avanzados para ramificación y sincronización:** -Elección múltiple. -Estructura de unión sincronizada. -Unión múltiple. -Discriminador estructurado.
Iteraciones: -Ciclos arbitrarios. -Bucle estructurado. -Recursividad.	**Cancelación:** -Cancelación de actividad. -Cancelación del caso.
Basados en estados: -Elección diferida. -Milestone. -Enrutamiento entrelazado paralelo.	**Múltiples instancias:** -Múltiples instancias sin sincronización. -Múltiples instancias con conocimiento a priori del número de ejecuciones en la etapa de diseño. -Múltiples instancias con conocimiento en el tiempo de ejecución.

	-Múltiples instancias sin conocimiento en el tiempo de diseño.

- ***Patrones de control básico de flujo***: Estos patrones presentan los aspectos básicos del flujo de procesos que modelan aspectos secuenciales, paralelos y condicionales.
- ***Patrones avanzados para ramificación y sincronización***: Este tipo de patrones extrapolan los patrones de control básico de flujo para representar tipos avanzados de divisiones y uniones.
- ***Patrones de iteraciones***: En los leguajes de programación las estructuras iterativas son bastante comunes, pero en modelamiento de procesos, esta situación es bastante limitada. Razón por la cual, se presentan tres patrones que emulan una estructura y que permiten un diseño menos rígido.
- ***Patrones de cancelación***: La ocurrencia de un evento puede dar lugar a la cancelación de actividades. En algunas situaciones, este tipo de eventos puede causar la cancelación de todo el proceso.
- ***Patrones basados en estados:*** Los sistemas de flujo de trabajos típicos se centran solamente en las actividades y en los eventos. De tal suerte, que se termina limitando la expresividad del lenguaje de procesos, ya que no se toman en cuenta a los estados. Al igual que los patrones de iteraciones, los patrones basados en estados pretenden dar más expresividad y permitir un diseño más flexible.
- ***Patrones de múltiples instancias:*** En el contexto de un solo caso, las mismas partes de un proceso necesitan ser instanciadas varias veces. Por ejemplo, En el contexto de una solicitud de movilización, se necesita varias instancias de aprobación.

Tabla 2.2: Patrón de Sincronización *(Bizagi, 2013)*

Nombre:	Sincronización
Categoría:	Patrones de control básico de flujo
Descripción:	La sincronización es un punto en el proceso donde dos o más ramas del proceso se unen en una sola. Se llama sincronización porque se espera a que todas las ramas entrantes se completen antes de continuar con la siguiente actividad.
Ejemplo:	Cuando un nuevo empleado llega a la compañía es necesario realizar varias actividades. Por ejemplo, darle acceso a la información de su cargo, firmar algunos documentos legales y alistar su puesto de trabajo. El empleado no puede empezar a trabajar hasta que todas las actividades se hayan completado.
Implementación:	Para este ejemplo se utiliza una compuerta paralela como elemento convergente para lograr la unión de todas las ramas.
Diagrama:	



2.2.4. Clasificación de Procesos de Negocio

De acuerdo a la literatura referente a procesos de negocio, existen clasificaciones que dependen mucho del contexto que desea el autor. Es por tal razón, que para tratar objetivamente al tema se presentan cuatro grandes clasificaciones: En una primera clasificación, los procesos de negocio se enfocan de acuerdo a la relevancia o función que tienen cada uno de ellos dentro de la organización (Hernandez, 2010). Otra clasificación enmarca a los procesos de negocio organizados según su grado de automatización (Weske, 2012). También se muestran los procesos de negocio dependiendo de las soluciones utilizadas para el alineamiento entre los procesos modelados y su implementación mediante las Tecnologías de Información. (Hertis & Juric, 2013) Por último, se presenta a los procesos de negocio clasificados de acuerdo a su comportamiento interno o interacción entre tareas. (Melao & Pidd, 2000)

Clasificación mediante funciones dentro de la organización

Esta clasificación de procesos de negocio (Hernandez, 2010) es bastante relativa, ya que dependerá mucho del tipo de organización y del tipo de clientes al que va dirigido sus productos y servicios. Dentro de esta clasificación se encuentran tres tipos de procesos: núcleo, soporte y gerencial.

- ***Núcleo**:* Los procesos de negocio núcleo, deben considerar el valor que reciben los actores del negocio primario. Es decir, al ser procesos que interactúan directamente con el cliente, entonces estos procesos núcleo son los encargados de crear los productos y/o servicios de la organización.
- ***Soporte**:* Son procesos que no intervienen directamente con los clientes, pero a su vez son de vital importancia para mantener la estructura organizacional de la empresa. De tal suerte que procesos financieros, contables o de nómina se podrían considerar como procesos de soporte.
- ***Gerencial**:* Son procesos que tienen que ver con el actor propietario de la empresa o con la gerencia de la organización. Dentro de estos procesos, se podría encontrar a los relacionados con la de toma de decisiones, manejo de informes para inversionistas, administración de indicadores de la organización, presupuestos a largo plazo, etc.

Clasificación según el grado de automatización

El grado de automatización de un proceso de negocio clasifica a los mismos en totalmente automatizados, parcialmente automatizados o manuales (Weske, 2012). La automatización dependerá del grado de involucramiento de una persona en el proceso.

- ***Totalmente automatizados***: Son procesos que carecen enteramente de un direccionamiento discrecional o manual del lado del dueño del proceso, ya que el mismo está completamente automatizado. Por ejemplo, para la "Reserva de vuelos", todo el proceso se lo realiza por interno y automáticamente mediante una interface web, aunque se sobreentiende que el cliente debe ingresar los parámetros o entradas requeridas por el proceso como fechas, origen, destino, etc.
- ***Parcialmente automatizados*:** En estos procesos se automatizan ciertas tareas, pero el momento en que existan actividades manuales o con un grado de discrecionalidad por parte del usuario, se podría considerar al proceso parcialmente automatizado. Por ejemplo, un proceso para "Solicitud de capacitación" dependerá de las decisiones del jefe inmediato para considerar la factibilidad de permiso o del jefe superior para verificar la pertinencia de la capacitación o de la aprobación financiera por parte del encargado de presupuesto. En este caso, las tareas automáticas como cálculos contables o de nómina se realizan automáticamente, pero existen tareas manuales de aprobaciones que deberán ser ejecutadas por los diferentes involucrados del proceso.
- ***Manuales***: Son procesos netamente transaccionales, en los que el usuario está a cargo del registro de la información y del procesamiento de la misma. En realidad, no tendría sentido crear procesos de este tipo ya que volviendo al ejemplo de la "Solicitud de capacitación" el proceso sería el mismo aunque las actividades de registro de permisos, erogación de dinero y demás se realizarían como procesos aislados, situación que desencadenaría en la pérdida de las ventajas de efectividad, ahorro y productividad de BPM.

Clasificación mediante soluciones de alineamiento entre las TI y los procesos negocio

El alineamiento de las TI[22] con los procesos negocio es un campo de estudio que se ha venido trabajando en los últimos años desde el punto de vista de BPM. En realidad, es una de las mejores alternativas, pero el problema comienza a surgir el momento en que las especificaciones utilizadas para el diseño de modelos de procesos de negocio difieren de las especificaciones de ejecución de los mismos. Esta clasificación, se centra en los enfoques de solución que presentan los autores Hertis y Juric (Hertis & Juric, 2013) para reducir el gap entre modelos y ejecución de procesos de negocio desde un punto de vista práctico:

- ***Enfoque metodológico*:** Son procesos modelados con un diseño sistemático, integral y mediante el uso de patrones de procesos de negocio estándares que permitan utilizar al modelo directamente en la implementación de los procesos.
- ***Enfoque de transformación***: Estos procesos necesitan de transformaciones para que se pueda generar el desarrollo de la implementación de las especificaciones de los procesos de negocio. Es decir, son procesos que se encuentran en niveles de abstracción muy altos de diseño y que necesitan ser refinados a modelos dependientes de la plataforma de implementación.

[22] **TI**: **T**ecnologías de la **I**nformación

- ***Enfoque de mejora:*** Son procesos que requieren de la introducción de ciertas mejoras en su especificación o en ciertos casos necesitan de una nueva especificación para su implementación mediante TI.

Clasificación según el comportamiento interno del proceso

Más que una clasificación, es un conjunto de perspectivas de los procesos de negocios, en donde influyen en mayor o menor grado los parámetros de recursos, ambiente, relaciones organizacionales, necesidades del cliente, etc. (Melao & Pidd, 2000) A continuación, se detallan las dos principales perspectivas al respecto:

- ***Procesos de negocio como máquinas deterministas***: "El determinismo es una teoría que supone que la evolución de los fenómenos naturales están completamente determinados por las condiciones iniciales" (Real-Academia, 2011). Gracias a esta definición, fácilmente se puede establecer que este criterio engloba a todos los procesos estáticos que utilizan una secuencia fija de actividades o de tareas bien definidas como se muestra en la Figura 2.7. Además, estos procesos convierten a los parámetros de entrada en salidas que permiten conseguir los objetivos del proceso sujetos a la restricción de satisfacer las necesidades de los clientes.

  El principal criterio de estos procesos radica en la premisa de que un buen diseño de un proceso permitirá la eficiencia en el uso de dinero, recursos y tiempo.

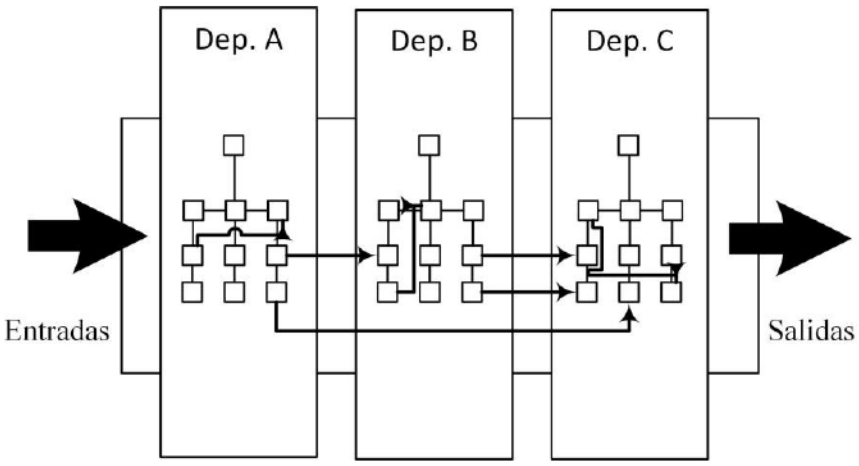


Figura 2.7: Máquinas deterministas *(Melao & Pidd, 2000)*

- ***Procesos de negocio como sistemas dinámicos complejos***: Este tipo de procesos, ajustan una realidad determinista del proceso de negocio con las características dinámicas del entorno. Es decir, a las entradas, procedimientos y salidas se aumenta los parámetros de frontera y entorno; los mismos que dependerán de las relaciones externas del proceso con el ambiente dinámico en el que se desenvuelve. En la práctica, generar o modelar este tipo de procesos resultaría demasiado desgastante para una organización que pretenda trabajar con procesos reales. Dicho de otra forma, es mejor que la organización trate de enfocarse en definir procesos deterministas que permitan un mejor control de las condiciones, en lugar de pretender la interacción con entornos cambiantes, ya que se necesita una inversión fuerte en tiempo, costo y habilidades para crear e implementar un modelo dinámico.

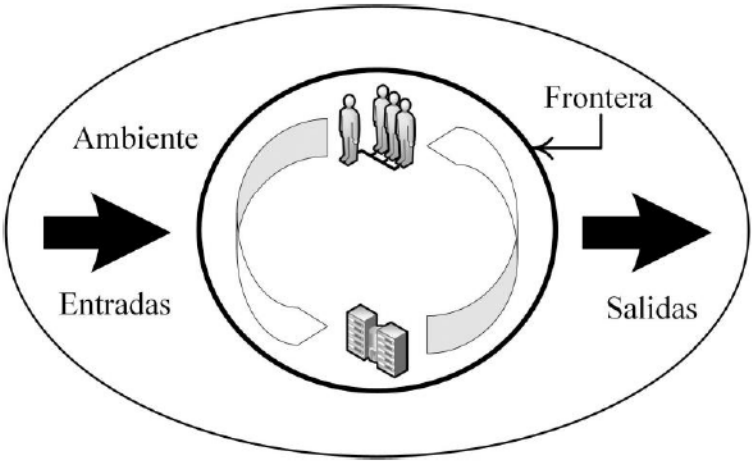


Figura 2.8: Sistemas dinámicos complejos *(Melao & Pidd, 2000)*

2.2.5.Suite[23] para la Gestión de Procesos de Negocio (BPMS)

En los apartados anteriores se definió el estado del arte de la Gestión por Procesos de Negocio y se planteó varias definiciones; dentro de ellas se presentó el ciclo de vida de un proceso de negocio con cinco fases, en donde también se planteaba la recomendación de usar herramientas que permitan la automatización de cada una. Es así, que en este punto se presenta ciertas herramientas utilizadas en la actualidad en conjunto con sus definiciones, conceptos y características de evaluación. A este conjunto de herramientas se les denomina Suite o Sistema. De ahí que "*un sistema de gestión de procesos de negocio (BPMS) es un software genérico que soporta el modelado, análisis y aprobación de los procesos de negocio. Este tipo de software está a menudo bajo la noción de un sistema de información consciente de los procesos*" (Poelmans, et al., 2013).

Con la aparición de los diferentes sistemas de información integrados como: ERP[24], CRM[25] o los SCM[26], también aparecen los Sistemas de Workflow; los mismos que son los precursores de los BPMS y que permiten definir un flujo de actividades humanas en conjunto con los diferentes documentos asociados a estas. (Abdelahad & Riesco, 2012) De esta forma, se obtiene el seguimiento de los diferentes procesos, pero en la praxis, la deficiencia de un Workflow es que no interactúa directamente con otros sistemas de información. Es decir, si una actividad del Workflow es registrar la información en un ERP, entonces se debe abandonar temporalmente la aplicación para registrar la información necesaria y volver nuevamente al Workflow. Los BPMS permiten que todas las actividades se desarrollen dentro de un mismo sistema y por lo tanto, el seguimiento del proceso y actualización de los diferentes sistemas de información relacionados con las tareas o actividades se lo hagan en un solo ambiente de una forma más natural, práctica y eficiente (Díaz, et al., 2009).

[23] **Suite**: Conjunto de programas informáticos que interactúan entre sí para la ejecución de tareas relacionadas
[24] **ERP**: **E**nterprise Resource Planning
[25] **CRM**: **C**ustomer **R**elationship **M**anagement
[26] **SCM**: **S**upply **C**hain **M**anagement

Componentes de un BPMS

En la mayoría de los casos, este tipo de sistemas o suites poseen componentes definidos según la organización que los desarrolle o los autores que los estudien. De esta forma y aunque muy genéricamente se pueden dividir a un BPMS en cuatro componentes: interfaces de negocio, componentes técnicos, gestión del sistema y componentes de integración representados en la Figura 2.9 (Labrador-Tovar & Sanchez, 2013).

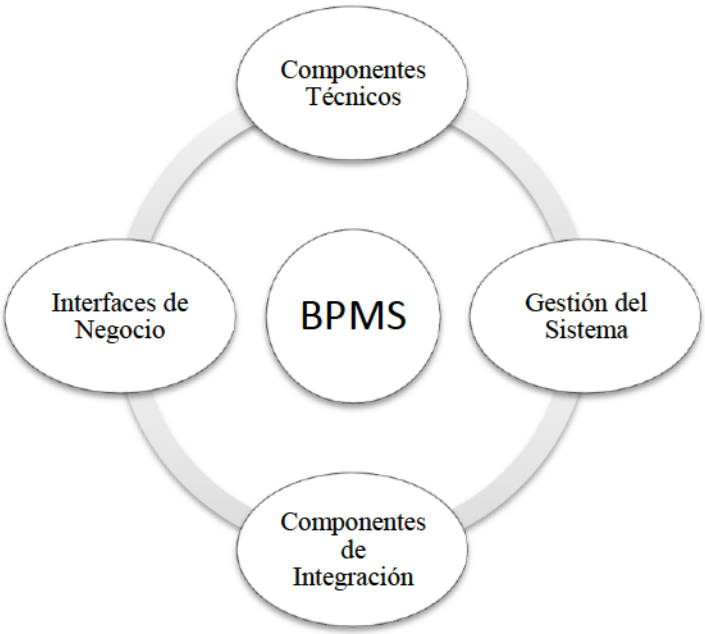


Figura 2.9: Componentes de un BPMS

Otra clasificación más detallada es la utilizada por Weske (Weske, 2012) (Martinez, et al., 2010) y que integra siete componentes que debe poseer un BPMS:

- ***Motores de orquestación***: Este componente permite coordinar la secuencia de actividades que se han definido según las reglas de modelo de procesos.
- ***Modelación de procesos***: La suite debe disponer como parte central una manera de modelar los procesos de negocio en una notación definida.
- ***Herramientas de análisis e inteligencia de negocio***: Este componente permite el estudio de la información generada a partir de la ejecución del proceso en tiempo real. Por otro lado, es importante que una suite BPM pueda implementar administración avanzada de usuarios y permisos, además de proveer estadísticas e informes personalizados y relacionados con las actividades, datos y rendimiento de la aplicación.
- ***Motores de reglas***: Permite ejecutar las reglas abstraídas de las políticas y decisiones de negocio. En este mismo contexto, también se puede utilizar un motor transaccional o diversas herramientas de migración y ejecución de procesos síncronos y asíncronos.
- ***Repositorios***: Mantiene los componentes y recursos de los procesos que estarán disponibles para su reutilización en otros procesos.
- ***Herramientas de Simulación y Optimización***: Permite comparar el nuevo diseño de procesos con el desempeño operacional actual.

- ***Herramientas de Integración***: Finalmente, este componente se relaciona con la habilidad que tenga el BPMS para integrar el modelo con otros sistemas de información.

Parámetros de Evaluación

En el mercado existen varias opciones de BPMS como se aprecia en la Tabla 2.3. La principal diferencia se basa en el criterio de licencia, ya que en algunos casos son de índole propietaria y otros bajo licencia de software libre.

Tabla 2.3: Listado de principales BPMS del mercado

Herramienta	**Organización**
Oracle BPM Suite	Oracle
IBM WebSphere Process Server	IBM
Bizagi BPM Suite	Bizagi
jBPM	JBOSS Community
Bonita BPM (Alvarado, 2011)	BonitaSoft
Intalio BPMS	Intalio
ProcessMaker BPM	Colosa
Activity BPM Platform	Activity Community
Facilis BPMN	Statum
WebRatio BPM	WebRatio
AuraPortal BPMS	Aura

Los diferentes parámetros de evaluación, permitan entender las relaciones, ventajas y desventajas entre unos y otros suites. Además, estos parámetros deben estar alineados con las características y componentes que se detallaron anteriormente. De esta forma, se definen dos niveles de parámetros de evaluación (Duarte & Costa, 2013):

- ***Nivel 1: Criterios específicos de un BPMS***: En un primer nivel, se definen criterios específicos de un sistema de BPMS y las características primordiales que deben poseer:
  - Módulo de modelado y orquestación
  - Soporte para reglas de negocio complejas
  - BAM (Monitoreo de Actividades de Negocio)
  - Seguridad
  - Portabilidad
  - Usabilidad
- ***Nivel 2: Requisitos específicos***: El segundo nivel, está formado por los requisitos de evaluación. Los requisitos representan directrices específicas para cada criterio definido

en el primer nivel. Por ejemplo, el criterio de "seguridad" se puede descomponer en los requisitos técnicos a fin de proporcionar más detalles y garantizar una mayor seguridad para los sistemas de BPM.

Los requisitos de segundo nivel de la jerarquía se deben priorizar, porque a partir de ellos se llevarán a cabo funciones específicas para cada criterio como se aprecia en la Tabla 2.4. En general, este nivel se basa en el hecho de que los requisitos planteados representan directrices relevantes para la evaluación de un BPMS.

Tabla 2.4: Criterios y requerimientos específicos de un BPMS *(Duarte & Costa, 2013)*

Criterios (Nivel 1)	**Requisitos (Nivel 2)**
Módulo de modelado y orquestación	• Lenguaje de notación • Gestión de usuarios • Gestión de roles • Gestión de auditoría
Soporte para reglas de negocio complejas	• Aplicación workflow cliente • Workflow para datos relevantes
BAM (Monitoreo de Actividades de Negocio)	• Funciones de supervisión de procesos • Funciones de los estados de los procesos • Repositorio de metadatos
Seguridad	• Confiabilidad • Disponibilidad
Portabilidad	• Adaptabilidad • Coexistencia • Posibilidad de sustituir • Capacidad para ser instalado
Usabilidad	• Facilidad de aprendizaje • Operatividad • Atractivo

2.2.6.Notación para el modelamiento de Procesos de Negocio (BPMN)

Como se ha venido exponiendo durante los diferentes apartados de BPM, una de sus principales ventajas es la reducción del gap entre los analistas de procesos de negocio y quienes implementarán en un software las diferentes políticas, procedimientos y reglas de la organización. Una forma para esta reducción, es definir un lenguaje común como una notación[27] que permita esta disminución del gap a nivel de modelado de los procesos negocio, haciendo que las partes involucradas puedan comunicar los procesos de forma clara, completa y eficiente. Es por tal motivo, que a lo largo del tiempo se han venido definiendo un amplio conjunto de notaciones en el ámbito de los negocios; esta dispersión ha hecho que en un principio los diferentes BPMS hayan adoptado diferentes notaciones de modelado, muchas veces utilizando una notación propia. Esto ha hecho que dentro de los múltiples esfuerzos y en un afán de estandarización el Grupo de Gestión de Objetos OMG por sus siglas en ingles (OMG, 2014) haya adoptado como estándar general de modelado de procesos de negocio al BPMN.

*"**Business Process Model and Notation** es una notación gráfica que describe la lógica de los pasos de un proceso de negocio. Esta notación, ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades"* (Bizagi, 2013)

El estándar BPMN se ha vuelto clave para el modelado de procesos de negocio, ya que se puede representar gráficamente las distintas etapas y componentes de un proceso de negocio, pero sobre todo la característica principal encontrada y que fue la causa de su estudio por separado, es que la mayoría de los BPMS en sus versiones actuales soportan representaciones de modelos de procesos de negocio bajo esta notación. Lo cual permite que la representación de las actividades, eventos, compuertas, conexiones y otros elementos puedan ser migrados o desplegados en diferentes BPMS. Además el estándar BPMN (OMG, 2011), ha sido desarrollado teniendo en cuenta el objetivo de superar las limitaciones de otras notaciones que impedían su aplicación en las aplicaciones de negocio, científicas y de ingeniería. (Abdelahad, et al., 2013)

En la versión 2.0 de BPMN existen más de 50 variantes (BPMN, 2011) de artefactos y representaciones gráficas pertenecientes a esta notación. A continuación, se presentan los componentes más representativos organizados en seis categorías: contenedores, actividades, objetos de flujo, compuertas, eventos y datos.

Contenedores

- ***Los contenedores (pools) y los compartimientos (lanes)*** representan a las entidades responsables de las actividades de un proceso.
    - Los contenedores pueden anidarse en contenedores y compartimientos.
    - Las entidades pueden ser: organizaciones, roles, usuarios, sistemas, etc.

[27] Notación: Representación por medio de signos o símbolos de una cosa.

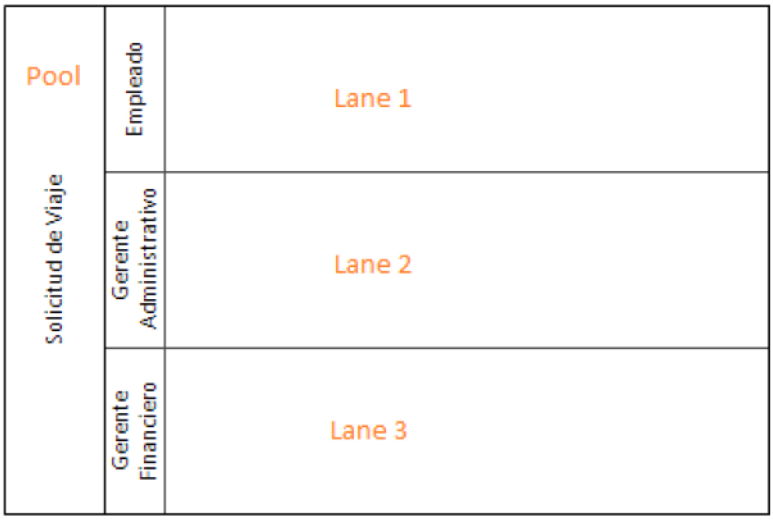


Figura 2.10: Contenedores BPMN2

Actividades

- ***Tarea*:** Es la unidad de trabajo realizado dentro de una organización.

Figura 2.11: Tarea BPMN2

- ***Tipos de tareas***: Especifican la naturaleza de la tarea que se llevará a cabo.
    - *Tarea Manual*: Es una forma de representar que la actividad no se realiza automáticamente.
    - *Tarea de Usuario*: Mediante este tipo se indica que la tarea debe ser ejecutada por un ser humano.
    - *Tarea de Envío*: De esta manera se representa que la tarea envía un mensaje con contenido de comunicación.
    - *Tarea de Recepción*: Viceversa al anterior, este tipo de tareas reciben mensajes con contenido de comunicación.
    - *Tarea de Regla de Negocio*: Son tareas que proporcionan un mecanismo, en el que a partir de un motor de reglas de negocio se pueden enviar y recibir datos.
    - *Tarea de Invocación de Servicio*: Son tareas que utilizan algún tipo de servicio, como por ejemplo un Web Service o una aplicación automatizada.
    - *Tarea de Ejecución de Script*: Representan tareas que ejecutan archivos de procesamiento por lotes o un conjunto de instrucciones ordenadas.

Figura 2.12: Tipos de Tareas BPMN2

- ***Marcadores de actividad*:** Son decoraciones o marcadores que permiten especificar el comportamiento particular de las actividades durante su ejecución.
    - *Subproceso*: Representa a una actividad que puede ser refinada y descomponerse a su vez en varias tareas.
    - *Ciclo*: Las tareas o subprocesos pueden estar decorados con una flecha circular que indica que la instancia puede realizar un número definido de repeticiones según condiciones específicas.
    - *Instancias Múltiples*: Las tres barras de este marcador representan que la tarea o el subproceso de ser el caso, pueden ejecutar la instancia generada una o más veces de forma serial o en paralelo.

Figura 2.13: Marcadores de Actividad BPMN2

Objetos de Flujo

- ***Flujo de secuencia***: Define el orden de ejecución entre dos actividades.
- ***Flujo por defecto***: Especifica el camino a seguir en el caso de que las condiciones de los caminos alternativos sean evaluados como falso.
- ***Flujo condicional***: Tiene una condición asociada que permite determinar si la relación o camino será activado o no. Si bien la expresión asociada no se muestra en el modelo, está debe ser definida en la herramienta de modelado que se haya elegido.

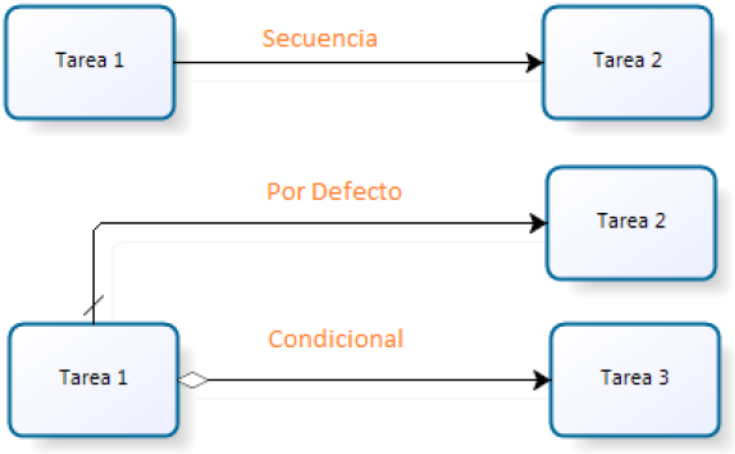


Figura 2.14: Objetos de Flujo BPMN2

Compuertas

- ***Exclusiva***: Esta compuerta puede utilizarse como un punto de bifurcación o como un punto de convergencia. En el primer caso, selecciona exactamente un flujo de secuencia de entre las alternativas existentes, mientras que como punto de convergencia, la compuerta espera a que un flujo incidente se complete para activar el flujo saliente.
- ***Paralela***: Cuando se ve a la compuerta como un punto de bifurcación, entonces todos los caminos salientes serán activados simultáneamente. Mientras que como punto de convergencia, la compuerta espera a que todos los flujos incidentes se completen antes de activar el flujo saliente.
    - *Basada en Eventos*: Esta compuerta siempre será seguida por eventos o tareas de recepción, y sólo activará un flujo saliente dependiendo del evento que ocurra en primer lugar.
    - *Inclusiva*: En un punto de bifurcación, al menos un flujo es activado. Mientras que en un punto de convergencia, espera que finalicen todos los flujos que fueron activados para activar el saliente.

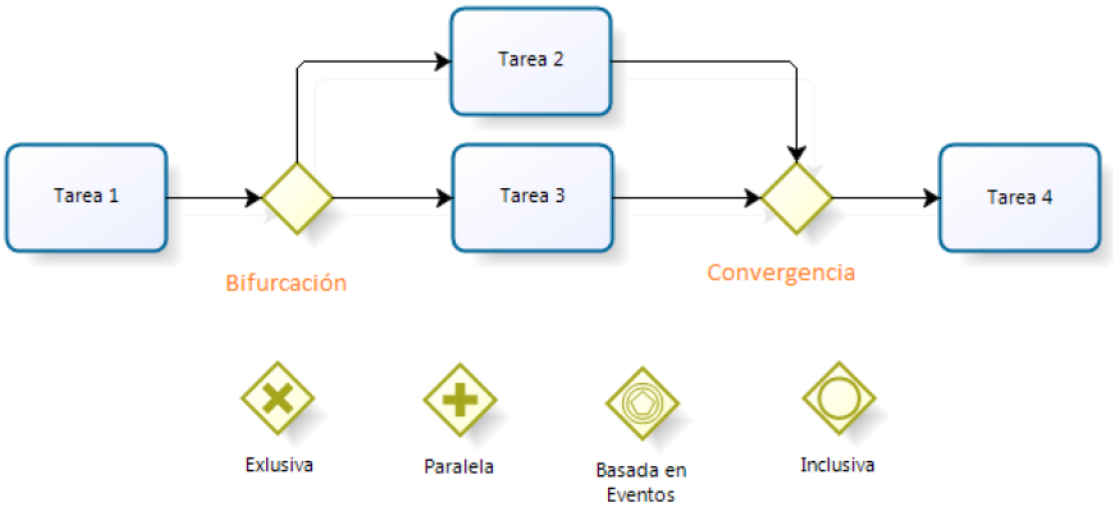


Figura 2.15: Compuertas BPMN2

Eventos

Los eventos representan algo que ocurre o puede ocurrir al inicio, en el transcurso o al finalizar el flujo del proceso. De ahí que la mayoría de eventos pueden ser vistos desde una clasificación genérica de eventos de inicio (verde), intermedio (amarillo) y fin (rojo). Los eventos intermedios tiene la particularidad de que pueden utilizarse dentro del flujo de la secuencia o adjunto a los límites de una tarea.

- ***Simple***: Esta representación no especifica ningún comportamiento en particular del evento.
- ***Mensaje***: Este tipo de eventos puede representar varios comportamientos dependiendo si es de inicio, intermedio o fin. En el primer caso, representa que el proceso debe iniciar con la recepción de un mensaje, mientras que si es intermedio significa que el proceso no puede continuar el flujo sino envía o recibe un mensaje. En el caso de que el evento sea de finalización, entonces significa que el proceso termina cuando se envía un mensaje.
- ***Temporal***: Utilizar este tipo de evento, sirve para indicar que se debe esperar una fecha definida para iniciar el proceso, o en su defecto un tiempo definido para continuar el flujo del proceso si se ha precisado un evento temporal intermedio.
- ***Error***: Los eventos intermedios de error solo se ejecutan cuando están adjuntos a una tarea, ya que si la tarea falla o lanza algún tipo de excepción, entonces la idea de este evento intermedio es que capture la misma y dirija su flujo en otro sentido. En el caso de que el evento sea un error de finalización, entonces esto significa que el proceso termina el momento que se genere y se capture la excepción.
- ***Terminación***: Este evento representa la finalización general del proceso. Esto significa que cuando algún camino llegue a este evento, entonces el proceso terminará completamente sin importar que existan más caminos del flujo pendientes.

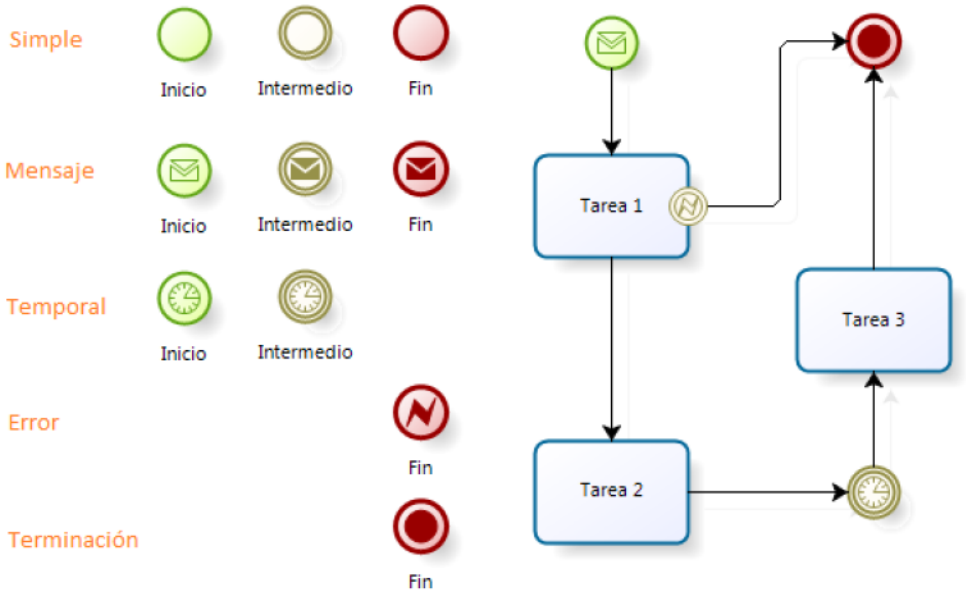


Figura 2.16: Eventos Generales BPMN2

Datos

- ***Almacén***: Es un lugar en donde el proceso puede leer o escribir datos. La información en el almacén debe permanecer persistente luego de la finalización de la instancia del proceso. Un típico ejemplo de estos almacenes son las bases de datos.
- ***Dato objeto***: Esta representación identifica a información que fluye a través del proceso como documentos, correos electrónicos, cartas, etc.
- ***Colección de datos objetos***: Representa una colección de objetos que almacenan información en conjunto. Como ejemplo, se puede tener una lista de documentos asociados.

Figura 2.17: Datos BPMN2

Claves en la Construcción de un modelo con BPMN2

La construcción de un modelo de proceso de negocio es clave para comenzar con la implementación del proceso de negocio. Es por tal motivo, que a continuación se presenta una metodología básica y práctica (Márquez, 2010) para la creación de un proceso:

- Crear un proceso inicial solamente utilizando los símbolos de tareas y compuertas simples en un solo contenedor.
- Luego de obtener un consenso sobre el flujo del proceso, entonces se debe agregar los participantes en los compartimientos respectivos.
- Considerar cambiar los nombres de las tareas según el contexto que manejan los participantes y sobre todo especificar mejor la naturaleza de cada actividad mediante los tipos de tareas.
- Definir explícitamente la consulta realizada en cada bifurcación y la etiqueta respectiva para cada flujo.
- Utilizar los diferentes tipos de compuertas para detallar mejor el proceso, teniendo en cuenta los conceptos de bifurcación y convergencia.
- Asignar los eventos con las variantes respectivas.
- Utilizar las representaciones de datos para agregar el almacenamiento de los productos generados.
- Finalmente y mediante anotaciones se debe tratar de documentar todo el proceso o al menos las ambigüedades que puedan presentarse.

La importancia de un modelo de proceso de negocio radica en la reducción del gap entre el analista del negocio y el analista de software, razón por la cual la notación debe ser entendida por ambas partes y sobre todo en similar profundidad, ya que puede suceder que existan modelos en los cuales la utilización de todas las bondades de la notación para definir un proceso, puedan ser causa de resistencia entre las partes por la diferencia de profundidad de

conceptos entre los involucrados o interesados en la modelación del proceso, por lo que el gap puede crecer en lugar de reducirse.

2.2.7. Otras Notaciones de Procesos de Negocio

Si bien existen varias alternativas, metodologías y estudios sobre notaciones. En la actualidad, existen cada vez menos notaciones de procesos de negocio en uso, ya que BPMN se ha convertido en un estándar muy utilizado y de gran popularidad en el mercado de los BPMS. Esta situación, ha hecho que las demás notaciones pierdan terreno en cuanto a su uso, aunque no hayan perdido vigencia. Es más, por la gran diversidad de notaciones, BPMN ha sido concebido con la posibilidad de migrar los procesos desarrollados en otras notaciones.

A continuación, se presenta un conjunto de notaciones gráficas o textuales que van de la mano con el estudio que se viene realizando:

- ***UML***: El Lenguaje Unificado de Modelado es un estándar completo para describir sistemas de software y facilita el acercamiento entre el diseño de soluciones más favorables para la empresa y el diseño detallado de sistemas de software. Dentro del modelo UML se puede encontrar a los diagramas de actividad; los mismos que pueden representar modelos de procesos básicos (Bera & Evermann, 2014).
- ***EPC***[28]**:** Las cadenas de procesos controladas por eventos fueron el estándar líder en la industria para el modelado de procesos de negocio, ya que es una notación orientada al negocio y permite a los usuarios sin conocimientos técnicos documentar y optimizar los flujos de trabajo con facilidad y rapidez. Esta notación fue la antesala de BPMN por lo que sus componentes de modelado son bastante similares. La diferencia se basa en la motivación, ya que si bien EPC fue concebido para el análisis y modelo de procesos de negocio, BPMN fue extendido para el análisis, modelo y despliegue de los procesos.
- ***BPEL***[29]**:** Más que una notación, el lenguaje de ejecución de procesos de negocio es un lenguaje técnico y estandarizado que ejecuta las funciones de negocio gestionándolas a través de servicios Web. Los procesos de negocio modelados en BPMN pueden ser traducidos directamente a lenguaje BPEL.
- ***XPDL***[30]: El lenguaje de definición de procesos XML, es otro estándar técnico utilizado para describir modelos de procesos orientados al usuario y aunque la ejecución de las funciones de negocio suelen gestionarlas los usuarios, también pueden realizarla los servicios Web. Las notaciones gráficas como BPMN o EPC pueden ser traducidas a este lenguaje.

[28] **EPC**: Event-driven Process Chains
[29] **BPEL**: Business Process Execution Language
[30] **XPDL**: XML Process Definition Language

2.3. Requerimientos de Software

El Cuerpo del Conocimiento de la Ingeniería de Software o SWEBOK[31] por sus siglas en inglés, es promovido por la IEEE Computer Society y define un consenso a nivel mundial sobre el conocimiento de la Ingeniería de Software. Este consenso, tiene por objetivo definir los diferentes documentos y estándares relacionados con el estado del arte de la disciplina. En la actualidad, se ha publicado la versión 3.0 (Bourque & Fairley, 2014); en la cual se han definido un conjunto de áreas de conocimiento que abarcan la Ingeniería de Software y que se muestran en la tabla 2.5.

Tabla 2.5: Áreas del Conocimiento del SWEBOK

Cuerpo del Conocimiento de la Ingeniería de Software (SWEBOK)	Requerimientos de Software
	Diseño de Software
	Construcción de Software
	Testing de Software
	Mantenimiento de Software
	Gestión de la Configuración de Software
	Gestión de Ingeniería de Software
	Proceso de Ingeniería de Software
	Modelos y Métodos de Ingeniería de Software
	Calidad de Software

El área de conocimiento de los requerimientos de software es el primer capítulo abordado por el SWEBOK y a su vez será el tema que se detallará en esta sección.

La importancia de los requerimientos de software radica en que esta área interviene tanto al inicio, como al final del proceso de desarrollo de software. En la práctica, esta intervención se realiza porque los requerimientos se convierten en la descripción de lo que se debe diseñar y por otro lado, son la base para verificar que la funcionalidad del producto de software sea la correcta.

Los requerimientos de software vistos como un área de conocimiento, han venido siendo estudiados desde inicios de los ochenta y por lo tanto se han convertido en una etapa bastante madura. Esta madurez, ha hecho que en la mayoría de casos se hable de ellos no solamente como una etapa inmersa en la Ingeniería de Software, sino como una ciencia o disciplina que enmarca el análisis y documentación de requerimientos de software denominada *Ingeniería de Requerimientos* (ISO/IEC/IEEE, 2011).

2.3.1. Definiciones

Introducir las definiciones principales para el manejo de los temas de requerimientos, tiene gran importancia porque la investigación científica está regida por el lenguaje y por lo tanto este debe ser claro, preciso y sin lugar para las ambigüedades. Lastimosamente, para iniciar

[31] **SWEBOK**: **S**oftware **E**ngineering **B**ody **o**f **K**nowledge

con las definiciones se debe presentar el problema existente entre diferentes autores en español; quienes utilizan dos términos diferentes para representar lo mismo: ***requerimiento y requisito***. En ambos casos, lo que se pretende definir es el término en inglés ***requirement*** como la "c*ondición o capacidad que necesita un usuario para resolver un problema o alcanzar un objetivo*" (ISO/IEC/IEEE, 2010).

En nuestro caso, y a lo largo de este trabajo se traducirá la palabra inglesa requirement como requerimiento, esto para evitar confusiones y no entrar en detalles de semántica que se encuentran fuera del alcance de esta investigación.

- ***Requerimiento***: "*Condición o capacidad que debe cumplir o poseer un sistema, componente, producto o servicio para satisfacer un acuerdo, estándar, especificación u otro documento formalmente definido*" (ISO/IEC, 2013).

  Así como la definición de requerimiento de la ISO 25064 habla de un requerimiento en general, también se sabe que existen diferentes fuentes y contextos en los cuales se pueden obtener los requerimientos. Por tal motivo, se encuentra en la literatura un conjunto de palabras que se asocian a los requerimientos y que son utilizadas en diferentes escenarios (Katina, et al., 2014); los mismos que dependerán de las características, ámbitos o audiencia (Durán Toro, 2000). A continuación, se definirán los términos más utilizados asociados a requerimientos:
- ***Requerimientos del sistema***: Los requerimientos del sistema engloban a todo tipo de requerimientos necesarios para el funcionamiento de un sistema en integral. "*Es una combinación de requerimientos de software, hardware, soporte, gente, información, técnicas, instalaciones, servicios y otros elementos de apoyo*" (Haskins, et al., 2006).
- ***Requerimientos funcionales***: Se refiere a las capacidades o funcionalidades que debe realizar el sistema. Este tipo de requerimientos tienen la capacidad de ir *evolucionando* desde descripciones textuales a transformarse en requerimientos con niveles de detalle como modelos conceptuales complejos, diagramas de procesos, diagramas de actividades, casos de uso, referencias a tipos de datos de atributos, rigurosas especificaciones, etc.
- ***Requerimientos no funcionales***: Como su nombre lo indica, son requerimientos que no están ligados directamente con las funcionalidades o servicios que se proveerá al usuario, sino más bien a las propiedades embebidas en un sistema. Por este motivo, más que una definición de requerimientos no funcionales, su profundización se podría dar mediante una clasificación de los mismos en requerimientos de seguridad, eficiencia, rendimiento, usabilidad, espacio, etc. La ventaja de este tipo de requerimientos es que en su mayoría pueden estar asociados a métricas que permiten una fase de validación más oportuna.
- ***Requerimientos de software:*** Los requerimientos de software estudian o se relacionan con las necesidades, deseos y expectativas del usuario en torno al desarrollo del sistema. Los requerimientos de software están acotados o tienen su alcance solamente con los requerimientos funcionales y no funcionales. Este tipo de requerimientos, son quienes se encuentran documentados y detallados en una SRS[32] formal.

[32] **SRS**: **E**specificación de **R**equerimientos de **S**oftware

- ***Ingeniería de requerimientos***: Es la ciencia o disciplina que establece los métodos, procesos y técnicas que permiten la elicitación[33], análisis, especificación, verificación, validación, comunicación, documentación y gestión de requerimientos durante el denominado proceso de requerimientos.

2.3.2. El proceso de requerimientos

En apartados anteriores, se definió al proceso como un *conjunto de actividades mutuamente relacionadas; las cuales transforman elementos de entrada en resultados*. Con esta definición, se puede entender y no confundir los dos procesos necesarios para este estudio: el ***proceso de software,*** que define la forma de interacción de las diferentes áreas de conocimiento planteadas en la tabla 2.5 y el ***proceso de requerimientos,*** que indica cómo interactúan el conjunto de actividades que tienen como propósito la recolección, estudio y análisis de datos sobre el dominio y reglas del negocio para producir una especificación detallada que permita validar la información recolectada.

Actores del proceso

Son los individuos u organismos involucrados de cualquier manera dentro del proceso de requerimientos. Generalmente, son las entradas o las fuentes de información, aunque en realidad pueden ser clasificados en varios tipos (Loucopoulos, 1995): usuarios, clientes, consultores, expertos, analistas, reguladores, desarrolladores, etc. En este contexto, lo primero que se debería hacer dentro del proceso de requerimientos es identificar, clasificar y priorizar los diferentes actores, ya que éstos son los encargados de darle vida al mismo y ser quienes produzcan entradas y/o reciban los resultados generados (Ballejos & Montagna, 2011).

Etapas del proceso

Así como el proceso de software posee sus diferentes etapas, el proceso de requerimientos también puede fraccionarse en diferentes etapas; las mismas que favorecen el estudio del proceso en general.

Las diferentes etapas, así como su subdivisión y profundización dependen en gran medida del autor que las propone, pero en general, se puede plantear un proceso genérico dividido en tres etapas: elicitación, especificación y validación como se aprecia en una adaptación del proceso de Loucopoulos presentada en la Figura 2.18.

El propósito de la etapa de ***elicitación*** es recolectar y analizar toda la información en torno a la lógica y reglas de negocio relacionadas con el producto de software a desarrollar. Una vez que se tiene toda la información relacionada al respecto, es importante formalizar estos datos de una forma rigurosa para que los mismos puedan ser utilizados por los diferentes involucrados del proceso. Esta formalización, se la levanta en la fase de ***especificación***. En esta fase, se pueden utilizar estándares como las normas: IEEE 830 o IEEE 29148 para que dicha especificación sea creada de una manera correcta, completa, consistente, verificable y sin ambigüedades. Una vez creada la especificación, se debe verificar que cada uno de los requerimientos sea el correcto, para que de esta forma se pueda realizar una ***validación*** de toda la especificación en integral.

[33] Si bien no es un término traducido exactamente al español, es muy utilizado en la informática para referirse a la extracción o captura inicial de la información.

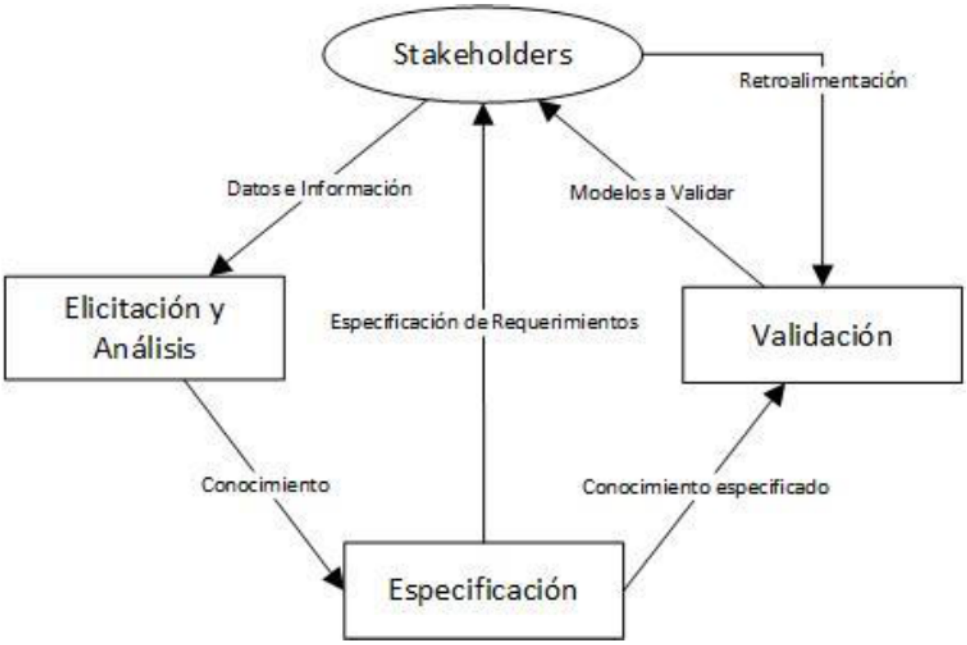


Figura 2.18: Proceso de requerimientos

En lo concerniente a la interacción entre las etapas; estas pueden ser vistas de una manera separada, pero en realidad cada una depende de la otra y la mejor forma de abordar este proceso es mediante un enfoque paralelo, en el cual se tenga una elicitación inicial, pero a medida que se vaya profundizando en el dominio del negocio se pueda a su vez especificar y validar cada requerimiento. En este caso, como se presenta en la Figura 2.19, si el proceso se encuentra en la fase de especificación, esto no significa que no se pueda volver a recolectar y analizar información que no esté clara. Además, en la medida de lo posible se puede ir verificando los diferentes requerimientos especificados.

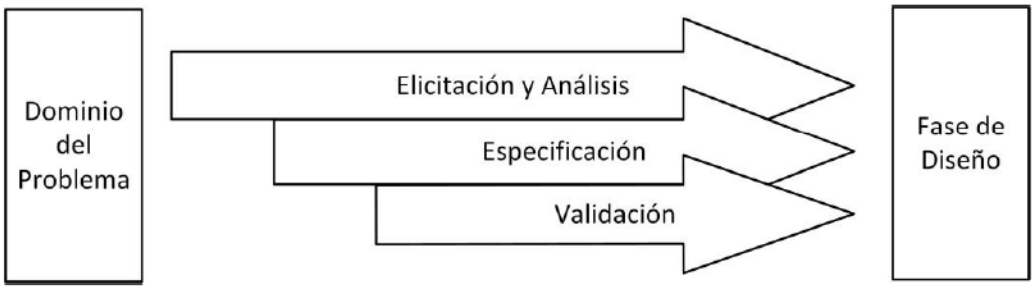


Figura 2.19: Interacción del proceso de requerimientos

Autores como Wiegers (Wiegers, 2003) subdividen a la etapa de elicitación en dos y por ende el proceso de requerimientos trabajará con cuatro etapas: elicitación, análisis, especificación y validación de requerimientos. Por otro lado, van Lamsweerde (van Lamsweerde, 2009) especializa al proceso de requerimientos en seis fases: comprensión del dominio, elicitación de requerimientos, evaluación de requerimientos, especificación y documentación de requerimientos, aseguramiento de calidad de requerimientos y evolución de requerimientos.

2.3.3. Elicitación y Análisis de requerimientos de software

Como se ha venido planteando, esta etapa es la encargada de capturar toda la información referente al nuevo sistema a construir. De esta forma, se utilizan un conjunto determinado de fuentes de conocimiento que tendrán que interactuar con el analista de software para que éste "elicite" el conocimiento relacionado con el proceso que se desea automatizar. La interacción entre el analista y las fuentes debe realizarse mediante un conjunto de técnicas bien definidas.

Fuentes

- ***Entorno organizacional*:** El software; al ser una herramienta que debe dar soporte y apoyar los procesos de la organización, no puede tener requerimientos que vayan en contra de la cultura, estructura y política interna de la organización. Es por tal motivo, que el analista debe tener en cuenta las fuentes de conocimiento institucionales como: misión, visión, planes operativos, organigramas, cartas de navegación, resoluciones, etc.
- ***Metas*:** Habitualmente, se refiere al propósito o a los objetivos generales que se persigue con la creación del software. Las metas son definidas de una forma muy superficial, por lo que no son una fuente de detalle de los requerimientos. El no ser una fuente de detalle, no significa que no deban ser tomadas en cuenta ya que las metas permiten tener una visión global y una cierta priorización que luego ayudará a realizar una clasificación de los diferentes requerimientos según su alineamiento con las metas determinadas.
- ***Stakeholders*:** Son los involucrados que interactúan de una u otra forma con el sistema en desarrollo y que permitirán ayudar en la definición de la información necesaria para la captura del conocimiento. La mayoría de las veces, estos involucrados son relacionados como los expertos del dominio del proceso.
- ***Reglas del negocio*:** Son enunciados que determinan las condiciones específicas que se deben cumplir y por lo tanto definen las restricciones "no negociables" a las que debe regirse el sistema. Por otro lado, se define que un mismo requerimiento puede tener varias reglas de negocio asociadas.
- ***Conocimiento del Dominio*:** Si bien, lo óptimo sería que los stakeholders brinden todo el conocimiento al analista, muchas veces el conocimiento está disperso en varios stakeholders y por lo tanto es necesario que el analista articule los diferentes datos de una manera coherente. Para esta articulación, se puede utilizar otras fuentes como: consultores, artículos, libros, estándares, software legado relacionado, software externo pero con funciones o metas similares a las solicitadas, leyes, reglamentos, ordenanzas, etc.

Técnicas

- ***Entrevistas*:** Sin lugar a dudas, es la técnica de elicitación más utilizada por los analistas, ya que se tiene la concepción de que no requieren de una preparación compleja previa a su aplicación. Esto hace que los analistas abusen de esta técnica y que no se considere las ventajas de utilizar otras alternativas. La ventaja de una entrevista adecuadamente preparada y conducida (Hadar, et al., 2014), radica en el hecho de que se obtiene el conocimiento directamente del experto asociado al requerimiento. Como principal desventaja de las entrevistas, se encuentra que solo se pueden utilizar como fuentes de conocimiento a los stakeholders, dejando de lado las diversas fuentes que necesitan de otro tipo de técnicas para ser elicitadas.
- ***Escenarios*:** Un escenario plantea una situación o un contexto definido que permite realizar preguntas como: "Qué pasa si" o "Cómo se hace esto". La representación de escenarios más difundida son los casos de uso (El-Attar & Miller, 2012).
- ***Prototipos*:** Este tipo de técnicas ayudan al analista cuando existe ambigüedad en el requerimiento o cuando el stakeholder no puede transmitir los requerimientos por existir dudas sobre los mismos. Similar a los escenarios, los prototipos proveen un contexto al usuario pero necesitan herramientas y lenguajes apropiados. Las características de los prototipos son: desarrollo rápido del prototipo, énfasis en la interfaz del usuario, comparativamente baratos.
- ***Reuniones*:** Las reuniones bien conducidas, son una valiosa técnica que permite elicitar requerimientos que demandan ser refinados o que se encuentran en conflicto y por lo tanto, necesitan de un consenso entre los stakeholders relacionados con el requerimiento. Dentro de las reuniones, se pueden encontrar diversas metodologías o estrategias de conducción como: talleres, lluvia de ideas, diseño de aplicación conjunta, grupos de discusión, etc.
- ***Observación*:** Es una técnica bastante costosa, ya que el analista debe sumergirse o convertirse en el "experto" del proceso que desea detallar, para esto no debe tener ningún tipo de dependencia del stakeholder y actuar con la mayor objetividad del caso en torno a la percepción del proceso o actividades profesionales relacionadas, ya que muchas veces el proceso en el papel o como lo define el usuario, no se parece a su funcionamiento en la práctica. La complejidad surge en saber identificar las fuentes y en segundo lugar, la forma de representar las tareas fruto de la observación a un usuario o al sistema relacionado al requerimiento.
- ***Otras técnicas*:** Existen una diversidad de técnicas que permiten la obtención del dominio del negocio y que dependen de las metodologías de desarrollo utilizadas o de las fuentes de elicitación. Dentro de estas técnicas se pueden encontrar los Análisis dirigidos por metas (De la Vara, et al., 2013), Gráficos de decisiones, Storyboarding, Léxicos extendidos del lenguaje (Antonelli, et al., 2012), Análisis de Tareas (De la Vara & Sánchez, 2010), Análisis de Políticas, etc.

Cabe destacar que cualquier técnica utilizada dependerá del contexto que se maneje y las fuentes que se hayan definido. Lo que perciba o documente el analista luego de practicar alguna técnica debe ser detallado y verificado en las fases de especificación y validación respectivamente.

Análisis

El análisis consiste en procesar el conocimiento capturado y examinarlo en integral para: detectar conflictos en los requerimientos, elaborar requerimientos de sistema, derivar requerimientos de software, descubrir los límites de los requerimientos y determinar la interacción de los requerimientos con el ambiente organizacional. (Bourque & Fairley, 2014)

- ***Clasificación**:* Una clasificación, no es más que una agrupación de ciertos objetos con características o parámetros similares. En este contexto, todos los requerimientos son diferentes, pero como punto de partida para el análisis de los mismos, se puede clasificarlos según un conjunto de parámetros definido. Es decir, para un análisis preliminar se debe clasificar todos y cada uno de los requerimientos de acuerdo a:
    - *Requerimiento funcional o no funcional*: La clasificación principal dependerá si es funcional o no. Estos términos se definieron en el punto 2.3.1.
    - *Requerimiento derivado*: Indica si el requerimiento es una subdivisión o se deriva de uno o más requerimientos padre.
    - *Prioridad del requerimiento*: No todos los requerimientos tendrán la misma prioridad, por tal motivo se puede fijar una escala cuantitativa o cualitativa.
    - *Volatilidad del requerimiento*: Esto significa que se debe determinar la posibilidad de cambio del requerimiento una vez que este sea definido.

Tabla 2.6: Plantilla para clasificación de requerimientos

Código:	**2021**		
Descripción:	Gestión de estructura y ofertas académicas		
Descripción Detallada:	Permite poseer un mecanismo a nivel de diseño que permita utilizar los contenidos de las diferentes estructuras y ofertas académicas registradas en el Sistema Nacional Académico.		
Funcional:	X	**No Funcional:**	
Derivado de:	1008, 1010, 2013		
Prioridad:	Crítica	**Volatilidad:**	Baja

- ***Modelo conceptual***: Si bien los modelos conceptuales son generalmente representaciones gráficas; esto no se debe confundir con los diseños de la solución del requerimiento, ya que el *modelo conceptual* pretende representar el funcionamiento actual del requerimiento, las entidades del dominio del problema y las relaciones del requerimiento con el mundo real.

  Existen varios tipos de modelos conceptuales que se pueden adaptar al análisis de requerimientos: diagramas de casos de uso, modelos de flujo de datos, modelos de estados, modelos basados en metas, modelos de datos y muchos más. De hecho, una hipótesis en el planteamiento de este trabajo de tesis es definir a BPMN2 como modelo conceptual en la elicitación de requerimientos.

  En la práctica, la definición de un modelo conceptual debe convertirse en una forma de reducir el gap existente entre el analista de software y el usuario. Es por tal motivo, que se vuelve a recordar lo definido en la subsección *"Claves en la Construcción de un*

modelo con BPMN2" dentro del punto 2.2.6; en donde se plantea que para utilizar adecuadamente un modelo específico, debe existir la misma profundidad de conocimientos sobre la notación, tanto en el analista como en el usuario que validará el modelo.

- ***Diseño arquitectónico y asignación de requerimientos**:* En esta sección de análisis se debe precisar la arquitectura básica del nuevo sistema. Esta definición básica debe constar al menos con la división de subsistemas, componentes y módulos.

  El fin de la división es ir asignado los requerimientos a los diferentes subsistemas, componentes y módulos, para de esta forma clasificar y profundizar las relaciones entre requerimientos. Comúnmente, la separación arquitectónica de los requerimientos se representa mediante notaciones gráficas contextuales, por lo que está sección va íntimamente ligada con el modelo conceptual definido anteriormente.

  Una definición completa de arquitectura se la puede realizar en la fase de diseño y utilizando estándares propios de definición de arquitectura como el IEEE 1471 (Maier, et al., 2001) en donde se define la arquitectura, concerns[34], puntos de vista, vistas, modelos, stakeholders, etc.

- ***Negociación*:** En la elicitación se trata de recolectar todos y cada uno de los requerimientos de los diferentes involucrados. Esta captura "indiscriminada" de conocimiento puede ocasionar que en algunos casos los requerimientos se contradigan o se registren diferentes puntos de vista referentes a un mismo requerimiento. Es aquí, en donde se debe entender a la negociación como la característica del análisis que efectúa la búsqueda de consensos para lograr la resolución de un conflicto.

  Es deber del analista detectar e informar los conflictos, para a su vez mediar consensos que eviten en la medida de lo posible tomar una decisión unilateral al respecto.

Si bien no existe un estándar que defina exactamente la formalidad que debe tener una fase de elicitación. No está por demás, recomendar llevar la formalidad del caso utilizando modelos conceptuales adecuados, evitando tener información dispersa y elaborando plantillas que permitan la recolección de información y una mejor lectura de esta etapa.

2.3.4. Especificación de requerimientos de software

En el área de conocimiento de los requerimientos se habla de dos tipos de documentos: el "*documento de especificación de requerimientos del sistema*" y el "*documento de especificación de requerimientos de software"*. En esta sección de la investigación, se hará referencia exclusivamente al segundo documento también denominado SRS por sus siglas en inglés.

Independientemente de la informática o de la computación, una especificación es un documento técnico y formal que denota todas las características funcionales y de calidad que debe poseer un producto. En este marco, se podría decir que la principal tarea de la fase de especificación de requerimientos de software es la presentación ordenada, estandarizada y coherente de los requerimientos obtenidos en la fase de elicitación y análisis. Esta presentación, debe tener la forma de un documento que permita ser revisado, evaluado y aprobado por los diferentes involucrados en el proceso de requerimientos.

[34] Concern: Asunto, Concerniente, Referido a

Documento de Especificación de Requerimientos de Software (SRS)

La principal razón para la creación de un documento técnico-formal es evidenciar de una manera clara los requerimientos obtenidos en la fase de elicitación y análisis. Por otro lado, se encuentran las motivaciones para su creación. En donde, una SRS se convierte en primer lugar en una herramienta tangible de comunicación entre la lógica del negocio, stakeholders y la fase de diseño del sistema. Mientras que desde un punto de vista comercial, la SRS se convierte en parte del acuerdo contractual con el cliente; permitiendo establecer las bases del acuerdo consumidores-proveedores sobre qué debe hacer el producto.

- ***Beneficios*:** Dentro de los beneficios de la creación de la SRS están la reducción del esfuerzo de desarrollo, ya que los diseños estarán basados en una especificación revisada, evaluada y aprobada. Además, sirve como línea de base para la validación y verificación del software, mientras que en la fase de mantenimiento será la base para la creación de mejoras. Finalmente, la SRS también se convierte en una herramienta muy apropiada para definir estimaciones en los costes y tiempos tanto del diseño como de la construcción del software. (Oliveros, 2012)
- ***Usuarios*:** El documento de especificación puede ser utilizado por varios usuarios y en distintas etapas del proceso de software. A continuación, el conjunto de usuarios propuesto por Sommerville (Sommerville, 2011).
    - *Clientes*: Son quienes deberán leer y comprobar que el documento se ajusta a sus necesidades. Además, son quienes podrán pedir cambios en los requerimientos planteados.
    - *Gerentes*: Los gerentes utilizan la especificación para planificar el proyecto de construcción en lo relacionado a las actividades, recursos y tiempos.
    - *Ingenieros de sistemas*: Es utilizado por los ingenieros de sistemas para entender lo que se quiere construir y así planificar una solución mediante un diseño adecuado. En el proceso de la construcción, también es utilizado el documento de especificación para profundizar reglas de negocio que no se hayan plasmado en el diseño.
    - *Ingenieros de pruebas*: Las pruebas de validación del software deben basarse en los requerimientos detallados en la especificación.
    - *Ingenieros de mantenimiento*: En la etapa de mantenimiento se debe tener un adecuado control de cambios, por esta razón la especificación inicial permitirá ser una línea base que permita entender al sistema y las diferentes relaciones existentes entre sus partes.

Estándares para escribir una SRS

Un estándar es un modelo, norma, patrón o referencia para seguir un proceso o generar un producto. En este caso, los estándares existentes para la creación de una SRS permiten utilizar un marco de referencia estructural para el documento. Este marco, permite a su vez una mejor legibilidad del documento por parte de los diferentes usuarios.

En la Figura 2.20, se aprecia la estructura que debe tener un documento de Especificación de Requerimientos de Software según el estándar IEEE 830 (*Especificación de*

requerimientos de software), mientras que en la Figura 2.21, se presenta la disposición de una SRS mediante la ISO/IEC/IEEE 29148 (*Ingeniería de requerimientos*).

1. Introducción
   1.1. Propósito
   1.2. Alcance
   1.3. Definiciones, Acrónimos y Abreviaturas
   1.4. Referencias
   1.5. Visión General del Documento
2. Descripción General
   2.1. Perspectiva del Producto
   2.2. Funciones del Producto
   2.3. Características de los Usuarios
   2.4. Restricciones
   2.5. Suposiciones y Dependencias
3. Requerimientos Específicos
   3.1. Interfaces Externas
   3.2. Funciones
   3.3. Requerimientos de Rendimiento
   3.4. Requerimientos de la base de datos lógica
   3.5. Restricciones de Diseño
   3.6. Atributos del Sistema
   3.7. Otros Requisitos
4. Apéndices

Figura 2.20: SRS (IEEE 830:1998)

1. Introducción
   1.1. Propósito
   1.2. Alcance
   1.3. Visión General del Producto
   1.4. Definiciones
2. Referencias
3. Requerimientos específicos
   3.1. Interfaces externas
   3.2. Funcionalidades
   3.3. Requerimientos de Usabilidad
   3.4. Requerimientos de Rendimiento
   3.5. Requerimientos de la base de datos lógica
   3.6. Restricciones de Diseño
   3.7. Atributos del Sistema
   3.8. Información de soporte
4. Verificación (Paralela al punto 3)
5. Apéndices
   5.1. Supuestos y dependencias
   5.2. Acrónimos y abreviaturas

Figura 2.21: SRS (ISO/IEC/IEEE 29148:2011)

En ambas figuras, se presentan conceptos y estructuras similares en lo relacionado a las *partes de una SRS*; esto no significa que ambos estándares sean similares, sino más bien, que mientras el estándar IEEE 830 fue creado en 1993 con la visión concreta de ser una estructura para la especificación de requerimientos, por otro lado el estándar ISO/IEC/IEEE-

29148 fue creado en 2011 como un compendio[35] de todos los procesos y productos involucrados en la Ingeniería de requerimientos. Este compendio, ha hecho que la estructura vigente de la SRS sea una adaptación del estándar IEEE 830, pero con una sección adicional dedicada a la verificación de los requerimientos como parte de la SRS y en paralelo con los requerimientos específicos.

A continuación, una breve descripción de cada una de las partes de una SRS según propone el estándar ISO/IEC/IEEE 29148:2011 y que se muestra en la Figura 2.21:

1. **Introducción**
   - **1.1.** ***Propósito:*** En esta subsección se debe precisar el propósito del software a ser especificado.
   - **1.2.** ***Alcance:*** El alcance identifica que productos de software van a ser generados a partir de la SRS. Además, se puede presentar los beneficios, objetivos y metas esperadas del producto.
   - **1.3.** ***Visión General del Producto:*** En esta subsección no se debería detallar, pero si se debe presentar un resumen del conjunto de características relacionadas con el producto de software y que incluyen: perspectiva del producto, funciones principales, limitaciones, supuestos y dependencias.
   - **1.4.** ***Definiciones:*** Esta subsección debe tener un listado de todas las palabras utilizadas en la especificación y que necesiten de una definición explícita, con el fin de evitar ambigüedades o desconocimiento del significado por parte de la audiencia del documento.
2. **Referencias:** Es una lista completa de todos los documentos asociados, citados o consultados durante la creación de la SRS.
3. **Requerimientos específicos**: Esta sección es la más importante de la especificación, ya que es aquí en donde se describirá con un nivel de detalle suficiente todos y cada uno de los requerimientos de software.

   En esta sección, se debe definir como mínimo las funcionalidades del sistema, teniendo en cuenta que cada requerimiento especificado debe ser único y rastreable.
   - **3.1.** ***Interfaces externas*:** Es una lista de las funcionalidades y formas de comunicación de cada interface del producto con sistemas externos.
   - **3.2.** ***Funcionalidades*:** En esta subsección se definen los requerimientos funcionales. Es decir, las acciones fundamentales que deben llevarse a cabo en el software para el procesamiento de las entradas y la generación de las salidas. En la Tabla 2.7, se presenta un ejemplo de requerimiento funcional.

Tabla 2.7: Ejemplo de un requerimiento funcional

Código:	**2003**
Descripción:	Gestión de calificaciones
Descripción Detallada:	Permite registrar o actualizar las calificaciones de los aspirantes que hayan rendido el examen de admisión.

[35] **ISO/IEC/IEEE 15289** - Content of life-cycle information products (documentation), **ISO/IEC TR 19759** - Guide to the Software Engineering Body of Knowledge, **IEEE Std 830** - Recommended Practice for Software Requirements Specifications., **IEEE Std 1362** - Guide for Developing System Requirements Specifications.

Precondiciones:	• Debe existir un listado con la información de cada aspirante con n calificaciones referentes a un eje temático
Entradas:	• Período: Número entero (Tipo) • Sede: Número entero (Tipo) • Área de Conocimiento: Número entero (Tipo) • Campus: Número entero (Tipo) • Carrera: Número entero (Tipo) • Modalidad: Número entero (Tipo) • Calificación mínima: Número entero • Eje Temático: Número entero (Tipo) • Porcentaje de Eje Temático: Número entero (Tipo) RENDIMIENTO ALUMNO • Valor calificación por eje: Número decimal
Proceso:	1. Inicio 2. Se ingresa los códigos de Período, Sede, Área de Conocimiento, Campus, Carrera, Modalidad, Calificación mínima. 3. Se registra los datos de Eje Temático, Porcentaje de eje y Valor de la Calificación las veces que sea necesario. 4. Se multiplica los porcentajes por cada valor 5. Se suma los productos y se obtiene la calificación final. 6. Se cambia el estado del examen de admisión (aprobado o reprobado). 7. Fin
Salidas:	• Calificación final: Número decimal • Estado Examen de Admisión: Número entero (Tipo)
Poscondiciones:	• Se registra o actualiza un nuevo registro en la base de datos relacional.
Fuentes:	Vicerrector Académico Coordinador de Admisión y Nivelación de Sede
Roles involucrados:	Administrador Coordinador de Admisión y Nivelación de Sede

3.3. ***Requerimientos de usabilidad*:** Define los requerimientos relacionados con la calidad de uso. Estos requerimientos de satisfacción dependerán de contextos específicos.

3.4. ***Requerimientos de rendimiento*:** Los requerimientos específicos se relacionan con tasas y valores cuantificables como el número de terminales soportadas, el número de usuarios simultáneos o la cantidad y tipo de información que puede almacenar.

3.5. ***Requerimientos de la base de datos lógica*:** Especifica los requerimientos lógicos para cualquier información que se vaya a almacenar en una base de datos: tipos de información, frecuencia de uso, entidades de datos, relaciones entre entidades de datos, restricciones de integridad y requerimientos de conservación de datos.

3.6. ***Restricciones de diseño*:** Esta subsección especifica restricciones para la fase de diseño del sistema. Estas restricciones pueden ser impuestas por estándares externos, estándares internos, reglamentos o limitaciones del proyecto.

3.7. ***Atributos del sistema*:** En esta subsección se puede especificar requerimientos relacionados con: fiabilidad, disponibilidad, seguridad, portabilidad, etc.

3.8. ***Información de soporte*:** Una vez definido o especificado al menos los requerimientos funcionales, es importante que en esta subsección se presente alguna información de soporte como: ejemplos de los formatos de entrada / salida, descripciones de los estudios de análisis, resultados de encuestas a los usuarios, descripciones textuales, actas de reuniones, etc.

4. **Verificación:** Esta sección es la principal novedad del estándar ISO/IEC/IEEE 29148:2011, ya que al ver a los requerimientos de una manera integral, entonces dicho estándar propone que la verificación se vaya realizando en paralelo con la sección anterior. Es decir, que se vaya verificando cada uno de los requerimientos y a su vez documentando dicha verificación como parte de la SRS.
5. **Apéndices:** No es una sección que maneje información referente del contexto principal de la especificación, pero permite añadir información extra relacionada.
   - **5.1.** ***Supuestos y dependencias*:** Se debe enumerar la lista de factores que afectarían a los requerimientos del sistema en el caso de que no estén presentes. Por ejemplo, se asume que se dispondrá de una base de datos relacional; en el caso de que esta no exista, entonces necesariamente la SRS se verá afectada.
   - **5.2.** ***Acrónimos y abreviaturas*:** Esta subsección define una lista con el significado de los diferentes acrónimos y abreviaturas utilizados en el documento.

Finalmente, es importante comentar que si bien se debe verificar cada uno de los requerimientos, también es posible apreciar la calidad integral del documento utilizando indicadores generales como: tamaño, comprensibilidad, nivel de detalle y estructura del texto.

2.3.5. Validación de requerimientos de software

El propósito de esta etapa es chequear que los requerimientos estén definiendo el software correcto y por lo tanto que se encuentren alineados a las necesidades, deseos y expectativas de los usuarios. La validación debe incluir actividades que permitan comprobar la corrección, completitud, consistencia, realismo y verificabilidad de los requerimientos. (Dragicevic & Celar, 2013)

Dentro del proceso de requerimientos, la validación se superpone tanto a la fase de elicitación, como a la fase de especificación. Con respecto a la elicitación, en la sección de análisis se debe negociar con los usuarios y validar la calidad de los modelos conceptuales en fases tempranas, utilizando técnicas de validación como revisiones, prototipos, etc. La importancia de ir validando tempranamente, radica en que se debe intentar evitar errores que deriven en altos costos de modificación. Por otro lado, en la especificación y según el estándar ISO/IEC/IEEE 29148:2011, la verificación debe presentarse como parte del documento de especificación de requerimientos.

Técnicas de Validación

En el contexto de los requerimientos, una técnica se presenta como un procedimiento o conjunto de reglas que permiten comprobar que los requerimientos son los adecuados. Dentro de las principales técnicas de validación se encuentran:

- ***Revisiones***: Esta técnica, es una de las más comunes y consiste en una lectura, inspección y análisis sistemático de los requerimientos por parte de los revisores asignados. Lo revisores pueden ser identificados dentro de los diferentes stakeholders definidos en la fase de elicitación y análisis.

  En lo concerniente a tipos de revisiones, se plantea que si la inspección se realiza a nivel de analistas, es recomendable que se utilice una *revisión por pares*. Es decir, que el analista no revise los requerimientos que especificó, sino que inspeccione los requerimientos relevados por su colega y viceversa. Cuando es necesario que las revisiones sean conducidas por parte de instancias superiores, entonces se denominan *revisión de dirección* y en el caso de que las revisiones las realice personal externo para asegurar la objetividad en la revisión, entonces se denominan *revisiones de auditoría*.
- ***Prototipos***: También definidos como una técnica de elicitación, los prototipos tienen el propósito de ser un software construido tan rápido y económicamente como sea posible, para que en etapas de requerimientos los involucrados tengan una visión clara de sus necesidades mediante un modelo ejecutable que puedan probar y que haga énfasis en la interfaz de usuario.
- ***Generación de casos de prueba***: *Los requerimientos son el principio y el fin de del proceso de software*. Partiendo de esta premisa, se debe considerar que los requerimientos son la base de las áreas de conocimiento de diseño y de pruebas. Luego, los requerimientos estarán bien especificados si a partir de ellos se puede realizar un correcto diseño y además se puede construir casos de prueba. Es por tal motivo, que la construcción de casos de prueba en conjunto con los requerimientos es una buena práctica que permite a su vez realizar una validación implícita, ya que un requerimiento que posea un caso de prueba, se convierte en un requerimiento que se podrá testear en el futuro.

3. Descripción del problema

Este capítulo establece la justificación para la realización del presente trabajo de tesis. Es decir, es aquí en donde se presenta una serie de argumentos relacionados con la necesidad de integrar BPM en el proceso de requerimientos de software para la modernización de sistemas legados. La descripción del problema se ha dividido en cuatro partes: introducción, identificación de sistemas legados, necesidades de modernización en sistemas legados y el rol de los requerimientos en la modernización.

3.1. Introducción

Este capítulo presenta un conjunto de argumentos que permiten entender la necesidad de integrar BPM en el proceso de requerimientos para la modernización de sistemas legados. Dentro de estos argumentos y como inicio a la descripción del problema, se presenta la justificación del porqué se debe identificar a un sistema de software como legado y priorizarlo para su modernización. Una vez identificado el sistema de software como legado, entonces se debe justificar la modernización del mismo. En este sentido, se presenta un conjunto de necesidades que argumentan la modernización.

Por otro lado, en este capítulo se explica la relación existente entre el proceso de requerimientos y la modernización de sistemas legados. Esta relación, se basa en el estado del arte defino en el capítulo dos (Bourque & Fairley, 2014) (Sommerville, 2011) y se presenta con la finalidad de convertirse en el fundamento para demostrar *que las limitaciones de representación del modelo conceptual podrían resolverse mediante la integración de BPM en el proceso de requerimientos*.

3.2. Identificación de sistemas legados

En el estado del arte de los sistemas legados (sección 2.1), se presenta un conjunto de características, clasificaciones y metodologías inherentes a los sistemas legados. En este sentido, se debe utilizar los parámetros abstraídos en la Figura 3.1 y catalogarlos de tal forma que en conjunto puedan establecer eficientemente dos particulares asociadas a un sistema de software. En primer lugar, *identificar si es un software legado* y en segundo lugar *definir la prioridad* que tendría el mismo dentro de un plan integral de modernización (Khadka, et al., 2011).

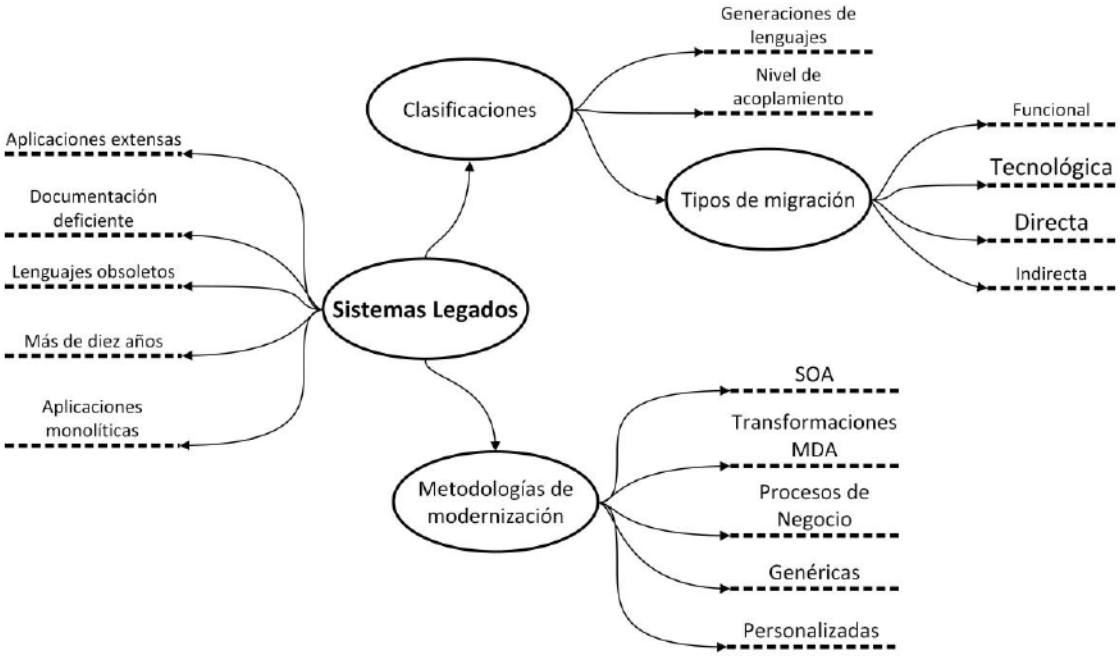


Figura 3.1: Sistemas legados

La modernización de sistemas legados es una práctica bastante compleja y costosa. Es por tal motivo, que antes de decantarse por una modernización, se debe generar un plan de modernización que comience identificando los sistemas de software legados que dispone la organización.

Para la creación de un plan de modernización se debe disponer de un *catálogo de servicios de software* actualizado, en donde se puedan detallar las características de los diferentes sistemas de software existentes.

Las características de los diferentes sistemas catalogados deberán estar asociadas a ciertos parámetros que permitan definir si un software es legado desde el punto de vista funcional o tecnológico (Matos & Heckel, 2009).

A continuación, un conjunto de posibilidades que se pueden utilizar para definir un sistema como legado:

- *Juicio experto*: Son opiniones, informes o resoluciones de profesionales que han sido consultados sobre el tema y que tienen amplia experiencia en la identificación de sistemas legados. Los expertos, pueden ser personas internas o en su defecto pueden ser entidades externas relacionadas con la organización mediante contratos de auditoría.
- *Decisiones políticas:* Desde la parte ejecutiva se definen planes operativos, normativas o reglamentos en donde se especifica el listado de aplicaciones de software que se deben modernizar.
- *Usuarios exigentes:* Muchas veces, las decisiones de modernización pueden verse afectadas por usuarios que *presionan* para que los sistemas relacionados con sus procesos o actividades tengan un trato preferencial. En teoría, no debe darse esta situación, pero la falta de directrices adecuadas para el proceso de identificación de software legado, hace que las decisiones tomadas en base a exigencias de usuarios terminen convirtiéndose en una práctica común.
- *Criterio técnico:* Esta catalogación debe estar apegada a criterios definidos o relacionados con las propiedades inherentes a los sistemas legados.

Como se puede apreciar en el listado anterior, la definición de un criterio técnico es la posibilidad más adecuada para identificar sistemas legados. En este sentido, la presente tesis y sobre todo el siguiente capítulo, propone un conjunto de criterios técnicos que permiten una identificación adecuada y objetiva.

Una vez analizados e identificados los diferentes sistemas de software legados, otro punto de discusión estará en la priorización de los mismos mediante cuatro niveles de acción básicos: dar de baja, mantenimiento normal, mantenimiento condicional, y modernización. (Dedeke, 2012)

3.3. Necesidades de modernización

Como se plantea en uno de los objetivos específicos de esta tesis, se evidencia la necesidad de identificar y priorizar los diferentes sistemas legados. Lograr esta categorización, significaría que se ha conseguido describir un conjunto de características propias de los sistemas legados y que las mismas se han convertido en parte de los sistemas analizados (Blunden, 2012) (Brodie & Stonebraker, 1995) (Gold, 1998). Es

decir, implícitamente se justifica la necesidad de modernización si solo se toma en cuenta las *características negativas* de un sistema legado, ya que existen ciertos atributos que pueden considerarse como ventajas desde el punto de vista financiero, funcional y procedimental. Puntualmente, las ventajas de un sistema legado pueden resumirse en:

- Posibilidad de funcionar por muchos años.
- Cumplimiento de un rol esencial dentro de las actividades de la empresa.
- Plasman los requerimientos funcionales de una manera adecuada.

En las siguientes subsecciones, se presenta las características que pueden considerarse como *desventajas* en un sistema legado y que a su vez se convierten en las *necesidades* que permiten argumentar una modernización.

3.3.1. Aplicaciones complejas

En este punto, la complejidad de una aplicación se presenta como *el volumen de funcionalidades, estructuras lógicas y de datos que se ejecutan bajo una misma aplicación*. En realidad, no es un problema que una aplicación sea compleja en tamaño, ya que no se puede acotar la dimensión de la misma, si las funcionalidades vienen determinadas desde la fase de requerimientos. El problema real, surge cuando una aplicación compleja carece de conceptos arquitectónicos de descomposición y/o modularización en su diseño e implementación. (Massari & Mecella, 2002) La falta de descomposición, hace que el sistema se vea como una sola pieza de software y por lo tanto no sea manejable desde puntos de vista más pequeños. Esta situación, deriva en problemas de comprensibilidad a nivel de código, de funcionalidad y de estructura que a la final se verán reflejados en costes de mantenimiento.

La complejidad de un software no debería determinarse de una manera arbitraria o empírica. Al contrario, se debe realizar una medición técnica, cuantificable y objetiva mediante la utilización de un conjunto de métricas probadas en la medición de la complejidad del software. Las métricas y mediciones dependen de ciertos criterios tradicionales que se encuentran relacionados en la Tabla 3.1 (Visconti, 2011).

Tabla 3.1: Medición de software

Criterios de Medición	Métricas
Tamaño	Líneas de código, Número de tokens, Caracteres efectivos, Número de funciones, Puntos función, Puntos de casos de uso
Estructura de Datos	Cantidad de datos, Variables vivas, Spans[36], Peso de la especificación de De Marco
Estructura Lógica	Número de decisiones, Complejidad ciclomática.

[36] Separaciones o amplitudes de variables

3.3.2. Documentación inadecuada

Quizá la documentación inadecuada o la falta de esta, sea la principal característica de los sistemas legados. La falta de documentación, hace que la lógica del negocio no este clara ni para los usuarios, ni para el equipo de mantenimiento del sistema. Esta situación, puede generar que se tema a un mantenimiento correctivo, adaptativo, preventivo o de mejora en el proceso (ISO/IEC, 2006).

La deficiencia en la documentación del software legado puede deberse a varias situaciones:

- *Falta de un marco de referencia de T.I*: Una de las mejores maneras de gestionar el Departamento de Tecnologías de Información de una organización en general, es mediante la utilización de marcos de referencia como COBIT[37], ITL[38], ISO 20000, etc. En el caso de que la organización además posea un equipo de desarrollo de aplicaciones, es importante definir otro tipo de marcos de referencia como el CMMI[39] para enfocarse en el desarrollo. En cualquiera de los casos y dependiendo del tipo de organización, la elección de un modelo de referencia precisa que se maneje una rigurosa documentación de procesos y productos relacionados con T.I.
- *Utilización de estándares empíricos*: Las diferentes fases del proceso de software generan artefactos y documentos que deben estar estandarizados de alguna manera. En el afán de lograr una documentación de software adecuada, el director del desarrollo crea diversos tipos de plantillas que se adecúan en ciertos casos a la información necesaria y conveniente para el proponente de la plantilla. Partiendo de este escenario, se puede dar el caso que exista en un futuro otro encargado del proceso que no crea pertinente la información recabada y por lo tanto puede considerar que la información es inadecuada y obsoleta. Para evitar estos inconvenientes, se debe mantener la objetividad a nivel de plantillas de documentación apegándose a estándares internacionales probados.
- *Gestión de proyectos inadecuada*: La gestión inadecuada parte de la falta de un proceso de gestión de proyectos que deviene en problemas como: alcances no realistas, modificaciones del alcance durante la ejecución del proyecto, personal no asignado adecuadamente, etc. Es decir, la falta de un proceso de proyecto se convierte en una bola de nieve que tiene como desenlace el retraso en la entrega de artefactos y por lo tanto los pocos recursos que quedan se direccionan a la implementación de los requerimientos funcionales, prestándose poca o ninguna atención a la generación de documentación del proyecto, proceso, productos y artefactos.
- *No existe un control de versiones*: Esta situación se vuelve crítica cuando se necesita realizar procesos de mantenimiento y no se dispone de herramientas de

[37] **COBIT**: Control Objectives for Information and related Technology
[38] **ITIL**: Information Technology Infrastructure Library
[39] **CMMI:** Capability maturity model integration

control de versiones que permitan hacer el seguimiento respectivo a los cambios y adaptaciones del sistema en torno a nuevos procesos y reglas de negocio de la organización.

3.3.3. Lenguajes o suites legados

Lo que pretende argumentar esta subsección es la necesidad de modernizar un sistema legado por el tipo de lenguaje en el que está implementado. (Langer, 2012)

En muchos casos, las argumentaciones dependerán puntualmente del lenguaje en que se implementa la solución. Es decir, las desventajas que presenta Fortran no son las mismas que COBOL, ANSI C, PowerBuilder u Oracle Developer Suite. Por esta razón, se presenta un conjunto de desventajas comunes de los sistemas implementados mediante lenguajes o suites obsoletos:

- *Adaptación a "nuevos" paradigmas*: En la mayoría de lenguajes obsoletos no se dispone de implementaciones como la programación orientada a objetos (Ortiz & Plaza, 2013) o la programación orientada a aspectos (Mortensen, et al., 2012). Esta situación, se da porque los lenguajes aparecen luego de surgir el paradigma y por lo tanto, cuando nacieron estos lenguajes todavía no se planteaba nuevas metodologías de soporte al diseño y construcción de aplicaciones.
- *Mayor dificultad en el rastreo de la lógica*: La dificultad se da sobre todo a nivel de redundancia en la codificación, ya que al carecer de conceptos de herencia y/o polimorfismo se repite muchas veces funcionalidades similares con diferentes identificadores a nivel de métodos, procedimientos o funciones.
- *Falta de capacitación formal:* La idea de un lenguaje es que vaya adaptándose a los cambios tecnológicos y aprovechar los avances o nuevas metodologías de desarrollo. Por lo tanto, las organizaciones o empresas propietarias de la suite, brindarán el soporte a sus productos por un tiempo limitado. Dentro de este soporte, se encuentra las capacitaciones y actualizaciones a las suites o lenguajes.

  La falta de capacitación formal, a su vez degenera en la falta de profesionales aptos para el mantenimiento de la aplicación y por lo tanto al existir una menor demanda puede ser muy costoso el reclutamiento de personal.
- *Aumento del riesgo a nivel de hardware:* Así como el software tiene su tiempo de soporte limitado, también el hardware tendrá un tiempo de vida útil, por lo tanto mantener la misma infraestructura para el mantenimiento de la aplicación puede conllevar un alto riesgo el momento que comience a fallar la infraestructura.
- *Ineficacia de nuevas infraestructuras*: Las aplicaciones desarrolladas con estos lenguajes podrían no adaptarse o no utilizar eficientemente las inversiones en nuevas infraestructuras ya que en ciertos casos, los lenguajes obsoletos pueden carecer de soporte para hilos, redes, asignación dinámica de memoria, nuevos sistemas operativos, cloud computing (Martinez & Bazán, 2013), etc. Luego, a

pesar de que se invierta ingentes recursos en infraestructura, la organización no tendrá un mejor rendimiento a nivel de aplicación.

3.3.4. Dispersión de la arquitectura de software

El término arquitectura que se pretende utilizar se refiera a los "*conceptos fundamentales o propiedades que relacionan a un sistema con su entorno en lo concerniente a elementos, relaciones, principios de diseño y evolución*" (ISO/IEC/IEEE, 2011). Es decir, las interrelaciones existentes con otras aplicaciones y la estructura contextual a nivel de subsistemas, módulos, tareas, etc. (Maier, et al., 2001)

La dispersión de la arquitectura, trata de reflejar dos puntos de vista relacionados con:

- *Modelo arquitectónico inadecuado*: Existen sistemas legados que no fueron concebidos con estilos arquitectónicos que eviten los problemas ocasionados por la falta de una clara distribución de componentes y de una efectiva separación entre la implementación de la aplicación y la capa de presentación. Un ejemplo claro de este problema son las aplicaciones desarrolladas mediante un estilo arquitectónico monolítico (Méndez & Tinetti, 2011).
- *Interrelaciones desordenadas*: En la búsqueda de mejorar la interacción y funcionalidad de un sistema legado, se pueden utilizar diferentes técnicas enmarcadas dentro de iniciativas como SOA, MDA, procesos de negocio, personalizaciones, etc. Si estas técnicas, han sido implementadas careciendo de una metodología en integral, en lugar de convertirse en una solución adecuada a largo plazo, pueden terminar convirtiéndose en un sistema con demasiados parches tanto a nivel tecnológico como a nivel funcional.

En ambos casos, la arquitectura puede llegar a ser un punto débil de los sistemas legados y por lo tanto se convierten en un argumento de peso y una necesidad real de la organización para proceder con la modernización.

3.4. Rol de los requerimientos en la modernización

Como introducción a esta sección, se puede echar un vistazo al estado del arte de los requerimientos de software (sección 2.3) en donde se muestran las relaciones existentes entre la elicitación, especificación y validación de los requerimientos de software, así como las metodologías, actores, estándares y actividades del proceso de requerimientos. (Bourque & Fairley, 2014) (Loucopoulos, 1995) (Wiegers, 2003)

Una motivación práctica para tener un proceso de requerimientos adecuado, se puede ejemplificar con las solicitudes de creación o mejora de aplicaciones, ya que el usuario solamente brinda una idea general de lo que quiere. Entonces, si los requerimientos provienen solamente de una conceptualización global, lo que sucederá es que el analista tendrá que comenzar a tomar decisiones con respecto a detalles que no se cubrieron por parte de los usuarios en la "idea general".

Dentro de la modernización (Baghdadi & Al-Bulushi, 2013), se presenta un problema similar al descrito en el párrafo anterior, ya que al convertirse el *sistema legado en la fuente principal de información dentro del proceso de requerimientos,* puede ocurrir que si bien se puede tener una idea general de lo que hace el sistema legado, en realidad no

se logre profundizar en los detalles funcionales que permitan una especificación adecuada.

Por otro lado, y esperando respaldar otro de los objetivos específicos detallados en el primer capítulo, se argumenta la necesidad de definir metodológicamente la captura de la información, datos o procesos a partir del sistema legado, para que estos datos o modelos conceptuales se conviertan en el punto de partida para la especificación y validación los requerimientos obtenidos.

3.4.1. Elicitación, análisis y modelo conceptual de un sistema legado

Entendiendo a la modernización desde un punto de vista de redefinición y/o refinamiento de las características de un sistema legado. Entonces, el rol de los requerimientos en la modernización iniciará desde la etapa de elicitación, ya que es aquí en donde se define el análisis, fuentes y técnicas de elicitación de requerimientos.

Desde el punto de vista específico de la elicitación, si bien el dominio del problema es la fuente transcendental en la captura de requerimientos, también es cierto que en los sistemas legados el dominio ha sido absorbido dentro del sistema y por lo tanto *la funcionalidad actual del sistema se convierte puntualmente en el usuario, stakeholder o fuente principal de elicitación* dentro de un proceso de modernización. Partiendo de esta premisa, se debe utilizar un conjunto de técnicas o métodos que permitan el análisis de los diferentes requerimientos y sobre todo su especificación de una manera estándar para que puedan ser validados adecuadamente.

Una vez definidas las fuentes de elicitación del sistema legado, se debe utilizar diferentes técnicas para obtener la información que permita el análisis respectivo de los requerimientos. (Hadar, et al., 2014) (El-Attar & Miller, 2012) (Antonelli, et al., 2012) Como una de las principales formas de realizar el análisis, se encuentran la representación de requerimientos mediante los diferentes ***modelos conceptuales***[40] (Bourque & Fairley, 2014); los mismos que *pretenden representar el funcionamiento actual del requerimiento, las entidades del dominio del problema y las relaciones del requerimiento con el mundo real*.

En la Figura 3.2, se puede apreciar como interviene el modelo conceptual dentro de la elicitación de requerimientos de software y sobre todo en su etapa de análisis. Además, este mapa conceptual resume el capítulo dos de esta tesis, en conjunto con las relaciones existentes entre las tres ideas principales: sistemas legados, BPM y requerimientos de software.

[40] **Modelos conceptuales**: BPM, casos de uso, flujos de datos, estados y metas

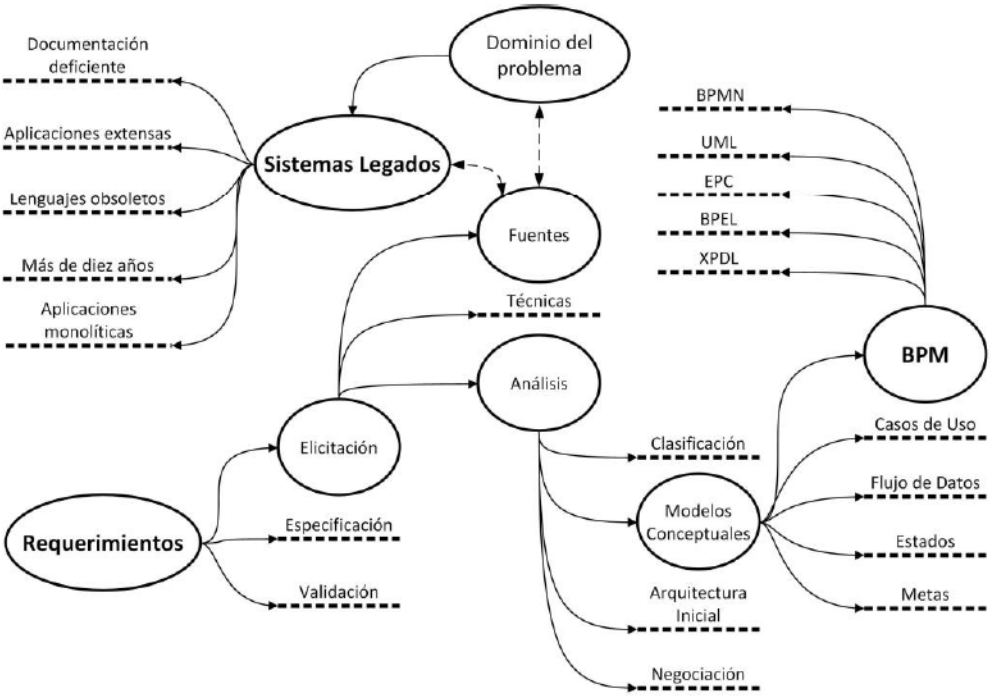


Figura 3.2: Rol de los requerimientos en los sistemas legados

3.4.2. Limitaciones del modelo conceptual

Entendiendo la relación existente entre el análisis de requerimientos de sistemas legados y el modelo conceptual, a continuación, se presenta las ventajas y desventajas generales de cada uno de los modelos conceptuales relacionados con: casos de uso, flujos de datos, estados y metas, ya que en teoría cualquier requerimiento o funcionalidad de un sistema legado podría representarse en alguno de los modelos conceptuales anteriormente citados. (Bourque & Fairley, 2014)

En esta subsección, además se presenta una conclusión por cada modelo conceptual. Conclusión que pretende resumir las *limitaciones de cada uno* de estos modelos en el contexto del análisis de requerimientos de software a partir de sistemas legados. La presentación de estas limitaciones preparará el camino para argumentar la necesidad de construir los modelos conceptuales de sistemas legados mediante la utilización de BPM en sus diferentes notaciones como: BPMN, UML, EPC, BPEL o XPDL.

- **Casos de Uso**: Sin lugar a dudas, los casos de uso son el modelo conceptual más utilizado dentro del análisis de requerimientos.
    - *Ventajas*
        - Amplia difusión dentro de la Ingeniería de requerimientos.
        - Facilidad de interpretación tanto desde el punto de vista del analista como del usuario.
        - Presenta o se enfoca en las actividades exactas que realiza el actor frente al proceso.

- *Desventajas*
  - No posee información de reglas de negocio como cálculos, decisiones y otros detalles de análisis.
  - No se puede analizar los tipos de datos que se pueden registrar en el caso de uso.
  - Cuando se pretende realizar un análisis de sistemas complejos mediante casos de uso, se puede terminar dificultando la descripción y entendimiento por el extenso número de documentación generada.
  - No existe un estándar oficial de referencia, más bien la diagramación de los casos de uso se han convertido en un estándar de facto.
  - Resulta complicado incluir relaciones entre los diferentes casos de uso.
- *Conclusión*: Si bien los casos de uso mostrarían de manera fácil la interacción entre el actor y el "sistema legado", también es cierto que dentro de las desventajas que presenta este tipo de modelo conceptual se encuentra la falta de detalle en: reglas de negocio, interacciones, tipos de datos, etc. Estos criterios son claves en un proceso de modernización, ya que no se puede profundizar en el requerimiento y por lo tanto no se podrá realizar una especificación de requerimientos de software adecuada.

- **Flujos de Datos**: Los Diagramas de flujo y los Diagramas de actividades (UML) son las principales muestras de este tipo de modelos conceptuales.
  - *Ventajas*
    - La línea gráfica permite una fácil comprensión del proceso.
    - Se detallan cada uno de los pasos del proceso y se puntualiza o documenta exactamente las condiciones o reglas de negocio.
  - *Desventajas*
    - En los diagramas de flujo, existen inconvenientes en la representación de diferentes caminos de una condición y sobre todo en la complejidad al tratar de representar todo el funcionamiento del negocio de esta forma.
    - Los diagramas de actividades están siendo reemplazados por los diagramas de procesos en BPMN, ya que en la actualidad, la OMG es la organización que trabaja en el desarrollo de ambas notaciones, pero decantándose por la segunda (OMG, 2011).
  - *Conclusión*: Se descartaría modelar mediante diagramas de flujo, ya que existiría demasiada complejidad de representación y además no se manifestaría la interacción del usuario con el proceso. Por otro lado, una forma adecuada de modelar los requerimientos de un sistema legado

serían los diagramas de actividades, pero ya que estos están siendo reemplazados por diagramas de procesos BPMN, quedaría también esta posibilidad como obsoleta.

- **Estados**: Los diagramas de estado UML son importantes para modelar la vida de un objeto. Es decir, el comportamiento de un objeto frente a estímulos externos.
    - *Ventajas*
        - Representan la condición que puede tomar un solo objeto y las transiciones del mismo.
        - Permite simbolizar las acciones, eventos y mensajes que pueda generar un cierto estado del objeto.
    - *Desventajas*
        - Son difíciles de leer sobre todo para los usuarios finales.
        - No permiten describir la interacción de colaboración entre objetos.
    - *Conclusión*: Como se puede apreciar en las viñetas anteriores, los diagramas de estados no representan tareas, actores, relaciones o reglas que permitan elicitar un sistema legado. Por esta razón, no se recomendaría utilizar este tipo de diagramas como base del modelo conceptual en el análisis.
- **Metas**: Si bien las metodologías basadas en metas como KAOS[41], GRAM[42] o i* pueden ser utilizadas en todo el proceso de requerimientos de software, en esta subsección solo se los abordará desde el punto de vista de modelos conceptuales para el análisis. (De la Vara, et al., 2007) (De la Vara, 2008) (Decreus, et al., 2009)
    - *Ventajas*
        - Las metas permiten definir el fundamento del sistema y las pautas principales que se deben cubrir.
        - Facilidad de definición por parte de los usuarios.
        - Motiva al usuario y demuestra la necesidad de la creación del sistema.
        - Identifica los actores y su relación con el ambiente y el sistema
        - Determinan las relaciones entre requerimientos mediante la construcción de escenarios.

[41] **KAOS**: **K**nowledge **A**cquisition in aut**O**mated **S**pecification
[42] **GBRAM**: **G**oal-**B**ased **R**equirements **A**nalysis **M**ethod

- *Desventajas*
  - Cada metodología dispone de su propia notación, lo cual ha hecho que estos modelos no se popularicen en el ámbito empresarial.
  - Demasiadas adaptaciones a las notaciones y al concepto en integral.
- *Conclusión*: Los modelos basados en metas tienen su punto fuerte en la definición de los objetivos y la motivación que genera demostrar el involucramiento de los mismos en el cumplimiento de las actividades diarias de los usuarios. Con relación a la elicitación de los sistemas legados, esta fortaleza quedaría sin piso ya que la funcionalidad definida por los objetivos se encuentra implementada en el sistema y por lo tanto la fuente principal de información no es un usuario común, sino el sistema legado en integral.

4. Propuesta de solución

En este capítulo, se presenta la propuesta del autor para solventar algunos problemas referentes a los sistemas legados, especificar sus requerimientos y responder a las limitaciones de ciertos modelos conceptuales para representar procesos dentro del análisis. Para plantear dicha solución, se presenta un método dividido en dos etapas: la primera permite la identificación de un sistema legado y la segunda presenta las acciones propuestas para analizar y especificar requerimientos utilizando modelos conceptuales con BPMN2.

4.1. Método de integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados

El método de integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados, es una propuesta desarrollada a partir de una extensa investigación que aspira responder a las dificultades planteadas en el capítulo anterior y que en la praxis se convierte en una serie de pasos puntuales que se deberían tener en cuenta tanto por parte de los jefes de un proyecto, así como por parte de los diferentes ingenieros de software que pretendan dar una solución a los problemas que se van presentando en la identificación y modernización de los sistemas legados de una organización.

La propuesta en integral, se fundamenta en la idea de presentar una modernización como un conjunto de actividades bien definidas y que puedan ser consideradas como un proceso. Esta idea puede verse plasmada por diferentes autores, aunque centrada en temas puntuales como: SOA (Khadka, et al., 2011) (Baghdadi & Al-Bulushi, 2013) (Zhang, et al., 2010) (Xu, 2010) (Cruz, et al., 2013), transformaciones MDA (Zhang, et al., 2009) (Huang & Chu, 2012), procesos de negocio (do Nascimento, et al., 2009) (De la Vara, et al., 2009) (Bazan, et al., 2012) y modernizaciones genéricas (Langer, 2012) (Dedeke, 2012). En esta tesis, se toma la idea de dichos autores y se plantea un método que permite a más de la modernización, la identificación previa de un sistema como legado mediante la interacción de dos etapas denominadas: ***identificación de sistemas legados*** e ***integración de BPM en el proceso de requerimientos***.

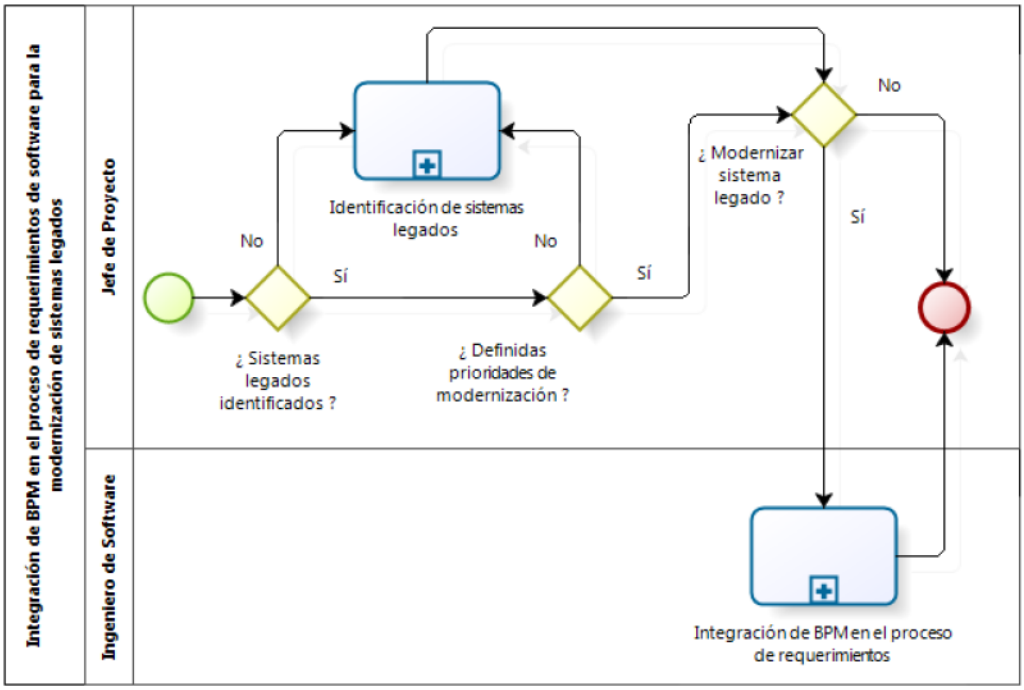


Figura 4.1: Propuesta de solución

En la Figura 4.1, se presenta un diagrama de procesos BPMN2 que sintetiza la propuesta de este capítulo y que se centra en dos actores, dos subprocesos y en tres decisiones exclusivas.

- **Subprocesos**
  - *Identificación de sistemas legados*: Este subproceso permite definir dos tareas principales: catalogar sistemas de software y especificar las prioridades de modernización para cada sistema catalogado.
  - *Integración de BPM en el proceso de requerimientos*: Este subproceso de la propuesta define un conjunto de tareas a seguir para obtener como salida una especificación de requerimientos de software.
- **Decisiones exclusivas**
  - *¿Sistemas legados identificados?*: El proceso general inicia con una condición, ya que se puede omitir el primer subproceso o una parte del mismo, en el caso de que hubiese existido alguna manera en la cual se haya decidido o se haya catalogado los diferentes sistemas legados de la organización.
  - *¿Definidas prioridades de modernización?*: En el caso de que se haya identificado los sistemas legados, esto no significa que se haya decidido modernizarlos. Por esta razón, esta consulta permite verificar que se haya definido prioridades de modernización como: dar de baja, mantenimiento normal, mantenimiento condicional y modernización. (Dedeke, 2012)
  - *¿Modernizar sistema legado?*: Solamente en el caso de que esta condición se evalúe como positiva se procederá con la ejecución del segundo subproceso. La evaluación positiva de esta condición significa que se ha tomado la decisión de modernizar el sistema legado. Caso contrario, se terminaría el proceso general de la propuesta de este capítulo.
- **Actores**
  - *Jefe de proyecto:* Actor o actores encargados de llevar a cabo el subproceso de Identificación de sistemas legados y sobre todo de tomar las decisiones exclusivas en torno a la interacción entre los dos subprocesos.
  - *Ingeniero de software:* Actor o actores responsables del subproceso de Integración de BPM en el proceso de requerimientos. Con respecto a este actor, es necesario precisar que ser el responsable del proceso no significa que desarrollará todas las actividades, ya que en realidad dentro del subproceso existen más actores como los analistas de negocios que se encargarán de la definición de los procesos o los analistas de software que se encargarán de la especificación de los requerimientos.

Este capítulo, no pretende definir exactamente cómo responder a las diferentes decisiones exclusivas presentadas, simplemente muestra las relaciones existentes entre los subprocesos y sobre todo detalla el funcionamiento de cada uno de ellos mediante la división del método en dos etapas:

- **Etapa de identificación de sistemas legados**
  - *Definir parámetros de identificación*: Esta actividad es la encargada de mantener la objetividad al definir si un sistema es legado o no, ya que es aquí en donde se debe especificar cuáles son los parámetros que permitirán identificar a los diferentes sistemas de software de la organización.
  - *Catalogar sistemas de software*: Dependiendo de los parámetros de identificación especificados, se debe catalogar y llevar un control de todos los sistemas de la organización. Este catálogo actualizado, será el referente que permita identificar exactamente cuáles son los sistemas legados y además será la base para especificar las prioridades de modernización.
  - *Especificar prioridades de modernización*: Es necesario que una vez identificados los sistemas legados se tome alguna acción al respecto. Dentro de las acciones posibles, en el presente trabajo de tesis se ha precisado cuatro niveles de acción: dar de baja, mantenimiento normal, mantenimiento condicional, y modernización. (Dedeke, 2012)

    Especificar prioridades de modernización se convierte en una tarea iterativa, ya que la misma debe ejecutarse por cada uno de los sistemas de información identificados en la etapa anterior.

- **Etapa de integración de BPM en el proceso de requerimientos**
  - *Identificar fuentes de conocimiento*: Esta actividad es el primer paso a tomar en cuenta para la definición de los procesos del sistema legado. Es aquí en donde se debe identificar la mayoría de fuentes válidas de información como: sistema legado en producción, usuarios expertos, reglamentos, políticas, etc.
  - *Identificar arquitectura funcional*: Esta actividad debe comenzar identificando la arquitectura genérica asociada al sistema legado. Dentro de las principales arquitecturas genéricas de software se encuentran: monolíticas, distribuidas, interactivas y adaptables (Bourque & Fairley, 2014). En segundo lugar y como el principal aporte de esta actividad a la etapa de integración, se encuentra la identificación de los diferentes subsistemas, módulos, tareas, librerías y relaciones con el entorno funcional.
  - *Agrupar o segmentar funcionalidades*: Las funcionalidades se refieren a las tareas que puede realizar el sistema legado y que se encuentran formando parte de los diferentes módulos identificados en el paso anterior. *Agrupar,* significa organizar las funcionalidades del sistema legado que se encuentran formando parte de un mismo proceso a pesar de encontrarse en diferentes tareas dentro de un módulo, mientras que por otro lado, *segmentar* significa separar funcionalidades que se encuentren en un mismo formulario, pero que pueden ser fraccionadas en varios procesos. Esta actividad, dará la pauta para que se puedan

diseñar los procesos que han sido agrupados o segmentados a partir del sistema legado.

- *Diseñar procesos asociados a funcionalidades*: Como su nombre lo indica, en esta sección de la etapa de integración se lleva a cabo el diseño de las funcionalidades de los sistemas legados como procesos de negocio; los mismos que serán representados en BPMN2 y que serán validados para que permitan el posterior desarrollo de una especificación de requerimientos de software. Esta tarea, es la más compleja y extensa de la segunda etapa, ya que a su vez consta de cuatro pasos denominados: proceso básico, asociación de participantes, diseño detallado y validación del proceso. (Langer, 2012) (do Nascimento, et al., 2009)
- *Especificar requerimientos del sistema legado*: Esta es la última fase de la segunda etapa y permite obtener como producto final una Especificación de requerimientos de software (SRS) levantada a partir del análisis de los procesos de negocio identificados en la etapa anterior. La SRS final (ISO/IEC/IEEE, 2011) es una extensión de la estructura IEEE definida en el capítulo dos (sección 2.3.4).

4.2. Etapa de identificación de sistemas legados

El objetivo principal de esta etapa es identificar los sistemas legados de la organización, mediante la definición de un conjunto de parámetros que permiten la creación de un *Catálogo de Sistemas de Software;* catálogo que permite a su vez identificar cuáles sistemas se consideran legados y si dichos sistemas necesitan además entrar en un plan de modernización.

En la Figura 4.2, se muestra un diagrama de procesos que indica las tres tareas involucradas, las interrelaciones entre estas y un producto de trabajo denominado en esta tesis como Catálogo de Sistemas de Software.

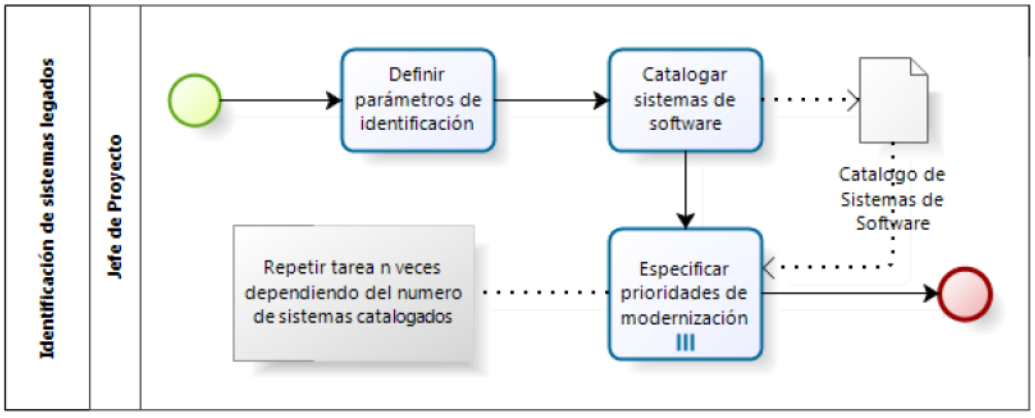


Figura 4.2: Etapa de identificación de sistemas legados

Antes de continuar con el detalle de cada una de las actividades involucradas en el proceso, es importante precisar que esta etapa amalgama aportes creativos del autor en

lo referente a parámetros de identificación, la estructura ISO oficial ISO/IEC/IEEE 29148:2011 (ISO/IEC/IEEE, 2011) para catalogar los sistemas de software y las prioridades de modernización presentadas por Adrew Sage y citadas por Adenekan Dedeke (Dedeke, 2012).

4.2.1. Definir parámetros de identificación

Dentro de la primera etapa del método, un parámetro es considerado como un valor variable que permite almacenar una característica específica de un sistema de información. El código asignado a un sistema, su descripción detallada, versión o número de tablas, pueden ser considerados como parámetros de identificación.

Esta tesis, plantea que la definición de un conjunto de parámetros permitirá la identificación objetiva y uniforme de los diferentes sistemas de información de una organización. Por otro lado, todos los parámetros que se definan deben estar relacionados con una *categoría tecnológica*, así como con *categorías transversales* de parámetros. Ambas categorías son aportes innovadores de la tesis, ya que tratan de plantear de una manera formal las experiencias empíricas del autor.

En la Figura 4.3 se muestra la relación existente entre los parámetros y las dos categorías presentadas. Entendiendo que un parámetro debe tener o estar asignado a una categoría transversal específica. Por otro lado, un parámetro podría ser utilizado opcional u obligatoriamente en varias categorías tecnológicas, mientras que una categoría tecnológica puede tener varios parámetros asociados (*Parámetro por Categoría Tecnológica*).

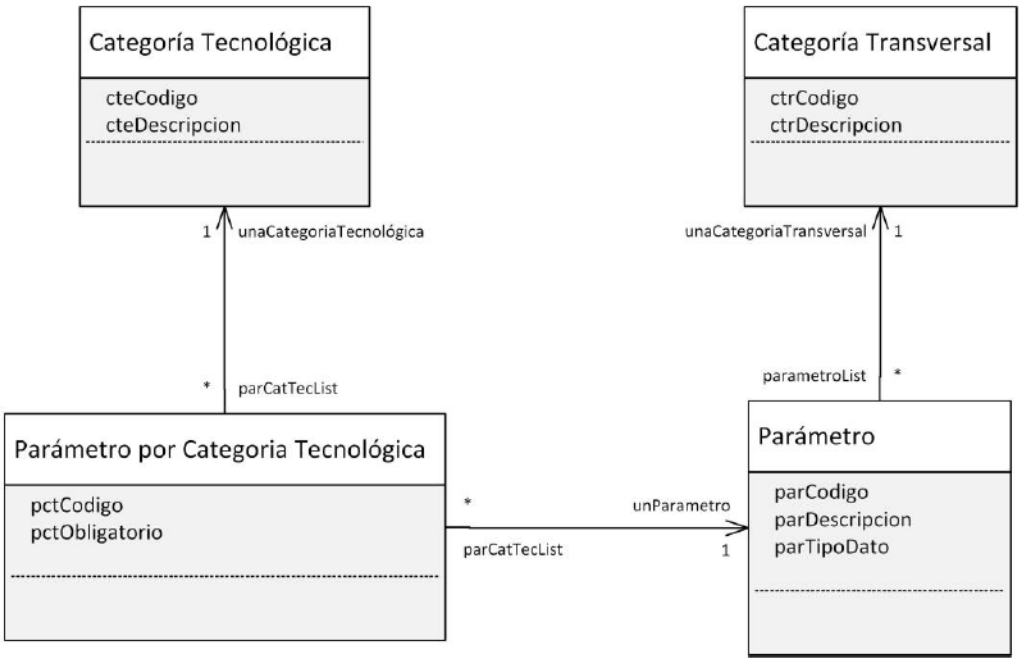


Figura 4.3: Categorías de parámetros de identificación de sistemas legados

Categorías tecnológicas

Una categoría tecnológica, consiste en la descripción de la principal plataforma de implementación de un grupo de sistemas o aplicaciones. Es decir, el Jefe de proyecto debe identificar las diferentes plataformas tecnológicas que dispone la organización y realizar una clasificación somera de las diferentes aplicaciones.

En la Tabla 4.1 se presenta un ejemplo de Categorías tecnológicas definidas por el Jefe del proyecto de modernización de una organización.

Tabla 4.1: Ejemplo de categorías tecnológicas

Categorías tecnológicas
Aplicaciones COBOL
Aplicaciones Foxpro
Aplicaciones Forms & Reports
Aplicaciones PHP
Aplicaciones APEX
Aplicaciones JSF+EJB+JPA

Si bien pueden existir otros tipos de clasificaciones, como una identificación funcional o una identificación por responsables. Se ha preferido utilizar las categorías tecnológicas como primera clasificación, ya que es una teoría de este trabajo de tesis y además permite identificar de manera fácil y rápida el carácter de obsolescencia de los sistemas de información.

Categorías transversales

Las categorías transversales hacen referencia al conjunto de parámetros que posee un sistema o aplicación de software. Esta tesis, plantea cuatro categorías transversales denominadas: *identificación funcional, proceso de software, características técnicas y medición de artefactos*. A continuación, un listado de parámetros elegibles asociados a cada una de las categorías:

- ***Identificación funcional**:* Son parámetros independientes de la categoría tecnológica y que en conjunto permiten tener una lectura rápida de la aplicación. Dentro de esta categoría de parámetros pueden encontrarse:
    - **Código:** La aplicación debe tener asignado un código único; el mismo que permitirá una lectura más ágil en declaraciones posteriores.
    - **Nombre:** Se define un nombre relacionado a las principales funciones que realiza. En lo posible, se debe utilizar nombres únicos pero puede utilizarse nombres compuestos si el caso lo amerita.
    - **Versión:** Si se mantiene un control de cambios, entonces es aquí en donde se indica la versión de la aplicación en producción.

- **Sistema:** De ser una aplicación parte de un sistema general, este campo corresponde al nombre del sistema al que pertenece.
- **Descripción corta:** Es una descripción opcional y se puede utilizar para especificar casos en los que el nombre del sistema sea un acrónimo. Por ejemplo, si la aplicación se llama PCI, entonces en este parámetro se puede escribir Sistema de producción científica.
- **Descripción detallada:** Esta descripción detalla las funcionalidades de la aplicación y ciertos aspectos relevantes que se necesiten describir para una compresión integral de la aplicación.
- **Ruta de acceso en producción:** En este campo se debe definir la ruta en la cual se accede a la aplicación en producción.
- **Fecha en producción:** La fecha en la que la aplicación entró en producción. Para la definición de esta fecha, no es necesario identificar si el proceso de software, la documentación o la capacitación fueron las pertinentes.
- **Ubicación física de la documentación:** La ruta de la documentación oficial relacionada a la aplicación.
- **Ubicación física de las fuentes:** La ruta de los códigos fuentes y/o ejecutables de la aplicación.
- **Responsable:** Se debe definir la persona que estuvo a cargo de la puesta en producción de la aplicación.
- **Fecha última modificación:** La fecha en que se ha registrado la última modificación de la aplicación.
- **Modificado por:** Se debe indicar la persona que realizó la última modificación a la aplicación.

- ***Proceso de software**:* De este conjunto de parámetros depende la calidad del proceso de software que tiene la aplicación. Se ha definido que la aplicación tenga un valor cuantitativo en esta categorización ya que dependiendo de cada ítem, se puede definir un valor numérico de cero (0) a cuatro (4). Para la definición del proceso de software, se utilizó cinco áreas definidas en el Cuerpo del conocimiento de la Ingeniería de Software (Bourque & Fairley, 2014).
  - **Documentación de Requerimientos (REQ):** Este ítem representa que tan documentado se encuentran los requerimientos funcionales, no funcionales y reglas de negocio:

    0: No existe documentación de ningún tipo de requerimientos y las reglas del negocio se encuentran embebidas en la aplicación**.**

    1: Existe documentación relacionada con requerimientos funcionales; estos documentos pueden ser descripciones textuales.

    2: Existe documentación relacionada con requerimientos funcionales; estos documentos pueden ser casos de uso y/o descripciones textuales.

3: Existe documentación detallada de los requerimientos funcionales que especifican las reglas de negocio.

4: La documentación de requerimientos se encuentra definida mediante estándares oficiales y se mantiene actualizada mediante un correcto control de cambios.

- **Documentación de Diseño (DIS):** La documentación del diseño puede considerarse el corazón de los sistemas transaccionales, ya que define las estructuras de almacenamiento de la información a nivel de base de datos o muestra las funcionalidades de los objetos de ser el caso. En lo correspondiente a la calidad de la documentación se han definido cinco escalas:

  0: No existe documentación de ningún tipo de diseño de datos o de funcionalidades.

  1: Existen documentos relacionales con la estructura a nivel de base de datos como puede ser un diagrama entidad relación.

  2: Existe documentación detallada de la estructura de datos y un diccionario de datos que describe cada estructura.

  3: Existe documentación de interacción entre métodos y funcionalidades del sistema, así como la implementación de las reglas de negocio.

  4: La documentación de diseño se halla definida mediante estándares oficiales y se mantiene actualizada mediante un adecuado control de cambios.

- **Prácticas de desarrollo (DES):** En realidad, las buenas prácticas de desarrollo dependerán del lenguaje o lenguajes de programación definidos para la implementación de la aplicación. En este caso, las escalas que se presentan a continuación están definidas principalmente en lo concerniente a la compresión o legibilidad del código:

  0: El código está desordenado y no aplica estándares a los nombres de variables, métodos, procedimientos, etc.

  1: El código sigue estándares regulares a nivel sintáctico, pero no documenta el código, solamente las reglas de negocio más importantes.

  2: El código fuente sigue estándares en nombres y además documenta claramente el código fuente, aunque el mismo tiene personalizaciones que en ocasiones tiende a generar código complicado de entender, sobre todo porque no se apega a la documentación de requerimientos y de diseño.

  3: Existe una fuerte concordancia entre el código fuente y la documentación de requerimientos y de diseño.

  4: Existe un estándar o se utiliza una herramienta para el seguimiento de control de versiones.

- **Proceso de pruebas (PRU):** Un proceso de pruebas adecuado debe considerar tres tipos de pruebas: unitarias, funcionales y no funcionales. La capacidad cuantitativa del manejo de este proceso puede ser reflejada como:

  0: No existe una documentación o registro de que haya existido al menos pruebas funcionales.

  1: Existe un registro por medio de correos electrónicos o cartas de aceptación del usuario en los cuales se refleje que el sistema en producción cumple con las características funcionales requeridas.

  2: Existe un documento oficial que muestre al menos casos de prueba funcionales.

  3: Se ha realizado y documentado pruebas unitarias y funcionales registradas en un documento oficial.

  4: Mediante algún estándar definido se tiene un plan de pruebas considerando casos funcionales, no funcionales y pruebas unitarias.

- **Proceso de mantenimiento (MAN):** Cuando el sistema genera una incidencia (nuevo requerimiento, mejora o error) y debe ser actualizado por el equipo de desarrollo, entonces se procede a la respectiva actualización mediante una de estas estas formas:

  0: No existe forma de realizar el mantenimiento frente a una incidencia presentada.

  1: Se comunica la incidencia informalmente y es solucionado por el programador a cargo del proyecto o con más experticia sobre la aplicación.

  2: Se realiza una comunicación oficial que pasa por un proceso implícito de asignación de responsables y prioridades.

  3: Existe un sistema de gestión de incidencias que permite la asignación de responsabilidades y prioridades.

  4: Existe un proceso oficial y automatizado definido para la trazabilidad, seguimiento y solución de incidencias. Además, se mantienen actualizados los requerimientos, diseños y versiones del software.

- ***Características técnicas**:* En este punto, se vuelve a depender de las *categorías tecnológicas* que se hayan definido, ya que las características técnicas son diferentes para cada tipo de sistema. Es decir, el Jefe de Proyecto o responsable deberá definir los atributos o parámetros técnicos que se necesite para cada característica tecnológica definida.

  A continuación, se presenta a manera de ejemplo, los atributos técnicos para dos características tecnológicas: Aplicaciones Forms & Reports y Aplicaciones JSF+EJB+JPA.

  Aplicaciones Forms & Reports:

  - **Ruta del servidor de aplicaciones:** Es la IP y/o DNS en donde está instalado el servidor de aplicaciones.

- **Información del servidor de aplicaciones:** Se debe especificar el nombre y versión del servidor de aplicaciones en el cual están desplegados los formularios y los reportes.
- **Ruta del servidor de reportes:** Nombre de la dirección IP y/o DNS en donde se instalan los servicios del servidor de reportes.
- **Información del servidor de reportes:** Nombre y versión del servidor de reportes.
- **Base de datos:** Se debe especificar el nombre, edición y versión de la base de datos utilizada.
- **Ruta de los formularios en producción:** Dirección física de las fuentes de los formularios de aplicación.
- **Ruta de los reportes en producción:** Dirección física de las fuentes de los reportes de la aplicación.
- **Librerías extras:** Si los formularios o reportes de la aplicación en general necesitan librerías extras, entonces, esta sería la sección en donde se debe definir las mismas.
- **Formularios relacionados a la aplicación:** Listado de nombres de los formularios relacionados con la aplicación.
- **Reportes relacionados a la aplicación:** Listado de nombres de los reportes relacionados con la aplicación.

Aplicaciones JSF+EJB+JPA

- **Ruta del servidor de aplicaciones**: Es la IP y/o DNS en donde está instalado el servidor de aplicaciones.
- **Información del servidor de aplicaciones**: Se debe especificar el nombre y versión del servidor de aplicaciones en el cual está desplegada la aplicación.
- **Ruta del servidor de reportes:** Nombre de la dirección IP y/o DNS en donde se instalan los servicios del servidor de reportes.
- **Información del servidor de reportes:** Nombre y versión del servidor de reportes.
- **Base de datos:** Se debe especificar el nombre, edición y versión de la base de datos utilizada.
- **Nombre del DataSource:** En este campo se debe especificar el valor correspondiente al Pool y el JNDI.
- **Lenguaje de programación:** En este parámetro se pueden definir varios lenguajes de programación, aunque sería conveniente que se defina el lenguaje principal.
- **Framework para capa de vista:** En el caso de existir, en esta sección se define el nombre y versión de framework utilizado.
- **Implementación JPA:** Si la aplicación maneja persistencia, entonces se necesita definir el nombre y la versión de la implementación.

- **Librerías extras:** En este tipo de aplicaciones web, es común utilizar librerías extras para ciertas funcionalidades. Es por tal razón que es fundamental tener esta información actualizada al respecto.

- ***Medición de artefactos**:* En esa categoría se definirán los parámetros asociados a los diferentes artefactos que se podrán medir dependiendo de la característica tecnológica definida.

  Aplicaciones Forms & Reports

  - **Número de formularios:** Se refiere al número de archivos con extensión .frm asociados a la aplicación.
  - **Número de reportes:** Es el número de archivos con extensión .rdf asociados a la aplicación.
  - **Número de tablas:** Número de tablas a nivel de base de datos relacionados a la aplicación.

  Aplicaciones JSF+EJB+JPA

  - **Número de paquetes:** El número de agrupaciones de clases, pueden ser los paquetes en Java.
  - **Número de clases entidades:** La contabilización de todas las clases que manejen la persistencia de la base de datos ya sea mediante anotaciones u otro tipo de metodología.
  - **Número de clases bean:** Clases que interactúan con la lógica del negocio. Como ejemplo en java estas clases son definidas como @Stateless.
  - **Número de clases controladores:** Clases que interactúan con las páginas de la aplicación. Como ejemple en java este tipo de clases se definen como @ManagedBean.
  - **Número de clases:** El número de clases extras que se estén utilizando dentro de la aplicación.
  - **Número de reportes:** Número de archivos de reportes asociados a la aplicación
  - **Número de tablas:** Número de tablas a nivel de base de datos relacionados a la aplicación.
  - **Número de formularios/páginas:** Número de páginas creadas para el mantenimiento de la aplicación. Cuando una página es compuesta de varias páginas cuenta como una sola.

En resumen, la definición de los parámetros de identificación dependerá en gran medida de los activos de software que disponga la organización, pero sobre todo del criterio de selección del Jefe de Proyecto en torno a los lineamientos presentados en esta sección de la etapa.

Por último, recordar la importancia de definir el número de parámetros suficientes que permitan representar todas y cada una de las aplicaciones existentes.

4.2.2. Catalogar sistemas de software

Una vez definidas las categorías y parámetros de identificación de las diferentes aplicaciones, se debe proceder a recolectar toda la información referente y relacionada con dichas aplicaciones.

Esta sección, no pretende detallar ***cómo*** recolectar la información, sino ***qué*** información recolectar y como formalizarla de tal manera que permita la creación de un producto denominado *Catálogo de Sistemas de Software;* el mismo que debe incluir las valoraciones de cada aplicación en torno a los parámetros definidos en la primera fase de la etapa.

1. Introducción
   1.1. Propósito
   1.2. Alcance
   1.3. Control de cambios
   1.4. Definiciones, Acrónimos y Abreviaturas
   1.5. Referencias
   1.6. Visión General del Documento
2. Definición de parámetros de identificación
   2.1. Categorías Tecnológicas
   2.2. Categorías Transversales
   2.2.1. Identificación funcional
   2.2.2. Proceso de software
   2.2.3. Características técnicas
   2.2.4. Medición de artefactos
3. Catálogo de sistemas
   3.1. Categoría Tecnológica 1
   3.1.1. Aplicación 1.1
   3.1.2. Aplicación 1.2
   3.1.m. Aplicación 1.m
   ...
   3.n. Categoría Tecnológica n
   3.n.1. Aplicación n.1
   3.n.2. Aplicación n.2
   3.n.m. Aplicación n.m
4. Apéndices

Figura 4.4: Catálogo de Sistemas de Software

En la Figura 4.4, se presenta la estructura del documento entregable, pero a continuación, se detalla cada una de las secciones que debe incluir el Catálogo de Sistemas de Software:

1. **Introducción**
   - **1.1.** ***Propósito***: A pesar de que parezca implícito la creación del documento, es necesario que se detalle los antecedentes y contexto en el cual se realiza la creación del documento.
   - **1.2.** ***Alcance***: Esta subsección debe especificar exactamente qué información está contenida en el documento y que información quedará fuera del mismo.

1.3. ***Control de cambios***: En esta sección se debe indicar las diferentes versiones del documento teniendo como principales características la fecha, versión, descripción y responsable del cambio.

1.4. ***Definiciones, Acrónimos y Abreviaturas***: Debe existir un listado con el significado de todas las definiciones, acrónimos y abreviaturas que hayan estado presentes dentro del documento.

1.5. ***Referencias***: Es una lista completa de todos los documentos asociados, citados o consultados durante la creación del Catálogo de Sistemas de Software.

1.6. ***Visión General del Documento***: En esta subsección, no se debería detallar, pero si presentar un resumen general del documento y de cada una de sus secciones.

2. Definición de parámetros de identificación

En esta sección del Catálogo de Sistemas de Software, se debe registrar todos los parámetros de identificación definidos y organizados de acuerdo a las categorías tecnológicas y transversales presentadas en el punto 4.2.2 de la presente tesis.

2.1. ***Categorías Tecnológicas***

2.2. ***Categorías Transversales***

2.2.1. Identificación funcional

2.2.2. Proceso de software

2.2.3. Características técnicas

2.2.4. Medición de artefactos

3. Catálogo de sistemas

Esta sección es la más extensa del documento, ya que es aquí en donde se deben registrar todas y cada una de las aplicaciones de la organización según la categoría tecnológica a la que pertenezcan.

Dependiendo de cada categoría tecnológica se creará una subsección por cada una. Ejemplo 3.1 Aplicaciones COBOL, 3.2 Aplicaciones Forms & Reports, 3.3 Aplicaciones APEX, etc.

Dentro de cada subsección, se debe registrar cada aplicación en conjunto con los valores correspondientes a los parámetros de identificación definidos para dicha categoría.

En la Tabla 4.2 se ejemplifica los datos recolectados de una aplicación de software con la categoría tecnológica de Aplicaciones JSF+EJB+JPA.

Tabla 4.2: Identificación de una aplicación de software

Identificación Funcional	
Código	1.1
Nombre	Ficha Socioeconómica
Versión	No dispone
Sistema	Sistema Nacional de Inscripciones
Descripción Corta	Ficha socioeconómica
Descripción Detallada	Permite el ingreso de la ficha socioeconómica de los estudiantes

Ruta Producción	services.ups.edu.ec/sdjf
Fecha Producción	lunes, 03 de enero de 2011
Ubicación Documentación	No aplica
Ubicación Fuentes	\\172.16.1.159\FuentesAplicaciones\services.ups.edu.ec\FichaSocioeconomica
Responsable	Ing. Juan Pérez
Proceso de Software	
Análisis y Requerimientos (REQ)	1: Existen documentos relacionados con requerimientos funcionales como casos de uso o descripciones textuales.
Documentación de Diseño (DIS)	1: Existen documentos relacionales con la estructura a nivel de base de datos como puede ser un diagrama entidad relación.
Prácticas de desarrollo (DES)	1: El código sigue estándares regulares a nivel sintáctico, pero no documenta el código, solamente las reglas de negocio más importantes.
Proceso de pruebas (PRU)	1: Existe un registro por medio de correos electrónicos o cartas de aceptación del usuario que el sistema en producción cumple con las características funcionales requeridas.
Proceso de mantenimiento (MAN)	2: Se realiza una comunicación oficial que pasa por un proceso implícito de asignación de responsables y prioridades.
Fecha última modificación	viernes, 31 de enero de 2014
Responsable	Ing. Juan Pérez
Característica Técnicas	
Ruta del servidor de aplicaciones:	http://services.ups.edu.ec/sbe
Información del servidor de aplicaciones:	Glassfish Community Edition 3.0.1 build 22
Ruta del servidor de reportes:	No dispone
Información del servidor de reportes:	No dispone
Base de datos:	Oracle Edición Estándar 11.2.0.4
Nombre del DataSource	jdbc/sdjfpro
Lenguaje de Programación	Java, JSF, XHTML, Javascript
Implementación JPA	EclipseLink
Librerías extras	cas-client-3.2.0
	jasperreports-3.7.6
	itext-2.1.7
Medición de Artefactos	
Número de Paquetes	5
Número de Formularios / Páginas	5
Número de Clases Entidades	0
Número de Clases Bean	1
Número de Clases Controladores	2
Número de Clases	30
Número de Reportes	6
Número de Tablas	14

4. Apéndices

Esta es una sección opcional en la cual se puede registrar cualquier información que pueda aportar con detalles al documento en integral.

4.2.3. Especificar prioridades de modernización

Esta es la actividad final de la etapa de identificación de sistemas legados y consiste en la evaluación de cada una de las aplicaciones del *Catálogo de Sistemas de Software*. Esta evaluación se basa en un cuadrante de análisis compuesto de dos ejes: valor técnico y valor de negocio.

El cuadrante que se utiliza es propuesto en un principio por Andrew Sage y vuelve a ser presentado en un trabajo de Adenekan Dedeke (Dedeke, 2012). En esta tesis, se extiende dicho cuadrante para que pueda soportar escalas de valores que permitan una medición cuantitativa para el establecimiento de las prioridades de modernización.

Valor Técnico

El valor técnico de un activo de software puede ser definido cuantitativamente de acuerdo a cuatro parámetros:

- ***Calidad de características* CC**: Un sistema con características de alta calidad podría ser modificable o funcionalmente adecuado a los requerimientos y reglas de negocio.
    - 1: No cumple con los requerimientos ni reglas de negocio.
    - 2: Cumple con los requerimientos actuales, pero se avizora nuevos requerimientos que no se podrían implementar.
    - 3: Cumple con los requerimientos actuales, los cuales no sufrirán modificación a corto y mediano plazo. En el caso de que existan nuevos requerimientos, no se podría incrementarlos.
    - 4: Cumple con los requerimientos actuales, pero se avizora nuevos requerimientos que el sistema estaría en la capacidad de implementar.
    - 5: Cumple con los requerimientos actuales y podría soportar nuevos requerimientos a pesar de que en la actualidad no se estime modificaciones a corto y mediano plazo.
- ***Fiabilidad del servicio* FS**: Se refiere a la continuidad del servicio prestado por el sistema. Una aplicación que esté continuamente en mantenimiento tendrá una puntuación menor de fiabilidad.
    - 1: Se han venido realizando cambios frecuentes en la aplicación por errores presentados.
    - 2: Se han venido realizando cambios ocasionales por errores en la aplicación.
    - 3: Se han venido realizando cambios frecuentes por errores de la aplicación o cambios en los requerimientos.
    - 4: Se han realizado cambios ocasionales para la implementación de cambios por la mejora o la implementación de nuevos requerimientos.

- 5: Desde que ha entrado en producción no se ha realizado modificaciones ni presentado nuevos requerimientos.

- ***Costos de mantenimiento*** **CM**: Implica las erogaciones financieras necesarias para mantener al sistema en funcionamiento. Mientras mayor sea la erogación disminuirá el valor del sistema.
  - 1: Menos del 10% del presupuesto anual de mantenimiento de las aplicaciones.
  - 2: Del 11% al 20% del presupuesto anual de mantenimiento de las aplicaciones.
  - 3: Del 21% al 40% del presupuesto anual de mantenimiento de las aplicaciones.
  - 4: Del 41% al 60% del presupuesto anual de mantenimiento de las aplicaciones.
  - 5: Más de 61% del presupuesto anual de mantenimiento de las aplicaciones.

  Los rangos aquí expuestos son un ejemplo y podrían ser ajustados de acuerdo a la organización.
- ***Factor de degradación*** **FD**: Este factor dependerá de la escala que se fije en la organización. Dentro de esta escala puede encontrarse la antigüedad, licencias, usuarios adicionales, extensiones, etc. Mientras más alto sea el factor de degradación, menor será el valor del sistema.
  - 1: Soporte oficial, tecnología popular, actual y con una gran oferta de personal capacitado en las herramientas de desarrollo.
  - 2: Soporte oficial y tecnología actual, pero sin mucha oferta de personal capacitado.
  - 3: Tecnología obsoleta y popular con un equipo interno completo para el mantenimiento y con una gran oferta de personal externo que se podría reclutar.
  - 4: Tecnología obsoleta, falta de soporte oficial pero con profesionales internos capacitados en la herramienta de desarrollo.
  - 5: Tecnología obsoleta, falta de soporte oficial y profesionales internos sin capacitación en la herramienta de desarrollo.

Para el cálculo del valor técnico se utiliza la fórmula:

$$vt = {(CC * FS)}/{5 * (CM * FD)}$$

Valor de Negocio

Para determinar el valor de negocio de un software se debe medirlo en función de un conjunto de parámetros detallados a continuación:

- ***Ventaja Competitiva* VC**: El primer parámetro define en qué nivel el activo de software puede ayudar en el aprovechamiento de las oportunidades del mercado.
    - 1: El aporte de la aplicación para las actividades de la empresa es mínima y puede ser eliminado.
    - 2: La aplicación puede ser reemplazada por procesos manuales sin afectar el funcionamiento de la organización.
    - 3: La aplicación procesa información necesaria para la organización desde un punto de vista importante, pero no crucial.
    - 4: La aplicación procesa información crítica e invaluable para la organización.
    - 5: El momento en que la aplicación deja de funcionar, los procesos ligados a ella no se pueden realizar ni siquiera de una forma manual[43].
- ***Impacto de rentabilidad* IR**: Este parámetro indica cómo se comportan los costos de mantenimiento dentro del presupuesto general.
    - 1: La inversión en mantenimiento es mayor a los beneficios que genera. Es decir, realizar ciertas actividades manualmente, resulta más rentable que hacerlo mediante el sistema.
    - 2: La inversión no genera pérdidas ni ganancias al ejecutar las actividades mediante el sistema.
    - 3: Se ve una clara ventaja en rentabilidad al ejecutar las actividades mediante el sistema.
    - 4: Las actividades de la aplicación no se pueden realizar manualmente, aunque la rentabilidad sea negativa.
    - 5: Las actividades de la aplicación no se pueden realizar manualmente y la rentabilidad que genera la aplicación es alta.
- ***Interdependencia con otros sistemas* IS**: Obedece al porcentaje de aplicaciones que dependen del sistema legado en el ámbito de procesamiento funcional o de datos.
    - 1: La aplicación interactúa con un alto porcentaje de aplicaciones a nivel funcional y de datos.
    - 2: La aplicación interactúa con un bajo porcentaje de aplicaciones a nivel funcional y de datos.
    - 3: La aplicación interactúa con un alto porcentaje de aplicaciones a nivel de datos.
    - 4: La aplicación interactúa con un bajo porcentaje de aplicaciones a nivel de datos.

[43] **Tarea manual:** Se refiere a que las actividades se realizan sin la ayuda de la aplicación informática involucrada. Por ejemplo, una tarea manual se puede hacer mediante aplicaciones ofimáticas.

- 5: La aplicación funciona independiente de otros módulos.

- ***Seguridad* SG**: Dependerá de la robustez del sistema con respecto a las diferentes amenazas de seguridad.
  - 1: La aplicación frecuentemente tiene inconvenientes por cualquier situación sin ninguna explicación aparente.
  - 2: La aplicación ha tenido problemas que se han solucionado, pero no se han tomado medidas para evitar que esto sea recurrente.
  - 3: La aplicación tiene ciertos comportamientos erróneos, no muy frecuentes que han sido corregidos y que no han vuelto a suceder.
  - 4: La aplicación no ha generado nunca comportamientos extraños por aspectos no funcionales.
  - 5: El sistemas tiene un plan de pruebas en donde prevé un amplio conjunto de situaciones anormales.

Para el cálculo del valor de negocio se utilizará la fórmula:

$$vn = {}^{(VC + IR + IS + SG)}/_{4}$$

Cuadrante de análisis

Para cada activo de software, se debe calcular el valor en cada eje y graficarlo en el cuadrante de la Figura 4.5. (Dedeke, 2012) Luego, si un activo de software tiene un valor de negocio alto y un valor técnico bajo, entonces se puede considerar la modernización del activo de software.

Como otro ejemplo, pero siguiendo la misma matriz, si un activo tiene un valor técnico bajo y un valor de negocio bajo, entonces se debe considerar dar de baja el sistema legado.

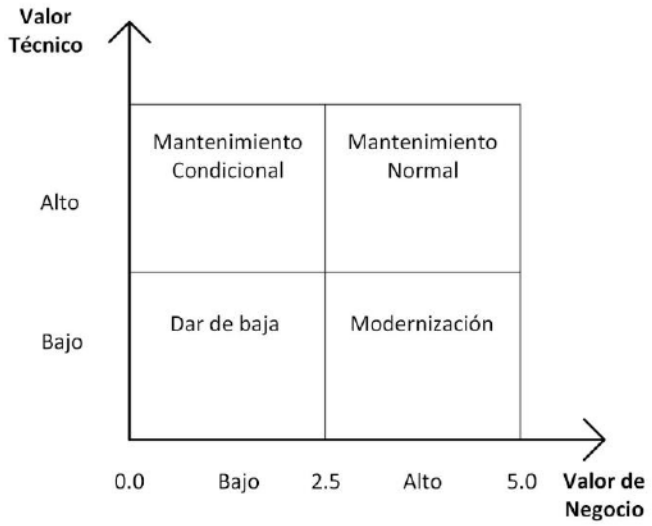


Figura 4.5: Cuadrante de análisis

Tanto el valor técnico, como el valor de negocio pueden generar un valor cuantitativo que permita sugerir la toma de decisiones referentes a:

- ***Dar de baja***: Son aplicaciones que han quedado relegadas y que no tiene sentido modernizarlas. En lugar de esto, se debería reemplazarlas o en su defecto darles de baja. La prioridad de estas aplicaciones se encuentran fuera de esta tesis.
- ***Mantenimiento Normal***: Significa que la aplicación puede seguir en funcionamiento, pero con un mantenimiento preventivo de ser el caso. Dentro de la propuesta de modernización se debería eliminar de la prioridad a estas aplicaciones.
- ***Mantenimiento Condicional***: Este tipo de aplicaciones no cumplen una actividad crucial en el negocio, pero el valor técnico alto hace que las mismas puedan soportar nuevas funcionalidades. El mantenimiento se debe condicionar al ajuste de las reglas de negocio.
- ***Modernización***: Las aplicaciones en este rango serán el objetivo o la prioridad de esta tesis, ya que son aplicaciones necesarias para el negocio, pero su valor técnico hace que vayan teniendo un carácter de obsolescencia. En este marco, para modernizar el sistema legado se puede recurrir a la siguiente etapa denomina: ***Etapa de integración de BPM en el proceso de requerimientos***.

4.3. Etapa de integración de BPM en el proceso de requerimientos

El objetivo principal de la segunda etapa del método, es generar una especificación de requerimientos de software en base al análisis de procesos de negocio modelados en BPMN2.

Esta etapa, asume que se ha definido exactamente la aplicación a modernizar y que por lo tanto se dispone de un sistema legado en producción. Para la definición de la aplicación a modernizar, se puede utilizar la etapa de identificación de sistemas legados de la sección anterior o cualquier otra técnica que defina la organización. He aquí, la importancia de separar las dos etapas dentro del método principal.

En la Figura 4.6, se muestra el diagrama de procesos de la etapa de integración de BPM en el proceso de requerimientos; la misma que presenta cuatro tareas, un subproceso, las interrelaciones existentes entre estos y el producto de trabajo denominado *especificación de requerimientos de software (SRS)*.

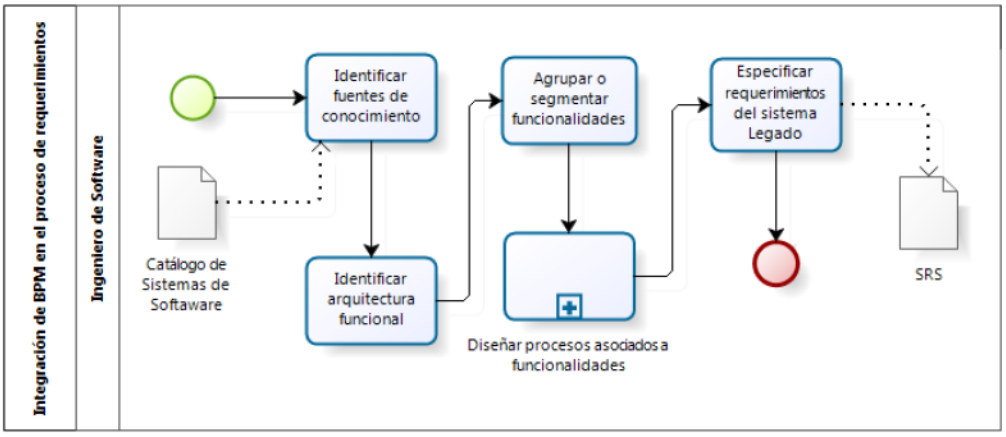


Figura 4.6: Etapa de integración de BPM en el proceso de requerimientos

Con respecto a la contribución de esta sección dentro de la tesis, la presente etapa ha sido establecida por el autor mediante la definición de un conjunto de actividades interrelacionadas que cumplen con el objetivo de producir una SRS a partir del análisis de modelos BPMN2. Con el fin de detallar el funcionamiento de cada una de las diferentes actividades de la etapa, se reutiliza la investigación del capítulo dos de la siguiente forma: identificar fuentes de conocimiento (Bourque & Fairley, 2014) (Ballejos & Montagna, 2011) (Loucopoulos, 1995), identificar arquitectura funcional (Maier, et al., 2001), agrupar o segmentar funcionalidades (Baghdadi & Al-Bulushi, 2013), diseñar procesos asociados a funcionalidades (Márquez, 2010) y especificar los requerimientos de software (Bourque & Fairley, 2014) (ISO/IEC/IEEE, 2011).

4.3.1. Identificar fuentes de conocimiento

Según la investigación realizada en el capítulo dos de esta tesis, la primera actividad del proceso de requerimientos de software se relaciona con la identificación de las fuentes de conocimiento.

Si bien existe un amplio estudio referente a requerimientos, en esta sección se acota las diferentes fuentes de conocimiento probables a: entorno organizacional, stakeholders y reglas de negocio.

- ***Entorno organizacional***: El sistema legado en producción debe cumplir con la misión, visión, planes operativos, organigramas, cartas de navegación o resoluciones que en su debido tiempo estuvieron vigentes. Por tal motivo, es necesario verificar y analizar la normativa anterior, alcances y/o normativa actual referente a los requerimientos funcionales del sistema.
- ***Stakeholders***: Como concepto, los stakeholders son las personas que interactúan de una u otra forma con el sistema; pero en el caso puntual de esta sección, los accesos del sistema legado en producción se convierten en la base para la identificación de los *stakeholders con categoría de usuario*. (Loucopoulos, 1995) (Ballejos & Montagna, 2011)

  En la práctica, mediante la utilización de los diferentes niveles de acceso y seguridad de la aplicación se puede identificar los stakeholders y la prioridad de cada uno según los niveles de permiso (administración, escritura, edición, solo lectura, etc.) que disponga el usuario dentro del sistema.
  Por otro lado, se puede utilizar la plantilla de la Tabla 4.3 para registrar los diferentes stakeholders del sistema legado.

Tabla 4.3: Matriz de Stakeholders

Código	Nombre	Departamento	Cargo	Categoría	Prioridad

- **Código**: Una identificación única para el actor involucrado.
- **Nombre**: Nombre de la persona que intervendrá en la toma de requerimientos.
- **Sector**: Área a la que pertenece el actor dentro de la organización.
- **Cargo**: Cargo que posee el actor dentro del sector.
- **Categoría**: Pueden existir diferentes categorías de actores: usuarios, clientes, patrocinadores, equipo de proyecto, etc.
- **Relevancia**: Es un número que comienza en uno y que aumenta según la serie de Fibonacci. Este aumento, servirá para representar de una mejor manera la relevancia del actor dentro del proceso de toma de requerimientos. (SCRUMstudy™, 2013)

- ***Reglas de negocio:*** Las reglas de negocio se consideran como enunciados o condiciones concretas que se deben cumplir y que por lo tanto se convierten en requerimientos funcionales específicos a tomar en cuenta para el análisis. Toda regla de negocio implementada dentro del sistema legado debe ser identificada y considerada como una fuente de conocimiento. (do Nascimento, et al., 2012)

  Una técnica básica propuesta por do Nascimento es la búsqueda de las condiciones de comparación (=, <>, <, >, >=, <=) y de operadores lógicos (AND, OR, NOT) en el código fuente de la aplicación legada.

En cualquiera de los casos, se da por hecho que se dispone del ***sistema legado en producción*** como una fuente de conocimiento transversal para el análisis.

4.3.2. Identificar arquitectura funcional

En esta actividad se define la arquitectura funcional del sistema legado en producción. (Maier, et al., 2001)

Para realizar el análisis arquitectónico se debe determinar conceptos estándares que permitan precisar las diferentes secciones de descomposición modular. Esta tesis, propone sistemas con tres descomposiciones:

- ***Tarea*:** Cuando se hable de tarea, se hará referencia al formulario o página que permita al usuario realizar acciones específicas sobre las estructuras o funcionalidades del sistema. Para ejemplarizar, la tarea *"Datos Personales"* consistiría en un formulario o página que implemente las reglas de negocio asociadas en la creación, lectura, actualización o eliminación de la tabla persona.
- ***Menú*:** Los menús, más que una forma de organización de las tareas a nivel lógico, son una categorización de las tareas con funcionalidades similares y que permiten mejorar la visibilidad del sistema. Por lo tanto, cada tarea debe poseer un menú; el mismo que puede ser de tres tipos:
  - **Parámetros**: Se refiere a los mantenimientos sobre tablas tipos o tablas base.

    Como ejemplo, se puede continuar con la tarea *"Datos Personales"*. En este caso, se podría identificar el parámetro *"Localidad"*; el mismo que se debe registrar de forma independiente, pero a su vez se convierte en pieza clave para realizar la tarea de *"Datos Personales"*, ya que con esta

información se puede registrar los campos de lugar de nacimiento, lugar de residencia, etc.

- **Funcionalidades**: A esta categoría pertenecen las tareas que utilizan un conjunto de parámetros y que permiten realizar alguna funcionalidad específica sobre los datos.
- **Reportes**: Esta categoría de tareas utiliza un conjunto de parámetros para llamar a reportes o tablas de visualización de datos. En este tipo de tareas no registran, actualizan o eliminan información.

- ***Módulo*:** Es un nombre que identifica o está compuesto de un conjunto de tareas; las mismas que son dependientes entre sí para el manejo de un proceso definido.

  Por ejemplo, dentro del módulo *"Inscripciones"* se puede integrar tareas cómo: "Datos Personales", "Registro de Inscripción", "Tipos de Admisión", "Aprobación de Aspirantes", etc.
  Para corroborar que las diferentes tareas pertenezcan al módulo, es necesario que a un módulo solo se le pueda asignar un proceso; el mismo que debe abarcar todas y cada una de las tareas definidas.
- ***Subsistema*:** Un subsistema es un conjunto de módulos interrelacionados que pueden funcionar independientemente de otros subsistemas y que a la vez pueden relacionarse con otros subsistemas generalmente a nivel de diseño de datos.

  Por ejemplo, el subsistema de *"Pregrado"* puede involucrar módulos como "Inscripciones", "Matriculación", "Calificaciones", "Oferta Académica", etc. "Pregrado" a su vez se puede relacionar con el subsistema de *"Admisión"* a nivel de diseño de datos para evitar redundancia de información e inconsistencias.
- ***Sistema*:** Un sistema es un conjunto de subsistemas relacionados que cumplen objetivos y funcionalidades delimitadas por un área o un departamento específico de la organización. De ser factible, puede darse interconexión entre sistemas a nivel de datos, pero para mantener la independencia se recomendaría que la interacción entre sistemas se maneje a nivel de arquitecturas orientadas a servicios. (Juric, et al., 2007)

  Para finalizar con un ejemplo, El sistema *"Académico"* compuesto por los subsistemas de "Admisión", "Pregrado" y "Posgrado" se puede relacionar con los sistemas *"Secretaría General"* y "*Bienestar Estudiantil"*.

En la Figura 4.7, se presenta un diseño que utiliza modelación UML y que permite entender la estructura modular genérica que se debe definir.

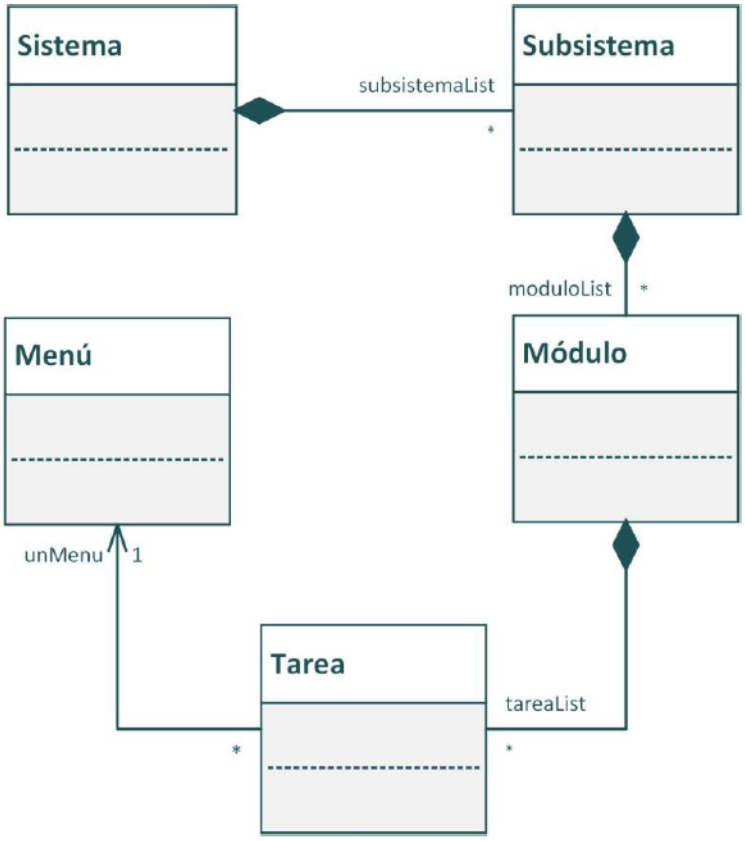


Figura 4.7: Estructura arquitectónica genérica

En la Figura 4.8, se presenta la enumeración para una estructura arquitectónica genérica, en donde se aprecia como un Sistema está compuesto de uno o más subsistemas. Además, un subsistema puede estar compuesto de varios módulos; los mismos que a su vez agrupan a un conjunto de tareas que son categorizadas mediante un menú.

1. Sistema 1
   1.1. Subsistema 1
   1.1.1 Módulo 1
   1.1.1.1 Parámetros
      Tarea 1.1.1.1.1
      Tarea 1.1.1.1.2

      Tarea 1.1.1.1.n
   1.1.1.2 Funcionalidades
      Tarea 1.1.1.2.1
      Tarea 1.1.1.2.2

      Tarea 1.1.1.2.n
   1.1.1.3 Reportes
      Tarea 1.1.1.3.1
      Tarea 1.1.1.3.2

      Tarea 1.1.1.3.n

Figura 4.8: Enumeración para una estructura arquitectónica genérica

4.3.3. Agrupar o segmentar funcionalidades

Esta sección, permitirá una reestructuración básica y elemental de las tareas definidas en la sección anterior. Esta reestructuración no significa que se vaya a modificar el proceso automatizado del sistema legado, sino que simplemente permite una definición más apegada a la realidad de los procesos que maneja el módulo del sistema legado. (Baghdadi & Al-Bulushi, 2013)

Dentro del contexto manejado, se puede entender que existen ciertos consensos por parte de los stakeholders en relación al funcionamiento de los procesos informatizados de los sistemas legados en producción. Además, existe una idea general por parte de los stakeholders sobre las mejoras básicas en torno a la interrelación entre las tareas actuales. Estos consensos, se pueden encontrar mediante la utilización de técnicas de elicitación como entrevistas y/o reuniones con los diferentes stakeholders definidos en la primera actividad de esta etapa.

- ***Agrupación de funcionalidades*:** Existen muchos casos en que las diferentes tareas de un mismo proceso se encuentran en diferentes formularios o páginas. En estos casos, se nota la dificultad del usuario al estar cambiándose de pantalla en pantalla para realizar el mismo proceso; estas situaciones propias del manejo continuo de la aplicación se deben identificar por parte de los stakeholders.
- ***Segmentación de funcionalidades***: Aunque menos frecuente que la situación anterior, puede darse el caso de que muchos procesos independientes se encuentren formando parte de una misma tarea. En estos casos, se debe segmentarla de tal forma que ciertas partes de la tarea sean independientes y pueden trabajar por separado.
- ***Definición de tareas integradoras:*** Definir una tarea integradora es el paso previo para la creación de un proceso básico. Esta tesis, plantea que una tarea integradora, debe estar correctamente documentada y validada por el responsable.

  En la tabla 4.4 se presenta una plantilla para registrar una tarea integradora a partir de la agrupación o segmentación de las funcionalidades del sistema legado.

Tabla 4.4: Tarea integradora

Tarea integradora	
Código	
Descripción	
Sistema	
Subsistema	
Módulos	
Descripción detallada	
Tareas asociadas	
Subtareas asociadas	
Responsable	

- **Código**: Una identificación única para la tarea integradora.
- **Descripción**: Identificación o nombre de la tarea integradora.
- **Sistema**: Nombre del sistema definido en la actividad anterior de esta etapa.
- **Subsistema**: Nombre del subsistema definido en la actividad anterior de esta etapa.
- **Módulo**: Nombre del módulo/s definido/s en la actividad anterior de esta etapa y que agrupan las tareas simples asociadas a la tarea integradora.
- **Descripción detallada:** Una descripción que representar la funcionalidad general del proceso asignado a la tarea.
- **Tareas asociadas**: El conjunto de tareas identificadas en la actividad anterior de esta etapa.
- **Subtareas asociadas**: El conjunto de subtareas identificadas a partir de la tarea asociada anterior.
- **Responsable**: Stakeholder que avala la definición.

4.3.4. Diseñar procesos asociados a funcionalidades

En las primeras tres actividades de la *etapa de integración de BPM en el proceso de requerimientos* se recolecta toda la información relacionada con la funcionalidad del sistema legado. Esta información, se convierte en la base de conocimiento para proceder con el modelamiento de las diferentes tareas de la aplicación como procesos de negocio. Los procesos de negocio, se deben desarrollar mediante la utilización de los conceptos de BPM y sobre todo mediante la utilización de la notación BPMN2 para su representación.

En la Figura 4.9, se presenta la actividad de *diseño de procesos asociados a funcionalidades* como un subproceso integrado en el proceso de la etapa principal. Además, se puede apreciar en detalle como el subproceso consta de cuatro actividades secuenciales denominadas: proceso básico, asociación de participantes, diseño detallado y validación del proceso (Márquez, 2010).

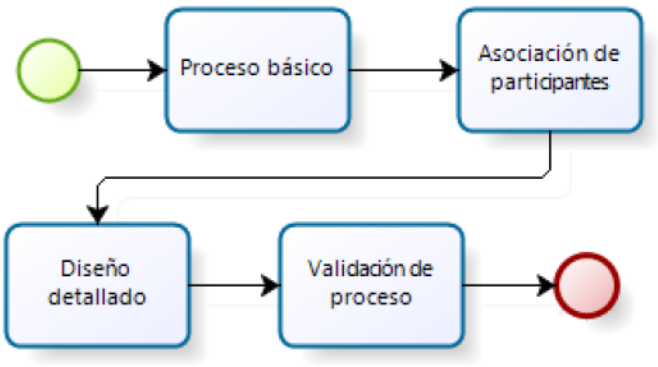


Figura 4.9: Diseñar procesos asociados a funcionalidades

Proceso básico

La idea de crear un proceso básico es disponer de un insumo global que permita el análisis rápido del proceso tanto desde la parte del usuario, así como desde el analista del proceso. Esta *"versión preliminar"*, debe ayudar en la representación del flujo de la información en el sistema legado.

Un proceso básico debe constar de cuatro tipos de componentes:

- ***Tarea simple:*** Es un rectángulo en donde se redacta la tarea que se pretende representar del sistema legado. En este punto, se debe realizar un alcance a las tareas propias[44] de BPMN, para que el registro del proceso de la tarea integradora se acote a cuatro tipos de tareas: seleccionar, registrar, calcular y mostrar.
    - **Seleccionar:** Se debe identificar acciones dentro del sistema legado en donde se seleccione un valor predefinido de un conjunto de datos o listas de valores.
      Un ejemplo típico de este tipo de tareas es la selección de localidades. Por lo tanto, cuando se identifique este tipo de tareas debe indicar *Seleccionar: país, provincia, ciudad*.
    - **Registrar:** Son acciones en donde se debe registrar valores en campos de texto con un dominio general: entero, decimal, booleano, texto, etc. Ejemplo: *Registrar nombre, apellido, altura, peso.*
    - **Calcular:** Son tareas del sistema legado en donde se identifica claramente un cálculo interno relacionado a una *regla de negocio*. Ejemplo: *Calcular: subtotal, impuesto, descuento, pago*.
    - **Mostrar:** Son etiquetas o resultados que muestra el sistema legado. Ejemplo: *Mostrar: promedio, estado de aprobación.*
- ***Condición exclusiva:*** Es un rombo que representa la bifurcación de un flujo de datos según cierta condición. Podría darse el caso de existir condiciones anidadas.
- ***Eventos simples:*** Simplemente se necesita de una representación para inicio y fin del proceso.
- ***Flujo de secuencia:*** Son flechas que representan el flujo del proceso y las relaciones existentes entre los diferentes componentes.

En la Figura 4.10, se puede apreciar un ejemplo de cómo construir un proceso básico utilizando el tipo de tareas definidas en esta propuesta.

[44] **Tipos de tareas BPMN:** manual, usuario, envío, recepción, regla de negocio, ejecución de script.

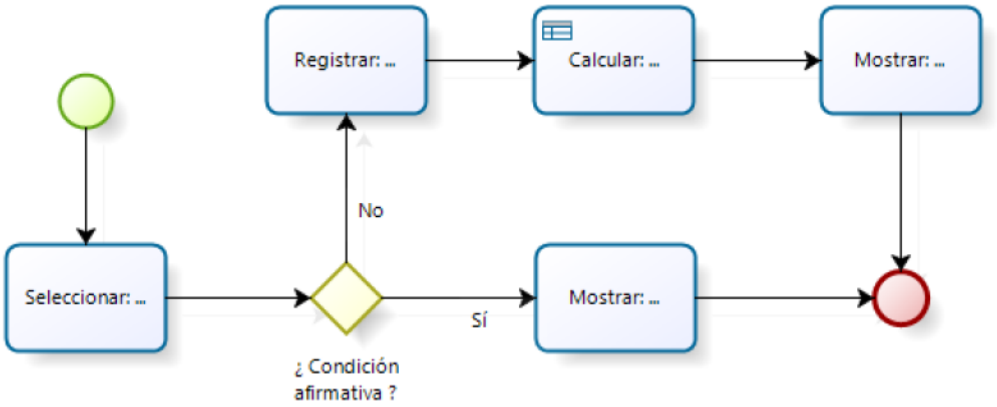


Figura 4.10: Proceso básico

Asociación de participantes

En segundo lugar y luego de haber definido el proceso básico del sistema legado, se debe identificar dentro de los diferentes stakeholders a los actores del proceso. En este punto, se debe aumentar al proceso básico los contenedores BPMN2.

Un contenedor puede agrupar un conjunto de compartimientos en los cuales se representa cada uno de los participantes identificados y a su vez se puede considerar cambiar el nombre de las tareas según el contexto de los participantes.

En la figura 4.11, se extiende un proceso básico con un conjunto de compartimientos que representan a cada uno de los participantes en las diferentes actividades.

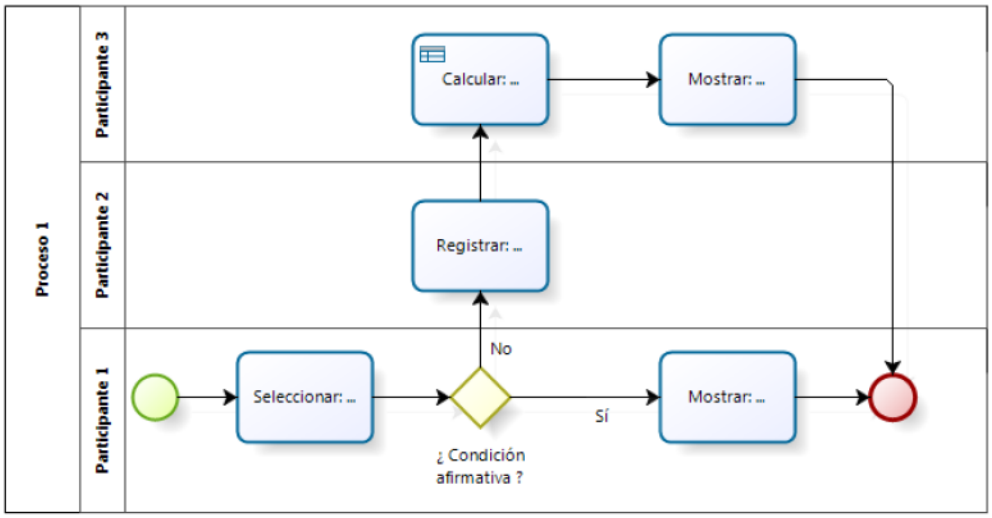


Figura 4.11: Asociación de participantes

Diseño detallado

En este punto, un diseño detallado se refiere al mismo proceso de negocio definido anteriormente, pero con la introducción de nuevos componentes BPMN2 que permitan una mejor lectura del mismo.

A continuación, se presenta una lista de los diferentes pasos que se propone en el desarrollo de un diseño detallado; dichos pasos se basan en el trabajo de Márquez (Márquez, 2010).

- Utilización de patrones de procesos de negocio en el diseño y definición de los mismos.
- Manejar conceptos de bifurcación y convergencia que permitan utilizar diferentes tipos de compuertas exclusivas, paralelas, inclusivas o basadas en eventos.
- Definir diferentes tipos de eventos de inicio y de fin.
- Asociar eventos intermedios en las tareas y flujos de datos.
- Documentar el proceso mediante las anotaciones respectivas.
- Utilizar artefactos de representación de datos que se puedan agregar el almacenamiento de los productos generados.
- Creación de artefactos personalizados.

Cabe destacar, que un proceso de negocio demasiado detallado no siempre es la mejor opción, ya que en la práctica, los procesos de negocio deben ser un medio para reducir el gap entre los analistas y los usuarios. Pero, si no existe el mismo nivel de profundidad en los conocimientos de los involucrados en su definición y validación, entonces el detalle del proceso puede convertirse en un obstáculo para el entendimiento.

Validación del proceso

Validar el proceso, significa examinar nuevamente el proceso final, con el propósito de obtener la aprobación por parte de los usuarios involucrados y comprobar que el proceso obtenido sea el adecuado.

Como técnica para la validación del proceso se puede utilizar a las ***revisiones***; las mismas que consisten en una lectura, inspección y análisis del proceso por parte de los revisores designados, pero sin la presencia de la persona que diseño el proceso, para mantener una objetividad que permita una revisión eficiente.

4.3.5. Especificar requerimientos del sistema legado

Esta sección, es la última parte de la segunda etapa y por lo tanto tiene como propósito la sistematización general de toda la información obtenida en las tareas anteriores; esta sistematización debe ser registrada mediante la creación de un producto de trabajo denominado Especificación de Requerimientos de Software o SRS por sus siglas en inglés.

Especificación de requerimientos de software

En el punto 2.3.4 de este trabajo de tesis, se plantea toda la información y estado del arte relacionado con una especificación de requerimientos de software estándar. Es aquí, en donde se puede profundizar en la motivación, beneficios, usuarios y estándares de redacción. Con esta información, se puede resumir que una SRS es *la presentación ordenada, estandarizada y coherente de los requerimientos obtenidos en la fase de elicitación y análisis*.

Dentro de esta propuesta, *no* se utilizará textualmente la estructura oficial ISO/IEC/IEEE 29148:2011 (ISO/IEC/IEEE, 2011) para una SRS, sino que se extenderá la misma para que dentro de un mismo documento se pueda examinar la sección de análisis, con la finalidad de obtener un solo producto que integre BPM en el proceso de requerimientos de software.

i. Revisión
ii. Tabla de contenidos
iii. Índice de figuras
iv. Índice de tablas
1. Introducción
   1.1. Propósito
   1.2. Alcance
   1.3. Visión general del documento
   1.4. Definiciones
2. Referencias
3. Fuentes de conocimiento
4. Análisis
   4.1. Estructura arquitectónica
   4.2. Procesos BPM
5. Requerimientos específicos
   5.1. Requerimientos funcionales
   5.2. Requerimientos No funcionales
6. Verificación (Paralela al punto 5)
7. Apéndices
   7.1. Supuestos y dependencias
   7.2. Acrónimos y abreviaturas

Figura 4.12: Esquema para Especificación de Requerimientos de Software

A continuación, se describe cada uno de los puntos del esquema propuesto en la Figura 4.12. Siguiendo estas pautas, se puede generar el producto de trabajo asociado a la tarea de *especificar requerimientos de software*.

i. Revisión

ii. Tabla de contenidos

iii. Índice de figuras

iv. Índice de tablas

1. Introducción

1.1. ***Propósito:*** En esta subsección se debe precisar el propósito de la creación del documento. El propósito, debe ser especificado de pendiendo de los diferentes puntos de vista existentes: usuario, patrocinador, equipo técnico, etc.

1.2. ***Alcance:*** El alcance identifica que productos de software van a ser generados a partir de la SRS. Además, se puede presentar los beneficios, objetivos y metas esperadas del producto.

1.3. ***Visión general del documento:*** En esta subsección no se debería detallar, pero si presentar un resumen general del documento y de cada una de sus secciones.

1.4. ***Definiciones:*** Esta subsección debe tener un listado de todas las palabras utilizadas y que necesiten de una definición explícita, con el fin de evitar ambigüedades o desconocimiento del significado por parte de la audiencia del documento.

2. **Referencias:** Es una lista completa de todos los documentos asociados, citados o consultados durante la creación de la SRS.
3. **Fuentes de conocimiento:** Esta sección está íntimamente relacionada con la primera tarea de la etapa de integración de BPM en el proceso de requerimientos denominada "*Identificar fuentes de conocimiento*".

   Es aquí en donde se detalla entorno organizacional, reglas de negocio y todos los actores relacionados con la especificación.
4. **Análisis**

   4.1. ***Estructura arquitectónica:*** En la tarea "*Identificar arquitectura funcional*" se plantea una descomposición modular; la misma que en esta sección debe ser registrada con el fin de organizar las funcionalidades del sistema legado.

   4.2. ***Procesos BPM:*** La tercera actividad denominada "*Agrupar o segmentar funcionalidades*" plantea un conjunto de tareas integradoras; las mismas que a su vez deben registrarse en esta sección y ser representadas como procesos de negocio siguiendo las pautas de la actividad "*Diseñar procesos asociados a funcionalidades*".

   La vista de las funcionalidades del sistema legado como procesos de negocio puede ayudar a los involucrados en la mejora o refinación del proceso actual.
5. **Requerimientos específicos**: Esta sección es la más importante de la especificación, ya que es aquí en donde se describirá con un nivel de detalle suficiente todos y cada uno de los requerimientos de software.

   En esta sección se debe definir como mínimo los requerimientos funcionales del sistema, teniendo en cuenta que cada requerimiento especificado debe ser único y rastreable.

   5.1. ***Requerimientos funcionales*:** Es una lista detallada de todas las acciones fundamentales relacionadas con los procesos de negocio y que deben llevarse a cabo en el software para el procesamiento de las entradas y la generación de las salidas.

   5.2. ***Requerimientos No funcionales*:** En esta subsección se definen los requerimientos No funcionales. Es decir, todos los requerimientos relacionados con rendimiento, seguridad, usabilidad, interfaces con sistemas externos, etc.
6. **Verificación:** En esta sección debe ser registrada en paralelo con cada uno de los requerimientos definidos en el punto anterior.
7. **Apéndices:** No es una sección que maneje información referente del contexto principal de la especificación, pero permite añadir información extra relacionada.

7.1. ***Supuestos y dependencias*:** Se debe enumerar la lista de factores que afectarían a los requerimientos del sistema en el caso de que no estén presentes.

7.2. ***Acrónimos y abreviaturas*:** Esta subsección define una lista con el significado de los diferentes acrónimos y abreviaturas utilizados en el documento.

5. Caso de Estudio

En este capítulo, se presenta una experiencia concreta de la aplicación del método de integración de BPM en el proceso de requerimientos para la modernización de sistemas legados. El contexto de aplicación de este método gira en torno a los sistemas informáticos de la Universidad Politécnica Salesiana del Ecuador.

5.1. Introducción

Dentro de la investigación cualitativa, un caso de estudio es considerado como una *herramienta que permite acercarse lo más posible al mundo real y por lo tanto, permitir su interpretación para la validación de una teoría*. (Reyes, 2005)

La definición del párrafo anterior, se utilizará como referencia para este capítulo. En donde se utiliza a los ***Sistemas informáticos de la Universidad Politécnica Salesiana*** como un caso de estudio (Díaz, 2014) (Runeson & Höst, 2009) para la validación de las etapas definidas en el presente trabajo de tesis.

Según su misión, la Universidad Politécnica Salesiana es una institución de educación superior humanística y politécnica, de inspiración cristiana con carácter católico e índole salesiana; dirigida de manera preferencial a jóvenes de los sectores populares; busca formar "honrados ciudadanos y buenos cristianos", con capacidad académica e investigativa que contribuyan al desarrollo sostenible local y nacional. (Politécnica Salesiana, 2014)

En el ámbito organizacional, la Universidad Politécnica Salesiana tiene presencia en tres ciudades del Ecuador: Quito, Guayaquil y Cuenca; dichas sedes en conjunto poseen cinco campus, que a su vez acogen alrededor de 24000 estudiantes distribuidos en veinte carreras de grado y cinco de postgrado. En cuanto a los colaboradores; los mismos se encuentran distribuidos en 1300 docentes y 450 administrativos a nivel nacional.

Como se puede apreciar en los datos presentados, la magnitud de la Universidad hace que exista una gran cantidad de datos e información generados a partir de procesos administrativos, académicos, financieros, contables y de talento humano. En la actualidad, la mayoría de esta información se almacena y gestiona de una manera automatizada mediante un conjunto de aplicaciones que han sido desarrolladas desde el año dos mil uno y que han venido cumpliendo en la medida de lo posible con los requerimientos funcionales de la Universidad.

Con el fin de mantener funcionales los diferentes sistemas de información, la organización dispone de un Departamento de Sistemas; el mismo que se divide en cuatro coordinaciones: desarrollo, explotación, base de datos e infraestructura y redes. Siendo la Coordinación de desarrollo, la encargada de recolectar las necesidades y peticiones de los diferentes usuarios con el fin de ejecutar el respectivo mantenimiento y/o creación de los sistemas informáticos que permitan la automatización de los diferentes procesos asociados a la gestión universitaria.

En resumen, este caso de estudio presenta la aplicación de las dos etapas del método propuesto en el capítulo cuatro. Esta aplicación, es realizada por parte de la Coordinación de desarrollo, con el fin de generar una propuesta para el Plan de modernización de los sistemas informáticos de la Universidad Politécnica Salesiana.

5.2. Antecedentes

Según el Plan Operativo Anual 2014 de la Universidad Politécnica Salesiana, la Secretaría Técnica de Tecnologías de la Información tiene como tarea la creación de un *Plan de modernización para los sistemas de información de la Universidad Politécnica Salesiana*. En este contexto, se designó a la Coordinación de desarrollo de software como la responsable de dicha tarea, y por lo tanto ser la encargada de generar un documento que sirva como línea de base para la creación de los diferentes proyectos asociados.

Con respecto a los sistemas núcleo de la gestión de la organización, se puede asegurar que más del 80% de los datos se gestionan mediante tres sistemas de información; los mismos que están siendo identificados, analizados y especificados utilizando los dos etapas propuestas en esta tesis. A continuación, una breve descripción de cada sistema de información:

- ***SNA***: El Sistema Nacional Académico es el encargado de gestionar todos los procesos académicos y en algunos casos administrativos de la Universidad.

  A falta de procesos académicos claros, la lógica del negocio de este sistema se ha venido modificando desde su puesta en producción, lo que ha hecho que se vaya perdiendo información documentada por un escaso control de cambios.

- ***SQUAD***: El Sistema de Nómina se ajusta a los procesos de Gestión de Talento Humano definidos por los usuarios.

  La lógica del negocio no se ha visto modificada de fondo, pero las reglas de negocio como fórmulas o funciones de cálculo han venido siendo alteradas por exigencias legales y reglamentarias emitidas por la entidad estatal encargada de regular el proceso de nómina.

- ***SIGAC***: El Sistema Integrado de Gestión Administrativa y Financiera es utilizado por las áreas financiera, presupuestaria y contable como parte medular de su gestión.

  La lógica y reglas de negocio se han mantenido incólumes, ya que los usuarios finales se han adaptado al proceso propuesto por el sistema.

Las tres aplicaciones han sido implementadas mediante el suite de desarrollo Oracle Forms & Reports.

Las dos primeras aplicaciones han sido desarrolladas con una documentación a nivel de diseño de datos y con una estructura similar, mientras que el sistema SIGAC fue desarrollado por una empresa externa denominada Carrasco & Asociados y por consiguiente posee una lógica de construcción diferente e inadecuada para los estándares de desarrollo actuales.

Para funcionalidades extras al núcleo del negocio, se han venido creando aplicaciones web que se conectan a estos tres sistemas, ya sea para presentar la información en la página web o en casos muy puntuales para ingresar datos al sistema académico y de nómina mediante servicios web o conexiones a nivel de base de datos.

5.3. Desafíos

Los desafíos hacen referencia a las limitaciones, problemas o retos que plantea la creación de un *Plan de modernización de los sistemas informáticos de la Universidad Politécnica Salesiana*. Estos desafíos, aparecieron cuando se pretendió dar una visión integral a todos los sistemas de información que se han venido desarrollando a lo largo de esta década.

Dentro de los principales desafíos se encuentran: complejidad en la estructura de datos, agrupación de funcionalidades, falta de documentación y tecnología de desarrollo obsoleta.

5.3.1. Complejidad en la estructura de datos

Al ser aplicaciones que interactúan con la base de datos, la manera más rápida de identificar la cantidad de información que posee, es mediante la lectura del número de esquemas que dispone la aplicación. En este sentido, la cantidad de datos va de la mano con el número de tablas creadas en los diferentes esquemas.

Como se puede apreciar en la Figura 5.1, se identificó tres esquemas asociados a cada uno de los sistemas núcleo de la organización. Los esquemas denominados SNA, SIGAC y SQUAD suman en conjunto 1810 tablas activas sin contar con las tablas de auditoría.

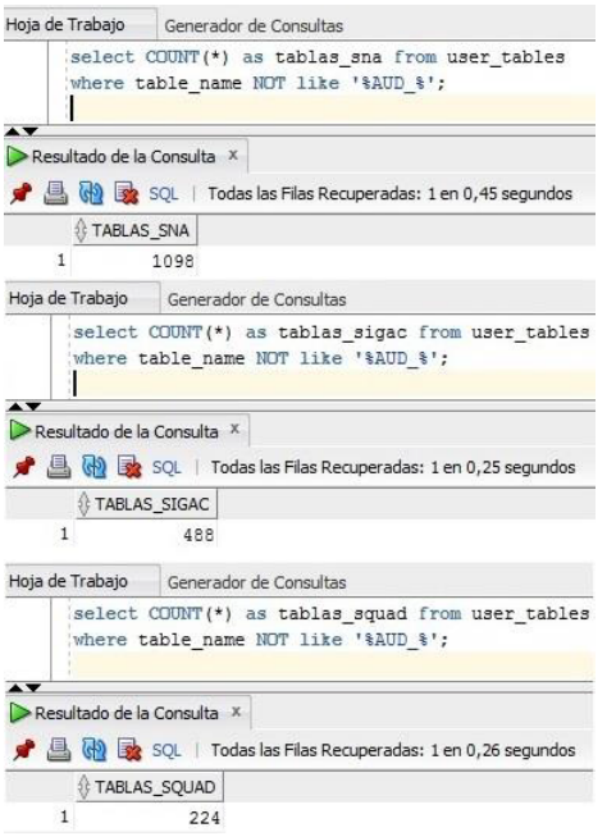


Figura 5.1: Número de tablas de los esquemas núcleo

Pretender realizar una ingeniería inversa para generar modelos o diseños actuales a partir de la base de datos, se convierte en una opción demasiado compleja, ya que además del alto número de tablas se debe considerar las relaciones existentes entre ellas.

Otro desafío referente a la complejidad de la estructura datos, se relaciona con una práctica de diseño utilizada en los sistemas de información analizados. Dicha práctica va creando con cada relación una clave primaria compuesta. Es decir, si existe una tabla denominada TAB_PERSONA que recibe como relación el TAB_TPO_PERSONA, entonces la clave primaria de la tabla TAB_PERSONA debe crearse como una clave compuesta del secuencial propio de la tabla y del código de la tabla relacionada.

Este tipo de prácticas es adecuado hasta cierto nivel y en ciertos casos, pero lastimosamente un porcentaje muy alto de tablas se encuentran definidas de esta manera y por lo tanto se puede llegar a puntos extremos como generar tablas con claves primarias compuestas de más de diez campos como se aprecia en la Figura 5.2.

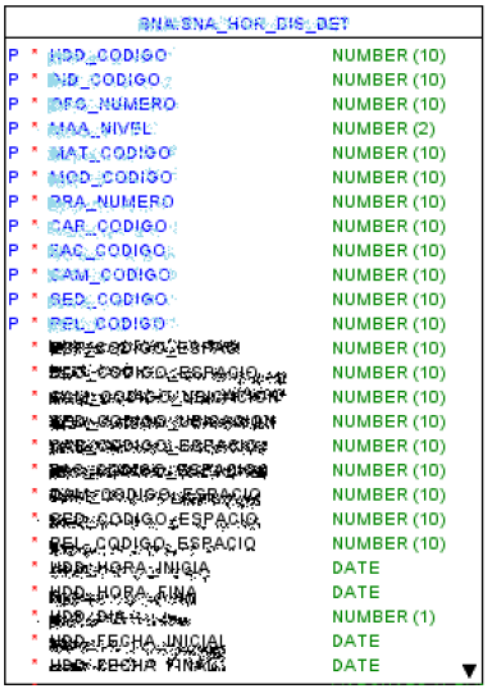


Figura 5.2: Tabla con clave primaria compuesta[45]

Si bien, la ventaja de utilizar esta técnica, radica en un desarrollo acelerado de aplicaciones y reportes utilizando específicamente la herramienta Forms & Reports, también es cierto, que como desventaja principal se encuentra el hecho de que esta excesiva "normalización" impide que el sistema pueda crecer a nivel de diseño, haciendo que cada vez sea más costoso y complejo realizar mejoras o aumentos de funcionalidad en los sistemas de información, ya que desde un principio el diseño fue atado a la herramienta de desarrollo.

[45] Imagen degradada intencionalmente.

5.3.2. Agrupación de funcionalidades

Con respecto a la agrupación de funcionalidades, lo que se considera como un desafío es intentar segmentar funcionalidades que en la actualidad se encuentran ligadas a nivel de aplicación y de diseño a pesar de ser consideradas como procesos diferentes. Esta agrupación impide que las diferentes funcionalidades se puedan ejecutar en procesos por separado. Como ejemplo, en el caso de estudio se encuentran parámetros claramente definidos como financieros que deben ser registrados por instancias académicas o de bienestar estudiantil.

En el caso de que se pueda definir otro tipo de estructura a nivel arquitectónico, se presenta un reto aún mayor; el mismo que consiste en migrar toda la información a los nuevos módulos o estructuras de almacenamiento.

5.3.3. Falta de documentación

En el punto 3.3.2 de esta tesis (Documentación inadecuada) se puede verificar las diferentes posibilidades o situaciones referentes a deficiencias en la documentación. A continuación, se presenta los problemas asociados al caso de estudio:

- *Falta de un marco de referencia de T.I*: El Departamento de Sistemas no utiliza ningún marco de referencia para su gestión, razón por la cual no se maneja una documentación formal de los procesos y productos relacionados con T.I.
- *Utilización de estándares empíricos*: Se han creado diversos tipos de plantillas empíricas que no se basan en estándares internacionales probados y que no se convierten en una fuente de información adecuada. Además, estas plantillas carecen de un registro de control de cambios.
- *Gestión de proyectos inadecuada*: No existe documentación relacionada con la gestión de tareas, recursos, hitos o entregables del proyecto inicial.
- *No existe un control de versiones*: Los cambios, mejoras y adaptaciones de los sistemas en torno a nuevos procesos y reglas de negocio de la organización, no mantuvieron por mucho tiempo un control de cambios a nivel de aplicación. Desde hace aproximadamente dos años la Coordinación de desarrollo dispone de un sistema para seguimiento de incidencias denominado JIRA®; en el cual se van registrando nuevos requerimientos, mejoras, errores y tareas relacionadas a las aplicaciones, pero a pesar de que estos cambios queden registrados, no existe un control de versiones de la aplicación.

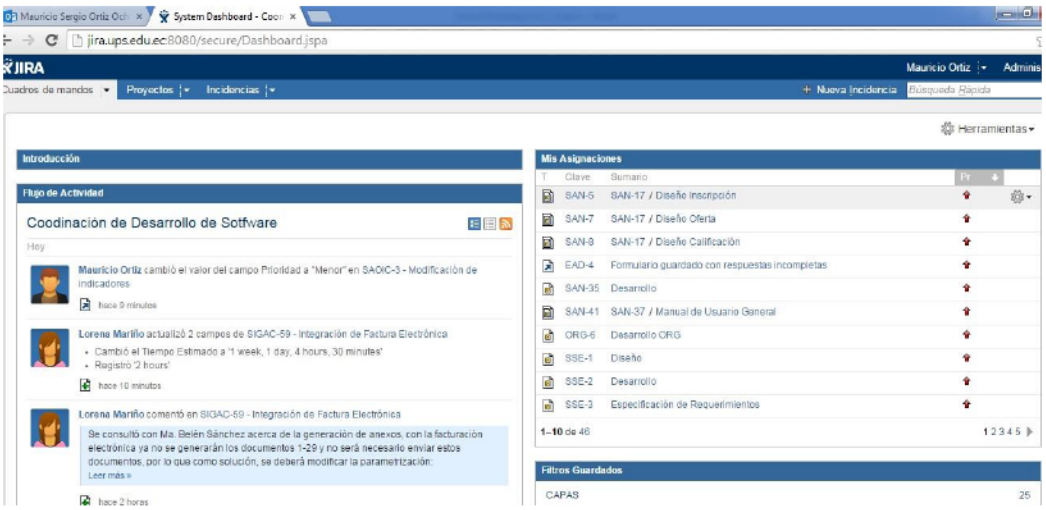


Figura 5.3: Sistema de seguimiento de incidencias JIRA®

La falta de documentación se convierte en otro reto complejo en la realización del Plan de modernización ya que "*No se puede analizar lo que no se conoce*".

5.3.4. Tecnología de desarrollo obsoleta

La herramienta tecnológica con la que se ha desarrollado los tres sistemas núcleo del caso de estudio presentan los siguientes retos:

- *Falta de soporte oficial para las aplicaciones Forms & Reports*: La última recomendación de Oracle fue dejar de lado la suite y migrar las aplicaciones a la versión final 11g en una plataforma web Java o ADF.
- *Falta de soporte oficial para el Oracle Application Server.*
- *Falta de soporte en los navegadores actuales de la versión de Java utilizada.*
- *Falta de personal capacitado en las herramientas de desarrollo*: Existe mucha complejidad en reclutar profesionales que tengan destreza o experiencia en estas herramientas.
- *Falta de capacitación oficial en las herramientas de desarrollo*: A nivel de pregrado o cursos de profesionalización no existen currículos que se adapten a las necesidades de la organización.
- *Implementación de la lógica del negocio a nivel de base de datos*: La programación de la lógica a este nivel, hace que se programe de una manera estructurada, haciendo que la lógica sea demasiado dependiente de la plataforma Oracle.
- *Limitaciones en la parte no funcional, para grupos con un extenso número de usuarios*: Las aplicaciones Forms & Reports fueron diseñadas para un grupo cerrado de usuarios, razón por la cual las aplicaciones para grupos de usuarios extensos, han tenido que ser desarrolladas en propuestas web.
- *Aparición de mejores prácticas de desarrollo no compatibles con la suite actual*: Al ser una programación estructurada, la documentación que se pueda generar en torno a otros paradigmas queda improcedente en cuanto a la implementación en sí.

La mayoría de los desafíos relacionados con la tecnología de desarrollo obsoleto, se pueden resumir en uno solo; este desafío consiste en identificar nuevas tecnologías de desarrollo que se adapten a las funcionalidades actuales de los sistemas.

5.4. Diseño del caso de estudio

5.4.1. Metodología de la investigación

A pesar de que en las siguientes secciones se presentará un estudio empírico, es importante enfocar una metodología de investigación que permita evidenciar el proceso científico en el diseño del caso de estudio. Dentro de esta búsqueda de metodologías, (Runeson & Höst, 2009) se presenta a la Investigación activa; la misma que "*tiene como propósito influenciar o cambiar algún aspecto real del objeto del caso de estudio*" (Robson, 2002).

Desde el punto de vista de esta tesis, el objeto del caso de estudio son los sistemas de información de la Universidad Politécnica Salesiana y el cambio real se encuentra focalizado en la forma de identificar los sistemas legados y de hacer el relevamiento de sus requerimientos funcionales.

5.4.2. Objetivos del caso de estudio

- Identificar los sistemas legados de la organización de una manera uniforme y objetiva.
- Identificar los sistemas legados que deben ser parte de un plan de modernización.
- Crear especificaciones de requerimientos de software que permitan una mejor lectura de las funcionalidades actuales por parte de los analistas de negocio.
- Aplicar una metodología que permita la refinación de los procesos de negocio.

5.4.3. Soluciones planteadas

De acuerdo a los retos definidos, las soluciones planteadas para el caso de estudio propuesto van de la mano con la implementación del ***"método de integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados"*** planteado en esta tesis.

La implementación del método se realiza mediante la ejecución de dos etapas:

La ***etapa de identificación de sistemas legados***, servirá para conocer y analizar los diferentes sistemas de información, ya que es menester tener un Catálogo de Sistemas de Software que permita identificar entre otras cosas las prioridades de modernización de los sistemas legados. Con esta etapa, se pretende atacar la falta de documentación y la complejidad de la estructura de los datos, ya que se pueden definir parámetros que permitan una visión integral de las aplicaciones.

La ***etapa de integración de BPM en el proceso de requerimientos***, pretende atacar la mayoría de los problemas asociados a los desafíos planteados en el caso de estudio.

En primer lugar, ayuda con la complejidad de la estructura de información, ya que permitirá visualizar las funcionalidades como *procesos de negocio* y por lo tanto las tablas y sus relaciones pueden pasar a segundo plano en el entendimiento de los procesos. Por otro lado, esta etapa permite agrupar o segmentar funcionalidades de una manera

estructurada, resolviendo así el problema presentado con la agrupación de funcionalidades que dispone el caso de estudio.

La falta de documentación es otro problema que se puede resolver con creces, ya que el usuario tendrá definidos procesos BPM en la especificación de requerimientos de software; dicha especificación servirá para la creación de un nuevo software o en su defecto como parte contractual para la compra de uno nuevo.

5.5. Etapa de identificación de sistemas legados

5.5.1. Definir parámetros de identificación

Según plantea la primera etapa del método, en esta actividad se debe definir los parámetros que identificarán a los diferentes sistemas. En este contexto, y luego de una breve explicación al equipo de desarrollo de lo que se pretendía determinar, se definió como consenso la Tabla 5.1. En donde se presenta las diferentes categorías tecnológicas en las cuales fueron desarrollados los sistemas de información que posee la Universidad Politécnica Salesiana.

Tabla 5.1: Categorías tecnológicas definidas por la Coordinación de desarrollo

CTE_CODIGO	CTE_DESCRIPCION
1	APLICACIONES FORMS & REPORTS
2	APLICACIONES PHP
3	APLICACIONES JSP
4	APLICACIONES JSF+EJB+JPA
5	APLICACIONES FOX PRO
6	APLICACIONES LIFERAY

Por otro lado, se utilizó las categorías transversales propuestas en la tesis y que se muestran en la Tabla 5.2.

Tabla 5.2: Categorías transversales predefinidas en la tesis

CTR_CODIGO	CTR_DESCRIPCIÓN
1	IDENTIFICACIÓN FUNCIONAL
2	PROCESO DE SOFTWARE
3	CARACTERÍSTICAS TÉCNICAS
4	MEDICIÓN DE ARTEFACTOS

En la Tabla 5.3, se aprecia los diferentes parámetros identificados por la Coordinación de desarrollo. Cada uno de los parámetros, se encuentra relacionado con una categoría transversal definida.

Tabla 5.3: Parámetros de identificación

PAR_CODIGO	PAR_DESCRIPCION	PAR_TIPO_DATO	PAR_UNA_CATEGORIA_TRANSVERSAL
101	CÓDIGO	ENTERO	1 IDENTIFICACION FUNCIONAL
102	NOMBRE	TEXTO	1 IDENTIFICACION FUNCIONAL
103	VERSIÓN	TEXTO	1 IDENTIFICACION FUNCIONAL
104	SISTEMA	TEXTO	1 IDENTIFICACION FUNCIONAL
105	DESCRIPCIÓN CORTA	TEXTO	1 IDENTIFICACION FUNCIONAL
106	DESCRIPCIÓN DETALLADA	TEXTO	1 IDENTIFICACION FUNCIONAL
…	…	…	…
201	DOCUMENTACIÓN DE REQUERIMIENTOS	ENTERO	2 PROCESO DE SOFTWARE
202	DOCUMENTACIÓN DE DISEÑO	ENTERO	2 PROCESO DE SOFTWARE
…	…	…	…
301	RUTA DEL SERVIDOR DE APLICACIONES	TEXTO	3 CARACTERÍSTICAS TÉCNICAS
302	RUTA DEL SERVIDOR DE REPORTES	TEXTO	3 CARACTERÍSTICAS TÉCNICAS
…	…	…	…
401	NÚMERO DE FORMULARIOS	ENTERO	4 MEDICIÓN DE ARTEFACTOS
402	NÚMERO DE REPORTES	ENTERO	4 MEDICIÓN DE ARTEFACTOS

Como punto final de esta actividad, en la Tabla 5.4, se relaciona los diferentes parámetros con las diferentes categorías tecnológicas, evitando de esta forma la duplicidad de definición de parámetros.

Como ejemplo, los parámetros código, nombre y versión que se encuentran relacionados con la identificación funcional, son utilizados a su vez en diferentes categorías tecnológicas.

Tabla 5.4: Parámetros por categoría tecnológica

PCT_CODIGO	PCT_UN_PARAMETRO	PCT_OBLIGATORIO	PCT_UNA_CATEGORIA_TECNOLÓGICA
1101	101 CÓDIGO / 1 IDENTIFICACION FUNCIONAL	S	1 APLICACIONES FORMS & REPORTS
1102	102 NOMBRE / 1 IDENTIFICACION FUNCIONAL	S	1 APLICACIONES FORMS & REPORTS
1103	103 VERSIÓN / 1 IDENTIFICACION FUNCIONAL	N	1 APLICACIONES FORMS & REPORTS
…	…	…	…
1201	201 DOCUMENTACIÓN DE REQUERIMIENTOS / 2 PROCESO DE SOFTWARE	S	1 APLICACIONES FORMS & REPORTS
…	…	…	…
1301	301 RUTA DEL SERVIDOR DE APLICACIONES / 3 CARACTERÍSTICAS TÉCNICAS	S	1 APLICACIONES FORMS & REPORTS
…	…	…	…
1401	401 NÚMERO DE FORMULARIOS / 4 MEDICIÓN DE ARTEFACTOS	S	1 APLICACIONES FORMS & REPORTS
…	…	…	…
2101	101 CÓDIGO/ 1 IDENTIFICACION FUNCIONAL	S	2 APLICACIONES PHP

2102	102 NOMBRE / 1 IDENTIFICACION FUNCIONAL	S	2 APLICACIONES PHP
2103	103 VERSIÓN / 1 IDENTIFICACION FUNCIONAL	N	2 APLICACIONES PHP
…	…	…	…
2201	201 DOCUMENTACIÓN DE REQUERIMIENTOS / 2 PROCESO DE SOFTWARE	S	2 APLICACIONES PHP
…	…	…	…
2301	301 RUTA DEL SERVIDOR DE APLICACIONES / 3 CARACTERÍSTICAS TÉCNICAS	S	2 APLICACIONES PHP
…	…	…	…
2401	401 NÚMERO DE FORMULARIOS / 4 MEDICIÓN DE ARTEFACTOS	S	2 APLICACIONES PHP

5.5.2. Catalogar sistemas de software

La estructura del documento entregable denominado Catálogo de Sistemas de Software está claramente definida en el capítulo cuatro. Razón por la cual, en el contexto del caso de estudio, se crea un documento con la siguiente organización:

Introducción

En esta sección se redactó información referida a: propósito, alcance, control de cambios, definiciones, acrónimos, abreviaturas, referencias y una visión general del documento.

Definición de parámetros de identificación

En esta sección, se especifica y enumera las diferentes categorías tecnológicas (Tabla 5.1) y transversales (Tabla 5.2) identificadas en la primera actividad de la etapa.

Catálogo de sistemas

El catálogo abarca la identificación de todas las aplicaciones organizadas en las diferentes categorías tecnológicas.

Tabla 5.5: Número de aplicaciones organizadas por categoría tecnológica

CODIGO	CATEGORÍA TECNOLÓGICA	NÚMERO DE APLICACIONES
1	APLICACIONES FORMS & REPORTS	36
2	APLICACIONES PHP	4
3	APLICACIONES JSP	25
4	APLICACIONES JSF+EJB+JPA	9
5	APLICACIONES FOX PRO	2
6	APLICACIONES LIFERAY	5

Una vez definidos los diferentes parámetros de identificación por categoría tecnológica, se comienza a cargar los valores correspondientes a las diferentes aplicaciones identificadas según la tabla de contenido de la Figura 5.4.

Figura 5.4: Catálogo de Sistemas de Software de la Universidad Politécnica Salesiana

En la Figura 5.5, se presenta un diagrama de clases que permite evidenciar como las aplicaciones son asignadas a una categoría tecnológica y por lo tanto a un contenido relacionado con cada parámetro predefinido en la primera actividad de esta etapa.

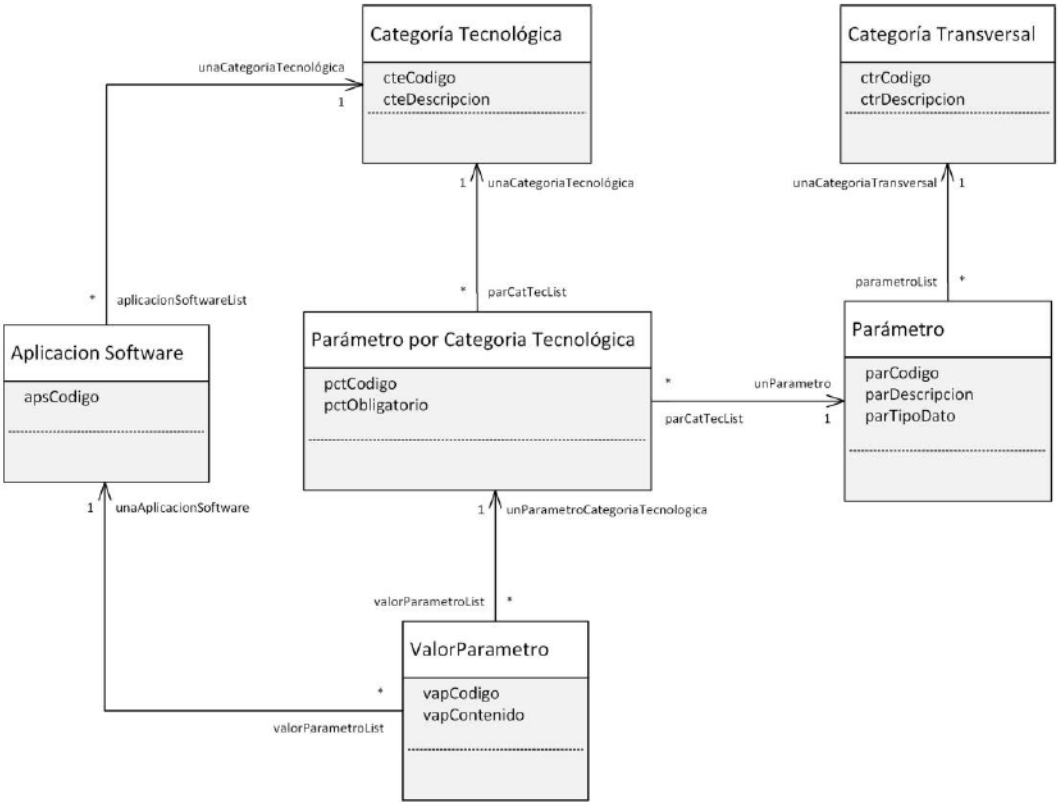


Figura 5.5: Valores para identificación de aplicaciones de software

5.5.3. Especificar prioridades de modernización

Para culminar esta etapa y según como se vaya cargando el contenido de identificación de cada aplicación se deberá especificar las prioridades de modernización según lo definido en el capítulo cuatro.

En el contexto del caso de estudio, se creó una matriz en una hoja de cálculo que permita obtener automáticamente el valor técnico y el valor de negocio de cada aplicación a partir de los diferentes parámetros definidos.

Tanto en la Tabla 5.6, como en la Figura 5.6 se puede apreciar el análisis del cuadrante realizado a la aplicación de Bienestar Estudiantil utilizada por la Universidad Politécnica Salesiana.

Tabla 5.6: Prioridades de modernización (Bienestar Estudiantil)

Código	1.14
Nombre	SNA_BIENESTAR_ESTUDIANTIL
Versión	N/D
Sistema	Sistema Nacional Académico

Identificación Funcional	
Descripción Corta	Bienestar Estudiantil
Descripción Detallada	Este subsistema es bastante utilizado ya que maneja servicios a los estudiantes como: ficha socioeconómica, visita domiciliaria, solicitudes, becas, pagos en diferido y garantes.
Ruta Producción	http://appswl.ups.edu.ec/forms/frmservlet?form=menusna.fmx&config=webutil
Fecha Producción	jueves, 01 de febrero de 2007
Ubicación Documentación	\\172.16.1.159\DesarrolloSNA\sna
Ubicación Fuentes	/u02/sna/formas ; /u02/sna/reportes ; /u02/sna/ayuda
Responsable	Ing. Verónica Clavijo
Proceso de Software	
Análisis y Requerimientos (REQ)	1: Existen documentos relacionados con requerimientos funcionales como casos de uso o descripciones textuales.
Documentación de Diseño (DIS)	2: Existe documentación detallada de la estructura de datos y un diccionario de datos que describa en detalle cada estructura.
Prácticas de desarrollo (DES)	3: Existe una fuerte concordancia entre el código fuente y la documentación de requerimientos y de diseño.
Proceso de pruebas (PRU)	2: Existe un documento oficial que muestre al menos casos de prueba funcionales.
Proceso de mantenimiento (MAN)	4: Existe un proceso oficial y automatizado definido para trazabilidad, seguimiento y solución de incidencias.
Fecha última modificación	martes, 01 de febrero de 2011
Responsable	Ing. Jessica Zúñiga
Característica Técnicas	
Ruta del servidor de aplicaciones:	http://apps.ups.edu.ec
Inf. del servidor de aplicaciones:	Oracle Application Server 10g Versión 2 (10.1.2)
Ruta del servidor de reportes:	http://rep_sna.ups.edu.ec
Inf. del servidor de reportes:	Reports Server Versión 10.1.2.3.0
Base de datos:	Oracle Edición Estándar 11.2.0.4
Ruta formularios en producción:	http://apps.ups.edu.ec/reportes
Ruta reportes en producción:	http://apps.ups.edu.ec/forms
Librerías extras	webutil
Formularios relacionados:	be010101 - be010413 / be020101 - be020108 / be030102 - be030501
Reportes relacionados:	be010101 - be010413 / be020101 - be020108 / be030102 - be030501
Medición de Artefactos	
Número de Formularios	Parámetros: 38 Funcionalidades: 8
Número de Reportes	Parámetros: 38 Funcionalidades: 8 Reportes: 19
Número de tablas	

Valor Técnico	
Calidad de características **CC**	2: Cumple con los requerimientos actuales, pero se avizora nuevos requerimientos que no se podrían implementar.
Fiabilidad del servicio **FS**	4: Se han realizado cambios ocasionales para la implementación de cambios por la mejora o implementación de nuevos requerimientos
Costos de mantenimiento **CM**	1: Menos del 10% del presupuesto anual de mantenimiento de las aplicaciones.
Factor de degradación **FD**	4: Tecnología obsoleta, falta de soporte oficial pero con profesionales internos capacitados en la herramienta de desarrollo.
Valor Técnico	0,4
Valor de Negocio	
Ventaja Competitiva **VC**	5: El momento en que la aplicación deja de funcionar, los procesos ligados a ella no se pueden realizar ni siquiera de una forma manual
Impacto de rentabilidad **IR**	5: Las actividades de la aplicación no se pueden realizar manualmente y la rentabilidad que genera la aplicación es alta.
Interdependencia con otros sistemas **IS**	1: La aplicación interactúa con un alto porcentaje de aplicaciones a nivel funcional y de datos.
Seguridad **SG**	3: La aplicación tiene ciertos comportamientos erróneos, no muy frecuentes que han sido corregidos y que no han vuelto a suceder.
Valor de Negocio	3,5

Cuadrante de Análisis	MODERNIZACIÓN

$$vt = {(2 * 4)}/{5 * (1 * 4)} \qquad vn = {(5 + 5 + 1 + 3)}/{4}$$

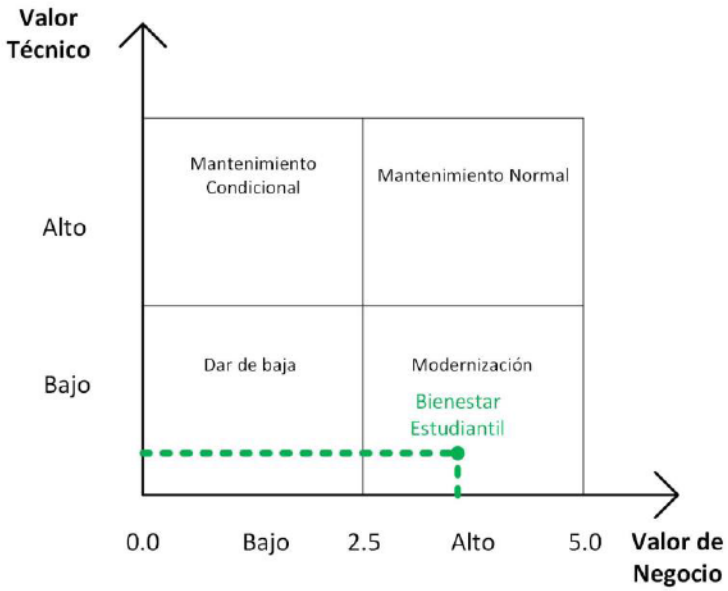


Figura 5.6: Cuadrante de análisis (Bienestar Estudiantil)

Según el cuadrante de análisis, se recomienda una modernización para la aplicación de Bienestar Estudiantil. De esta forma, se debe ir definiendo las prioridades de modernización de cada una de las aplicaciones registradas en el Catálogo de Sistemas de Software de la Universidad Politécnica Salesiana.

5.6. Etapa de integración de BPM en el proceso de requerimientos

Sabiendo que en la primera etapa se identifica las diferentes aplicaciones a modernizar, es en esta segunda etapa, en donde se genera una Especificación de requerimientos de software (SRS) a partir del análisis de procesos de negocios del sistema legado.

Para el presente caso de estudio, se continúa con la exposición de la aplicación de Bienestar Estudiantil; la misma que se recomendó modernizar en la etapa anterior y que posee un alto valor de negocio y un valor técnico bajo.

5.6.1. Identificar fuentes de conocimiento

Como primera actividad de la segunda etapa del método se propone tres tipos de fuentes de conocimiento:

Entorno organizacional

Dentro del entorno organizacional se pudo determinar tres documentos referentes a las normativas vigentes dentro del Departamento de Bienestar Estudiantil:

- ***Reglamento general de Bienestar Estudiantil***: Aprobado el 20 de noviembre del 2006.
- ***Reglamento general de becas***: Aprobado el 11 de septiembre del 2014.
- ***Reglamento interno de régimen académico de la Universidad Politécnica Salesiana***: Aprobado el 19 de febrero del 2014.

Stakeholders

Los stakeholders son las personas que interactúan de una u otra forma con el sistema. En este caso de estudio, se identifica los stakeholders principales según la normativa del Reglamento general de Bienestar Estudiantil y por otro lado, se recurre a los diferentes niveles de acceso y seguridad de la aplicación legada para identificar los diferentes usuarios y sus respectivas prioridades.

Tabla 5.7: Matriz de stakeholders (Bienestar Estudiantil)

Código	Nombre	Departamento	Cargo	Categoría	Prioridad
1	Lcdo. Fausto Sáenz	Secretaría Técnica B.E	Secretario Técnico	Cliente	1
2	Ing. Marcela Campoverde	Secretaría Técnica B.E	Asistente	Usuario	2
3	Lcda. Nancy Chumbay	Bienestar Estudiantil Cuenca	Director Técnico	Usuario	3
4	Lcda. Mónica Castro	Bienestar Estudiantil Guayaquil	Director Técnico	Usuario	3
5	Lcda. Irene Lema	Bienestar Estudiantil Quito	Director Técnico	Usuario	3

6	Lcda. Carmen Pauta	Bienestar Estudiantil Cuenca	Asistente	Usuario	5

Reglas de negocio

Para la extracción de las reglas de negocio se utilizó la herramienta de software denominada Finders Keepers™; la misma que permite buscar palabras o enunciados dentro de archivos de código fuente.

Para este caso de estudio, se realizó la búsqueda de condiciones de comparación y operadores lógicos en archivos .frm relacionados con la aplicación legada y así poder determinar las diferentes reglas de negocio.

A continuación, a manera de ejemplo se presenta tres reglas de negocio extraídas del sistema legado de Bienestar Estudiantil:

```
--CARGA DE VALORACIÓN POR DIRECCIÓN
IF (pv_ciudad_direccion != pv_ciudad_sede) OR
   (pv_ciudad_direccion != pv_ciudad_colegio) THEN
       ln_pro_codigo := 1; -- 25 puntos
ELSE
       ln_pro_codigo := 2; -- 50 puntos
END IF;

--VALIDAR FECHA DE SOLICITUD DE BECA
IF (SYSDATE BETWEEN ld_fec_ini AND ld_fec_fin) THEN
       pb_autorizar := TRUE;
       pv_fecha := 'EN RANGO';
ELSE
       pb_autorizar := FALSE;
       pv_fecha := 'FUERA DE EN RANGO';
END IF;

--VALIDAR QUE EL MONTO DE BECA NO EXEDA EL MONTO DE COBRO
IF (ln_monto_descuento > ln_monto_prefactura) THEN
       pn_execede := -1;
ELSE
       pn_execede := 0;
END IF;
```

5.6.2. Identificar arquitectura funcional

Luego de realizar el análisis arquitectónico propuesto en el *Método de integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados*, se identifica la arquitectura funcional de la aplicación de Bienestar Estudiantil; dicha estructura genérica se presenta en la siguiente enumeración:

4. Sistema de Bienestar Estudiantil
   - 4.1. General
     - 4.1.1.Parámetros
       - 4.1.1.1. Tipos de instituciones educativas
       - 4.1.1.2. Pensiones de colegios
       - 4.1.1.3. Rango de pensión
       - 4.1.1.4. Rango de pensiones por carrera
     - 4.1.2.Funcionalidades
     - 4.1.3.Reportes
       - 4.1.3.1. Historial de alumno
       - 4.1.3.2. Listado de alumnos por nivel
   - 4.2. Beca
     - 4.2.1.Parámetros
       - 4.2.1.1. Tipos de becas
       - 4.2.1.2. Porcentaje de beca
     - 4.2.2.Funcionalidades
       - 4.2.2.1. Beca
       - 4.2.2.2. Asignación de Beca con pre-factura generada
     - 4.2.3.Reportes
       - 4.2.3.1. Solicitudes de becas
       - 4.2.3.2. Becas asignadas
       - 4.2.3.3. Cuadro de resumen de becas
       - 4.2.3.4. Concesión de becas
       - 4.2.3.5. Distribución de becas por tipo
       - 4.2.3.6. Distribución de becas por categoría
   - 4.3. Pago diferido
     - 4.3.1.Parámetros
       - 4.3.1.1. Tipos de pago diferido
     - 4.3.2.Funcionalidades
       - 4.3.2.1. Pago diferido
     - 4.3.3.Reportes
       - 4.3.3.1. Número de solicitudes por tipo de pago
       - 4.3.3.2. Solicitudes de pago diferido
       - 4.3.3.3. Pagos diferidos asignados
       - 4.3.3.4. Concesión de pagos diferidos
       - 4.3.3.5. Distribución de pagos diferidos
   - 4.4. Crédito educativo con responsabilidad social
     - 4.4.1.Parámetros
       - 4.4.1.1. Porcentaje de CERS
       - 4.4.1.2. Tipo de clasificación de devolución
       - 4.4.1.3. Valor Hora
       - 4.4.1.4. Instituciones
       - 4.4.1.5. Supervisor
       - 4.4.1.6. Responsable
       - 4.4.1.7. Tipos de certificados
       - 4.4.1.8. Responsable de certificados
     - 4.4.2.Funcionalidades
       - 4.4.2.1. Registro de CERS
       - 4.4.2.2. Devolución CERS
       - 4.4.2.3. Certificado de no adeudar

4.4.2.4. Certificado de montos adeudados
4.4.3.Reportes
4.4.3.1. Estado de cuenta individual
4.4.3.2. Estado de cuenta general
4.4.3.3. Cartera por cobrar
4.4.3.4. Estado de devoluciones

5.6.3.Agrupar o segmentar funcionalidades

La agrupación o segmentación de funcionalidades tiene como finalidad la creación de diferentes tareas integradoras relacionadas con el sistema legado. Estas tareas integradoras, se convierten en la antesala de los procesos transversales asociados a la aplicación.

En el contexto del caso de estudio, se identificaron seis tareas integradoras: *Asignación de beca, Asignación de pago diferido, Asignación de crédito educativo, Registro de devolución de crédito educativo, Registro de Visita domiciliaria y Registro de ficha socioeconómica*. A continuación, en la Tabla 5.8 se presenta la definición de la tarea integradora Asignación de Beca:

Tabla 5.8: Tarea integradora (Asignación de beca)

Tarea integradora	
Código	SBE001
Descripción	Asignación de Beca
Sistema	Sistema Nacional Académico
Subsistema	Bienestar estudiantil
Módulo	Beca
Descripción detallada	Para asignar una beca, el estudiante debe tener una inscripción vigente y solicitar un tipo de beca específica. Por otro lado, se puede asignar o no la beca solicitada con el porcentaje específico. Este porcentaje, se convertirá en el descuento que obtendrá el estudiante cuando realice la matrícula.
Tareas asociadas	4.2.1.1 4.2.1.2 4.2.2.1 4.2.2.2 4.2.3.*
Subtareas asociadas	
Responsable	Ing. Marcela Campoverde

5.6.4. Diseñar procesos asociados a funcionalidades

Con toda la información relevada en las tareas anteriores de esta segunda etapa, se procedió con la diseño de un proceso por cada tarea integradora definida. Para el diseño del proceso, se tomó en cuenta todas las recomendaciones y actividades presentadas en la sección 4.3.4 (Diseñar procesos asociados a funcionalidades).

Para el caso de estudio propuesto, se procede con el diseño de la tarea integradora denominada "*Asignación de beca*", presentando el modelo del mismo en la Figura 5.7:

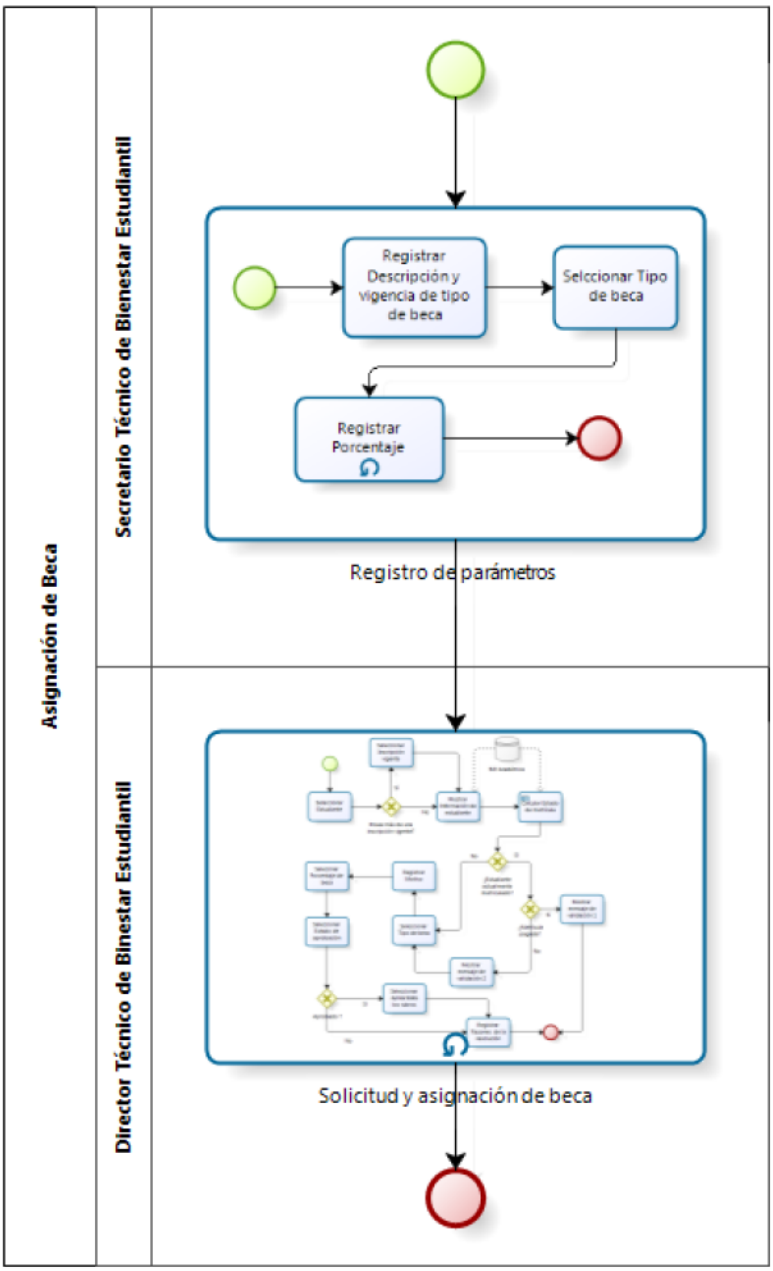


Figura 5.7: Proceso de Asignación de beca

El proceso de "Asignación de beca" consta a su vez de dos subprocesos denominados: "*Registro de Parámetros*" representado en la Figura 5.8 y "*Solicitud y aprobación de beca*" representado en la Figura 5.9.

Cada subproceso tiene un actor responsable y cada tarea es identificada con el prefijo que indica su tipo. A continuación, se describe cuatro ejemplos de tareas:

- ***Seleccionar*** *Estudiante*: Significa que existe una agregación con estudiantes registrados por separado y que la tarea los permite seleccionar de una lista de valores.
- ***Mostrar*** *Información de estudiante*: En esta tarea, existen etiquetas informativas que deben ser consultadas de un repositorio de información. La información a mostrarse, se encuentra detallada en la descripción de la tarea.
- ***Calcular*** Estado de matrícula: Es un cálculo o regla de negocio que recupera si el estudiante ha generado costos por la inscripción en los diferentes cursos académicos. Esta información, se encuentra en una base de datos ajena al proceso principal.
- ***Registrar*** *Motivo*: Significa que no existe un dominio previamente definido y que por lo tanto, se debe registrar un texto directamente por parte del usuario.

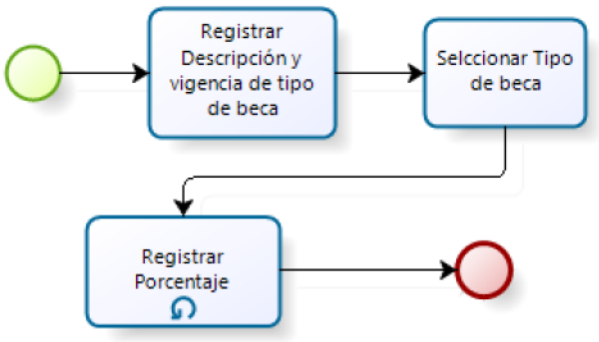


Figura 5.8: Subproceso de Registro de Parámetros

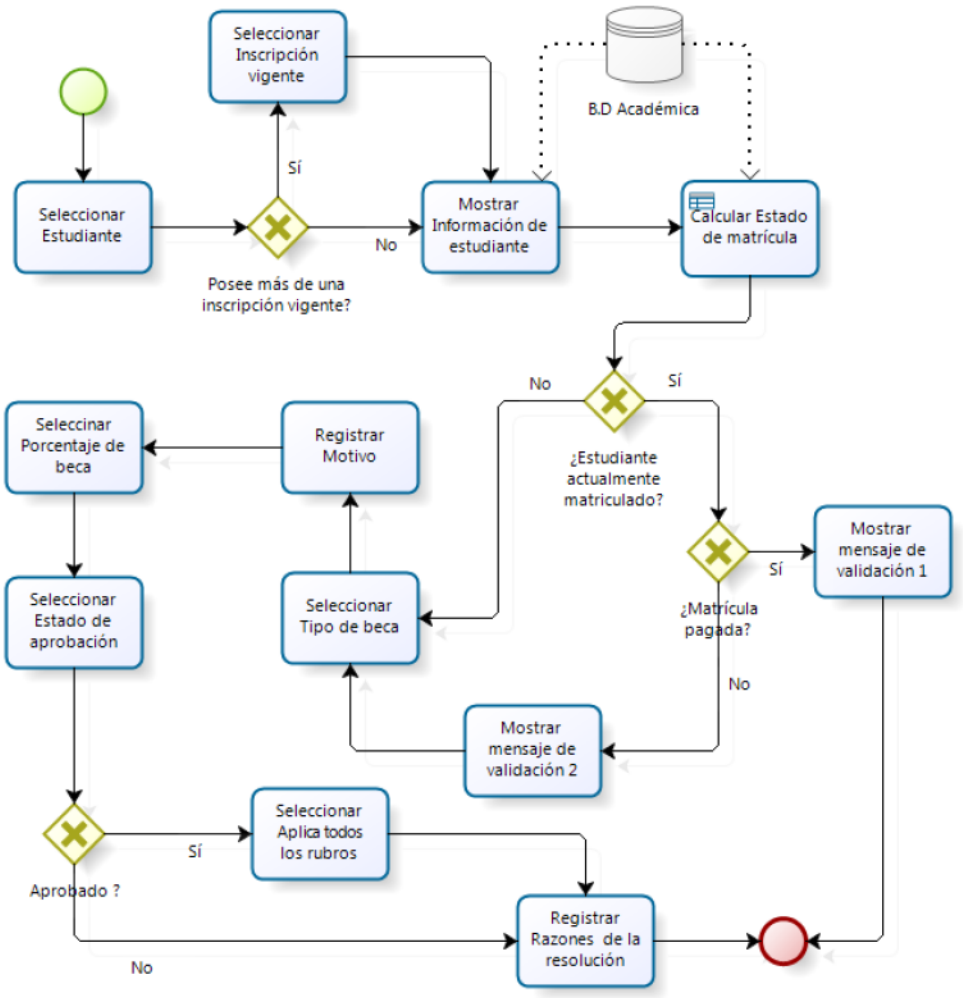


Figura 5.9: Subproceso de Solicitud y aprobación de beca

La definición del ***"Proceso de asignación de beca"***, fue validada por todos los stakeholders involucrados. Esta validación, se llevó a cabo luego de una refinación del mismo y llegando a un consenso con los usuarios.

5.6.5. Especificar requerimientos del sistema legado

En este punto, se redactó un producto de trabajo final denominado Especificación de requerimientos de software (SRS); el mismo que fue creado según la estructura propuesta en la sección 4.3.5 (Especificar requerimientos del sistema legado).

En la Tabla 5.9 se presenta un ejemplo relacionado con la descripción del requerimiento funcional presentado en la Figura 5.9 y que se encuentra formando parte de la SRS creada.

Tabla 5.9: Solicitud y asignación de beca

Código:	**1001**
Descripción:	Solicitud y asignación de beca
Descripción Detallada:	La asignación de beca, se puede ver como dos subprocesos integrados en uno solo (Figura 1). El primer subproceso, denominado *Registro de parámetros* (Figura 2) es el encargado de definir los diferentes tipos de becas y porcentajes asociados, mientras que el subproceso de *Solicitud y asignación de beca* (Figura 3) será el encargado de registrar los datos asociados a la beca; la misma que será traducida como un descuento en el costo de la matrícula.
Pre-condiciones	• Debe existir registrados tipos y porcentajes de beca. • El estudiante debe tener al menos una inscripción vigente.
Entradas:	• Estudiante: Número entero (Tipo) • Inscripción: Número entero (Tipo) • Tipo de beca: Número entero (Tipo) • Motivo: Texto(2000) • Porcentaje de beca: Número entero (Tipo) • Estado de aprobación: Texto(1) • Aplica todos los Rubros: Texto(1) • Razones de resolución: Texto(4000)
Proceso:	Figura 3
Salidas:	• Nombre del estudiante: Texto • Descripción de la inscripción: Texto • Descripción del tipo de beca: Texto • Tipo de beca anterior: Texto • Porcentaje de beca anterior: Número entero • Estado de beca anterior: Texto • Tipo de alumno: Texto • Quintil: Número entero • Nro materias aprobadas: Número entero • Nro materias reprobadas: Número entero • Promedio último ciclo: Número decimal • Promedio general: Número decimal
Post-condiciones:	• Solicitud de beca registrada. • Beca registrada en el caso de que el estado de aprobación sea Aprobado.
Fuentes:	Sistema legado de Bienestar estudiantil Director Técnico de Bienestar estudiantil Cuenca
Roles:	Director Técnico

5.7. Análisis y presentación de los resultados

Es esta sección, se analiza desde diferentes puntos de vista al caso de estudio y se plantea un conjunto de beneficios alcanzados como resultado de la implementación del método en el caso de estudio propuesto y como esta implementación ha contribuido directamente con la gestión de la Coordinación de desarrollo de software y otras áreas de tecnología de la Universidad Politécnica Salesiana:

- ***Definición de sistemas de información como legados:*** Definir un sistema como legado dentro de la Coordinación de desarrollo se ha venido realizando gracias al juicio experto, pero esta situación, ha generado sesgos por la intervención explícita del ser humano en las decisiones. La implementación del método en su primera etapa, permite evitar estos sesgos en la consideración de un sistema como legado, ya que a todas las aplicaciones se las analiza con la utilización de un conjunto de parámetros similares y previamente especificados.

  Los parámetros que se utilizaron en el caso de estudio fueron definidos de una manera transparente y objetiva por diferentes actores del área tecnológica de la Universidad Politécnica Salesiana: Director de sistemas, Secretario técnico de tecnologías de información, Coordinador de desarrollo y diferentes miembros del equipo de desarrollo. Es decir, existe un consentimiento y aprobación general de los parámetros que permiten a la primera etapa del método determinar si un sistema de información es considerado como legado.

- ***Catálogo de sistemas de software:*** Al crear un catálogo que organiza las aplicaciones de acuerdo a las categorías tecnológicas propuestas en el caso de estudio, se logra determinar información verificable y sujeta a un control de cambios de todas las aplicaciones existentes dentro de la Coordinación de desarrollo.

  Por otro lado, el Catálogo de sistemas de software (Figura 5.4), ha sido un insumo fundamental para que la Coordinación de explotación[46] pueda definir a los sistemas de información como servicios transversales en la implementación de una herramienta de helpdesk.

- ***Especificar prioridades de modernización***: Especificar las prioridades de modernización se convierte en la clave para encarar una modernización objetiva de los sistemas legados y sobre todo en el caso de estudio, permite ser la base para la creación del *Plan de modernización para los sistemas de información de la Universidad Politécnica Salesiana* que se presenta en la sección 5.2 de antecedentes. Cada uno[47] de los sistemas de la organización han pasado por esta actividad de la primera etapa y por lo tanto se ha especificado exactamente en cada caso si necesitan ser dados de baja o ser candidatos al plan de modernización.

- ***Fuentes de conocimiento:*** Con la implementación de la segunda etapa del método, se concluye que toda aplicación que pretenda ser modernizada debe tener stakeholders "*clientes*" que sean responsables ejecutivos de la modernización y

[46] La Coordinación de explotación de la Universidad Politécnica Salesiana se encarga de la formación y soporte a usuarios en lo referente a los sistemas de información.
[47] Hasta el momento se ha utilizado el método en los Sistemas de Gestión de talento humano, Admisión y Nivelación, Gestión de la investigación y Bienestar estudiantil.

stakeholders "*usuarios*" que son los directamente involucrados en el uso del sistema legado.

La identificación de las reglas de negocio es una parte más compleja si se pretende que los stakeholders sean los encargados de detallarlas, ya que en muchos casos las reglas parecen implícitas para los stakeholders y no son expuestas en la definición del proceso. Es por tal motivo, que se utilizó buscadores de código fuente que encuentren operadores relacionales y extrapolarlos en reglas de negocio.

- ***Arquitecturas funcionales***: Mediante el uso de la arquitectura funcional genérica propuesta en la sección 5.6.2 (sistema, subsistema, módulo, menú y tarea) se puede estandarizar las funcionalidades de los sistemas legados.

  La ventaja principal de tener una estandarización a nivel funcional, radica que el stakeholder establece explícitamente un patrón de tareas (parámetros, funcionalidades y reportes) que puede realizar dentro de los diferentes sistemas de información.

- ***Diseñar procesos asociados a funcionalidades***: Al convertir a BPMN2 en el estándar para el análisis de las funcionalidades de los sistemas legados y reconociendo que los stakeholders no conocían esta notación, entonces se comienza a brindar una capacitación básica *in situ*, para que los diferentes involucrados puedan reducir el gap con los analistas.

  Una ventaja de esta propuesta es la acotación de las tareas BPMN2 a: seleccionar, registrar, calcular y mostrar. Esta acotación, permite que los usuarios entiendan el concepto básico del proceso y puedan realizar una refinación del mismo mediante la identificación de cuellos de botella dentro del proceso implícito en el sistema legado. Una refinación del proceso actual no es obligatorio, pero de ser el caso debe generar una mayor eficiencia a nivel administrativo. A manera de ejemplo, en la actualidad el proceso de "asignación de beca" del sistema legado dispone de dos pantallas en donde se realizan 18 selecciones y 2 registros de información para un mismo proceso de negocio. Mientras que, con el nuevo proceso (Figura 5.9) se podría[48] realizar 6 selecciones y 2 registros de información en un mismo proceso y sin perder ningún tipo de información.

- ***Especificaciones de requerimientos***: Se crea especificaciones de requerimientos de software que se rigen a estándares internacionales y que se convierten en la base para una necesaria comprensión previa de los stakeholders y una abstracción bastante apegada al funcionamiento actual del sistema legado.

  El control de versiones se realiza de una manera óptima siendo bienvenidos los cambios que el usuario desee en el proceso.

  Los diferentes actores de la fase de diseño, implementación, pruebas y mantenimiento se ven beneficiados porque poseen de un documento clave que puede ser utilizado en todas las fases del proceso de software, ya que es aquí en donde se especifica las necesidades del usuario en torno al sistema legado en producción.

  Al considerar al proceso de negocio como una tarea transversal entre funciones; muchos de los procesos determinados en la especificación de requerimientos, se han

[48] Si bien esta propuesta gira en torno a la fase de análisis, el nuevo proceso puede ser implementado en cualquier herramienta informática.

convertido en la fuente para la definición de servicios de software transversales entre sistemas de software. Esta definición de servicios, se ajusta dentro de las buenas prácticas de ITIL que pretende implementar el departamento de sistemas.

6. Conclusiones

Las conclusiones son enunciados o descripciones breves que pretenden resumir y explicar en ciertos casos los puntos principales de la tesis. Además, en este capítulo se plantea futuras líneas de investigación que podrían ser derivadas de este trabajo.

6.1. Conclusiones

La mayoría de autores citados con respecto a los sistemas legados, coinciden más que en una definición, en un conjunto de características inherentes a la obsolescencia que un sistema de información debe poseer y que usualmente se relacionan con aplicaciones de más de diez años de existencia que se conservan en funcionamiento por la situación crítica que mantienen dentro de la organización. En el ámbito empresarial, se pueden encontrar un sinnúmero de aplicaciones de software que por sus condiciones pueden llegar a ser legadas. De hecho, "*las mejores soluciones actuales, serán los sistemas legados del mañana*". (Zhang, et al., 2010)

Con el objeto de estudiar los sistemas legados, se los pueden clasificar desde tres puntos de vista: mediante generaciones de lenguajes, mediante nivel de acoplamiento y mediante el tipo de migración requerida. En esta última categorización, destaca la *migración indirecta*, la misma que propone el reemplazo de los sistemas legados mediante la reestructuración de una nueva aplicación con plataformas tecnológicas actuales.

La modernización de los sistemas legados, es otro punto a tomar en cuenta, ya que las investigaciones sobre el tema, se pueden organizar en cinco grupos de metodologías de modernización: mediante SOA, a través de trasformaciones MDD, mediante metodologías genéricas, mediante la utilización de metodologías personalizadas dependiendo de un dominio específico y finalmente, mediante metodologías que usan procesos de negocio.

Como antecedente para la integración de BPM en esta tesis, se puede precisar que todavía existen algunas organizaciones que mantienen una gestión tradicional basada en la gestión funcional. Esto significa, que los diferentes empleados cumplen funciones inherentes a su cargo y que generan escenarios que van en contraposición con la transversalidad de tareas y actores que propone la gestión por procesos de negocio.

Con respecto a la clasificación de procesos de negocio, se puede presentar cuatro grandes categorizaciones: procesos de negocio enfocados en la relevancia o función que tienen dentro de la organización (Hernandez, 2010), procesos de negocio organizados según su grado de automatización (Weske, 2012), procesos de negocio dependientes de las soluciones para el alineamiento entre los procesos modelados e implementación (Hertis & Juric, 2013) y por último, procesos de negocio clasificados de acuerdo a su comportamiento interno o interacción entre tareas. (Melao & Pidd, 2000)

BPM provee una serie de beneficios tanto desde el punto de vista organizacional, así como desde el punto de vista del analista del negocio. En cuanto a la organización, BPM proporciona un gestión caracterizada por: efectividad, eficiencia, consistencia, productividad, ahorro y calidad (Noguera, 2011). Mientras que para el analista, la ventaja radica en el grado de abstracción, ya que esta característica admite trabajar independientemente de la plataforma de implementación y por ende permitirá agilizar el proceso de desarrollo e identificar posibles errores en fases tempranas.

La calidad, se ha convertido en una característica íntimamente ligada con los procesos de negocio (Beimborn & Joachim, 2011), a tal punto que el ciclo de vida de los procesos de negocio se encuentra basado en el círculo de mejora continua PDCA[49] de Deming.

Dentro de las principales ventajas de BPM y como una razón de peso para su utilización en esta tesis, se encuentra la adecuada visualización de los procesos de negocio en la etapa de modelamiento, para esta visualización se utiliza la notación estándar denominada BPMN2 (OMG, 2011); la misma que ha sido desarrollada teniendo en cuenta el objetivo de superar las limitaciones de otras notaciones que impedían su implementación en las aplicaciones de negocio, científicas y de ingeniería. (Abdelahad, et al., 2013)

En lo referente al área de conocimiento de los requerimientos de software, su importancia radica en la intervención de dicha área tanto al inicio, como al final del proceso de software, ya que en la práctica, los requerimientos se convierten en la descripción de lo que se debe diseñar y en la base para verificar que la funcionalidad del producto de software sea la correcta.

Otro punto a tomar en cuenta, es el proceso de requerimientos de software, entendiendo al mismo como la manera en la que interactúan un conjunto de involucrados, técnicas y actividades en la recolección, estudio y análisis del dominio del negocio. Dicho análisis, es la base para la producción de una especificación puntual que permita la validación de la información detallada. Dentro de dicho proceso, se identifican tres fases relativamente paralelas y que se denominan: elicitación y análisis, especificación y validación (Wiegers, 2003).

La fase de análisis de requerimientos de software, consiste en procesar el conocimiento capturado y examinarlo en integral para detectar conflictos en los requerimientos, elaborar requerimientos de sistema, derivar requerimientos de software, descubrir los límites de los requerimientos y determinar la interacción de los requerimientos con el ambiente organizacional. (Bourque & Fairley, 2014)

En la misma línea del análisis de requerimientos, se encuentran los modelos conceptuales, generalmente representaciones gráficas que no se deben confundir con los diseños de la solución del requerimiento, ya que el *modelo conceptual* pretende representar el funcionamiento actual del requerimiento, las entidades del dominio del problema y las relaciones del requerimiento con el mundo real.

La principal problemática que motivó la profundización del estado del arte referente a los sistemas legados, se centra en la identificación de cuatro necesidades de modernización: documentación inadecuada, complejidad de las aplicaciones, lenguajes o suites legados de implementación y la dispersión de la arquitectura de software. Por otro lado, y en lo que respecta a la representación de las funcionalidades del sistema legado, en esta tesis se presenta un conjunto de limitaciones asociadas a los principales modelos conceptuales: descripciones textuales, casos de uso, flujos de datos, actividades, metas o estados.

[49] **PDCA**: Plan Do Check Act

Ante las necesidades de modernización y las limitaciones en ciertas representaciones del modelo conceptual, se propone como solución central al "*método de integración de BPM en el proceso de requerimientos de software para la modernización de sistemas legados*", el mismo que consta a su vez de dos etapas denominadas: identificación de sistemas legados e integración de BPM en el proceso de requerimientos.

Dentro de la etapa de *identificación de sistemas legados* se definen tres tareas principales, la primera es la definición de parámetros de identificación, la segunda es la catalogación de los sistemas de software y en tercer lugar se especifica las prioridades de modernización para cada sistema previamente catalogado. En lo que respecta a la definición de parámetros, se consigue identificar un conjunto de características comunes que pueden ser utilizadas en la recolección de toda la información referente a los sistemas a ser analizados; esta recolección, es la segunda tarea de esta etapa y tiene como fin la creación de un Catálogo de sistemas de software generado a partir de un punto de vista técnico, objetivo y ligado a parámetros comunes que evitan recurrir a otros conjuntos de posibilidades de identificación como el juicio experto, las decisiones políticas o los usuarios exigentes. Finalmente, la etapa de identificación de sistemas legados apunta a la utilización del valor técnico y el valor del negocio (Dedeke, 2012) como una medida cuantitativa para el establecimiento de las prioridades de modernización y sugerir la toma de decisiones referentes a: dar de baja, realizar un mantenimiento normal, realizar un mantenimiento condicional o modernizar la aplicación analizada.

La segunda etapa del método propuesto en esta tesis se denomina *integración de BPM en el proceso de requerimientos* y tiene como objetivo principal la generación de una especificación de requerimientos de software en base al análisis de procesos de negocio modelados en BPMN2. Dicha especificación es el fruto de la interacción de cinco tareas secuenciales denominadas: identificar fuentes de conocimiento (Ballejos & Montagna, 2011) (do Nascimento, et al., 2012), identificar arquitectura funcional (Maier, et al., 2001), agrupar o segmentar funcionalidades (Baghdadi & Al-Bulushi, 2013), diseñar procesos asociados a funcionalidades (Márquez, 2010) y especificar requerimientos del sistema legado (ISO/IEC/IEEE, 2011).

Diseñar los procesos asociados a funcionalidades es una actividad clave en la *etapa de integración de BPM en el proceso de requerimientos,* ya que mediante la utilización de la notación BPMN2 y la ejecución de cuatro tareas secuenciales denominadas: proceso básico, asociación de participantes, diseño detallado y validación del proceso (Márquez, 2010) se realiza un modelado adecuado de los procesos de negocio asociados al sistema legado en estudio. El diseño de los diferentes procesos de negocio, se convierte en la base para la sistematización general de toda la información recolectada y analizada. Esta sistematización se ve reflejada en la creación de un producto de trabajo denominado Especificación de Requerimientos de Software, producto que a su vez permite cumplir con el objetivo principal planteado en la conceptualización de este trabajo de tesis.

En lo que respecta a la validez de la propuesta, no se puede asegurar que sea completamente aplicable en otros ambientes de modernización, ya que se ha realizado solamente un caso de estudio y de cierta manera con el sesgo del autor por su calidad de Coordinador de desarrollo. En otro sentido, es loable comentar que existe un balance positivo en la implementación de la propuesta, ya que si antes no se tenía una metodología clara para modernizar los sistemas legados, en esta ocasión existe un

método soportado en el marco teórico del capítulo dos y que hace posible tener una propuesta con fundamentos teóricos y científicos sólidos.

Basados en el caso de estudio y continuando con la descripción de las diferentes ventajas que se generaron con la implementación de la propuesta, se puede testificar que ahora la especificación de requerimientos se ha convertido en parte crucial para el diseño y verificación de los diferentes entregables o productos de software, mientras que antes la especificación de requerimientos era un formalismo contractual que quedaba archivado. En lo referente a los procesos de negocio, se ha planteado una cultura de especificación de funcionalidades transversales y de que cualquier sistema legado puede refinar su proceso.

Dentro de las principales desventajas de la propuesta, está la falta de profundidad en el concepto de BPM, ya que al acotar la propuesta a la etapa de requerimientos, se pierden muchas funcionalidades ligadas íntimamente con el diseño de la solución. Por otro lado, la falta de conciencia del concepto de procesos por parte de los diferentes actores de la organización, hace que la notación BPMN2 no pueda ser aprovechada en su totalidad, ya que en lugar de reducir el gap usuario/analista, terminaría aumentándolo y por lo tanto creando una especificación demasiado compleja, difícil de entender y que podría terminar nuevamente en el archivo.

6.2. Futuras líneas de investigación

En ningún caso se asume que los contenidos tratados en este trabajo han sido abordados en su totalidad, ya que los sistemas legados, gestión por procesos de negocio y proceso de requerimientos son temas muy extensos y por lo tanto han sido acotados para que puedan ser adecuados a la propuesta principal de la tesis.

Tomando en cuenta el párrafo anterior, se puede afirmar que permanecen en el tintero ciertos temas relacionados con este trabajo y que a su vez pueden ser considerados en un futuro como líneas de investigación.

En esta tesis, se realiza el análisis y especificación de toda la información relacionada con el sistema legado. Una vez generada la especificación, se podría crear una metodología para la utilización de esta información en el diseño de una solución que responda a la especificación presentada. Es decir, en base a la especificación de debería continuar en el desarrollo de la fase de diseño e implementación del Plan de modernización.

Con los procesos de negocio modelados en BPMN2, se puede profundizar en el diseño e implementación de los mismos mediante la investigación de automatizaciones BPMS.

Otra línea de investigación, puede direccionarse en la determinación de cómo la especificación de requerimientos se relaciona con la gestión del proyecto de modernización, ya que la idea de relacionar la gestión con los requerimientos viene de ciertos autores que plantean que "*muchas veces los requerimientos sirven solamente como un contrato, pero nada más en profundidad*" (Ralph, 2013). Es decir, una vez que los requerimientos se conviertan en la base contractual, investigar cómo puede esta especificación ayudar en la gestión del proyecto. Por ejemplo, en la definición de product backlogs (SCRUMstudy™, 2013) de proyectos ágiles.

Referencias

Aalst, W. & Hofstede, A., 2012. Workflow patterns put into context. *Software & Systems Modeling,* Volumen 11, pp. 319-323.

Abdelahad, C. & Riesco, D. E., 2012. *Transformación de Workflows Científicos a BPMN2.* Posadas, s.n.

Abdelahad, C. y otros, 2013. *Extendiendo BPMN2 para soportar workflows científicos de ESTECO..* Paraná, s.n.

Alexander, I. & Beus-Dukic, L., 2009. *Discovering requirements: how to specify products and services.* s.l.:John Wiley & Sons.

Alvarado, P., 2011. *BONITA SOFT: Gestor de procesos de negocios BPM.* s.l.:s.n.

Antonelli, L., Rossi, G., do Prado Leite, J. C. S. & Oliveros, A., 2012. Deriving requirements specifications from the application domain language captured by Language Extended Lexicon. *WER.*

Aseeva, N., Babkin, E. & Kozyrev, O., 2012. Architecture and Language for Semantic Reduction of Domain-Specific Models in BPMS. En: *Lecture Notes in Business Information Processing.* s.l.:Springer Berlin Heidelberg, pp. 70-84.

Baghdadi, Y. & Al-Bulushi, W., 2013. A guidance process to modernize legacy applications for SOA. *Service Oriented Computing and Applications,* pp. 1-18.

Ballejos, L. & Montagna, J., 2011. Modeling stakeholders for information systems design processes. *Requirements Engineering,* 16(4), pp. 281 - 296.

Barnes, N. & Jones, D., 2011. Clear Climate Code: Rewriting Legacy Science. *Software,* Volumen 28, pp. 36-42.

Bazán, P., 2009. *Un modelo de integrabilidad con SOA y BPM.* La Plata: Universidad Nacional de La Plata - Facultad de Informática.

Bazan, P., Perez, G., Giandini, R. & Diaz, J., 2011. *Process-Service Interactions Using a SOA-BPM-Based Methodology.* Curico, s.n.

Bazan, P. y otros, 2012. *Services conceptualization within SOA/BPM methodology.* Medellin, s.n.

Beimborn, D. & Joachim, N., 2011. The joint impact of service-oriented architectures and business process management on business process quality: An empirical evaluation and comparison. *Information Systems and e-Business Management,* 9(3), pp. pp 333-362.

Bera, P. & Evermann, J., 2014. Guidelines for using UML association classes and their effect on domain understanding in requirements engineering. *Requirements Engineering,* 19(1), pp. 63 - 80.

Berón, M., Salgado, C. H., Peralta, M. & Saez, F., 2013. *Comprensión de especificaciones de procesos de negocios escritas en BPMN.* Paraná, s.n.

Bizagi, 2013. *Patrones de Modelado de Procesos.* [En línea]
Available at:
http://www.bizagi.com/docs/Workflow_Patterns_using_BizAgi_Process_Modeler_Esp.pdf
[Último acceso: 24 02 2014].

Blunden, R. B., 2012. *Software exorcism.* s.l.:Apress.

Booch, G., 2012. Facing Future. *Software, IEEE,* 29(2), pp. 20-21.

Bourque, P. & Fairley, R., 2014. *Guide to the Software Body of Knowledge SWEBOK.* 3.0 ed. s.l.:IEEE Computer Society.

BPMN, 2011. *BPMN 2.0 Poster.* Berlín: s.n.

Brodie, M. & Stonebraker, M., 1995. *Migrating legacy systems: Gateways, interfaces & the incremental approach.* s.l.:Morgan Kaufmann Publishers Inc..

Canfora, G., Fasolino, A. R., Frattolillo, G. & Tramontana, P., 2008. A wrapping approach for migrating legacy system interactive functionalities to service oriented architectures. *Journal of Systems and Software,* 81(4), pp. 463--480.

Canning, R., 1984. Rejuvenate your old systems. *EDP Analyzer,* 22(3), pp. 1 - 16.

Castillo, P., 2011. *BONITA SOFT: Gestor de procesos de negocios BPM.* [En línea] [Último acceso: 2013 08 21].

Cataldi, Z. & Lage, F. J., 2004. *Diseño y organización de tesis.* Buenos Aires: Nueva Libreria.

Conover, H. y otros, 2013. Introducing Provenance Capture into a Legacy. *Geoscience and Remote Sensing,,* 51(11).

Cruz, D., Fontana, J., Rivadeneira, S. & Vilanova, G., 2013. *Un acercamiento en la integración entre BPMN y SOA.* Paraná, s.n.

Davenport, T., 1993. *Process innovation: reengineering work through information technology.* s.l.:Harvard Business Press.

Dayal, U., Hsu, M. & Ladin, R., 2001. Business Process Coordination: State of the Art, Trends, and Open Issues. En: *VLDB.* s.l.:s.n., pp. 3-13.

De la Vara, J. L., 2008. *Captura de Requisitos de Sistemas de Información a partir de procesos de negocio y metas.* Valencia: Tesis Magister .

De la Vara, J. L., 2011. *Business process-based requirements specification and object-oriented conceptual modelling of information systems.* Valencia: Tesis PHD.

De la Vara, J. L., Alcolea, D. A. & Diaz, J. S., 2007. *Descomposición de árboles de metas a partir de modelos de procesos..* Toronto, s.n.

De la Vara, J. L., Alcolea, D. A. & Sanchez, J., 2007. *Construcción de modelos de requisitos a partir de modelos de procesos y de metas.* Isla de Marguerita, s.n.

De la Vara, J. L., Diaz, J. S. & Lopez, O. P., 2007. *Integración de un entorno de producción automática de software en un marco de alineamiento estratégico.* Toronto, s.n.

De la Vara, J. L. y otros, 2009. *Modelado de Requisitos de Datos para Sistemas de Información basados en Procesos de Negocio.* Medellín, s.n.

De la Vara, J. L. y otros, 2012. Towards a model-based evolutionary chain of evidence for compliance with safety standards. En: *Computer Safety, Reliability, and Security.* Magdeburg: Springer, pp. 64 - 78.

De la Vara, J. L. & Sanchez, J., 2008. *Improving requirements analysis through business process modelling: a participative approach.* Innsbruck, Austria, Springer, pp. 165-176.

De la Vara, J. L. & Sánchez, J., 2009. BPMN-based specification of task descriptions: Approach and lessons learnt. En: *Requirements Engineering: Foundation for Software Quality.* s.l.:Springer, pp. 124-138.

De la Vara, J. L. & Sánchez, J., 2010. *System Modeling from Extended Task Descriptions.* Redwood City, s.n.

De la Vara, J. L., Sánchez, J. & Pastor, O., 2008. *Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems.* Montpellier, s.n.

De la Vara, J. L., Sánchez, J. & Pastor, O., 2013. On the Use of Goal Models and Business Process Models for Elicitation of System Requirements. En: *Enterprise, Business-Process and Information Systems Modeling.* s.l.:Springer, pp. 168 - 183.

De la Vara, J. L. y otros, 2011. *An Empirical Study on the Importance of Quality Requirements in Industry.* Miami Beach, s.n.

Decreus, K., Snoeck, M. & Poels, G., 2009. Practical Challenges for Methods Transforming i* Goal Models into Business Process Models. *Requirements Engineering Conference, 2009. RE '09. 17Th IEEE International,* pp. 15 - 23.

Dedeke, A., 2012. Improving Legacy-System Sustainability: A Systematic Approach. *IT Professional,* 14(11), pp. 38-43.

Díaz, J. F. y otros, 2009. *Entornos para usar BPM en aplicaciones JAVA: Un análisis comparativo.* San Juan, s.n.

Díaz, J. P. J. G. J., 2014. A model for tracing variability from features to product-line architectures: a case study in smart grids. *Requirements Engineering,* pp. 1-21.

do Nascimento, G. y otros, 2012. *Identifying business rules to legacy systems reengineering based on bpm and soa.* Ho Chi Minh, Springer.

do Nascimento, G. S., Iochpe, C., Thom, H. L. & Reichert, M., 2009. *A Method for Rewriting Legacy Systems using Business Process Management Technology.* Milan, s.n.

Dragicevic, S. & Celar, S., 2013. Method for elicitation, documentation and validation of software user requirements (MEDoV). *Computers and Communications (ISCC), 2013 IEEE Symposium on.*

Duarte, N. & Costa, N., 2013. A Set of Requirements for Business Process Management Suite (BPMS). En: *Advances in Information Systems and Technologies.* s.l.:Springer, pp. 487--496.

Dumas, M., Van-der-Aalst, W.-M. & Ter-Hofstede, A., 2005. *Process-aware information systems: bridging people and software through process technology.* s.l.:Springer.

Durán Toro, A., 2000. *Un entorno metodológico de ingeniería de requisitos para sistemas de información.* Sevilla: Tesis Doctoral.

El-Attar, M. & Miller, J., 2012. Constructing high quality use case models: A systematic review of current practices. *Requirements Engineering,* 17(3), pp. 187 - 201.

Fleischmann, S. O. A., 2012. *S-BPM ONE-Education and Industrial Developments.* s.l.:Springer.

Funes, A., Reinoso, E., Castro, M. & Dasso, A., 2012. *Creación y evaluación de modelos LSP en un contexto MDA.* s.l., s.n.

Giandini, R., Perez, G. & Pons, C., 2010. *Un lenguaje de Transformación específico para Modelos de Proceso del Negocio.* San Lorenzo, s.n.

Gold, N., 1998. *The Meaning of "legacy Systems".* s.l.:Department of Computer Science, University of Durham.

Gomez, H., 2010. *Elaboración de una guía para la migración de sistemas legados OracleForms6i hacia una arquitectura multi-capa.* Cali: s.n.

Gu, Q. & Lago, P., 2011. Guiding the selection of service-oriented software engineering methodologies. *Service Oriented Computing and Applications,* 5(4), pp. 203-233.

Haberecht, T., Muller, J. & De Angelis, G., 2011. The Challenge of Dynamic Services in Business Process Management. *International Journal of Digital Society (IJDS),* 2(3), pp. 539 - 546.

Hadar, I., Soffer, P. & Kenzi, K., 2014. The role of domain knowledge in requirements elicitation via interviews: an exploratory study. *Requirements Engineering,* 19(2), pp. 143-159.

Haskins, C. y otros, 2006. *Systems engineering handbook.* s.l.:INCOSE.

Hernandez, A., 2010. Identificacion de Procesos de Negocio. *Ingenieria Industrial,* 25(3), p. 6.

Hertis, M. & Juric, M. B., 2013. Ideas on improving the Business-IT alignment in BPM enabled by SOA. *Information and Communication Technology (ICoICT), 2013 International Conference of,* pp. 55 -- 60.

Holschke, D.-I., Rake, D.-I. J., Offermann, P. & Bub, U., 2010. Improving software flexibility for business process changes. *Business & Information Systems Engineering,* 2(1), pp. 3--13.

Huang, Y.-C. & Chu, C.-P., 2012. Legacy System User Interface Reengineering Based on the Agile Model Driven Approach. En: *Recent Advances in Computer Science and Information Engineering.* s.l.:Springer, pp. 309-314.

Hurwitz, J., Bloor, R., Baroudi, C. & Kaufman, M., 2007. *Service oriented architecture for dummies.* Indianapolis: Willey.

ISO/IEC/IEEE, 2010. *24765:2010 Systems and software engineering--Vocabulary.* Ginebra: s.n.

ISO/IEC/IEEE, 2011. *ISO/IEC/IEEE 29148:2011 Systems and software engineering--Life cycle processes--Requirements engineering.* Ginebra: s.n.

ISO/IEC/IEEE, 2011. *ISO/IEC/IEEE 42010:2011 Systems and software engineering - Architecture description.* s.l.:s.n.

ISO/IEC, 1. S. E. L. C. P., 2006. *ISO/IEC.* s.l.: ISO/IEC, 14764, 2006..

ISO/IEC, 2013. *ISO/IEC 25064:2013 Systems and software engineering--Software product Quality Requirements and Evaluation (SQuaRE)--Common Industry Format (CIF) for usability: User needs report.* Ginebra: s.n.

ISO, N., 2004. 9000: 2000. *Sistemas de gestión de la calidad. Fundamentos y vocabulario.*

Juric, M.-B., Loganathan, R., Poornachandra, S. & Frank, J., 2007. *SOA Approach to Integration: XML, Web services, ESB, and BPEL in real-world SOA projects.* s.l.:s.n.

Katina, P., Keating, C. & Jaradat, R., 2014. System requirements engineering in complex situations. *Requirements Engineering,* 19(1), pp. 45-62.

Khadka, R. y otros, 2011. *A method engineering based legacy to SOA migration method.* Williamsburg, VA, USA, s.n.

Khadka, R., Saeidi, A. & Idu, A., 2012. *Legacy to SOA Evolution: Evaluation Results,* Utrecht, The Netherlands: Technical Report UU-CS-2012-006.

Kocbek, A. & Juric, B., 2010. *Using advanced Business Intelligence methods in business process management.* Ljubljana, Slovenia, s.n.

Koschmider, A., De la Vara, J. L. & Sanchez, J., 2010. *Measuring the Progress of Reference Model-Based Business Process Modeling.* Leipzig, s.n.

Labrador-Tovar, M. & Sanchez, R., 2013. *Definición de la arquitectura de una solución BPMS para la implementación de un proceso de solicitud de pedidos en una compañía de empaques plásticos.* s.l.:s.n.

Langer, A., 2012. *Guide to software development.* s.l.:Springer.

Langer, A., 2012. Legacy Systems and Integration. En: *Guide to Software Development.* s.l.:Springer, pp. 179-212.

Lehman, M. M., 1980. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE,* 68(9), pp. 1060 - 1076.

Li, B. & Li, M., 2009. Research and Design on the Refinery ERP and EERP Based on SOA and the Component Oriented Technology. *Networking and Digital Society, 2009. ICNDS '09. International Conference on,* Volumen 1, pp. 85 --88.

Li, B. & Zhou, W., 2008. Research and Design of EERP: End-to-End Resource Planning Based on SOA and BPM. *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4Th International Conference on,* pp. 1 --3.

Lins, F., Medeiros, R., Silva, B. & Souza, A., 2011. SSC4Cloud Tooling: An Integrated Environment for the Development of Business Processes with Security Requirements in the Cloud. *Services (SERVICES), 2011 IEEE World Congress on,* pp. 53 --60.

Lockerbie, J. y otros, 2012. Exploring the impact of software requirements on system-wide goals: A method using satisfaction arguments and i* goal modelling. *Requirements Engineering,* 17(3), pp. 227 - 254.

Loucopoulos, P., 1995. *System Requirements Engineering.* s.l.:McGraw-Hill .

Luna, E. R., Rossi, G. & Garrigos, I., 2011. WebSpec: A visual language for specifying interaction and navigation requirements in web applications. *Requirements Engineering,* 16(4), pp. 297 - 321.

Magliano, V., Bazán, P. & Martínez, J., 2013. *Análisis metodológico para la utilización de Process Mining como tecnología de optimización y respaldo de la implementación de procesos de negocio bajo el marco de BPM.* Paraná, s.n.

Maier, M., Emery, D. & Hilliard, R., 2001. Software architecture: introducing IEEE Standard 1471. *Computer,* 34(4), pp. 107-109.

Marjanovic, O., 2010. *Business value creation through business processes management and operational business intelligence integration.* Hawaii, s.n.

Márquez, M., 2010. *Introducción a la Notación BPMN.* s.l.:Universidad Central de Venezuela.

Martinez, J. & Bazán, P., 2011. *Lineamientos para la integración de BI y BPM: Retroalimentación y mejora continua de procesos basada en decisiones..* Rosario, s.n.

Martinez, J. & Bazán, P., 2013. *Conceptos de dinamismo aplicados a servicios y workflows en BPMS basados en Cloud Computing.* Paraná, s.n.

Martinez, J., Bazán, P. & Lorenzón, E., 2010. *Comparación del entorno IBM Websphere BPM y sus equivalentes funcionales en código fuente abierto.* El Calafate, s.n.

Massari, A. & Mecella, M., 2002. Il trattamento dei legacy system. En: *del libro Sistemi informativi.* s.l.:s.n., p. Capitolo 2 .

Matos, C. & Heckel, R., 2009. Migrating Legacy Systems to Service-Oriented Architectures. *Scientific and open access journal,* p. 16.

McGee, S. & Greer, D., 2012. Towards an understanding of the causes and effects of software requirements change: Two case studies. *Requirements Engineering,* 17(2), pp. 133 - 155.

Melao, N. & Pidd, M., 2000. A conceptual framework for understanding business processes and business process modelling. *Information Systems Journal,* 10(2), pp. 105--129.

Méndez, M., Overbey, J. & Tinetti, F. G., 2012. *Legacy fortran software: Applying syntactic metrics to global climate models.* Bahía Blanca, s.n.

Méndez, M. & Tinetti, F. G., 2011. *First steps towards a tool for legacy systems.* La Plata, s.n.

Minor, M., Bergmann, R. & Gorg, S., 2011. *Adaptive workflow management in the cloud-Towards a novel platform as a service.* Greenwich, s.n.

Mircea, M., 2010. SOA, BPM and cloud computing: Connected for innovation in higher education. *Education and Management Technology (ICEMT), 2010 International Conference on,* pp. 456 - 460.

Mortensen, M., Ghosh, S. & Bieman, J. M., 2012. Aspect-oriented refactoring of legacy applications: An evaluation. *IEEE Transactions on Software Engineering,* 30(1), pp. 118-140.

Nahuel, L. y otros, 2012. *Integración de modelos BPMN en ambientes MDA.* Bahía Blanca, s.n.

Noguera, M., 2011. *Sistemas colaborativos y procesos de negocio.* Granada: s.n.

Oliveros, A., 2012. *Ingeniería de Requerimientos [diapositivas de PowerPoint].* La Plata, s.n.

OMG, 2011. [En línea]
Available at: http://www.omg.org/spec/BPMN/2.0/PDF

OMG, 2014. [En línea]
Available at: http://www.omg.org/gettingstarted/gettingstartedindex.htm

Ortiz, M. & Plaza, A., 2013. *Programación orientada a objetos con Java y UML.* Cuenca: Ediciones Universitarias Universidad Politécnica Salesiana.

Oxford-English, 2012. *Oxford English Dictionary, Online-source.* s.l.:s.n.

Park, K.-S., Young-Pil, P. & Park, N.-C., 2011. Prospect of Recording Technologies for Higher Storage Performance. *Magnetics, IEEE Transactions on,* 47(3), pp. 539 - 545.

Perez-Castillo, R., de Guzman, I.-R., Piattini, M. & Ebert, C., 2011. Reengineering Technologies. *Software, IEEE,* 28(6), pp. 13-17.

Petter, S. & Ward, K., 2013. Countdown to Y2Gray. *IT Professional,* 15(6), pp. 49-55.

Poelmans, S., Reijers, H. & Recker, J., 2013. Investigating the success of operational business process management systems. *Information Technology and Management,* pp. 1-20.

Politécnica Salesiana, U., 2014. *Universidad Politécnica Salesiana.* [En línea]
Available at: http://www.ups.edu.ec/razon-de-ser
[Último acceso: 2014 10 13].

Pollice, G., 2003. *Using the Rational Unified Process for Small Projects: Expanding Upon eXtreme Programming.* [En línea]
[Último acceso: 15 08 2013].

Pourshahid, A. y otros, 2008. Toward an integrated user requirements notation framework and tool for business process management. *E-Technologies, 2008 International MCETECH Conference on,* pp. 3 -- 15.

Pourshahid, A., Amyot, D. & Peyton, L., 2009. Business process management with the user requirements notation. *Electronic Commerce Research,* 9(4), pp. 269 - 316.

Rajabi, B. A. & Lee, S. P., 2009. Change Management in Business Process Modeling Survey. *Information Management and Engineering, 2009. ICIME '09. International Conference on,* pp. 37 -- 41.

Ralph, P., 2013. The illusion of requirements in software development. *Requirements Engineering,* 18(3), pp. 293 - 296.

Real-Academia, E., 2011. *Diccionario de la Lengua Española,[en línea]. RAE.* s.l.:s.n.

Reyes, R., 2005. Metodología de la Investigación. En: *Fusiones y Adquisiciones de la Industria Automotriz Mundial.* s.l.:s.n.

Salvatierra, G., Mateos, C., Crasso, M. & Zunino, A., 2012. *Towards a computer assisted approach for migrating legacy systems to SOA.* Salvador de Bahia, Brazil, s.n.

Santorum, M., 2011. A serious game based method for business process management. *Research Challenges in Information Science (RCIS), 2011 Fifth International Conference on,* pp. 1 --12.

SCRUMstudy™, 2013. *Cuerpo del Conocimiento de SCRUM.* 2013 ed. Phoenix: VMEdu.

Sneed, H. M. & Erdoes, K., 2013. *Migrating AS400-COBOL to Java: A Report from the Field.* s.l., s.n.

Sneed, H. M., Schedl, S. & Sneed, S. H., 2012. Linking legacy services to the business process model. *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the,* pp. 17 - 26.

Sommerville, I., 2011. *Software Engineering.* 9 ed. s.l.:Pearson.

Thomas, O. & Fellmann, M., 2009. Semantic process modeling--design and implementation of an ontology-based representation of business processes. *Business & Information Systems Engineering,* 1(6), pp. 438--451.

Torres, V., Giner, P. & Pelechano, V., 2012. Developing BP-driven web applications through the use of MDE techniques. *Software \& Systems Modeling,* 11(4), pp. 609 - 631.

van Lamsweerde, A., 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications.* Viena: Wiley.

Visconti, M., 2011. *Métricas tradicionales de software.* s.l.:Universidad Ténica Federico Santa María.

Walk, S., 2010. Projecting Technology Change to Improve Legacy System Support Strategies. *Naval Engineers Journal,* 122(3), pp. 113-120.

Weske, M., 2012. *Business Process Management Architectures.* Berlín: Springer.

White, S., 2004. *Introduction to BPMN.* s.l.:IBM Cooperation.

Wiegers, K., 2003. *Software Requirements.* 2 ed. s.l.:Microsoft Press.

Xu, Y., 2010. *Bussines Rules based Legacy System Evolution towards Service-Oriented Architecture.* Lencester: Software Technology Research Laboratory / Faculty of Computing Sciences and Engineering / De Montfort University.

Yaxiong, T., Guoquan, Z., Zhen, X. & Boqing, L., 2008. A research on BPM system based on process knowledge. *Cybernetics and Intelligent Systems, 2008 IEEE Conference on,* pp. 69 --75.

Yousef, R., Odeh, M., Coward, D. & Sharieh, A., 2009. Translating RAD business process models into BPMN. *Applications of Digital Information and Web Technologies, 2009. ICADIWT'09. Second International Conference on the,* pp. 75 - 83.

Yue, T., Briand, L. & Labiche, Y., 2011. A systematic review of transformation approaches between user requirements and analysis models. *Requirements Engineering,* 16(2), pp. 75-99.

Zhang, L., Ge, M., Bi, X. & Chen, S., 2010. A SOA-BPM-based architecture for intelligent power dispatching system. *Power and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific,* pp. 1 --4.

Zhang, W., Berre, A., Roman, D. & Huru, H., 2009. *Migrating legacy applications to the service Cloud.* Orlando, s.n.

Zhang, Z., Zhou, D.-D., Yang, H.-J. & Zhong, S.-C., 2010. A service composition approach based on sequence mining for migrating e-learning legacy system to SOA. *International Journal of Automation and Computing,* 7(4), pp. 584-595.

Zhou, Y. C. y otros, 2010. Business Process Centric Platform-as-a-Service Model and Technologies for Cloud Enabled Industry Solutions. *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on,* pp. 534 - 537.

Zorzán, F. & Riesco, D. E., 2007. *Transformación de Actividades SPEM por medio de su transformación en relations a subprocesos BPMN.* Corrientes y Resistencia, s.n.

Zorzán, F. & Riesco, D. E., 2010. *Transformación de procesos BPMN a su implementación en BPEL utilizando QVT..* El Calafate, s.n.