Universidad de Cuenca

Facultad de Ingeniería

Maestría en Gestión Estratégica de Tecnologías de la Información

Proyecto de Tesis

ANÁLISIS Y DISEÑO DE SOFTWARE EN LA APLICACIÓN DE TDABC PARA EL SECTOR PRODUCTIVO MEDIANTE EL USO DE METODOLOGÍAS ÁGILES

Autor:

Ing. Eduardo Patricio Merchán Sempértegui

C.I. 0102836103

Director:

Ing. Carlos Villie Morocho Zurita, PhD.

C.I. 0300930328

Co-Directora:

Ing. Lorena Catalina Sigüenza Guzmán, PhD.

C.I. 0102659687

2018

Resumen

Un correcto manejo de los costos y la gestión estratégica de procesos son herramientas indispensables hoy en día en el normal desenvolvimiento de las industrias debido a su impacto directo en los objetivos de toda empresa, estando entre los primordiales, el minimizar costos y maximizar utilidades. Por un lado, el manejo de los costos nos ayudara a controlar que los insumos estén siendo utilizados de una manera eficiente, lo mismo sucede con la gestión estratégica de los procesos brindando las directrices necesarias para mejorar la gestión empresarial. En este sentido, un sistema de costos nos facilitará información adecuada, precisa y detallada para una mejor toma de decisiones.

El presente trabajo consiste en documentar el análisis y diseño de una plataforma informática para el apoyo a la gestión de costos y procesos en una industria del sector productivo, específicamente el de ensamblaje mediante la aplicación del sistema de costes basados en el tiempo invertido por actividad (TDABC), mediante el uso de metodologías ágiles se busca verificar si el desarrollo de la plataforma permite superar algunas de las limitaciones del sistema de costeo, así como acentuar sus ventajas. Para esto, se establece primero un marco teórico que se centra en describir los principales sistemas de costeo, así como las metodologías de desarrollo ágiles existentes. Seguidamente, el estudio hace referencia al estado actual de la empresa, documenta los componentes de análisis, diseño y desarrollo de una plataforma informática, haciendo una alineación con respecto a metodologías ágiles. Para finalizar el sistema planteado supone una evolución de la experiencia previa de desarrollo de un prototipo que aplica TDABC para el manejo de procesos en bibliotecas al alinearlo a empresas de producción. El trabajo finaliza extrayendo las conclusiones más importantes del análisis efectuado

Palabras Claves: gestión, procesos, costos, TDABC, BPMN

Abstract

Cost management and strategic process management are indispensable tools today in the development of industries given their direct impact on the objectives of every company, among the most important, minimizing costs and maximizing profits. On one hand, cost management helps to ensure that inputs are being used in an efficient way, along with the strategic management of the processes, which provides the necessary guidelines to improve business management. A cost system is critical to provide us with adequate, accurate and detailed information for better decision making in businesses.

This article presents the analysis and design of a software that serves to manage cost management and assembly processes using the ¨Time Driven Activity Based Costing¨ (TDABC) system. This study supports the academic field by using agile methodologies to verify if the development of the platform allows its clients to overcome limitations of TDABC, while simultaneously accentuating its advantages. First, a theoretical framework is established that focuses on presenting the main costing systems and existing agile development methodologies. Next, the study reviews the current state of business in order to assess the the components of analysis and design that are essential for the development of a software that aligns with agile methodologies. Lastly, this article explores the evolution of past experiences with a TDABC prototype that was designed for libraries in order to assess its appropriateness for production companies. This article concludes by arguing the groundbreaking role of the software which promotes the advantages of managing TDABC, while also preventing and limiting the downfalls of the methodology

Keywords: management, processes, costs, TDABC, BPMN

Tabla de Contenido

Índice de Figuras

Índice de tablas

Cláusula de Licencia y Autorización para Publicación en el Repositorio Instituciona

Universidad de Cuenca

Cláusula de Licencia y Autorización para Publicación en el Repositorio Institucional

Eduardo Patricio Merchán Sempértegui en calidad de autor/a y titular de los derechos morales y patrimoniales del trabajo de titulación "Análisis y Diseño de Software en la Aplicación de TDABC para el Sector Productivo mediante el uso de Metodologías Ágiles", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el Repositorio Institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, Enero del 2018

Eduardo Patricio Merchán Sempértegui

C.I: 0102836103

9

Ing. Eduardo Patricio Merchán Sempértegui

Cláusula de Propiedad Intelectual

Universidad de Cuenca

Cláusula de Propiedad Intelectual

Eduardo Patricio Merchán Sempértegui, autor del Trabajo de Titulación "Análisis y Diseño de Software en la Aplicación de TDABC para el Sector Productivo mediante el uso de Metodologías Ágiles", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, Enero 2018

Eduardo Patricio Merchán Sempértegui

C.I: 0102836103

10

Ing. Eduardo Patricio Merchán Sempértegui

Agradecimientos

Agradezco a mi Director, Dr. Villie Morocho, y mi Co-Directora, Dra. Lorena Sigüenza, por la paciencia, tiempo brindado en la dedicación y apoyo a este proyecto…

… Y con toda mi alma a la razón de mi vida Eduardo, Isabella y Rafaella, mis hijos. …

Capítulo 1: Generalidades

Introducción

El alto nivel de competitividad que se da en el entorno empresarial ha motivado a que estas, en su afán por conseguir los objetivos de minimizar costos y maximizar ganancias, enfoquen su atención en la búsqueda de nuevas alternativas para la mejora de sus servicios y procesos. Implicando, entre otras cosas, una buena implementación de sistemas de contabilidad de costos, y de igual forma su traducción a sistemas informáticos (Cámara de Industrias Producción y Empleo. 2016).

En relación a la contabilidad se ha establecido como un medio imprescindible para las empresas en el conocimiento de su situación económica y financiera en determinado punto del tiempo. La contabilidad de costos, según Mesías Jaime León (Mesías, J. L. 2006), ha evolucionado en tres generaciones: determinación de costos unitarios, predeterminación de costos, y control de costos; surgiendo en cada una de ellas distintos sistemas de costeo. En la tercera generación, específicamente, nace el sistema de costos basados en actividades (ABC), ABC es un sistema de costos promovido por Robert S. Kaplan y Robin Cooper en 1998 (Robert S. Kaplan, & Robin Cooper. 1998). Posterior a este revolucionario sistema de costeo, surge un nuevo enfoque basado en el tiempo invertido por actividad denominado TDABC, TDABC es un sistema de costeo desarrollado por Kaplan y Anderson en 2003 para superar las limitaciones de sus antecesores (Kaplan, R. S., & Anderson, S. R. 2003) como: son el tratar de identificar los diferentes departamentos, los costos involucrados en cada uno de ellos y su capacidad normal.

La traducción de requerimientos a sistemas informáticos implica el análisis, diseño, arquitectura de software y sus patrones que son una parte fundamental para una correcta organización de las secciones que involucran el desarrollo y puesta en marcha de estos sistemas. Todos los conceptos implicados en los sistemas informáticos, estructuras, el código y sus módulos, almacenamiento de datos, roles, responsabilidades y relaciones entre cada uno de estos puntos, surgen gracias a la necesidad de una base modular para el manejo de sistemas en crecimiento tanto en tamaño como en complejidad como nos menciona (Arias Chaves, M. 2005).

Desafortunadamente, pocos son los estudios que documentan el desarrollo e implementación de sistemas informáticos para TDABC. Dos de ellos es el presentado por Cabrera y Ordoñez (Cabrera Encalada, P., & Ordoñez Parra, C. 2012) y Siguenza-Guzman (Siguenza-Guzman et al., 2014) que describen el desarrollo de un módulo denominado TD-ABC-D, orientado a la gestión de costos en bibliotecas universitarias.

De igual forma, el escoger una correcta metodología para un futuro desarrollo de software es complementario e importante luego de obtener el resultado de los análisis de patrones de arquitectura correctos para las características del software que se propondría para el desarrollo (Parra Castrillón, E. 2011). Hace varios años existía la opinión general entre los programadores de que la mejor forma de desarrollar software era planificando cuidadosamente el proyecto y utilizando métodos de análisis y diseño cerrados y de igual

forma rigurosos. No obstante, estos pensamientos provenían de los ingenieros implicados en el desarrollo de sistemas informáticos demasiado grandes y complejos. Todos estos enfoques tienden a volverse pesados y dejan de ser efectivos en sistemas medianos y pequeños, ya que el esfuerzo invertido es demasiado grande para las ventajas que realmente se tendrían. Como resultado de esto se tenían proyectos con demasiadas horas en planificación y muy pocas en programación y pruebas; de tal manera que fue necesario generar un sistema de análisis, diseño y desarrollo evolutivo, iterativo e incremental sentando, de esta forma, las bases para las metodologías ágiles.

Es así que tanto análisis, patrones de arquitectura de software como metodologías de desarrollo ágiles (García Rodríguez, M. J. 2015) se enfocaron en un inicio en sistemas medianos y pequeños obteniendo sistemas más robustos, flexibles, adaptables, probados y enfocados a los requerimientos funcionales. Por esta razón se realizaron algunas modificaciones básicas en los conceptos para poder dividir los sistemas grandes en sistemas funcionales o entregables más pequeños.

Justificación del proyecto

Luego de realizada una investigación exploratoria, no se ha logrado encontrar ninguna empresa en el sector productivo y específicamente en el de ensamblaje que maneje una plataforma informática capaz de gestionar los costos mediante el uso del enfoque basado en el tiempo invertido por actividad denominado TDABC. Las empresas en el sector productivo, en su mayoría, se han limitado a gestionar los costos de forma manual o registros en hojas de cálculo sin lograr una administración y control adecuado de los mismos. Métodos de costeo tradicionales por la facilidad y rapidez en la implementación.

Objetivo general

Estudiar la aplicación de TDABC en el sector productivo para generar un modelo de software usando metodologías ágiles

Objetivos específicos

- Realizar una revisión sistemática de literatura y estado del arte de las aplicaciones para TDABC o relacionados.
- Determinar el modelo conceptual para la arquitectura a proponer.
- Desarrollar un estudio comparativo de las herramientas que mejor se adapten al modelo.
- Generar una propuesta de arquitectura de software para la aplicación de TDABC en el sector productivo.

Alcance del proyecto

Este trabajo forma parte del proyecto de investigación **"Modelo de Gestión para la Optimización de Procesos y Costos en la Industria de Ensamblaje" ganador de la XV Convocatoria de Concursos Universitarios de Investigación de la DIUC**. El proyecto tiene como objetivo *desarrollar un modelo de gestión para la optimización de procesos y costos en las industrias de ensamblaje*. Debido a su naturaleza multidisciplinaria, el proyecto

abarca aspectos académicos-científicos de la administración, costeo, finanzas y aspectos de ingeniería relacionadas con Tecnologías de Información; gracias a esta amplitud existe la posibilidad de vinculación con la sociedad y el sector productivo. Para lograr este fin, el proyecto se propone entre sus objetivos específicos implementar una plataforma informática de soporte del modelo de gestión. Es así que, el objetivo de esta propuesta de tesis es el análisis y diseño para un futuro despliegue e implementación del software que posibilite una adecuada puesta en marcha de un sistema informático para el manejo de procesos y costos en industrias del sector productivo y de manera específica el de ensamblaje. Para lograr este objetivo, primero, se efectuará un análisis sistémico de los patrones de arquitectura de software tomando en cuenta diferentes características y variables necesarias para el desarrollo del software.

Posteriormente se hará un estudio comparativo de las herramientas que mejor se adapten al modelo. Para ello, se tomará en cuenta algunos parámetros funcionales como escalabilidad, interoperabilidad, flexibilidad y eficiencia. Entre las herramientas disponibles para este análisis, partiendo de los patrones de arquitectura existentes, se encuentran: Software basado en modelos, Programación por capas, Dirigida por eventos, Orientada a servicios y Basada en el espacio. Sistemas de almacenamiento de datos ya sean relacionales o no relacionales, por ejemplo: PostgreSQL, MySQL y MongoDB, lenguajes de programación: PHP, Ruby, Javascript y Java. Frameworks de desarrollo enfocados a cada uno de los lenguajes analizados: CakePHP, Ruby on Rails, Meteor y Angular. En el estudio comparativo de las herramientas se pretende extraer los puntos importantes de cada una de ellas y así fortalecer los puntos más problemáticos a nivel de desarrollo de software, instalación y pruebas.

En el presente trabajo de tesis, además, se incluye el análisis del prototipo generado como parte de la tesis de pregrado: Desarrollo de un Módulo TDABC, aplicado al Centro de Documentación Regional "Juan Bautista Vázquez" (Cabrera, Ordoñez, 2012). Así también, se analizará el modelo de software desarrollado como parte de la tesis de doctorado: *Optimal Resource Allocation and Budgeting in Libraries* (Siguenza-Guzman et al.,2015). Este análisis permitirá utilizar como base, el trabajo realizado en dichas tesis para formar parte del prototipo planteado como entregable en el presente trabajo de titulación.

Una vez realizado todo este análisis, se determinará una metodología de desarrollo ágil que sirva para la gestión y desarrollo de proyectos mediante el uso de procesos iterativos y graduales teniendo como principales objetivos, por un lado el aumento de la productividad en el desarrollo del software y por otro lado conocer si el desarrollo de este software elimina algunas de las limitaciones y acentúa las ventajas del TDABC, considerando para ello todos los requerimientos solicitados por la empresa. Algunas de las metodologías más utilizadas son Scrum, Kanban o XP; existiendo la posibilidad de combinación entre ellas para trabajar con una metodología híbrida enfocada a las necesidades específicas de la empresa. En este caso específico, el trabajo de esta tesis se desarrollará tomando como caso de estudio a la empresa Motsur. Esta empresa ha sido seleccionada por el proyecto de investigación DIUC para validar los resultados tanto en el modelado de la plataforma como en el uso de la misma.

Producto de esta investigación se dispondrá de la metodología y herramientas de desarrollo, arquitectura de software, análisis de las ventajas, y limitaciones de la metodología TDABC para la construcción de un un software de soporte del modelo de gestión de procesos, costos que automatice y mejore el análisis TDABC en industrias de producción. Este modelo deberá integrar las características esenciales del costeo TDABC con las particularidades del sector productivo. Mediante el uso del sistema se pretende determinar si sobrepasan las limitaciones vinculadas al TDABC, y se potencia los beneficios que este posee y aprovecha las oportunidades de análisis de costos en este tipo de empresas.

De igual manera la utilización de herramientas para optimización de los datos utilizando modelos matemáticos permitirá mostrar múltiples resultados posibles. La idea es visualizar un gran número de escenarios futuros posibles, e indicar las probabilidades y riesgos asociados con cada uno. Mediante este análisis se pretende facilitar la toma de decisiones a un nivel gerencial.

Método de trabajo

Esta tesis contempla su realización metodológica en 4 etapas generales, las conceptualizaciones de estas etapas se realizaron en base al método ágil SCRUM, con el cual se gestionará el proyecto de desarrollo de software. Cada una de las etapas contiene capítulos desarrollados del 1 al 8; el capítulo 1 menciona de manera general todos los aspectos necesarios para contextualizar este proyecto, el alcance y la metodología de trabajo. El capítulo 2 contiene los aspectos teóricos necesarios dentro de la investigación desde el punto de vista administrativo, económico y tecnológico. El capítulo 3 nos brinda un resumen de cómo se encuentran las empresas de producción actualmente desde el punto de vista de la utilización de los métodos de costeo. El capítulo 4 se enmarca en el aspecto tecnológico, específicamente, en el desarrollo de software, su evolución y características actuales para el manejo en proyectos. Los capítulos subsiguientes y de manera específica 5 y 6 corresponden al análisis de herramientas. Desde el aspecto tecnológico y puramente de desarrollo, metodologías para desarrollo e implementación y finalizando con la alineación del software con métodos de costeo específicamente TDABC en empresas de producción. El capítulo 7 pretende vincular todo el conocimiento adquirido durante los capítulos anteriores y presentar la forma en la que se podría sobrepasar las limitaciones, potenciar los beneficios del uso de TDABC y aprovechar las ventajas de todas las herramientas utilizadas para el cumplimiento de los objetivos planteados. El capítulo 8 realiza comparativos entre las limitaciones y las bondades del desarrollo de software usando las metodologías mencionadas en capítulos anteriores, permite conocer si estas bondades sobrepasan algunas de las limitaciones planteadas.

Por otro lado, a continuación, se mencionará la correspondencia de los capítulos descritos arriba con cada una de las etapas de la metodología Scrum.

Etapa 1: Visualización

En esta primera etapa, de todos los elementos mencionados, se hará hincapié en el estado actual de las empresas de producción con relación a los métodos de costeo, dentro de esta etapa corresponde el capítulo 3.

Etapa 2: Planeación

Esta etapa comprende la planificación del trabajo que consiste en proponer literatura para la arquitectura de la solución y listar los componentes del desarrollo de software partiendo del análisis hasta la gestión de riesgos es decir la revisión de la literatura de desarrollo y el modelo conceptual a proponer, en esta etapa se encuentra el capítulo 4.

Etapa 3: Análisis

Dentro de esta etapa se realiza el análisis comparativo de herramientas de desarrollo y metodologías ágiles enfocadas a la aplicación de TDABC en la empresa, y una alineación entre estos dos aspectos brindando una visión global de la funcionalidad y ventajas de este desarrollo. A esta etapa corresponden los capítulos 5 y 6.

Etapa 4: Propuesta

En esta etapa se pretende generar el análisis y diseño de software enfocado a la aplicación de TDABC en las empresas del sector productivo y de manera específica el de ensamblaje. La idea principal en esta etapa es proporcionar la guía completa tanto a desarrolladores como administrativos de cómo aplicar los conceptos y conocimiento generado para la aplicación empresas de este tipo, dentro de esta última etapa se encuentran los capítulos 7.

Capítulo 2: Marco Teórico

Introducción

El marco teórico que fundamenta esta investigación proporcionará una idea más clara acerca de los temas involucrados dentro de este proyecto. Se encontrarán los conceptos muy preliminares, los complementarios y específicos organizados de la siguiente forma: sistemas de costeo en el que se mencionara un comparativo entre dos de ellos; modelos de procesos y una introducción al manejo de TDABC con el uso de BPMN; metodologías agiles en donde se pretende mencionar aspectos generales de esta metodología; metodologías de desarrollo detallando las características de cada una de ellas. Todos estos puntos brindaran una base fuerte para el desarrollo de los siguientes capítulos.

Las empresas en el sector productivo

Las empresas dentro del sector de la producción requieren la correcta utilización de un método de costeo en el análisis de los costos incurridos en cada uno de los pasos del proceso productivo, de esta manera se podrá determinar de mejor manera el precio de los productos, controlar las operaciones y estados financieros como afirma (Salinas 2011).

Existen algunos puntos a tomar en cuenta en empresas que forman parte del sector productivo como son la planeación y programación de la producción (Gómez Niño, O. 2011). La planeación de la producción es un elemento importante en cualquier empresa, puesto que, ella se apropia de la utilización de los recursos existentes. Dentro de la planificación se tiene en cuenta varios aspectos, tales como: los materiales, la mano de obra, la maquinaria y equipos, y el método de producción. En cuanto a los materiales hace referencia al abastecimiento, control y existencia de la materia prima y elementos que deben estar disponibles con sus especificaciones de calidad y cantidad para asegurar que todas las operaciones productivas comiencen a su debido tiempo. La mano de obra involucra funciones, cantidad y perfil que se va a utilizar en el proceso de producción. En lo que se refiere a la maquinaria y equipos nos menciona políticas de reposición, mantenimiento y procedimientos de ejecución para evitar paradas del sistema; por último se hace referencia al método de producción proporcionando, posibles formas de optimización teniendo en cuenta los recursos disponibles y la capacidad de producción.

La programación de la producción, por su parte, es una tarea difícil por ser la que determina la utilización de mano de obra, maquinaria, materiales y ruta de producción. El rendimiento de la planta depende directamente de la programación de la producción, este rendimiento cumplirá las solicitudes del departamento de ventas (Meléndez, 2004).

La capacidad de producción en una empresa responde al número de productos que puede fabricar en un periodo determinado, teniendo en cuenta los recursos disponibles, representados en: recursos económicos, físicos, tecnológicos, humanos, entre otros. Todos ellos deben estar equilibradamente distribuidos, pues de qué le sirve a la empresa tener una excelente planta física si no cuenta con los recursos económicos o financieros para producir o viceversa. Es conveniente que la administración analice los factores de producción con los que cuenta y las necesidades de producción de acuerdo al pronóstico de ventas para ajustar

la programación de la producción a necesidades concretas y evitar un mal uso de los mismos (Gómez Niño, O. 2011).

Sistemas de costeo

Los métodos de costeo podrían definirse como el conjunto de reglas, procesos y procedimientos, que hacen posible el cálculo sistemático de datos relacionados con el consumo de recursos necesarios para producir un bien. El objetivo de estos métodos es el de suministrar información relevante encaminada a facilitar la toma de decisiones por parte de la dirección de la empresa, así como proceder a la valoración de la producción antes de ingresar al inventario. Los sistemas de costos satisfacen dos propósitos, el primero, está relacionado con la planificación y el control el cual se materializa en uno de los objetivos de costos como son los centros o unidades organizativas. En esta planificación y control se acumulan los costos por centros de responsabilidad, lo que permite valorar la contribución que cada uno de ellos realiza con la consecución de los objetivos globales de la empresa. El segundo propósito, está relacionado con el cálculo del costo de los productos y se materializa en el objetivo de las unidades de producto, que permite valorar existencias y calcular resultados (Tafur, J. C., & Osorio, J. A. 2016).

Un sistema de costeo se compone de tres partes fundamentales como nos menciona (Tafur, J. C., & Osorio, J. A. 2016): los métodos de acumulación de costos, los métodos de asignación de costos o filosofías de costeo, y las bases de costeo que como lo define este mismo autor hace referencia a la fuente u origen de los costos que sirven de base para la valoración de los productos o servicios. Los métodos de acumulación de costos hablan de la forma en que se recopila la información de costos y que posteriormente va a dar origen a la forma como se calcula el costo. Los **métodos de asignación de costos o filosofías de costeo** asignan costos a los productos; es decir, son los conceptos o elementos que se incluyen dentro del costo del producto y, por ende, del valor de los inventarios, y los elementos que se deben llevar directamente al periodo sin pasar por los inventarios de gastos.

Los costos son un factor que contribuye a la competitividad. Existen sistemas de costeo tradicionales y modernos. Los sistemas tradicionales de costos consideran que su única misión es la de determinar correctamente el costo del producto o servicio, ignorando que actualmente, lo que demandan los usuarios de costos es información para ver que se puede hacer para reducirlos (Ramírez, 2008). Es decir, se requiere un sistema de información que determine qué actividades agregan valor y cuáles no, con el fin de lograr el mejoramiento continuo. Esta herramienta tiene, entre otras, la ventaja de reducir al mínimo el prorrateo de los gastos indirectos de fabricación, así como realizar una identificación de los gastos de administración y venta entre los diferentes clientes, zonas, productos, etc., lo cual permite una correcta toma de decisiones.

Sistemas de Costeo Tradicional

Los sistemas de costeo tradicionales distribuyen los costos fijos de la empresa a través de centros de costos y de éstos a los productos o servicios. Los sistemas tradicionales no entregan el verdadero costo del producto terminado, sino una estimación errónea que se

transmite al cálculo final de la rentabilidad de los distintos productos y, por lo tanto, de la empresa (Contreras, H., & McCawley, A. 2006).

Los sistemas tradicionales de costeo al imputar los costos fijos en base a volumen, arrojan resultados inválidos para los distintos productos que la empresa produce (Contreras, H., & McCawley, A. 2006).

Los sistemas de costeo tradicionales distribuyen los costos fijos de la empresa a través de centros de costos y de estos a los productos. Esto implica que se transmitan a estos de acuerdo al volumen producido, generando un costo medio, y no por las necesidades del proceso de acuerdo al nivel de actividades, generando un costo variable (Contreras, H., & McCawley, A. 2006).

Costeo Basado en Actividades (ABC)

Es una filosofía desarrollada por Robin Cooper y Robert Kaplan en la década de los 80 según la cual, se incluye dentro del costo del producto, tanto los costos de producción como los gastos administrativos y de ventas incurridos. Este concepto parte de la premisa que todos ellos deben ser recuperados con la venta y que la estructura administrativa y comercial son necesarias para que el producto llegue al consumidor final (Tafur J et al., 2016)

Las etapas del ABC, que lista de Arbulo López, P. R., y Santos, J. F. (2011), y cuya fuente principal corresponde a (Everaert, P., et al., 2008) son las siguientes:

1. Identifica las actividades
2. Asigna los costes indirectos a las distintas actividades a través de los inductores de coste
3. Identifica los inductores de cada actividad
4. Calcula el coste de los inductores dividiendo el coste total de cada actividad entre su volumen de actividad normal
5. Multiplica el coste del inductor por los inductores consumidos para obtener el coste de los objetos de coste (productos, clientes, etc.)

En resumen, para el cálculo ABC, primero identifica las actividades que tiene lugar en la empresa. A continuación, localiza las actividades en las secciones o centros de coste de la empresa. Una actividad puede desarrollarse en más de un centro. Los costes directos se asignan directamente a los productos. El siguiente paso es localizar las cargas indirectas en cada sección y asignarlas a las actividades (Cooper, R., & Kaplan, R. S., 1988).

En algunos casos, se identifica fácilmente qué actividad ha generado dicho coste y su imputación es directa, pero en otros casos se genera una bolsa de coste indirecto en la sección que tiene que repartirse entre las actividades por medio de algún criterio. Para cada actividad hay que asignar un portador de costes o inductor (en inglés, *cost driver*). Un inductor es la unidad de una actividad que causa cambios en el coste de dicha actividad. El inductor realiza la función de unidad de medida de la actividad, por lo que debe estar directamente relacionado con los recursos consumidos. Tanto puede ser una entrada de la actividad, una

salida u otro indicador físico de la misma. El coste total de cada actividad se divide para encontrar

el coste unitario del inductor (motivo por el que esta técnica se denomina también rate-based ABC). A partir de ahí, el coste de cada producto u objeto de coste se obtiene en función del consumo de unidades de inductor (Cooper, R., & Kaplan, R. S., 1988), más los costes directos correspondientes, tal como se muestra en la Figura 1.

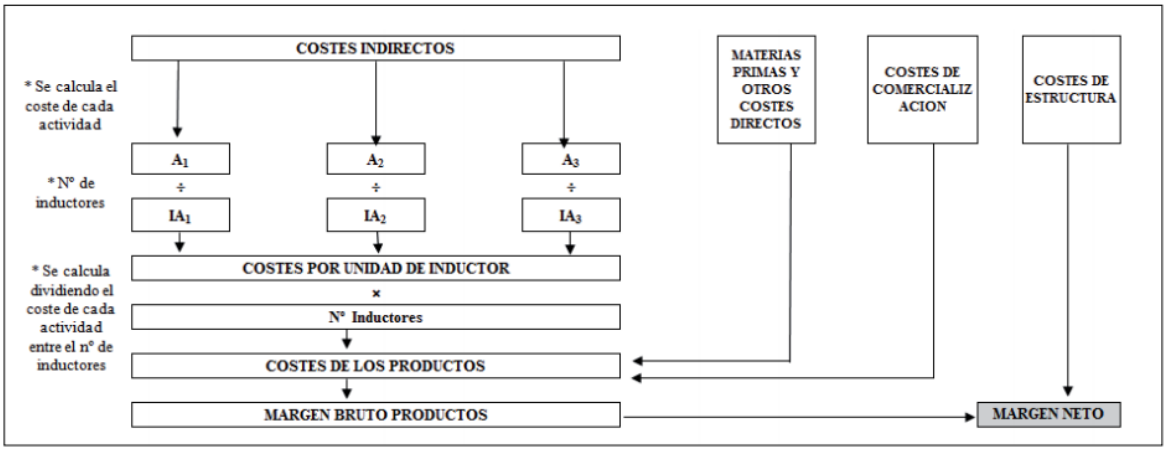

Figura 1 - Metodología de imputación del coste según el sistema ABC (de Arbulo López, P. R., & Santos, J. F. 2011)

Costes basados en el tiempo invertido por actividad (TDABC)

Los orígenes del hoy conocido como método de costes basados en el tiempo invertido por actividad o TDABC iniciaron en 1995, año en el que Steven R. Anderson, estudiante de Dirección y Administración de Empresas en la Harvard Business School, acudió a un curso impartido por Robert Kaplan. Dicho curso versaba sobre el cálculo de los costes basándose en un sistema ABC. Junto con otro compañero, Anderson ideó un software a través del cual se podía analizar la información de los costes y beneficios de las empresas. Al finalizar la carrera y después de trabajar como asesor en McKinnsey & Co., Anderson fundó Acorn Systems, Inc., una compañía de asesoramiento y software que ayudaba a pequeñas y medianas empresas a automatizar sus procesos y sistemas de implantación del ABC. A lo largo de los meses y después de trabajar con varias empresas, Anderson se dio cuenta de que el software implantado por su compañía no reproducía fielmente el análisis de los costes que había llevado a cabo en la universidad, por lo que tras varios planteamientos formuló el primer esbozo de costes basados en el tiempo invertido por actividad.

En el año 2001, Kaplan entró a formar parte del equipo directivo de Acorn Systems y colaboró activamente con Anderson para convertir su planteamiento del TDABC en el modelo riguroso que es actualmente. El enfoque TDABC identifica los diferentes grupos de recursos o departamentos, sus costes y su capacidad normal. Por ejemplo, para el departamento de recepción de material la capacidad normal se calcula multiplicando el número de empleados que trabajan en dicho departamento por su jornada laboral mensual,

restándole, a continuación, a dicho producto el tiempo no productivo o de descansos. A continuación, se divide el coste total de dicho departamento entre la capacidad normal y se obtiene el coste por unidad de tiempo (generalmente, coste por minuto). Finalmente, los costes son asignados a cada pedido de compra multiplicando el coste por unidad de tiempo por el tiempo necesario para completar las operaciones de recepción de los materiales (de Arbulo López, P. R., & Santos, J. F. 2011)

Las etapas del TDABC que lista (de Arbulo López, P. R., & Santos, J. F. 2011), y cuya fuente principal corresponde a (Everaert, P., et al., 2008) se listan a continuación:

1. Identifica las actividades que son realizadas con los mismos medios para constituir los "grupos de recursos"
2. Estima los recursos consumidos por cada «grupo de recursos»
3. Estima la capacidad normal de cada grupo de recursos en términos de horas de trabajo
4. Calcula los costes unitarios de los inductores (el más habitual es el minuto de trabajo) de cada grupo de recursos, dividiendo el coste de los recursos consumidos entre la capacidad normal
5. Para cada tarea, determina el tiempo necesario de acuerdo con sus características
6. Para valorar cada tarea, multiplica el coste unitario de los recursos por el tiempo necesario para llevarla a cabo.

En comparación con un sistema ABC convencional, Namazi (Namazi, M. 2016) describe al menos seis diferencias significativas: Primera, extrae el tiempo como el manejador de costo primario para el costo de los objetos. Segundo, TDABC, inicialmente, elimina el primer paso correspondiente a la metodología ABC tradicional en el proceso de implementación, el cual es la determinación de diferentes actividades como nos menciona (Siguenza-Guzman et al., 2014), Tercero, TDABC simplifica el proceso de cálculo de costos al eliminar la necesidad de entrevistar y encuestar a los empleados para asignar los costos de los recursos a las actividades antes de derivarlos a los objetos de costos. En cambio, permite a los gerentes "estimar" el tiempo requerido para realizar las actividades. Cuarto, TDABC determina inequívocamente la "capacidad utilizada", así como la "capacidad no utilizada", basando las tasas de costos generales predeterminadas en la "capacidad práctica", que se supone que es del 80 al 85% de la capacidad ideal. Quinto, TDABC puede acomodar sistemas complejos de producción reales e incorpora variaciones en la utilización de recursos mediante la formulación de diferentes modelos de ecuación de tiempo. Sexto, el sistema ABC tradicional es un modelo de gestión de costos "*push*". Es decir, los costes se asignan primero a "actividades y, a continuación, los costes de actividad se atribuyen a los "objetos de coste". Para una mejor comprensión se describe de manera gráfica las premisas teóricas graficas del TDABC en la Figura 2.

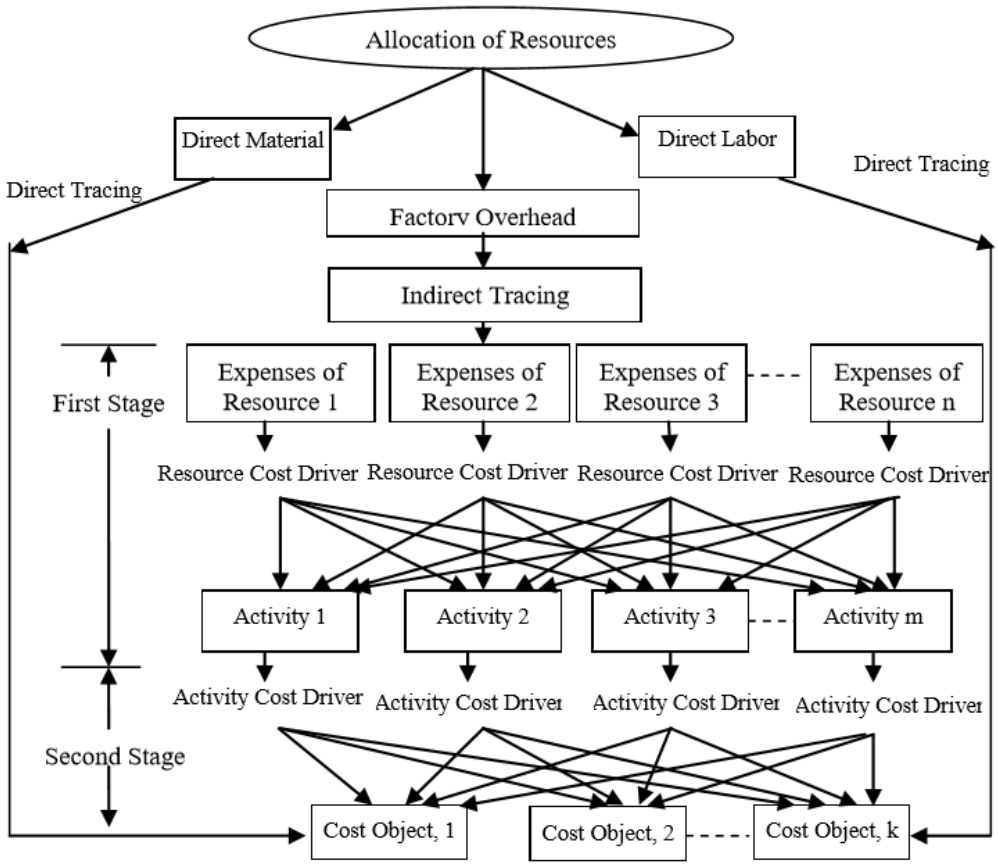

Figura 2 - Elementos principales de un sistema ABC (Namazi, M. 2016)

Ventajas de la metodología TDABC

Tanto Siguenza-Guzman (Siguenza Guzman, L., et al,. 2013), como Namazi (Namazi, M. 2016), identifican los siguientes beneficios del uso de TDABC en varios trabajos analizados. Simplicidad: TDABC simplifica la construcción de modelos de costeo, ya que solamente se requiere conocer el tiempo que toma desarrollar una actividad y el costo por unidad de tiempo de desarrollar la actividad. Reducción de complejidad en operaciones: Otro beneficio es el uso de las ecuaciones de tiempo lo que permite una reducción de complejidad en los cálculos requeridos para el diseño de modelos de costos. Desagregación de costos de procesos: Al crear modelos de costeo basados en actividades, TDABC permite desagregar los costos por transacción basados en procesos. Esto permite obtener un mayor detalle de costos y tiempos pudiendo comprender la influencia de todos los elementos sus elementos. Visualización de la capacidad de utilización: es otra ventaja mencionada por (Siguenza Guzman, L., et al,. 2013). Versatilidad del sistema: La versatilidad del sistema TDABC permite que pueda ser aplicado en cualquier industria con objetos de costos complejos. En los artículos presentados por (Siguenza Guzman, L., et al,. 2013), y (Namazi, M. 2016) se analiza su aplicación en distintos ámbitos empresariales, lo cual ha demostrado las ventajas que han surgido en su aplicación. Modularidad y versatilidad de modelos de costeo: Un modelo de costos basado

en TDABC puede ser actualizado más fácilmente que uno basado en ABC. Esto, ya que TDABC no requiere que se hagan entrevistas nuevamente al momento de alterar procesos. Capacidad de simulación de procesos: TDABC puede usarse para modificar el comportamiento de los diferentes procesos modelados y simular las diferentes situaciones que pueden darse al cambiar valores de costos o flujos de procesos.

Limitaciones de la metodología TDABC

Dentro de las limitaciones que nos presenta (Namazi, M. 2016) en su artículo podemos encontrar las enumeradas a continuación:

1. Aunque TDABC se puede aplicar a varias industrias, su aplicación se limita a situaciones en las que el "tiempo" puede ser ejercido como el único factor de costeo.
2. La falta de identificación de la actividad, en la primera etapa, desvía TDABC significativamente de los mayores y principales fundamentos del ABC. Si las "Actividades" no se identifican claramente al principio, y se calcula una sola tasa de costos holística para todo el departamento, equivale a volver a los sistemas tradicionales de contabilidad de costos basados en el volumen como lo cita (Cooper, R. 1989)
3. Aunque aparentemente TDABC aborda la simplicidad, la determinación exacta de los costos de capacidad práctica, la tasa de capacidad y la absorción de una tasa de costo de capacidad uniforme para todas las actividades del departamento han surgido como nuevos obstáculos, así como lo menciona el mismo autor en otro de sus artículos (Namazi, M. 2009)
4. Aunque parece que la construcción de un modelo TDABC es más fácil que la creación de un modelo CABC, que no siempre es el caso porque "ABC cronometrado requiere tanto la recopilación de datos como el modelo ABC tradicional. Cada vez que se actualiza y recalcula un modelo, se deben actualizar los controladores de duración. " (BARRET, R. 2005:39)
5. TDABC añade un paso más que podría considerarse innecesario para el proceso de implementación de tiempo del ABC porque requiere que el administrador se involucre en el proceso de la estimación del tiempo. Este proceso no sólo aumenta los costos de recolección de la información requerida, sino que también hace que consuma mucho tiempo y crea disimetría de información.
6. Al calcular el costo de la capacidad no utilizada, los estudios de la TDABC no consideraron la capacidad de los recursos y el comportamiento de los costos completamente.
7. La ventaja de TDABC para la toma de decisiones es limitada por las siguientes razones:
   a) TDABC asume que la relación entre las actividades y los recursos consumidos es lineal, absoluta y determinada
   b) TDABC ignora las restricciones sobre los recursos de la actividad y los cuellos de botella.

c) Principalmente, las decisiones gerenciales deben seguir el concepto de "información relevante", que no es equivalente al concepto de la información sobre costes de absorción
d) La información de TDABC es útil sólo cuando el tipo de decisión se define sin ambigüedad

8. TDABC, al derivar modelos de ecuación de tiempo de "actividad", puede no necesariamente generar información de costo más precisa, porque:
   a) Los modelos de ecuaciones de tiempo se basan en la hipótesis de linealidad y certeza. Estos supuestos, por supuesto, son muy restrictivos.
   b) Las ecuaciones de tiempo no reducirán las complejidades del proceso (Tse, M., & Gong, M. 2009) y corren el riesgo de una estimación de la transparencia de costos.
   c) Las ecuaciones de tiempo no consideran los efectos de un "proceso" designado en la estimación de costos,
   d) Los modelos de ecuaciones de tiempo se desarrollan individualmente para cada actividad y se ignoran los efectos de interacción de las actividades

TDABC vs ABC

La ventaja del TDABC sobre el ABC radica en la estimación del tiempo. En efecto, el uso del parámetro tiempo como principal inductor de costes permite a TDABC evitar la compleja fase de asignar los costes de los recursos a las actividades, antes de vincularlos a los objetos de coste. El tiempo de realización de una actividad es estimado para cada caso concreto (mediante cronómetro, entrevistas a las personas, etc.). En el modelo convencional de ABC, el responsable de costes pide a las personas que respondan a una serie de cuestionarios sobre cómo reparten su tiempo entre las actividades que realizan. En TDABC las encuestas se abordan de forma diferente. El responsable de costes solicita el tiempo necesario para realizar los pasos concretos de un proceso. Por ejemplo, al analizar el proceso de introducir un pedido de compras, el responsable de TDABC solicita el tiempo que supone dar de alta al proveedor y el tiempo de procesar cada línea del pedido (de Arbulo López, P. R., & Santos, J. F. 2011). Un ejemplo del cálculo del Tiempo del proceso lo podemos observar en las siguientes líneas para una mejor comprensión.

Tiempo del proceso = suma de tiempos de actividades individuales = $(\beta 0 + \beta 1 \cdot X1 + \beta 2 \cdot X2 + \beta 3 \cdot X3 + \ldots + \beta i \cdot Xi) = \beta 0 + \Sigma \beta i \cdot Xi$ (1)

- Donde $\beta 0$ es el tiempo estándar para realizar la actividad básica.
- βi es el tiempo calculado para la actividad incremental i.
- Xi es la cantidad de actividad incremental i

Otra de las ventajas que nos menciona (Siguenza-Guzman, et al., 2013) es que TDABC asigna los costos de los recursos directamente al objeto del costo usando dos parámetros: 1)

el costo por unidad de tiempo de suministro de capacidad de recursos y 2) una estimación de las unidades de tiempo requeridas para realizar un proceso, una actividad o un servicio.

Modelos de procesos

Como se expresa en (Salgado, C. H., et al., 2016), se puede decir que un modelo de proceso de negocio muestra las actividades que se deben realizar para alcanzar los objetivos establecidos por la organización en su negocio. También en (Salgado, C. H., et al., 2016) se menciona que, en el campo del modelado de procesos de negocio se pueden encontrar numerosas propuestas de lenguajes de modelado, como IDEF0, IDEF3, UML, UML 2.0 y BPMN, por mencionar algunos, de todas estas opciones nos enfocaremos en dos de las más populares que corresponden a BPMN e IDEF.

Modelo de procesos IDEF

Los métodos o modelos IDEF son utilizados para, crear representaciones gráficas de varios sistemas, analizar el modelo de procesos, crear el modelo de una versión deseada del sistema y ayudar en la transición desde uno a otro (Ávila-Gutiérrez, M. J et. a., 2017). Dependiendo del método IDEF utilizado, se pueden determinar diferentes sintaxis para representar los modelos (Ávila-Gutiérrez, M. J et. a., 2017). Además, se encuentran herramientas software que proponen o prescriben una metodología específica para ser usada (Ávila-Gutiérrez, M. J et. a., 2017).

La familia IDEF, consiste en un gran número de técnicas, entre las cuales se destaca IDEF0 e IDEF3, que son aquellas relacionadas con los procesos de negocio, aunque existen otras versiones como IDEF1, IDEF1X, IDEF2, IDEF4 e IDEF5 (Sanchis, R. et al., 2009). La técnica IDEF0, está diseñada para modelar las decisiones, acciones y actividades de una organización u otro sistema, y representa la perspectiva funcional de modelado (Mayer et al., 1995). Es considerada una técnica sencilla pero poderosa, ampliamente usada en la industria durante la etapa de análisis en la reingeniería de procesos. Permite identificar apropiadamente los procesos y sus interfaces; así como, elaborar los documentos que permitan su control en cualquiera de sus etapas de desarrollo. IDEF0 utiliza solo un tipo de anotación en sus representaciones gráficas conocido como ICOM (Input-Control-Output-Mechanism). La representación estática de sus diagramas no permite visualizar las perspectivas de modelado de comportamiento o informacional. Para vencer dichas limitaciones, se desarrolló IDEF3 (*Process Description Capture*), que describe a los procesos como secuencias ordenadas de hechos o actividades, representando el cómo, y mostrando la visión dinámica o de comportamiento (Mayer et al., 1995).

Gestión de procesos de negocio (BPM)

El *Business Process Management* (BPM) es una metodología corporativa cuyo objetivo es mejorar el desempeño (eficiencia y eficacia) dela organización a través de la gestión de los procesos de negocio, que se deben diseñar, modelar, organizar, documentar y optimizar de forma continua (Quispe, H. G. M., et al., 2017). Esta metodología se concentra en la administración de los procesos de negocio. Se entiende como tal a la metodología que orienta los esfuerzos para la optimización de los procesos de la empresa, en busca de mejorar la

eficiencia y la eficacia por medio de la gestión sistemática de los mismos. Estos procesos deben ser modelados, automatizados, integrados, monitoreados y optimizados de forma continua. La filosofía BPM se ve como un sistema completo de información y comunicación, a través de un marco documental que permite publicar, almacenar, crear, modificar y gestionar procesos, así como acceder a ellos en cualquier momento y lugar (Díaz Piraquive, F. N. 2008).

Por lo tanto, se puede decir que, el enfoque de las tecnologías BPM es el análisis de la administración de los procesos de una empresa. Es decir, este análisis, es la convergencia de plataformas de gestión, tecnologías y aplicativos de colaboración y gestión, y de metodologías de gestión empresarial existentes en la organización. Esta unión tiene como objetivo mejorar la productividad y la eficacia de la organización a través de la optimización de sus procesos de negocio (Díaz Piraquive, F. N. 2008).

Según (Laurentiis Gianni Renato, 2005), la tecnología BPM es considerada como la evolución de los flujos de datos y dentro de sus características se pueden contemplar las reglas de negocio robustas y flexibles a través de motores de reglas de negocio, arquitectura basada en web, seguridad y autenticación de usuarios (LDAP u otros sistemas), asignación de actividades por "roles" y dinámica, gestión de *timers* dinámicos, ejecución paralela de una misma actividad, cambios a los procesos "On-the-Fly" o en línea, subprocesos y procesos articulados, ejecución y dinámica de subprocesos, el denominado "Process RollBack" en el que se devuelven los datos a algún estado previo, manejo robusto de excepciones, reportes estadísticos y de monitorización, y/o generador de reportes (datos del workflow). Organización (organigrama y localidades geográficas), calendario de negocio (festivos y horarios), integración con servidores de aplicaciones, servicios del motor a través de servicios web.

Modelos de Procesos BPMN

Un Proceso de Negocio (BP) es una secuencia de tareas realizadas para producir un resultado de valor para una organización. *Business Process Model and Notation* (BPMN) es una notación gráfica que describe la lógica de las tareas de un BP. BPMN permite coordinar la secuencia de las actividades y los mensajes que fluyen entre los participantes de las diferentes tareas (Vidal Duarte, E et al., 2014), además de hacer que las organizaciones puedan comunicar sus procesos de negocios de una manera estándar, no solo al interior de la organización, sino también a los colaboradores de ésta (Vidal Duarte, E et al., 2014). Comunicar procesos en una manera uniforme permite mejorar el control sobre la eficiencia. Por lo tanto, permite a las organizaciones mantener o aumentar su ventaja competitiva (Vidal Duarte, E et al., 2014).

BPMN permite crear diagramas de flujos de tareas de manera sencilla, permitiendo manejar la complejidad inherente a estos procesos de negocio; además, permite modelar los procesos de una manera unificada y estandarizada para una clara comprensión a todas las personas de una organización (Vidal Duarte, E et al., 2014). La siguiente tabla muestra los elementos descritos por Vidal se tiene:

Elemento	Descripción
Proceso	BPMN permite encapsular las diferentes áreas o participantes que intervienen dentro del proceso
Tareas	Representa la tarea que se realiza en un punto del proceso
Subprocesos	Es una tarea compuesta de un conjunto de subtareas. Aquí se puede apreciar el control de granularidad que ofrece BPMN para manejar la complejidad a través de los sub-procesos
Eventos	Permiten identificar el inicio y fin de un proceso
Compuertas	Representan elementos de decisión. Indican un punto de división en el flujo del proceso

Tabla 1 - Elementos de BPMN (Vidal Duarte, E et al., 2014)

TDABC con el uso de BPMN

El modelo BPM propuesto consta de una serie de fases para el modelado de los procesos de negocio en las empresas del sector productivo. El desarrollo de cada fase permite la descripción, análisis, evaluación y el modelado de los procesos de negocio.

Las características o requerimientos principales de la metodología de BPM propuesta para mejorar los procesos de negocio en empresas del sector de la producción y tomando como base lo que nos menciona (Reynoso, R. N., & Esteban, F. C. L. 2010):

1. Debe mejorar el proceso de negocio de las empresas del sector productivo teniendo en consideración y respetando, en la medida que se pueda y se necesite, las características del proceso actual, para que el alcance de la mejora no sea contraproducente respecto a otros parámetros de eficiencia o efectividad (coste o tiempo).
2. Identificación del tiempo de cada una de las actividades y el uso del grupo de recursos por cada una de ellas.
3. Debe identificar los riesgos en la producción de las actividades de los procesos de negocio, por medio de un parámetro fundamentado en un análisis de riesgos; que nos permita obtener una medida del grado de vulnerabilidad del proceso respecto a una contaminación intencional, y así tener un punto de referencia para la mejora.
4. Debe contemplar la elección de un lenguaje, técnica o herramienta de modelado común para facilitar la comparativa de los procesos de negocio.

La metodología BPM está compuesta de una serie de fases las cuales nos servirán para satisfacer los requerimientos de los puntos mencionados anteriormente y tener una mejor organización en el desarrollo de cada uno de estos requerimientos es por eso que tomando

como base a (Reynoso, R. N., & Esteban, F. C. L. 2010) las fases se enumeran a continuación y se resumen en la Figura 3:

Preparación: esta etapa es realmente, una fase previa a la ejecución de la metodología, en la cual se realizan los preparativos para la ejecución del resto de fases, y consta fundamentalmente de una introducción que incluye: la descripción de la empresa; el ámbito de actuación; la formación de equipos de trabajo para la asignación de responsabilidades, determinación de costos, método de cálculo usando la metodología TDABC, la definición y reparto de tareas (correspondientes a las fases posteriores).

Análisis y determinación de cambios: en esta fase; básicamente, se establecen las bases para analizar los procesos de negocio de las empresas de producción que se quieren mejorar, y se determinan los cambios que hay que hacer a las situaciones actuales o presentes para llegar a la situación deseada.

Evaluación de cambios: una vez determinados los cambios que hay que hacer, en esta tercera fase genérica, se evalúan dichos cambios, midiéndolos y cuantificándolos, para poder dar soporte a la siguiente y última fase.

Toma de decisiones: una vez determinados los cambios y evaluados, se podrá decidir si la situación es finalmente interesante para la entidad y conviene, por lo tanto, iniciar los trabajos y las modificaciones de las metodologías.

Implementación de los cambios: fase que organizará y planificará las acciones a realizar para alcanzar las modificaciones.

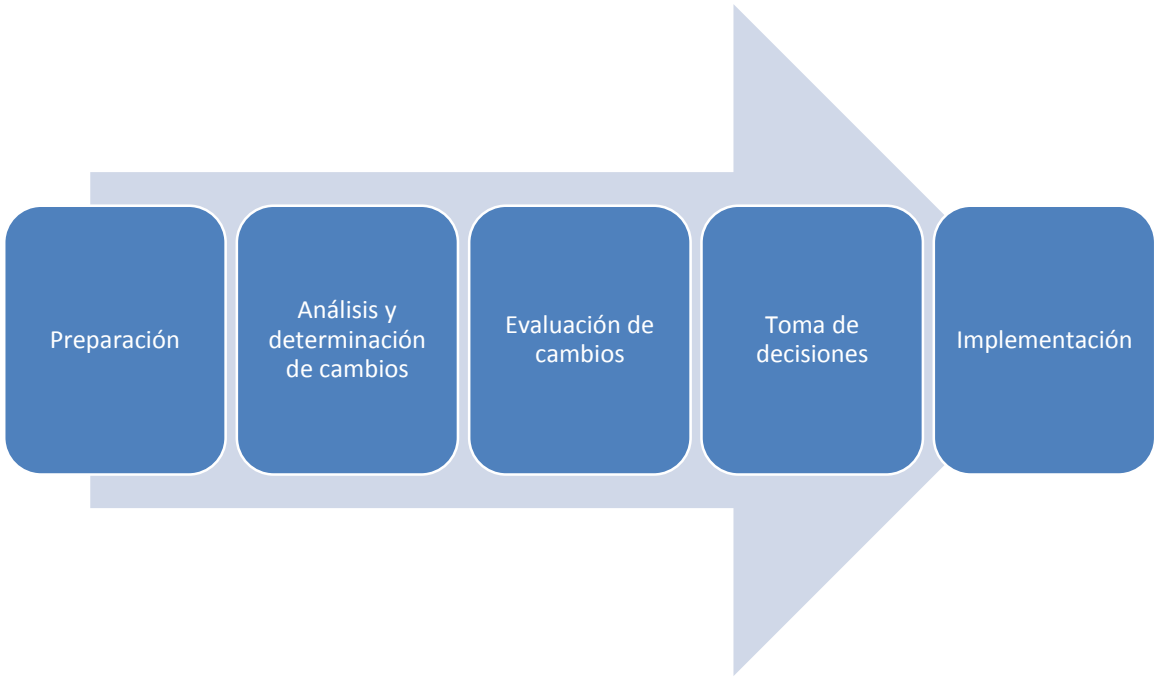

Figura 3 - Fases de la Metodología BPM propuesta usando TDABC

Luego de poseer un marco conceptual correspondiente al modelo de procesos BPM es posible adaptar la técnica de modelado BPMN a la propuesta sin mayores inconvenientes ya que se tendrían cubiertos los requerimientos principales de la metodología.

Metodologías tradicionales

De manera general se realizará una breve introducción y análisis de las metodologías tradicionales por encontrarse completamente alejadas de las metodologías agiles y poder brindar un punto de partida y comparación con el resto de metodologías denominadas ágiles.

Se estudiarán 3 metodologías PMBOK, ISO 21500, ICB, PRINCE2 y SWEBOK

PMBOK/ISO 21500

Estas dos metodologías se encuentran estrechamente relacionadas y poseen una gran similitud, por esta razón se ha decidido juntarlas y colocar su descripción.

La guía Project Management Body Of Knowledge (PMBOK) es el estándar de gestión de proyectos del Instituto de Gestión de Proyectos (Project Management Institute - PMI) y es el único estándar acreditado por la American National Standards Institute (ANSI) (organismo para la coordinación y el uso de estándares de EEUU). Proporciona un marco común para los gestores de proyectos, suministrando un léxico y unos procedimientos estructurados aplicables a cualquier tipo de proyecto. PMI se fundó en 1960 y es una organización internacional sin ánimo de lucro que asocia a profesionales relacionados con la gestión de proyectos. A finales de 1970 casi 2000 miembros formaban parte de la organización y en los 80 se realizó la primera evaluación para la certificación como profesional en gestión de proyectos, llamado Project Management Professional (PMP). Desde 2011 es la más grande del mundo por estar integrada por más de 700.000 miembros de cerca de 170 países (García Rodríguez, M. J. 2015).

La ISO 21500 es un estándar internacional desarrollado por la International *Organization for Standardization* (ISO) a partir de 2007 y publicado en 2012. Proporciona una orientación genérica, explica los principios básicos y lo que constituye unas buenas prácticas en la gestión de proyectos. Es un documento de orientación que no está destinado a ser utilizado con fines de certificación. ISO tiene previsto que esta norma sea la primera de una familia de normas de gestión de proyectos y la diseño también para alinearse con otros estándares relacionados, tales como la ISO 10006:2003, ISO 10007:2003 e ISO 31000:2009 (García Rodríguez, M. J. 2015).

Estructura

La guía PMBOK e ISO establece 47 procesos divididos en 5 grupos de proceso básicos y áreas de conocimiento.

1. Inicio. Aquellos procesos realizados para definir un nuevo proyecto o una nueva fase de un proyecto existente.
2. Planificación. Aquellos procesos requeridos para establecer el alcance del proyecto, refinar los objetivos y definir la estrategia para alcanzar los objetivos que se marcaron al comienzo del proyecto.

3. Ejecución. Aquellos procesos realizados para completar el trabajo definido en el plan del proyecto y así satisfacer sus especificaciones.
4. Monitorización y Control. Aquellos procesos requeridos para realizar el seguimiento, revisar y controlar el progreso y la eficiencia del proyecto. Se identifican las áreas en las que los cambios en el plan son necesarios y se inician los cambios correspondientes.
5. Cierre. Aquellos procesos realizados para finalizar formalmente todas las actividades de todos los procesos.

Según (Karaman, E., & Kurt, M. 2015), las 10 áreas de conocimiento son las siguientes.

- Gestión de la Integración. Se incluyen todas las actividades y procesos que hay que realizar para identificar, combinar y coordinar los diversos procesos y actividades de gestión dentro de los grupos de gestión de procesos.
- Gestión del Alcance. Se incluyen los procesos para asegurar que el proyecto tiene todo el trabajo necesario y sólo el necesario, para completar el proyecto de forma satisfactoria.
- Gestión de Plazos. Se incluyen los procesos requeridos para finalizar el proyecto de forma satisfactoria en el plazo previsto.
- Gestión de Costes. Se incluyen los procesos necesarios para poder planificar, estimar, presupuestar y controlar los costes de forma que se pueda finalizar dentro de los costes planificados.
- Gestión de la Calidad. Se determinan las políticas de calidad, objetivos y responsabilidades de forma que el proyecto satisfaga las necesidades previstas.
- Gestión de los RRHH. Se encarga de organizar y gestionar al equipo de proyecto, asignando los roles y responsabilidades correspondientes.
- Gestión de la Comunicación. Se asegura la generación temporal apropiada y la distribución, colección y almacenamiento de la información del proyecto.
- Gestión del Riesgo. Son los procesos que realizan la planificación, identificación, análisis cualitativo y cuantitativo de los riesgos, así como la planificación de las medidas a adoptar y su control.
- Gestión de Aprovisionamiento. Son los procesos que incluyen la adquisición de productos, servicios o resultados necesarios y que siendo ajenos al equipo del proyecto son necesarios para el trabajo a realizar.
- Gestión de Involucrados. Se incluye los procesos requeridos para identificar a todas las personas u organizaciones afectadas por el proyecto, analizando las expectativas y el impacto de las partes interesadas en el proyecto y el desarrollo de estrategias de gestión adecuada para la participación efectiva de los interesados en las decisiones y la ejecución de proyectos.

Las interacciones entre cada uno de estos grupos y áreas de conocimiento se describen en la Figura 4.

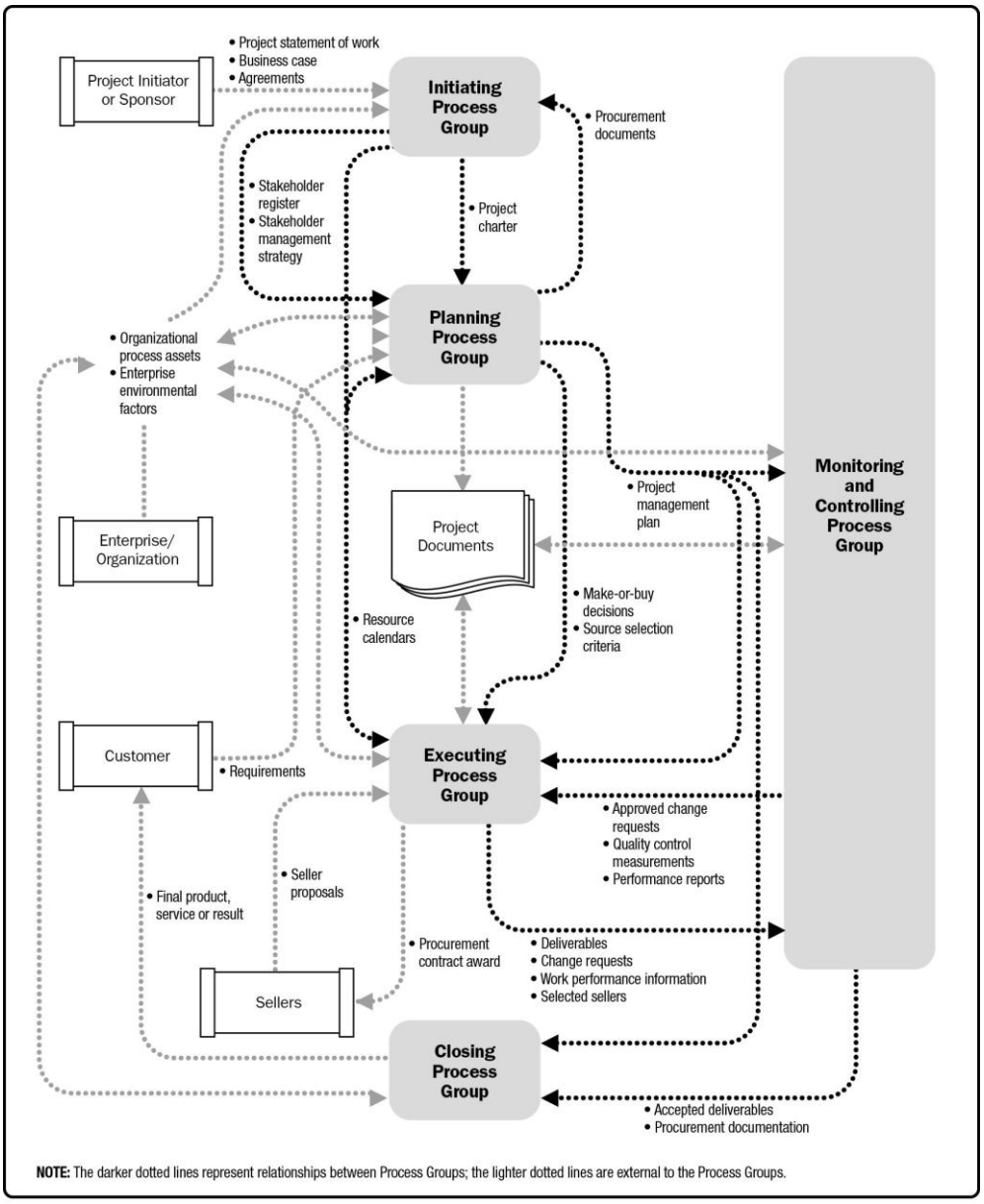

Figura 4 - Interacción de procesos en el manejo de proyectos (Project Management Institute. 2013)

Metodologías ágiles

En una reunión celebrada en febrero de 2001 en las montañas Wastch Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software (Letelier, P. 2006). En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software (Letelier, P. 2006). Su objetivo fue

esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto (Letelier, P. 2006). Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas (Letelier, P. 2006). Varias de las denominadas metodologías ágiles ya estaban siendo utilizadas con éxito en proyectos reales, pero les faltaba una mayor difusión y reconocimiento (Letelier, P. 2006).

De acuerdo a Beck K, et al. (2017) se lista los doce fundamentos del Manifiesto Ágil en el que se resume las mejores prácticas de desarrollo de software, procurando el desarrollo más rápido conservando su calidad:

1. La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Se acepta que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Se entrega software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajan juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos son desarrollados en torno a individuos motivados.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo debe reflexionar sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento.

Comparativo entre metodologías agiles y tradicionales

De acuerdo a lo que nos menciona (Cadavid, A. N., et al., 2013), las metodologías ágiles son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. Los proyectos ágiles se subdividen en proyectos más pequeños mediante una lista ordenada de características. Cada proyecto es tratado de manera independiente y desarrolla un subconjunto de características durante un periodo de tiempo corto, de entre dos y seis semanas. La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son altamente colaborativos y se adaptan mejor a los cambios; de hecho, el cambio en los requerimientos es una característica esperada y deseada, al igual que las entregas constantes al cliente y la retroalimentación por parte de él.

Tanto el producto como el proceso son mejorados frecuentemente. Un resumen comparativo es presentado en la Tabla 2.

Metodologías tradicionales	**Metodologías ágiles**
Predictivos	Adaptativos
Orientados a procesos	Orientados a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdividido en varios proyectos más pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de software al finalizar el desarrollo	Entregas constantes de software
Documentación extensa	Poca documentación

Tabla 2 - Metodologías tradicionales vs Metodologías Ágiles (Cadavid, A. N., et al., 2013)

Metodologías de desarrollo de software ágiles

De acuerdo a Avison, D., y Fitzgerald, G. (2003) una metodología de desarrollo es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo (Avison, D., y Fitzgerald, G. 2003). A partir de estos conceptos el término "Métodos Ágiles" fue creado en marzo de 2001 para definir a las nuevas metodologías que estaban surgiendo como alternativa a las tradicionales formales. Al final se resumió cuatro postulados que se denominó "Manifiesto Ágil" que son los valores sobre los que se asientan las metodologías ágiles.

La construcción de software es el evento fundamental de la ingeniería de software. Los programadores trabajan construyendo e integrando programas a través de técnicas de codificación, validación y pruebas. Pero ese carácter esencial no minimiza fases tan cruciales como la planeación del proyecto, el análisis de requerimientos, el diseño y la gestión de la calidad (Parra Castrillón, E. 2011).

En los proyectos de desarrollo de software es primordial la definición de la metodología. Esta se define según la forma como se asuman las distintas actividades para la consecución del

producto final de software. Las metodologías aplican distintos modelos de desarrollo tales como el de cascada, el incremental, el evolutivo y el de espiral (Aycart Pérez, D et. al, 2007). El modelo en cascada determina cuatro fases terminales del ciclo de vida, con unos hitos específicos al final de cada una (toma de requisitos, análisis, diseño e implementación). Los modelos incremental y evolutivo permiten la creación de productos en etapas parciales, donde cada etapa agrega funcionalidad a la anterior e incluye las fases del ciclo de vida. El modelo en espiral incluye la creación de prototipos del proyecto que pasan cíclicamente por las fases del ciclo de vida, hasta llegar paulatinamente al producto final, después de validarse repetidamente los requisitos y diseños del proyecto.

Con respecto a las metodologías se pueden listar las siguientes, cada una de ellas con una breve descripción brindando una idea global de las principales características:

SCRUM

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en el desarrollo mediante iteraciones y reuniones durante todo el proyecto (Schwaber, K., & Beedle, M. 2002). La primera que corresponde a el desarrollo de software se realiza mediante iteraciones, denominadas *sprints* cada uno de estos con una secuencia de fases (visualización, planeación, análisis y propuesta), con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (Schwaber, K., & Beedle, M. 2002).

Crystal Methodologies

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo (de ellas depende el éxito del proyecto) y la reducción al máximo del número de artefactos producidos (Cockburn, A. 2002). Han sido desarrolladas por Alistair Cockburn(Cockburn, A. 2002). El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar (Cockburn, A. 2002). El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas (Cockburn, A. 2002). Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores; por ejemplo, Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros) (Cockburn, A. 2002).

Dynamic Systems Development Method (DSDM)

Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo de crear una metodología RAD unificada (Stapleton, J. 1997). Sus principales características es ser un proceso iterativo e incremental, y el equipo de desarrollo y el usuario trabajan juntos (Stapleton, J. 1997). Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación

(Stapleton, J. 1997). Las tres últimas son iterativas, además de existir realimentación a todas las fases (Stapleton, J. 1997).

Adaptive Software Development (ASD)

Su impulsor es Jim Highsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios (Highsmith, J. 2013). El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje (Highsmith, J. 2013). En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente (Highsmith, J. 2013). La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo (Highsmith, J. 2013).

Feature-Driven Development (FDD)

Define un proceso iterativo que consta de cinco pasos (1) Estudio de Viabilidad, (2) Estudio del Negocio, (3) Modelado Funcional, (4) Diseño y Construcción y (5) Implementación (Mendes Calo, K., et al., (2010). Las iteraciones son cortas (hasta dos semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad (Coad, P., Luca, J. D., y Lefebvre, E. 1999).

Lean Development (LD)

Definida por Bob Charette's a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa (Poppendieck, M., & Poppendieck, T. 2003). En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente (Poppendieck, M., & Poppendieck, T. 2003). Su principal característica es introducir un mecanismo para implementar dichos cambios (Poppendieck, M., & Poppendieck, T. 2003).

Extreme Programming(XP)

La metodología XP se considera una metodología de desarrollo leve, al igual que el resto de metodologías agiles trata de resolver los problemas de entrega de software de calidad de manera rápida, satisfacer las necesidades y objetivos de negocio que siempre son cambiantes (Letelier, P. 2006). No es aplicable a todo tipo de proyecto más bien a proyectos con equipos pequeños o medianos es decir de dos a 12 personas (Letelier, P. 2006).

Kanban

De acuerdo a Acevedo et al. (2001), Kanban es una técnica de gestión de producción basada en un sistema pull (halar) que se fundamenta en la autogestión de los procesos, eliminando la programación centralizada. Se produce y se transporta lo que se demanda en los procesos consumidores, manteniendo en rotación sólo aquellas cantidades que garantizan la continuidad del consumo. Cuando se interrumpe el consumo se detiene la producción. Es una herramienta para conseguir la producción "Justo a Tiempo" (JIT).

La metodología Kanban es aplicable a entornos repetitivos de manufactura en donde el material fluye en rutas fijas y tasas constantes. Para estos casos existe gran variedad de técnicas que funcionan bajo el mismo esquema en el cual se aplica Kanban (Arango et.al 2015).

Conclusiones

Como se ha analizado hasta el momento, para el desarrollo de este proyecto es necesario involucrar varios aspectos considerados de mayor relevancia e impacto para el cumplimiento de nuestros objetivos. Resultado de este análisis se establecen las siguientes herramientas: dentro de la metodología de costeo se usará TDABC por involucrar de forma más detallada el factor tiempo para el análisis de los costos. Dentro del modelo de procesos se tiene a BPMN por su mayor difusión y documentación de ayuda. Para el desarrollo de software, la metodología a utilizar corresponde a SCRUM, por la forma en la que se manejan los prototipos, entregas y el manejo general del equipo de trabajo de una forma no jerárquica.

Capítulo 3: Estado actual de las empresas de producción en cuanto a métodos de costeo – Etapa 1(Visualización)

Introducción

En la experiencia profesional mantenida por el equipo de trabajo del proyecto de investigación y, luego de múltiples observaciones, entrevistas con gerentes, empleados de diferentes departamentos, personal que maneja los sistemas y bases de datos; se ha planteado la realización del análisis de costos en las empresas del sector productivo y específicamente el de ensamblaje, la idea principal es generar conocimiento adecuado del uso de las metodologías de costeo para su aplicación.

Historia

Gerardo Ortiz surgió en la ciudad de Cuenca en el año de 1953, cuando Don Gerardo Ortiz, reconocido comerciante y transportista adquirió una abacería ubicada en el Mercado Mayorista, 10 de Agosto. Don Gerardo y Doña Carmen, su esposa, ganaron la confianza de sus clientes y con ello la satisfacción de sentirse bien servidos. Un año más tarde, incursionaron en la producción y venta de café procesado el mismo que hasta hoy es conocido como Café Cubanito. Posteriormente con el desarrollo del sector de calzado en la provincia del Azuay, Don Gerardo decidió ampliar su negocio, a través de la venta de materias primas y accesorios para esta línea. En 1976 se constituyó la empresa Gerardo Ortiz e Hijos, a partir de aquí esta empresa incursiono en la industria química, con la puesta en marcha de la fábrica de pegamentos, plásticos y accesorios de calzado, ADHEPLAST. A partir de eso, el grupo Gerardo Ortiz e hijos, se diversifica en diferentes grupos económicos, uno de ellos es el denominado SURAMERICANA DE MOTORES MOTSUR CIA LTDA, con nombre comercial MOTSUR se encuentra en el ranking # 11, entre las 35 empresas de Grupo Ortiz, la cual se dedica al ensamblaje de televisores motocicletas y radios de coches.

Estado Actual

La empresa Suramericana de Motores Motsur Cia Ltda a la fecha no dispone de una metodología de costeo que permita la toma de decisiones y el análisis de datos de manera efectiva. Por otro lado, tampoco dispone de una herramienta tecnológica que permita facilitar el tratamiento de los ingresos para el manejo de datos históricos, análisis y toma de decisiones de manera efectiva.

Conclusiones

Es imperativo que una empresa que maneje grandes cantidades de datos como es la empresa Motsur organice, estructure y adopte una metodología de costeo acorde a sus necesidades. Son algunos los usos de los resultados de estas metodologías algunos de ellos corresponden al uso interno de la gerencia en la formulación de objetivos y programas de operación en la comparación del desempeño real con el esperado y en la presentación de informes que servirán para la toma de decisiones.

Los mandos altos de la empresa, gerentes y administrativos hacen frente a situaciones que afectan directamente el funcionamiento de Motsur, la información que se obtenga de los

costos y los gastos en que incurre la organización para realizar su actividad y que rige su comportamiento, son de vital importancia para la toma de decisiones de una manera rápida y eficaz, esto hace que actualmente una metodología de costeo tome gran importancia frente a las necesidades de los encargados del manejo de información.

Capítulo 4: Componentes de análisis y diseño para desarrollo de software – Etapa 2 (Planeación)

Introducción

En cualquier desarrollo de sistemas informáticos es de vital importancia seguir alguna especificación que permita a los analistas y desarrolladores tener una guía facilitando la documentación y el seguimiento de todas las etapas del desarrollo del sistema, provocando que cada una sea más coherente y formal. Este capítulo navega por todos los componentes del análisis y diseño para el desarrollo de software pasando por la ingeniería, modelado, documentación y riesgos. Pretende dar una visión global de los componentes para la futura adaptación a la metodología de desarrollo Scrum, seleccionado en el proyecto.

Análisis y Diseño

En todo desarrollo de software es primordial seguir alguna especificación que permita a los programadores y analistas tener una guía de cada una de las etapas del desarrollo del sistema, partiendo de los requerimientos hasta las pruebas finales, permitiendo que cada uno de los pasos tengan formalidad y coherencia.

El análisis y diseño para un futuro desarrollo de software que este proyecto propone un problema real, razón por la cual es imperativo seguir una guía que proporcione los cimientos necesarios para futuros desarrollos e implementaciones en distintas empresas u organizaciones.

Dentro del análisis y diseño se encuentran los modelos de ciclo de vida del software, de lo analizado en (Aycart, Ginestá & Hernández, 2007), todas coinciden implícitamente o explícitamente en cinco fases: requisitos, diseño, desarrollo, pruebas y mantenimiento. Como se encuentran relacionadas, su organización y la importancia que se dé a cada una de ellas dentro del ciclo de vida dependerá del modelo que se escogió para el desarrollo (Aycart, Ginestá & Hernández, 2007).

- **Requisitos:** Es el análisis y recopilación formal de los requerimientos tanto funcionales como no funcionales que deberá cumplir el proyecto. Es bastante común que los clientes o usuarios de las aplicaciones no posean una idea clara de las necesidades y resultado del proyecto, pero no necesariamente sobre las funciones que tendrá el software (Aycart, Ginestá & Hernández, 2007).
- **Diseño:** En esta etapa se establecerá la idea y estructura general del proyecto de software tomando en cuenta todos los puntos de la etapa anterior (Aycart, Ginestá & Hernández, 2007).
- **Desarrollo**: También conocida como codificación, en esta etapa los desarrolladores o ingenieros de software programan cada una de las secciones del software empezando por el análisis y diseño de las etapas anteriores (Aycart, Ginestá & Hernández, 2007).
- **Pruebas:** Se desarrolla el "testing" del software desarrollado mediante varias técnicas como pruebas de caja negra o blanca, todos estos errores pueden aumentar

considerablemente la dificultad en el desarrollo (Aycart, Ginestá & Hernández, 2007).

- **Mantenimiento:** Durante la utilización del software es bastante común realizar mantenimientos ya sea por errores o desarrollar nuevas características o funcionalidades (Aycart, Ginestá & Hernández, 2007).

Además de estas etapas se encuentran implícitas etapas como la documentación y guías del software.

Modelos de ciclo de vida

Dentro de este apartado se resumirán algunos de los distintos ciclos de vida del software que se mencionarán y usarán dentro del proyecto.

Cascada: como nos menciona (Aycart, Ginestá & Hernández, 2007) se considera el primer modelo de desarrollo de software, se derivó de procesos de ingeniería más generales; en este modelo cada una de las fases deberá ser completada para proceder a iniciar la siguiente como se muestra en la Figura 5.

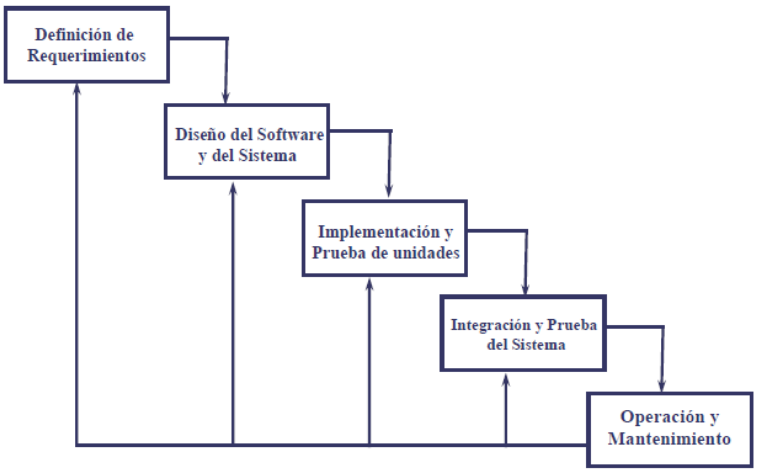

Figura 5 - Modelo de ciclo de vida de tipo Cascada (Aycart, Ginestá & Hernández, 2007)

Incremental y evolutivo: Los modelos incremental y evolutivo son una variación del modelo en cascada en la que éste se aplica a subconjuntos del proyecto. Dependiendo de si los subconjuntos son partes del total (modelo incremental) o bien versiones completas, pero con menos prestaciones (modelo evolutivo) se aplicará uno u otro (Aycart, Ginestá & Hernández, 2007).

Espiral: El modelo en espiral se basa en la creación de prototipos del proyecto, que pasan por las fases anteriores, y que van acercándose sucesivamente a los objetivos finales, como se puede observar la Figura 6. Así pues, se examina y valida repetidamente los requisitos y diseños del proyecto antes de acometer nuevas fases de desarrollo (Aycart, Ginestá & Hernández, 2007).

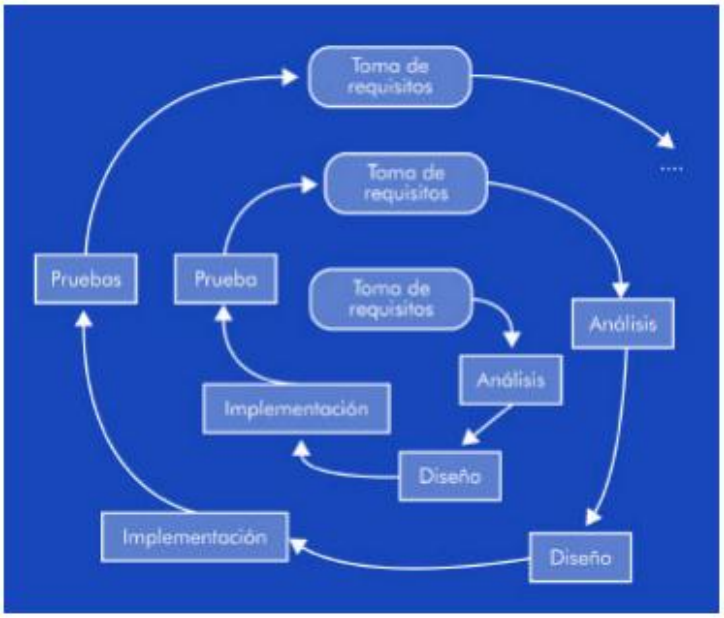

Figura 6 - Modelo de ciclo de vida de tipo Espiral (Aycart, Ginestá & Hernández, 2007).

Finalmente, el modelo iterativo o incremental es el que permite que las fases de análisis, diseño, desarrollo y pruebas se retroalimenten continuamente, y que empiecen lo antes posible como visualizamos en la Figura 7. Permitirá atender a posibles cambios en las necesidades del usuario o a nuevas herramientas o componentes que los desarrolladores descubran y que faciliten el diseño o proporcionen nuevas funcionalidades (Aycart, Ginestá & Hernández, 2007).

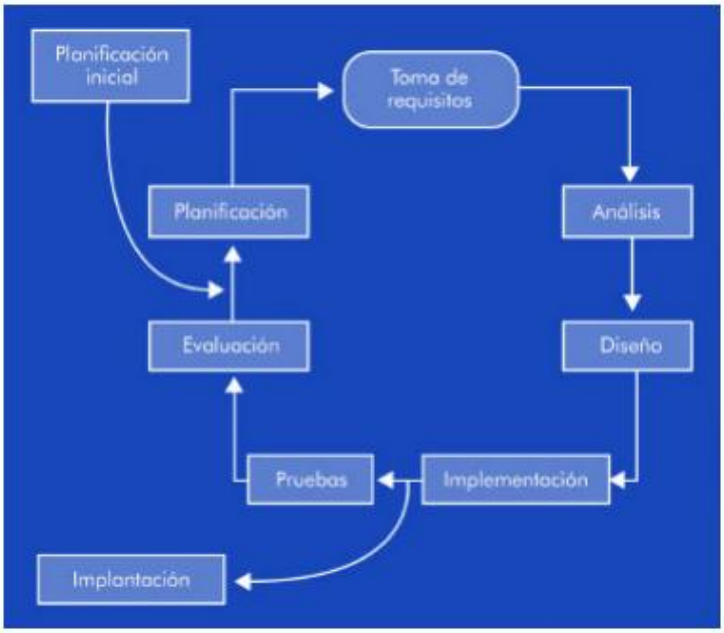

Figura 7 - Modelo de ciclo de vida(Incremental) (Aycart, Ginestá & Hernández, 2007).

Prototipado evolutivo: El uso de prototipos se centra en la idea de ayudar a comprender los requisitos que plantea el usuario, sobre todo si este no tiene una idea muy acabada de lo que desea. También pueden utilizarse cuando el ingeniero de software tiene dudas acerca de la viabilidad de la solución pensada (Cataldi, Z. 2000).

Esta versión temprana de lo que será el producto, con una funcionalidad reducida, en principio, podrá incrementarse paulatinamente a través de refinamientos sucesivos de las especificaciones del sistema, evolucionando hasta llegar al sistema final (Cataldi, Z. 2000).

Según Cataldi, Z. (2000), al usar prototipos, las etapas del ciclo de vida clásico quedan modificadas por el análisis de requisitos del sistema y software; diseño, desarrollo e implementación del prototipo; evaluación del prototipo; refinamiento iterativo del prototipo y de las especificaciones del mismo; diseño e implementación del sistema final y finalmente la explotación (u operación) y mantenimiento producto de ingeniería. Figura 8.

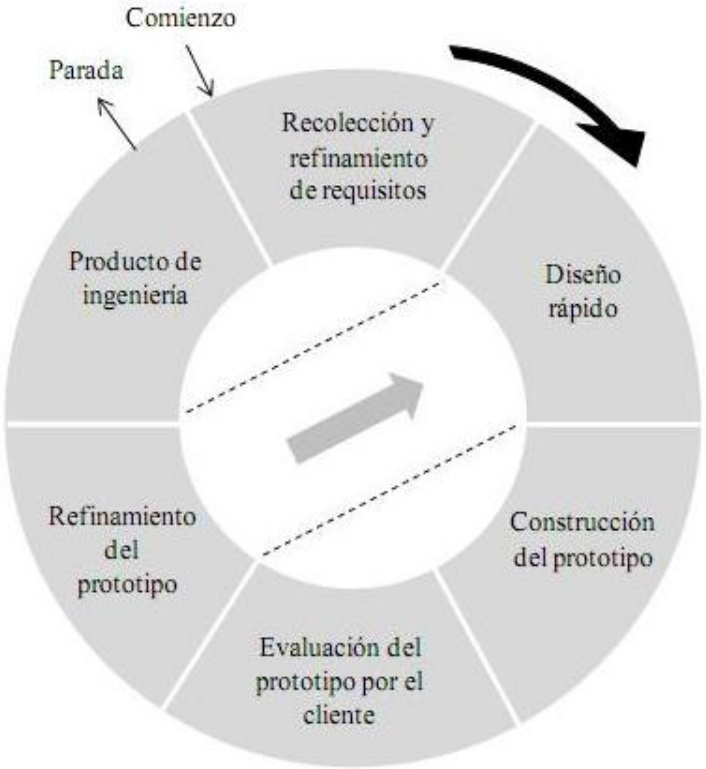

Figura 8 - Modelo de ciclo de vida (Prototipado evolutivo) (Cataldi, Z. 2000)

Ingeniería de software

Dentro del concepto de ingeniería de software se realizará un comparativo del concepto planteado por el mismo autor en dos ediciones distintas de su libro.

Como lo menciona (Pressman, R. S., & Troya, J. M. 1988), el software de computadora es el producto que construyen los programadores profesionales y al que después le dan

mantenimiento durante un largo tiempo. Incluye programas que se ejecutan en una computadora de cualquier tamaño y arquitectura, contenido que se presenta a medida que se ejecutan los programas de cómputo e información descriptiva tanto en una copia dura como en formatos virtuales que engloban virtualmente a cualesquiera medios electrónicos. La ingeniería de software está formada por un proceso, un conjunto de métodos (prácticas) y un arreglo de herramientas que permite a los profesionales elaborar software de cómputo de alta calidad.

De igual forma (Pressman, Roger. S. 2010) nos dice que la ingeniería de software es una tecnología compuesta de varias capas: herramientas, métodos, procesos, compromiso con la calidad; cualquier enfoque de ingeniería (incluso la de software) debe basarse en un compromiso organizacional con la calidad. La administración total de la calidad, Six sigma y otras filosofías similares alimentan la cultura de mejora continua, y es esta cultura la que lleva en última instancia al desarrollo de enfoques cada vez más eficaces de la ingeniería de software. El fundamento en el que se apoya la ingeniería de software es el compromiso con la calidad.

Gestionar un proyecto implica tener que adoptar una solución de compromiso entre las restricciones de alcance, calidad, plazos, presupuesto, recursos y riesgos. Además, en la Ing. de Software hay factores tecnológicos específicos que pueden dar lugar a más restricciones (García Rodríguez, M. J. 2015). Para el manejo de todos estos puntos pertenecientes a la ingeniería de software se ha planteado el uso de metodologías de desarrollo ágiles específicamente Scrum analizada con más detalle dentro del marco teórico.

Modelado del negocio

Para conseguir sus objetivos, una empresa organiza su actividad por medio de un conjunto de procesos de negocio. Cada uno de ellos se caracteriza por una colección de datos que son producidos y manipulados mediante un conjunto de tareas, en las que ciertos agentes (por ejemplo, trabajadores o departamentos) participan de acuerdo a un flujo de trabajo determinado. Además, estos procesos se hallan sujetos a un conjunto de reglas de negocio, que determinan las políticas y la estructura de la información de la empresa. Por tanto, la finalidad del modelado del negocio es describir cada proceso del negocio, especificando sus datos, actividades (o tareas), roles (o agentes) y reglas de negocio (García Molina, J., et, al 2007).

El primer paso del modelado del negocio consiste en capturar los procesos de negocio de la organización bajo estudio. La definición del conjunto de procesos del negocio es una tarea crucial, ya que define los límites del proceso de modelado posterior. En primer lugar, consideramos los objetivos estratégicos de la organización. Teniendo en cuenta que estos objetivos van a ser muy complejos y de un nivel de abstracción muy alto, serán descompuestos en un conjunto de sub-objetivos más concretos, que deberán cumplirse para conseguir el objetivo estratégico. Estos sub objetivos pueden a su vez ser descompuestos en otros, de manera que se defina una jerarquía de objetivos. En nuestro estudio, hemos experimentado que dos o tres niveles de descomposición son suficientes. Para cada uno de

estos sub-objetivos de segundo nivel definimos un proceso de negocio que deberá dar soporte a dicho sub-objetivo. Representamos cada proceso del negocio como un caso de uso del negocio, que inicialmente será descrito de forma textual.

Requerimientos del Negocio

De acuerdo a lo mencionado por (Arias Chaves, M. 2005) un requerimiento es una descripción de una condición o capacidad que debe cumplir un sistema, ya sea derivada de una necesidad de usuario identificada, o bien, estipulada en un contrato, estándar, especificación u otro documento formalmente impuesto al inicio del proceso.

Los requerimientos de software pueden dividirse en 2 categorías: requerimientos funcionales y requerimientos no funcionales.

Los requerimientos funcionales son los que definen las funciones que el sistema será capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Es importante que se describa el ¿Qué? y no el ¿Cómo? se deben hacer esas transformaciones. Estos requerimientos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema.

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

Es importante no perder de vista que un requerimiento debe ser:

- **Especificado por escrito**: Como todo contrato o acuerdo entre dos partes. Posible de probar o verificar. Si un requerimiento no se puede comprobar, entonces ¿cómo se sabe si se cumplió con él o no?
- **Conciso**: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple clara para aquellos que vayan a consultarlo en un futuro.
- **Completo**: Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
- **Consistente**: Un requerimiento es consistente si no es contradictorio con otro requerimiento.
- **No ambiguo**: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector

Como lo describe (Arias Chaves, M. 2005) mencionando a (Herrera, 2003), existen cuatro actividades básicas que se tienen que llevar a cabo para completar el proceso de determinación de requerimientos:

- **Extracción**: Esta fase representa el comienzo de cada ciclo. Extracción es el nombre comúnmente dado a las actividades involucradas en el descubrimiento de los requerimientos del sistema. Aquí, los analistas de requerimientos deben trabajar junto al cliente para descubrir el problema que el sistema debe resolver, los diferentes

servicios que el sistema debe prestar, las restricciones que se pueden presentar, etc. Es importante, que la extracción sea efectiva, ya que la aceptación del sistema dependerá de cuan bien éste satisfaga las necesidades del cliente.

- **Análisis**: Sobre la base de la extracción realizada previamente, comienza esta fase en la cual se enfoca en descubrir problemas con los requerimientos del sistema identificados hasta el momento. Usualmente se hace un análisis luego de haber producido un bosquejo inicial del documento de requerimientos; en esta etapa se leen los requerimientos, se conceptúan, se investigan, se intercambian ideas con el resto del equipo, se resaltan los problemas, se buscan alternativas y soluciones, y luego se van fijando reuniones con el cliente para discutir los requerimientos.
- **Especificación**: En esta fase se documentan los requerimientos acordados con el cliente, en un nivel apropiado de detalle.
  En la práctica, esta etapa se va realizando conjuntamente con el análisis, se puede decir que la especificación es el "pasar en limpio" el análisis realizado previamente aplicando técnicas y/o estándares de documentación, como la notación UML (Lenguaje de Modelado Unificado), que es un estándar para el modelado orientado a objetos, por lo que los casos de uso y la obtención de requerimientos basada en casos de uso se utiliza cada vez más para la obtención de requerimientos.
- **Validación**: La validación es la etapa final de la IR. Su objetivo es, ratificar los requerimientos, es decir, verificar todos los requerimientos que aparecen en el documento especificado para asegurarse que representan una descripción, por lo menos, aceptable del sistema que se debe implementar. Esto implica verificar que los requerimientos sean consistentes y que estén completos.
  Se puede apreciar que el proceso de ingeniería de requerimientos es un conjunto estructurado de actividades, mediante las cuales se obtiene, se valida y se logra dar un mantenimiento adecuado al documento de especificación de requerimientos, que es el documento final, de carácter formal, que se obtiene de este proceso. Es necesario recalcar que no existe un proceso único que sea válido de aplicaren todas las organizaciones. Cada organización debe desarrollar su propio proceso de acuerdo al tipo de producto que se esté desarrollando, a la cultura organizacional, y al nivel de experiencia y habilidad delas personas involucradas en la ingeniería de requerimientos. Hay muchas maneras de organizar el proceso de ingeniería de requerimientos y en otras ocasiones se tiene la oportunidad de recurrir a consultores, ya que ellos tienen una perspectiva más objetiva que las personas involucradas en el proceso.

Roles del Entorno del Negocio

Una vez se han identificado los procesos de negocio, es preciso encontrar los agentes involucrados en su realización. Cada uno de estos agentes o actores del negocio desempeña cierto papel (juega un rol) cuando colabora con otros para llevar a cabo las actividades que conforman dicho caso de uso del negocio. De hecho, identificaremos los roles que son jugados por agentes de la propia empresa (que incluyen trabajadores, departamentos y dispositivos físicos) o agentes externos (como clientes u otros sistemas). Por el momento nos

centraremos en este último tipo de roles, con los que la organización interactúa para llevar a cabo sus procesos de negocio. En nuestro ejemplo tenemos los roles Cliente y Proveedor, claramente externos al sistema (García Molina, J., et, al 2007).

Casos de Uso del Negocio

Los casos de uso permiten entonces describir la posible secuencia de interacciones entre el sistema y uno o más actores, en respuesta a un estímulo inicial proveniente de un actor, es una descripción de un conjunto de escenarios, cada uno de ellos comenzado con un evento inicial desde un actor hacia el sistema. La mayoría de los requerimientos funcionales, sino todos, se pueden expresar con casos de uso. Según el autor Sommerville, los casos de uso son una técnica que se basa en escenarios para la obtención de requerimientos (Arias Chaves, M. 2005).

Prototipos

Durante la actividad de extracción de requerimientos, puede ocurrir que algunos requerimientos no estén demasiado claros o que no se esté muy seguro de haber entendido correctamente los requerimientos obtenidos hasta el momento, todo lo cual puede llevar a un desarrollo no eficaz del sistema final. Entonces, para validar los requerimientos hallados, se construyen prototipos. Los prototipos son simulaciones del posible producto, que luego son utilizados por el usuario final, permitiéndonos conseguir una importante retroalimentación en cuanto a si el sistema diseñado con base a los requerimientos recolectados le permite al usuario realizar su trabajo de manera eficiente y efectiva. El desarrollo del prototipo comienza con la captura de requerimientos. Desarrolladores y clientes se reúnen y definen los objetivos globales del software, identifican todos los requerimientos que son conocidos, y señalan áreas en las que será necesaria la profundización en las definiciones. Luego de esto, tiene lugar un "diseño rápido". El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles al usuario (por ejemplo, entradas y formatos de las salidas). El diseño rápido lleva a la construcción de un prototipo (Arias Chaves, M. 2005).

Casos de Prueba

Para desarrollar software de calidad y libre de errores, el plan de pruebas y los casos de prueba son muy importantes como lo menciona (Aristegui, J. L. 2010). El Software Test Plan (STP) se diseña para determinar el ambiente de aplicación de los recursos y el calendario de las actividades de las pruebas, se debe identificar el dominio y sus características a probar, lo mismo que el tipo de pruebas a realizar. Un caso de prueba bien diseñado tiene gran posibilidad de llegar a resultados más fiables y eficientes, mejorar el rendimiento del sistema, y reducir los costos en tres categorías: productividad, menos tiempo para escribir y mantener los casos; capacidad de prueba, menos tiempo para ejecutarlos; y programar la fiabilidad estimaciones más fiables y efectivas

Otro concepto que seguiremos dentro de este trabajo (Puello, O. 2013) mostrado también en la Figura 9, es que los casos de prueba son especificaciones de las entradas para la prueba y la salida esperada del sistema más una afirmación de lo que se está probando. Los datos de prueba son las entradas que han sido ideadas para probar el sistema. Los datos de prueba a veces pueden generarse automáticamente. La generación automática de casos de prueba es

imposible. Las salidas de las pruebas solo pueden predecirse por personas que comprenden lo que debería hacer el sistema.

Las pruebas exhaustivas, en las que cada posible secuencia de ejecución del programa es probada, son imposibles. Las pruebas, por lo tanto, tiene que basarse en un subconjunto de posibles casos de prueba. Idealmente, algunas compañías deberían tener políticas para elegir este subconjunto en lugar de dejar solo al equipo de desarrollo. Estas políticas podrían basarse en políticas generales de pruebas, tal como una política en la que todas las sentencias de los programas deberían ejecutarse al menos una vez.

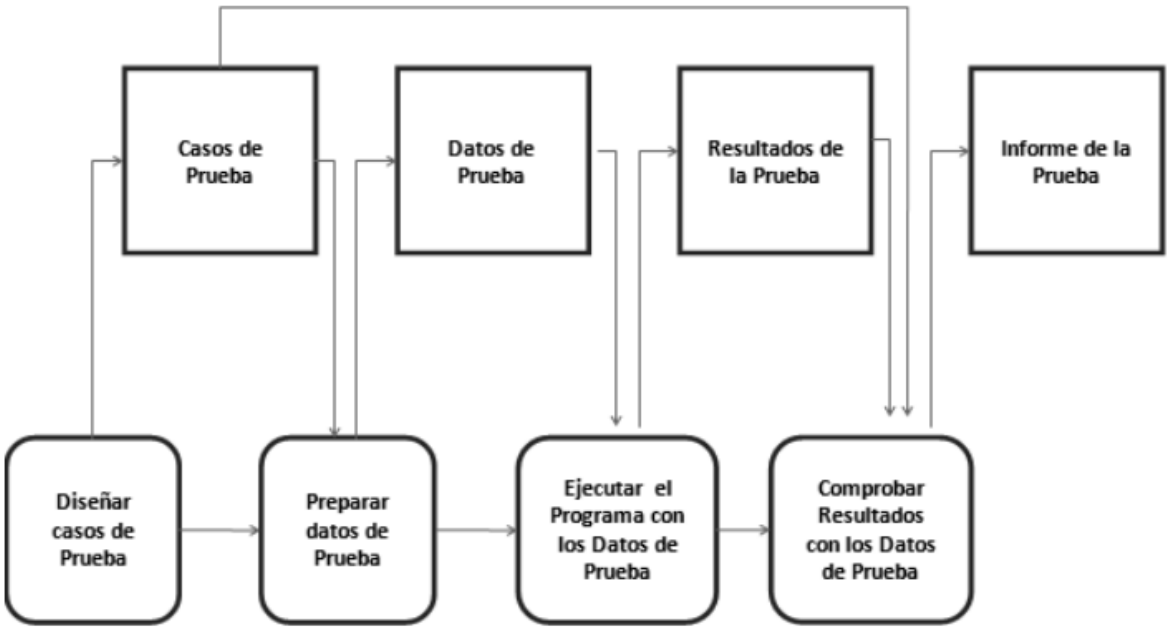

Figura 9 - Modelo de procesos de pruebas de software (Puello, O. 2013)

Documentación

Doxygen

Además de toda documentación que podría ser generada en base a los puntos anteriores es necesario la producción de documentación a nivel de código en donde el mantenimiento de la aplicación no requiera un esfuerzo grande por parte de los desarrolladores.

Es necesario abordar una automatización de generación de documentación legible entendiendo esta documentación, no solo como un contenido añadido al programa, sino más bien como una característica intrínseca durante la generación del programa. En este paradigma el programador escribe y tanto el humano como la maquina pueden leer. Puesto que la máquina y el ser humano leen de forma diferente es necesario proveer al programador de diferentes herramientas para generar paralelamente al código, distintas representaciones semánticamente representativas (Asensio, J. J. et al., 2013).

De acuerdo a las recomendaciones que presenta Doxygen tenemos los puntos en los que será conveniente documentar:

¿de qué se encarga una clase? ¿un paquete?
¿qué hace un método?
¿cuál es el uso esperado de un método?
¿para qué se usa una variable?
¿cuál es el uso esperado de una variable?
¿qué algoritmo estamos usando? ¿de dónde lo hemos sacado?
¿qué limitaciones tiene el algoritmo? ¿… la implementación?
¿qué se debería mejorar … si hubiera tiempo?
¿Cuándo hay que poner un comentario?

Por obligación deberá ser colocado al principio de cada clase, al principio de cada método y ante cada variable de clase. Por conveniencia (una línea) debera ser colocado al principio de fragmento de código no evidente y a lo largo del bucle. Finalmente, por si acaso (una línea) siempre que se realice algo fuera de los estándares y siempre que el código no sea evidente.

Javadoc

De acuerdo a lo que nos menciona el sitio web de Oracle, Javadoc es una herramienta para generar la documentación de la API en formato HTML desde los comentarios del doc en el código fuente, es considerado el estándar de la industria para documentar clases de java.

Esta herramienta se encuentra ampliamente documentada y su ayuda clasificada en los siguientes puntos: 1) Javadoc FAQ - Esta FAQ cubre desde dónde descargar la herramienta Javadoc, cómo encontrar una lista de errores conocidos y peticiones de características, soluciones para errores conocidos, cómo aumentar la memoria de Javadoc y algunos puntos más. 2)Documentación de Javadoc - Mejoras, Doclet (clases en Java que especifican el contenido y el fomato de salida de la herramienta javadoc) estándar, descripción de Doclet, API Doclet y Taglet. 3)Páginas de referencia de Javadoc. 4)Cómo escribir comentarios de Doc para convenciones de Javadoc - Sun y sus especificaciones para escribir comentarios de la documentación. 5)Requisitos para la escritura de especificaciones API - Requisitos estándar utilizados al escribir la especificación Java 2 Platform. Cubre los requisitos de paquetes, clases, interfaces, campos y métodos para satisfacer las aserciones.

Gestión de riesgos

Como cualquier actividad humana, el desarrollo de un proyecto incluye la ocurrencia de riesgos, los cuales son eventos o condiciones inciertas, que, si se producen, afectan de manera positiva o negativa al menos un objetivo del proyecto (Pressman, 2010). Una acertada gestión de riesgos es la mejor manera de mitigar un riesgo o un conjunto de estos, ya que permite identificar, analizar y responder a los riesgos a lo largo de la vida de un proyecto, con el propósito de aumentar la probabilidad y el impacto de los eventos positivos y disminuir la de los eventos adversos. Este proceso dentro del ciclo de desarrollo de un proyecto de software

es generalmente complejo, pues requiere seguimiento constante dependiendo en gran medida de las experiencias acumuladas, el conocimiento adquirido y los estándares definidos (Cordero Morales, D et al., 2013).

Aunque hay un considerable debate acerca de la definición adecuada de riesgo de software, existe un acuerdo general en que los riesgos siempre involucran dos características: incertidumbre (el riesgo puede o no ocurrir; es decir, no hay riesgos 100 por ciento probables) y pérdida (si el riesgo se vuelve una realidad, ocurrirán consecuencias o pérdidas no deseadas). Cuando se analizan los riesgos es importante cuantificar el nivel de incertidumbre y el grado de pérdidas asociados con cada riesgo. Para lograr esto, se consideran diferentes categorías de riesgos de acuerdo a (Pressman, R. S. 2010).

- Los riesgos del proyecto amenazan el plan del proyecto, es decir, si los riesgos del proyecto se vuelven reales, es probable que el calendario del proyecto se deslice y que los costos aumenten. Los riesgos del proyecto identifican potenciales problemas de presupuesto, calendario, personal (tanto técnico como en la organización), recursos, participantes y requisitos, así como su impacto sobre un proyecto de software. La complejidad, el tamaño y el grado de incertidumbre estructural del proyecto también se definieron como factores de riesgos para el proyecto (y la estimación).
- Los riesgos técnicos amenazan la calidad y temporalidad del software que se va a producir. Si un riesgo técnico se vuelve una realidad, la implementación puede volverse difícil o imposible. Los riesgos técnicos identifican potenciales problemas de diseño, implementación, interfaz verificación y mantenimiento. Además, la ambigüedad en la especificación, la incertidumbre técnica, la obsolescencia técnica y la tecnología "de punta" también son factores de riesgo. Los riesgos técnicos ocurren porque el problema es más difícil de resolver de lo que se creía.
- Los riesgos empresariales amenazan la viabilidad del software que se va a construir y con frecuencia ponen en peligro el proyecto o el producto. Los candidatos para los cinco principales riesgos empresariales son: 1) construir un producto o sistema excelente que realmente no se quiere (riesgo de mercado), 2) construir un producto que ya no encaje en la estrategia empresarial global de la compañía (riesgo estratégico), 3) construir un producto que el equipo de ventas no sabe cómo vender (riesgo de ventas), 4) perder el apoyo de los administradores debido a un cambio en el enfoque o en el personal (riesgo administrativo) y 5) perder apoyo presupuestal o de personal (riesgos presupuestales).

Identificación de riesgos

Un método para la identificación de riesgos es crear un listado enfocado a alguna de las siguientes categorías listadas por (Pressman, R. S. 2010).

- Tamaño del producto: riesgos asociados con el tamaño global del software que se va a construir o a modificar.

- Impacto empresarial: riesgos asociados con restricciones impuestas por la administración o por el mercado.
- Características de los participantes: riesgos asociados con la sofisticación de los participantes y con la habilidad de los desarrolladores para comunicarse con los participantes en forma oportuna.
- Definición del proceso: riesgos asociados con el grado en el que se definió el proceso de software y la manera como se sigue por parte de la organización desarrolladora.
- Entorno de desarrollo: riesgos asociados con la disponibilidad y calidad de las herramientas por usar para construir el producto.
- Tecnología por construir: riesgos asociados con la complejidad del sistema que se va a construir y con lo "novedoso" de la tecnología que se incluye en el sistema.
- Tamaño y experiencia del personal: riesgos asociados con la experiencia técnica y de proyecto global de los ingenieros de software que harán el trabajo.

Estimación del Riesgo

Para resumir este punto podemos referirnos a la Figura 10.

Componentes / Categoría		Rendimiento	Apoyo	Costo	Calendario
Catastrófico	1	La falla para satisfacer el requisito resultaría en fallo en la misión		La falla da como resultado aumento de costos y demoras en el calendario, con valores esperados en exceso de US$500K	
	2	Degradación significativa para no lograr el rendimiento técnico	Software que no responde o no puede tener apoyo	Significativos recortes financieros, probable agotamiento de presupuesto	IOC inalcanzable
Crítico	1	Falla para satisfacer el requisito degradaría el rendimiento del sistema hasta un punto donde el éxito de la misión sería cuestionable		La falla da como resultado demoras operativas y/o aumento de costos con valor esperado de US$100K a US$500K	
	2	Cierta reducción en rendimiento técnico	Demoras menores en modificaciones de software	Cierto recorte de recursos financieros, posible agotamiento	Posible deterioro en IOC
Marginal	1	Falla para satisfacer los requisitos resultaría en degradación de misión secundaria		Costos, impactos y/o calendario recuperable se deterioran con valor esperado de US$1K a US$100K	
	2	Reducción mínima a pequeña en rendimiento técnico	Apoyo de software receptivo	Suficientes recursos financieros	Calendario realista, alcanzable
Despreciable	1	Falla para satisfacer requisitos crearía inconvenientes o impacto no operativo		Error da como resultado costo menor y/o impacto en calendario con valor esperado de menos de US$1K	
	2	No reducción en rendimiento técnico	Software fácilmente soportable	Posible subejercicio de presupuesto	IOC alcanzable con facilidad

Figura 10 - Estimación del riesgo (Pressman, R. S. 2010)

Tabla de riesgos

Proporciona una técnica simple de proyección de riesgos, la estructura básica la podemos observar en la Figura 11:

Riesgos	Categoría	Probabilidad	Impacto	RMMM
Estimación de tamaño puede ser significativamente baja	PS	60%	2	
Mayor número de usuarios que el planificado	PS	30%	3	
Menos reuso que el planificado	PS	70%	2	
Usuarios finales que se resisten al sistema	BU	40%	3	
Fecha de entrega será apretada	BU	50%	2	
Pérdida de fondos	CU	40%	1	
Cliente cambiará requisitos	PS	80%	2	
Tecnología no satisfará las expectativas	TE	30%	1	
Falta de capacitación en herramientas	DE	80%	3	
Personal inexperto	ST	30%	2	
Alta rotación de personal	ST	60%	2	
Σ				
Σ				
Σ				

Figura 11 - Ejemplo de tabla de riesgos (Pressman, R. S. 2010)

Conclusiones

Al considerar los componentes del análisis y diseño para el desarrollo de software resumidos en este capítulo será posible adaptarlos al uso y aplicación dentro de la metodología de desarrollo Scrum.

Los componentes utilizados dentro de la metodología de desarrollo partiendo desde el modelo de ciclo de vida es el cascada ya que se realizarán diferentes prototipos o entregables para cada una de las secciones del aplicativo y mediante es modelo es posible recorrer todos los pasos para cada una de ellas; los componentes de modelado de negocio analizados son los recomendados dentro de la metodología de desarrollo por su facilidad de uso y amplia documentación. La documentación se generará mediante la plataforma Javadoc por su documentación y adaptabilidad. Por último, la gestión de riesgos será en base a lo mencionado por (Pressman, R. S. 2010) ya que se cuenta con la teoría y ejemplos necesarios para adaptarlos a la metodología.

Capítulo 5: Análisis de herramientas de desarrollo de software y almacenamiento de datos – Etapa 3(Análisis)

Introducción

En este capítulo se analiza y define todas las herramientas necesarias para el desarrollo del software planteado en este trabajo, partiendo desde el lenguaje de programación, el Framework necesario para construir la estructura en ese lenguaje, el almacén de datos necesario que se adapte a las necesidades del análisis, las herramientas Web necesarias de modelo de procesos que permitan ser lo suficientemente adaptables para manejar diferentes ambientes de desarrollo y finalmente lo referente al análisis de datos para su uso en proyección y análisis de datos.

Lenguajes de programación

Un lenguaje de programación es una notación o conjunto de símbolos y caracteres combinados entre sí de acuerdo con una sintaxis ya definida que posibilita la transmisión de instrucciones a la CPU (Catalinas, E. Q. 2002). Los lenguajes de programación según Benedetto et al. (2015) se clasifican en aquellos que permitan la posibilidad de manipular objetos persistentes, así como lenguajes de manipulación de datos en bases de datos que, además, provean facilidades para el manejo de interfaces gráficas, y los analiza desde un punto de vista de rendimiento. También hace referencia al desarrollo mencionando que cada uno de estos lenguajes, resulta más o menos adecuado que otro para producir software con ciertas cualidades, en función de las características que ellos presentan.

De acuerdo a estas bases conceptuales y al análisis comparativo entre los lenguajes de programación y su productividad informática propuesto por (Dávila, M. R. 2017) en el cual propone a Java y Javascript como lenguajes que han evolucionado e implementado nuevas características a lo largo de los años, proporcionando ventajas como son el soporte, comunidad y la posibilidad de vinculación de herramientas adicionales. Gran parte del análisis que el autor realiza fue tomando como base paginas como GitHub y TIOBE para sus recomendaciones como los describe con más detalle (Benedetto et al., 2015).

Frameworks de desarrollo

En general, el término Framework, hace referencia a una estructura software compuesta de secciones personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un Framework se puede considerar como una aplicación genérica incompleta y configurable a la que es posible añadir las últimas piezas para construir una aplicación concreta (Gutiérrez, J. J. 2014).

Las ventajas de usar un Framework a la hora de realizar un proyecto son diversas, entre otras, disminuye el tiempo de creación de las aplicaciones, facilita el mantenimiento del código y hace uso de patrones de desarrollo (Pantoja, L., & Pardo, C. 2016). El patrón más utilizado por casi todos los Frameworks es el conocido como Modelo Vista Controlador (MVC). El patrón MVC puede implementarse sin la necesidad de utilizar un Framework; no obstante, y

a diferencia de aplicarlo de forma manual, el Framework obliga al desarrollador a utilizarlo, creando de esta forma un código mucho más robusto (Pantoja, L., & Pardo, C. 2016).

Existe un gran número de Frameworks que pueden utilizarse; el abanico de opciones es tan grande que en ocasiones es bastante complejo decidir cuál es el más adecuado de acuerdo con el tipo de proyecto o producto que se desea desarrollar (Pantoja, L., & Pardo, C. 2016). Elegir un Framework de desarrollo de software no es una tarea sencilla. Se deben considerar diferentes variables (Pantoja, L., & Pardo, C. 2016). Entre estas, la más importante, es el lenguaje de programación por utilizar. A partir de este, es posible tener en cuenta otras variables como: características del proyecto, conocimientos previos del equipo en el uso de la tecnología, existencia de proyectos que hayan sido desarrollados con el Framework por elegir, nivel de madurez del Framework, soporte, curva de aprendizaje, tipo de licencia (libre o de pago), entre otras (Pantoja, L., & Pardo, C. 2016).

Un análisis detallado presentado por (Pantoja, L., & Pardo, C. 2016) nos presenta las características de diferentes Frameworks de desarrollo MVC considerados por el autor como los más populares en el mercado, los cuales son evaluados con relación a factores como el esfuerzo de aprendizaje, productividad y tamaño (referente a la cantidad de líneas de código producidas). Pantoja y Pardo (2016) concluyen su análisis con una evaluación global de los Frameworks de los cuales JSF en la posición 4 de manera general y número 1 en el lenguaje Java que servirá de punto de partida para este proyecto.

Bases de Datos

Las bases de datos NoSQL o No Relacionales son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación, es decir, con el conocido modelo relacional (Croce, L. D., & Salinas, J. A. 2016). El término NoSQL, independientemente del concepto "Base de datos" es interpretado comúnmente como "Not Only SQL" (no solo SQL) (Croce, L. D., & Salinas, J. A. 2016). También se lo puede conocer como una combinación de dos palabras: "No" y "SQL"; es decir, el término no posee un único significado (Croce, L. D., & Salinas, J. A. 2016).

Las bases de datos relacionales, por su parte, se basan en el uso de tablas (llamadas también relaciones). Las tablas se representan gráficamente como una estructura rectangular formada por filas y columnas. Cada columna almacena información sobre una propiedad determinada de la tabla (llamada también atributo), nombre, cédula, apellidos, edad, etc. Cada fila (llamada también tupla) posee una ocurrencia o ejemplar de la instancia o relación representada por la tabla (Sánchez, J. 2004).

Para el desarrollo de este proyecto se plantearon varias alternativas de motores de bases de datos enfocados dentro de la clasificación de software libre, principalmente por temas de licenciamiento como: MySQL, PostgreSQL, Cassandra y MongoDB. Estas decisiones se dieron tomando en cuenta el análisis realizado por Córdova y Cuzco (2013). La elección de las bases de datos NoSQL se realizaron en base a investigaciones realizadas por (Fiannaca, A. J., and Huang, J. 2015) para el manejo de grandes cantidades de datos.

El motor de base de datos escogido para el desarrollo del proyecto corresponde a PostgreSQL Para esto se consideró un análisis proporcionado por (Aránega Hernández, A. 2015) donde realiza un análisis detallado tanto de PostgreSQL y MySQL, si bien es cierto la conclusión en dicho documento es MySQL, en el presente trabajo se toma en cuenta puntos referentes a algunas ventajas como son: Multiplataforma, bloqueo a nivel de registro y la integridad referencial las cuales son consideradas de peso al momento de realizar gran cantidad de cálculos. Este mismo autor coloca estos puntos como características ganadoras de PostreSQL, sobre su rival en este estudio.

BPM

BPMN vs IDEF

Dentro del análisis presentado por Abdelhady (2015) en el que realiza un comparativo entre los modelos BPMN e IDEF describe ventajas y desventajas de cada uno de ellos; las cuales, de acuerdo al proyecto que se pretende desarrollar se tomaron en cuenta las ventajas de BPMN sobre IDEF. Por ejemplo, El modelo BPMN posee más detalle que el modelo IDEF, ayuda a identificar instantáneamente los problemas en la secuencia o asignación de actividades a los artistas, intérpretes o ejecutantes (Abdelhady, I. A. 2015). BPMN usa una notación de "*swim-lane*" (carril de natación), mostrando las actividades dentro de los *swimlanes* que indican la actividad de cada ejecutante, lo que da una visión clara sobre el flujo de trabajo y "lo que está pasando ¿en? ¿Y quién hace qué?" para modelar el análisis (Abdelhady, I. A. 2015). BPMN es un modelo más estructurado, compuesto, coherente y consistente, capaz de ejecutar y cambiar continuamente los procesos de negocios de extremo a extremo (Abdelhady, I. A. 2015).

Una vez establecido el modelo BPMN es imperativo generar un modelo automatizable de documentación de procesos en los que la empresa podrá estandarizarlos y automatizarlos. La Tabla 3, proporciona un formato guía para documentar la automatización de procesos. Es labor de los encargados del proyecto definir los campos necesarios dentro de la documentación para personalizar la estructura de la tabla a las necesidades específicas.

Introducción o motivación del manual
Autor del modelo/responsable de la documentación
Objeto del manual
Definición de los elementos del BPMN
Marco normativo
Definiciones
Estructura organizacional
Mapa de procesos
Clave o código, versión y fecha de generación de documento
Nombre del proceso
Responsable o dueño del proceso
Objetivo del proceso
Descripción o alcance del proceso

<table>
<tr><td colspan="5">Marco normativo especifico del proceso</td></tr>
<tr><td colspan="5">Recinto seguro</td></tr>
<tr><td colspan="5">Descripción de reglas de negocio asociadas al proceso</td></tr>
<tr><td colspan="3">Diagrama o modelo en BPMN</td><td rowspan="13">Aplica para procesos y subprocesos o procedimientos</td><td rowspan="13">Elementos del BPMN</td></tr>
<tr><td colspan="2">Nombre, código y descripción de la operatividad de los eventos</td><td>Eventos</td></tr>
<tr><td colspan="2">Nombre, código y descripción de la operatividad de las tareas personales, entradas salidas, reglas de negocio (si aplican), Formatos, responsable/roles o ejecutores</td><td rowspan="12">Tareas o actividades</td></tr>
<tr><td colspan="2">Tiempo de ejecución de las tareas personales</td></tr>
<tr><td>Nombre del Formulario/Formatos</td><td rowspan="4">Documentos asociados a las tareas</td></tr>
<tr><td>Descripción del Formulario/Formato - Nombre y tipo de campo</td></tr>
<tr><td>Descripción del contenido de inFormación del campo</td></tr>
<tr><td>Documento automático asociado generado en una tarea personal</td></tr>
<tr><td>Preparada o modifcada por:</td><td rowspan="3">Registro de modifcaciones</td></tr>
<tr><td>Fecha y hora de preparación o última modifcación</td></tr>
<tr><td>Calendario</td></tr>
<tr><td colspan="2">Nombre, código, tipo, tipo de Función y descripción de la operatividad de las tareas de sistema (notifcadora, invocadora, traspasadora, desviadora, ejecutora, entre otras)</td></tr>
<tr><td colspan="2">Documento automático asociado por una tarea de sistema</td></tr>
<tr><td colspan="2">Nombre, código, género o tipo y descripción de la operatividad de las compuertas</td><td>Compuertas o gateways</td></tr>
<tr><td colspan="5">Key Performance Indicator (KPI) del proceso</td></tr>
</table>

Tabla 3 - Modelo de automatización de procesos (López Supelano, K. 2015)

Análisis de datos

Dentro de esta sección se pretende generar los conocimientos teóricos necesarios para generar modelo para la inspección y transformación de datos ingresados dentro de la plataforma, permitiendo tomar decisiones de manera más efectiva.

Simulación Montecarlo

La simulación Monte Carlo es una técnica matemática computarizada que permite tener en cuenta el riesgo en análisis cuantitativos y tomas de decisiones. Esta técnica es utilizada en diferentes campos como nos menciona (Palisade 2000), por esta razón se generó un modelo de datos para el caso específico de manufactura, en el que se pretende realizar un análisis y tomar decisiones de datos referentes a nuevos productos, extensión de líneas de producto y

proyectos de ahorro en manufactura. La herramienta analizada para el uso de esta técnica es @Risk creada por Palisade ya que permite la utilización de los datos dentro de una hoja de cálculo.

Análisis del árbol de decisiones

Los árboles de decisión son diagramas cuantitativos con nodos y ramas que representan diferentes posibles rutas de decisión y sucesos aleatorios. Esto ayudará a identificar y calcular el valor de todas las posibles alternativas, para poder seleccionar con confianza la mejor opción (Palisade . 2000), la herramienta necesaria para ejecutar estos procesos corresponde a Presicion Tree de la empresa Palisade.

Optimización

De acuerdo a Palisade (2000) se han proporcionado varias definiciones con respecto a la optimización como por ejemplo encontrar la mejor solución al problema, el cual tiene muchas soluciones, ajustar variables de decisión para llegar al mejor resultado, calculado por una función objetivo, estadísticas vs. Condiciones determinantes (restricciones) y para finalizar variación y "ruido"

Informes y Reportes

De acuerdo al trabajo presentado por (Parra, V. M., et al., 2016) en el que se presenta un análisis experimental de la comparación de dos de los mejores programas de *Bussiness Intelligente (*BI) de código abierto presentes en el mercado: Pentaho y Jaspersoft. Su punto focal para este estudio lo divide en dos secciones la primera corresponde al procesamiento de datos y a los procesos denominados: extracción, transformación y carga con su representación en inglés *Extract Transform and Load* (ETL). La segunda sección hace referencia a la capacidad para realizar procesos de generación de informes midiendo su desempeño usando sistemas de álgebra computacional.

Las conclusiones que presenta (Parra, V. M., et al., 2016) brindaron la oportunidad de decidir Jaspersoft sobre Pentaho ya que posee Java / Perl nativo, permite desarrollar aplicaciones Web con Java j2EE 100% extensibles, adaptables y personalizables. Dentro de esta herramienta las modificaciones en la gestión se encuentran controladas perfectamente permitiendo realizarlo todo a través de la misma aplicación Web. El sistema Jaspersoft integra todos los recursos de información en una sola plataforma operativa; los informes Ad-hoc del editor y el comando Ad-hoc del Editor de cuadros son resueltos de una mejor manera. Finalmente, los informes son rápidos; posee una muy buena interfaz, con buena flexibilidad y potencia además de sencillo, intuitivo y fácil de usar.

Métodos de documentación sugeridos

Los javadoc son comentarios semiestructurados escritos en lenguaje natural que especifican las características de un elemento de código, como puede ser un método, una clase o un paquete. Esta herramienta ha sido recomendada ya que es la que se encuentra adaptada de manera simple dentro de los *Integrated Development Environment*(IDE´s) más populares del lenguaje como son: Eclipse, que es uno de los más famosos para el uso de lenguaje Java,

pudiendo integrar gran cantidad de paquetes y *plugins* por defecto y Netbeans, igualmente diseñado mayoritariamente para soportar el lenguaje Java sin embargo esta más enfocado a soportar aplicaciones basadas en MVC. Un ejemplo de este método se encuentra en la Figura 12.

```
1    /**
2     * Creates a new aspect for the given {@code method}.
3     *
4     * @param method method for which an aspect will be created
5     * @param aspectName name of the file where the newly created aspect is saved
6     * @throws NullPointerException if {@code method} or {@code aspectName} is null
7     */
8     private static void createAspect(DocumentedMethod method, String aspectName)
```

Figura 12 - Ejemplo de utilización de javadoc

Conclusiones

Este capítulo presenta un análisis comparativo de diferentes herramientas necesarias para el desarrollo del software, en base a las necesidades de desarrollo y a la aplicación de TDABC en empresas de producción. Para este fin, como las más opcionadas para el cumplimiento de los objetivos planteados se han escogido las siguientes herramientas: en el caso de lenguaje de programación se menciona Java; el Framework a usar y que complementa el lenguaje antes mencionado es JSF; el almacenamiento de datos se realizará en una base de datos PostgreSQL; para el manejo de los procesos de negocio se escogió BPMN; y, finalmente, para el análisis de datos y optimización se usará la herramienta @Risk de la empresa Palisade.

Capítulo 6: Alineación del desarrollo de software con el método de costeo TDABC en empresas de producción – Etapa 3(Análisis)

Introducción

Para entender la alineación del desarrollo de software con métodos de costeo fue imperativo partir del entendimiento de la metodología TDABC y su implementación. Luego de este entendimiento se procedió a analizar algunos planteamientos relacionados con la vinculación de TDABC al software de Planificación de Recursos Empresariales o por sus siglas en ingles ERP. Para finalizar se realizó un análisis de un módulo específico enfocado al área de bibliotecas con el objetivo de generar los requerimientos básicos de la metodología TDABC, además de la estructura tecnológica del aplicativo como punto de partida del nuevo sistema.

TD-ABC-D

De acuerdo al análisis presentado por Cabrera y Ordoñez (Cabrera Encalada, P., & Ordoñez Parra, C. 2012) y Siguenza-Guzman (Siguenza-Guzman et al., 2014) el módulo de "*Time-Driven Activity-Based Costing Software for libraries*" denominado por los autores como TD-ABC-D pretende brindar una herramienta Web de análisis en procesos de bibliotecas mediante el uso de la metodología TDABC. Las características principales de este módulo son la facilidad de uso para generar flujos de procesos, interfaz de usuario amigable y la posibilidad de información adecuada para el análisis de costos dentro de la biblioteca, todo esto de una forma rápida y amigable.

Módulo TDABC y su uso en bibliotecas

Un planteamiento importante que ha sido la base de este trabajo es el planteamiento y desarrollo de un módulo TDABC en el uso de bibliotecas presentado por Cabrera y Ordoñez (2012) denominado TD-ABC-D. El objetivo radicó en adaptarlo a un sistema con tecnología ISIS (Integrated Set for Information System, o en español, Conjunto Integrado de Sistemas de Información); la estructura de este que incluye módulos o funcionalidades relativamente independientes con la única finalidad de obtener una herramienta integrada para la gestión de bibliotecas. Este estudio fue realizado en base a investigaciones previas realizadas por (Sigüenza Guzmán, et al. 2014), en el que se plantea la implementación de la metodología TDABC en el proceso de catálogo de bibliotecas demostrando la aplicabilidad y usabilidad de TDABC para realizar análisis de costos de procesos en la sección de catálogos.

Software ERP y manejo de metodologías TDABC

Existe otro planteamiento, presentado por Ai-Min, et al. (2016), que usa sistemas ERP en la nube y el sistema de costeo TDABC de forma conjunta. La idea es colocar las ventajas tanto del ERP y el uso de sistemas en la nube obteniendo mejoras de la capacidad, velocidad e información de costos logísticos en las empresas. Discriminar el costo logístico que depende en gran medida del nivel de eficiencia de tiempo.

Conclusiones

Del análisis realizado en este capítulo se desprende que el uso de la metodología TDABC dentro de un software se basa en varias investigaciones y sobre todo de un prototipo denominado TD-ABC-D. Si bien es cierto en el mercado existen varios sistemas que permiten el cálculo y manejo de los datos no se encuentra información sobre implementaciones que permitan manejar el flujo completo de un sistema de producción. Es decir, adaptar las características y necesidades de la empresa a las metodologías TDABC mediante un software especialmente diseñado para los requerimientos y especificaciones en el sector productivo.

Capítulo 7: Propuesta de Análisis y Diseño – Etapa 4(Propuesta)

Introducción

En el mundo de desarrollo de software actual, cada vez son mayores las restricciones de tiempo durante el análisis, desarrollo y puesta en marcha de sistemas software. Los cambios en las reglas de negocio, modificaciones de requerimientos y los problemas para mantener sistemas obsoletos, así como el entorno cambiante en el que se mueven las empresas de producción, hacen que la construcción y el mantenimiento de sistemas evolucionen a gran velocidad para adecuarse a los nuevos requisitos.

Metodología para proyectos de desarrollo de software

Para el desarrollo de proyectos de software en empresas del sector productivo específicamente para el desarrollo de la propuesta TDABC se recomienda el uso de metodologías ágiles Scrum, ya que es un proceso metodológico que sirve para administrar y controlar la construcción de software de manera más efectiva. El desarrollo se realiza en forma iterativa utilizando ciclos cortos de construcción de manera repetitiva e incremental. Generando una secuencia de prototipos que permiten el desarrollo de entregables rápidos generando la retroalimentación de cada uno de los interesados y su aprobación. Scrum se enfoca en la priorización del trabajo en función del valor que tenga para la empresa, al maximizar la utilidad y el retorno de la inversión. Estos puntos son importantes en empresas de producción ya que el software debe ser capaz de insertarse de manera completamente transparente al proceso productivo (Cadavid, A. N., et al., 2013).

Scrum define tres roles: el Scrum master, el dueño del producto y el equipo de desarrollo. El Scrum master tiene como función asegurar que el equipo está adoptando la metodología, sus prácticas, valores y normas; es el líder del equipo, pero no gestiona el desarrollo. El dueño del producto es una sola persona y representa a los interesados, es el responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo; tiene entre sus funciones gestionar la lista ordenada de funcionalidades requeridas o Product Backlog. El equipo de desarrollo, por su parte, tiene como responsabilidad convertir lo que el cliente quiere, el Product Backlog, en iteraciones funcionales del producto; el equipo de desarrollo no tiene jerarquías, todos sus miembros tienen el mismo nivel y cargo: desarrollador. El tamaño óptimo del equipo está entre tres y nueve personas (Cadavid, A. N., et al., 2013).

Scrum define un evento principal o Sprint con una secuencia de fases (visualización, preparación, sprints y propuesta) y que se compone de los siguientes elementos: reunión de planeación del Sprint, Daily Scrum, trabajo de desarrollo, revisión del Sprint y retrospectiva del Sprint (Cadavid, A. N., et al., 2013). Las fases del sprint representadas de manera gráfica las podemos observar en la Figura 13.

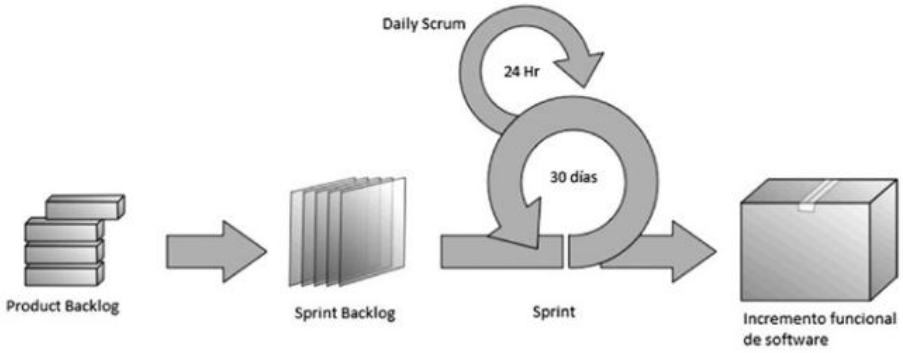

Figura 13 - Fases de un sprint (Cadavid, A. N., et al., 2013)

En la reunión de Planeación del Sprint se define su plan de trabajo: qué se va a entregar y cómo se logrará. Es decir, el diseño del sistema y la estimación de cantidad de trabajo. Esta actividad dura ocho horas para un Sprint de un mes. Si el Sprint tiene una duración menor, se asigna el tiempo de manera proporcional (Cadavid, A. N., et al., 2013).

El Daily Scrum es un evento del equipo de desarrollo de quince minutos, que se realiza cada día con el fin de explicar lo que se ha alcanzado desde la última reunión; lo que se hará antes de la siguiente; y los obstáculos que se han presentado. Este evento se desarrolla mediante una reunión que normalmente es sostenida de pie con los participantes reunidos formando un círculo, esto, para evitar que la discusión se extienda. La Revisión del Sprint ocurre al final del Sprint y su duración es de cuatro horas para un proyecto de un mes (o una proporción de ese tiempo si la duración es menor). En esta etapa: el dueño del proyecto revisa lo que se hizo, identifica lo que no se hizo y discute acerca del Product Backlog; el equipo de desarrollo cuenta los problemas que encontró y la manera en que fueron resueltos, y muestra el producto y su funcionamiento. Esta reunión es de gran importancia para los siguientes Sprints (Cadavid, A. N., et al., 2013).

El Product Backlog es una lista ordenada por valor, riesgo, prioridad y necesidad de los requerimientos que el dueño del producto define, actualiza y ordena. La lista tiene como característica particular que nunca está terminada, pues evoluciona durante el desarrollo del proyecto. El Sprint Backlog es un subconjunto de ítems del Product Backlog y el plan para realizar en el Incremento del producto. Debido a que el Product backlog está organizado por prioridad, el Sprint backlog es construido con los requerimientos más prioritarios del Product backlog y con aquellos que quedaron por resolver en el Sprint anterior. Una vez construido, el Sprint backlog debe ser aceptado por el equipo de desarrollo, pertenece a éste y solo puede ser modificado por él. Requerimientos adicionales deben ser incluidos en el Product backlog y desarrollados en el siguiente Sprint, si su prioridad así lo indica. El Monitoreo de Progreso consiste en la suma del trabajo que falta por realizar en el Sprint. Tiene como característica que se puede dar en cualquier momento, lo que le permite al dueño del producto evaluar el progreso del desarrollo. Para que esto sea posible, los integrantes del equipo actualizan constantemente el estado de los requerimientos que tienen asignados indicando cuánto consideran que les falta por terminar. El Incremento es la suma de todos los ítems terminados

en el Sprint backlog. Si hay ítems incompletos deben ser devueltos al Product backlog con una prioridad alta para que sean incluidos en el siguiente Sprint. Se considera que un ítem está terminado si es funcional. La suma de ítems terminados es el producto a entregar (Cadavid, A. N., et al., 2013).

Al estar diseñado para adaptarse a los cambios en los requerimientos tanto funcionales como no funcionales, permite que el producto pueda adaptarse en tiempo real a las necesidades de la empresa. Scrum podría definirse como un Framework para colaboración de equipo efectiva en el desarrollo de productos complejos resumido en la Figura 14.

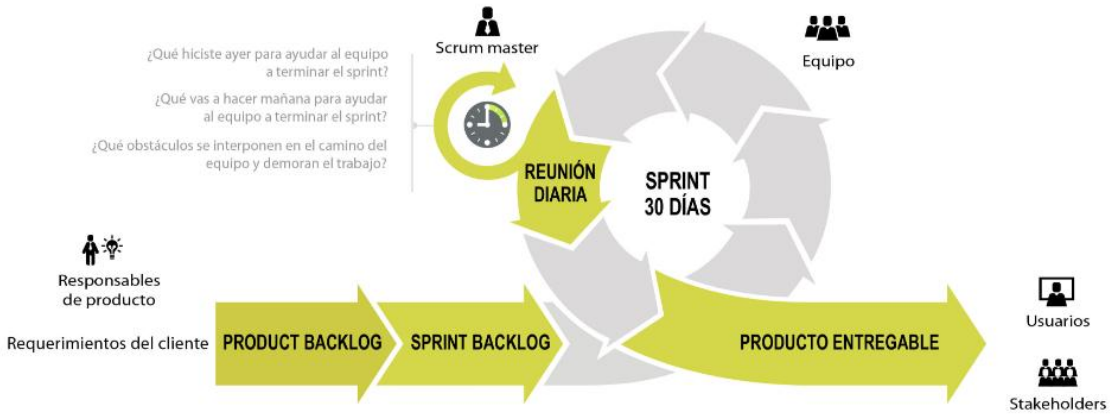

Figura 14 - SCRUM Framework (SCRUM.org, 2017)

El resultado del análisis de todos estos conceptos es la adaptación de la metodología a las necesidades específicas, por esta razón en la Figura 15 se visualiza los 4 pasos de la metodología Scrum cada una con sus actividades y productos resultantes. La sección de visualización realizara las siguientes actividades: análisis de requerimientos, módulo TD-ABC-D y diagnostico operativo en la empresa Motsur y como resultados la documentación de requerimientos y el modelo conceptual de datos. La segunda fase que para el caso de estudio se denominara preparación contiene: definición de funcionalidad, planeación del trabajo, determinación de herramientas, arquitectura y el esquema de bases de datos y como productos resultantes los casos de uso, productbacklog, ambiente de desarrollo, arquitectura de la plataforma, estructura de bases de datos y el plan de validación. La tercera etapa o Sprints contendrá el desarrollo de cada uno de los componentes o entregables de la aplicación y sus productos resultantes será los entregables funcionales para el desarrollo de pruebas. La última etapa corresponde a la estabilización en donde se ejecutará un plan de validación y el producto resultante será la plataforma funcional.

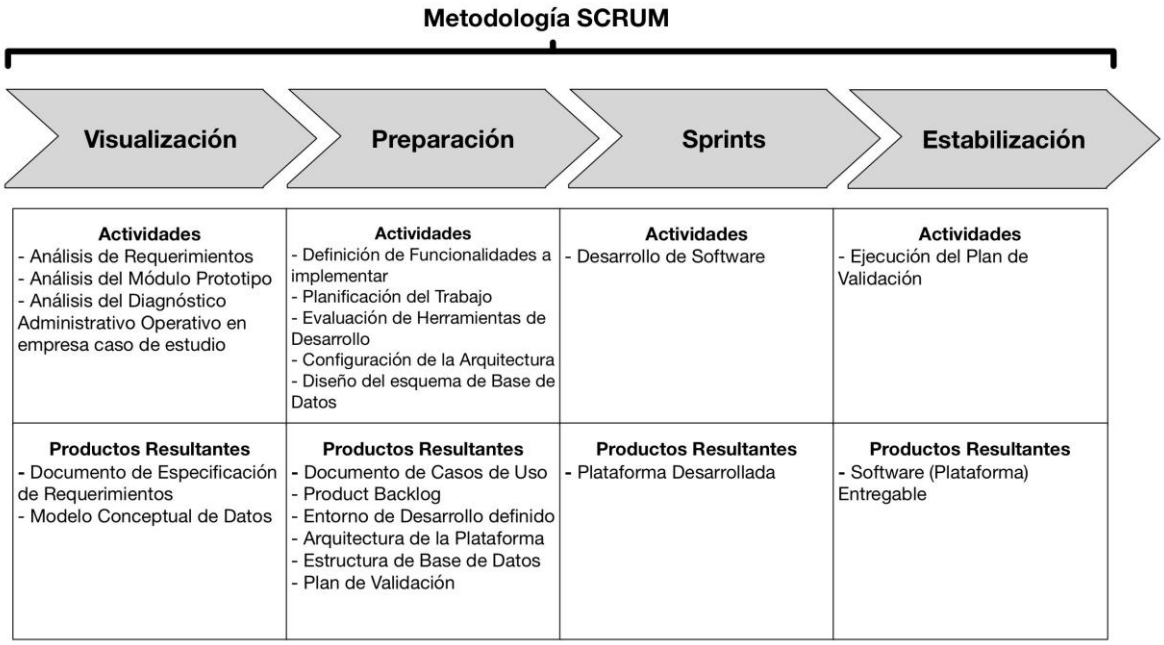

Figura 15 - Esquema de metodología Scrum empresa Motsur (autoria propia)

Componentes de análisis y diseño recomendados

Análisis de requerimientos

Para la realización del análisis de requerimientos se tomaron en cuenta tres puntos principales: el documento de levantamiento de procesos proporcionado por (Andradre, E. y Elizalde, B. 2017), documentación y el formato de mapa de procesos según el levantamiento de riesgos, así como el prototipo proporcionado por (Cabrera, Ordoñez, 2012), explicado en el Capítulo 7. El resultado de este análisis se puede observar en las Figuras 16 – 17. La primera Figura corresponde al formato necesario para la recopilación de los procesos, la parte principal que se tomó en cuenta para el análisis de los requerimientos corresponde a la interacción que tienen cada uno de los campos de este formato; brindándonos una visión de la estructura de la base de datos y la forma de interactuar entre cada uno de los datos. La Figura 17 proporciona los requisitos necesarios de cada una de las actividades, así como las políticas que se manejan para cada una de ellas.

MOTSUR Proyecto DIUC-Concurso XV 2017 "Modelo de gestión para la optimizacion de procesos y costos en la industria de ensamblaje"		MANUAL DE PROCESOS	FECHA:		INICIO PROCESO	
			FECHA DE REVISION:		FIN PROCESO	
DEPARTAMENTO	CODIGO	ENTRADAS	SALIDAS SI		SALIDAS NO	CONDICION
PROCESO	CODIGO	ENTRADAS	SALIDAS SI		SALIDAS NO	CONDICION
RECURSOS MATERIALES						
UNIDAD RESPONSABLE:		Compras	RESPONSABLE:			
MISION DEL SUBPROCESO DOCUMENTAL						
MISION DEL SUBPROCESO PRACTICO						
ACTIVIDADES						
INICIO DE LA TOMA DE DATOS					HORA	

CODIGO	DESCRIPCION	RECURSOS HUMANOS	RECURSOS TECNOLOGICOS	TIEMPOS(MIN)	COSTOS	HORA DEL LEVANTAMIENTO	FECHA DEL LEVANTAMIENTO	SALIDAS SI	SALIDAS NO	CONDICION
	Tomar fotos del contenedor									

FIN DE LA TOMA DE DATOS		Hora:	
DOCUMENTOS APLICABLES			

Figura 16 - Formato de levantamiento de procesos (referencia)

MOTSUR Cía. Ltda.

MAPA DE PROCESOS SEGÚN LEVANTAMIENTO DE RIESGOS

Mapa General de Procesos / .....

Área: Fecha de Revisión:

Elaborado Por: Aprobado Por:

NOMBRE DEL PROCESO

SUBPROCESO	ACTIVIDAD	Observaciones	Formas/Formularios/Formatos	Requisitos	MANUAL /POLITICAS	Reuniones	Frecuencia
F1	A1						
	A2						
	A3						
	A4						
F2	A1						
	A2						
	A3						
	A4						
F3	A1						
	A2						
	A3						
	A4						
F4	A1						
	A2						
	A3						
	A4						

Figura 17 - Mapa de procesos según levantamiento de riesgos (referencia)

Como resultado de los tres puntos mencionados anteriormente en las Tablas 4 y 5 se describen unos ejemplos con la especificación de requerimientos tanto funcionales como no funcionales. Este formato contiene la prioridad de la ubicación en el sprint para priorización del desarrollo y los criterios de aceptación, los cuales serán socializados y aprobados por el cliente. La codificación RFxx nos indica que es un requerimiento funcional, por otro lado, la prioridad nos ayudara a gestionar los requerimientos dentro de los sprint para el futuro uso de los desarrolladores. Cada uno de estos requerimientos se crearán de forma modular capaz de adaptar a la aplicación general fácilmente en cualquiera de las etapas del desarrollo.

Especificación de Requerimiento	
Código:	RF01
Nombre:	Administración de Procesos
Prioridad:	Alta
Descripción:	La Plataforma debe permitir crear, modificar, visualizar y eliminar un proceso. Para esto el usuario tendrá que asignar actividades al proceso y establecer relaciones de orden y condicionales entre cada una de las actividades.
Criterio de aceptación:	• Generación de flujos de secuencia entre subprocesos y actividades • Visualización correcta de flujos del proceso
Producto resultante:	• Módulo de Administración de Procesos (Procesos almacenados en la BD)

Tabla 4 - Especificación de requerimiento: Administración de procesos

Especificación de Requerimiento	
Código:	RF02
Nombre:	Administración de Subprocesos
Prioridad:	Alta
Solicitado por:	Gestor de Procesos
Descripción:	La Plataforma debe permitir agrupar las actividades de un proceso en subprocesos. Para esto el usuario podrá crear, modificar, visualizar y eliminar un subproceso.
Criterio de aceptación:	• Asignación de subprocesos dentro de un proceso • Generación de flujos de secuencia entre actividades
Producto resultante:	• Módulo de Administración de Subprocesos (Subprocesos almacenados en la BD)

Tabla 5 - Especificación de requerimiento: Administración de subprocesos

Documentación y dependencia de los casos de uso

El propósito principal de esta sección es ilustrar el funcionamiento de la plataforma, el objetivo principal de los casos de uso es indicar si todos lo requerimiento se cumplen mediante alguna de las funcionalidades descritas, además de brindar información importante para las estimaciones en el desarrollo.

Como ejemplo se describe un caso de uso y la interacción del mismo con la plataforma por medio de una secuencia de pasos y conexiones, la Figura 18 nos proporciona una idea del caso de uso de "Gestión de Procesos" partiendo de los actores "Analista de Procesos/Administrador del sistema". El Analista de Procesos es la persona que podrá

acceder a las todas las funciones de la plataforma referentes a la gestión y análisis de procesos. No podrá utilizar funcionalidades referentes a aspectos de configuración de la plataforma y creación de usuarios. El Administrador del Sistema es la persona tiene acceso a todas las funcionalidades de la plataforma. La principal característica de este perfil de usuario es que podrá administrar las cuentas de usuarios del sistema y alterar la configuración del sistema en aspectos como el idioma, la moneda y otras funciones de configuración.

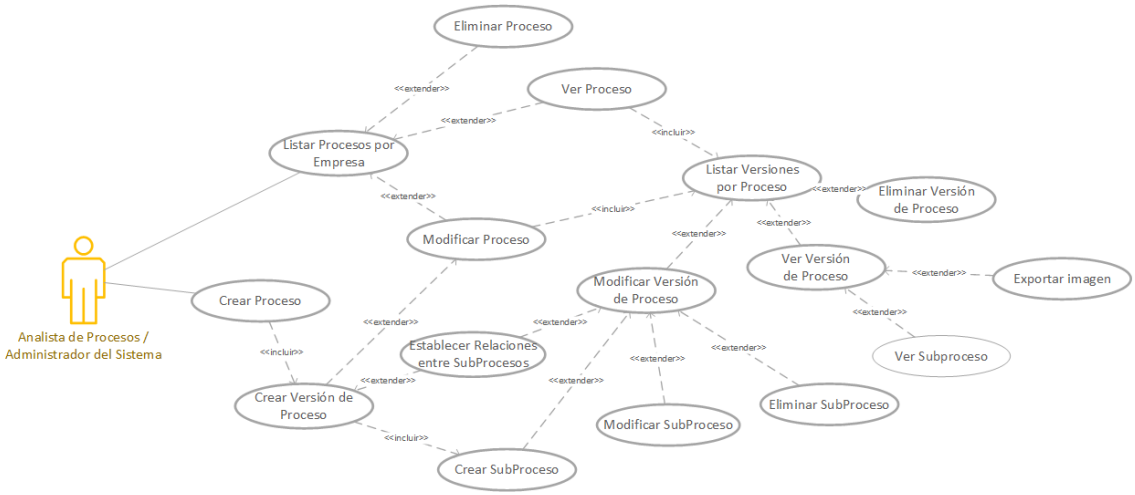

Figura 18 - Caso de uso de Gestión de procesos de la plataforma

Cada uno de los componentes de del caso de uso tendrá acompañado una tabla en la que se describe a detalle su funcionalidad como se muestra en la Tabla 6.

CDU-GP-001	Crear Proceso	
Prioridad:	Alta	
Descripción:	El usuario ingresa los datos de un proceso, creando subprocesos, asignando actividades a subprocesos y estableciendo relaciones de orden y condicionales entre las actividades y subprocesos.	
Disparador:	El usuario selecciona la opción de crear nuevo proceso	
Precondiciones:	- Datos de Empresas ingresados - Datos de Líneas de producción ingresados - Datos de Recursos Ingresados - Datos de Responsables Ingresados - Datos de Actividades Ingresados	
Actores:	- Analista de Procesos - Administrador de Sistema	
Secuencia normal:	**Paso**	**Acción**
	1	El usuario selecciona la opción de crear nuevo proceso
	2	La plataforma muestra el formulario de ingreso de un nuevo proceso y la opción de crear subprocesos

	3	El usuario ingresa los datos del proceso en los campos de texto del formulario. Los datos requeridos son: - Código - Nombre - Descripción - Misión - Línea de producción a la que pertenece
	4	El usuario ingresa subproceso(s) (“CDU-GP-011 Crear Subproceso”) y establece el orden y relaciones entre procesos (“CDU-GP-015 Establecer relaciones de orden y condición entre procesos”).
	5	El usuario selecciona la opción de Guardar proceso
	6	La plataforma verifica que todos los campos de datos requeridos hayan sido llenados
	7	La plataforma verifica que el proceso tiene al menos un subproceso creado
	8	La plataforma guarda los datos del proceso en la base de datos y muestra un mensaje de confirmación al usuario.
Post-condiciones:	Datos de proceso guardados en la Base de Datos	

Secuencias Alternas:	**Paso**	**Acción**	
	3	3.a	La plataforma no tiene información de ninguna línea de producción o de la línea a la que pertenece el proceso, y presenta la opción Crear Línea de Producción
		3.b	El usuario selecciona la opción Crear Línea de Producción
		3.c	La plataforma inicia el caso de uso “CDU-GE-011 Crear Línea de Producción”
	6	6.a	La plataforma encuentra que hay campos vacíos en el formulario
		6.b	La plataforma muestra un mensaje de error al usuario y marca los campos faltantes
		6.c	El usuario completa el llenado de los campos faltantes y regresa al paso 5 o cancela la creación del proceso.
	7	7.a	La plataforma encuentra que el proceso a guardar no tiene ningún subproceso creado
		7.b	La plataforma muestra una advertencia a usuario de que el proceso que desea guardar está vacío
		7.c	El usuario acepta la advertencia y continua con el guardado del proceso vacío o cancela la creación del proceso

Casos de Uso Asociados:	- CDU-GE-011 Crear Línea de Producción - CDU-GP-011 Crear Subproceso - CDU-GP-015 Establecer relaciones de orden y condición entre procesos

Tabla 6 - Detalle de funcionalidad por caso de uso

Modelo Conceptual de datos

Una vez analizado cada uno de los puntos mencionados en el análisis de requerimientos y los casos de uso se estableció el modelo conceptual de datos, con el cual se iniciará el desarrollo de la estructura de base de datos, así como la estructura MVC dentro de la plataforma. En la Figura 19 se encuentran cada uno de los componentes del modelo conceptual partiendo de las empresas clasificadas por grupo dando una apertura para el uso dentro de la aplicación en diferentes compañías. Todas estas empresas podrán mostrar información a partir de los departamentos los cuales están conectados con los procesos y a su vez con cada uno de los responsables. Los responsables forman el árbol y gestión de usuarios para acceso a la plataforma brindando la apertura de generar varias funciones y accesos para cada uno de ellos. La estructura medular corresponde a la conformada por las líneas, procesos, subprocesos, actividades y recursos, la cual permitirá dar el mantenimiento adecuado, acceso a los datos y organización de los mismo de todo el planteamiento de requerimientos de la metodología TDABC. El modelo conceptual fue planteado de tal forma que permita generar reportes gerenciales de manera sencilla respondiendo preguntas a nivel gerencial.

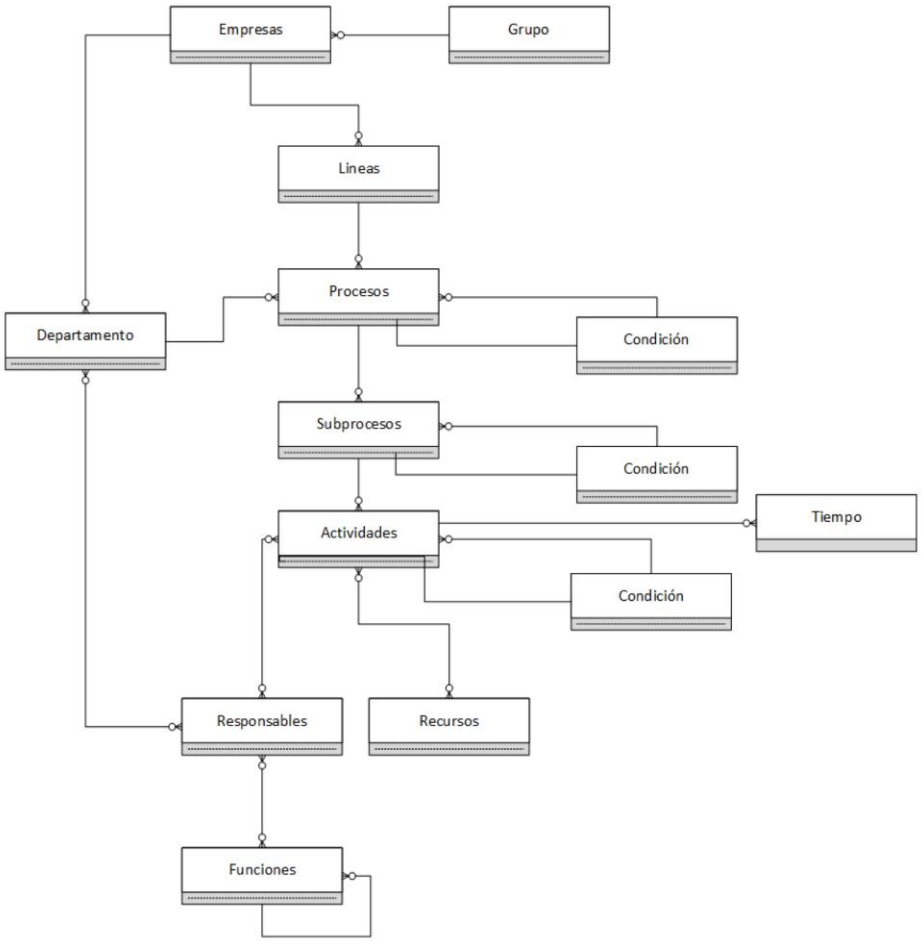

Figura 19 - Modelo conceptual de datos de la plataforma

Estructura MVC

Como patrón de arquitectura de software se ha planteado el denominado Modelo-Vista-Controlador (MVC); en el cual, se divide la estructura del programa en tres partes que se ejemplifican a continuación tomando como base el modelo conceptual de datos de la plataforma:

Modelo: existirá uno por cada uno de los componentes del modelo conceptual de datos: Tiempo, Recursos Materiales, Actividades, etc, cada uno de estos archivos corresponde al acceso a los datos.

Vista: corresponde a la interacción con el usuario, gestionando los datos tanto de entrada como de salida.

Controlador: permite la interacción y relación entre el modelo y la vista, dentro de estos archivos se encontrarán las clases, objetos y sus características

La ventaja principal de este tipo de arquitectura es que nos permite organizar de mejor manera el código y, desde el punto de vista de trabajo en equipo, dividir la tarea de cada uno de los integrantes del equipo de desarrollo.

El patrón asociado a la tecnología Web dentro del proyecto podría resumirse en la Figura 20.

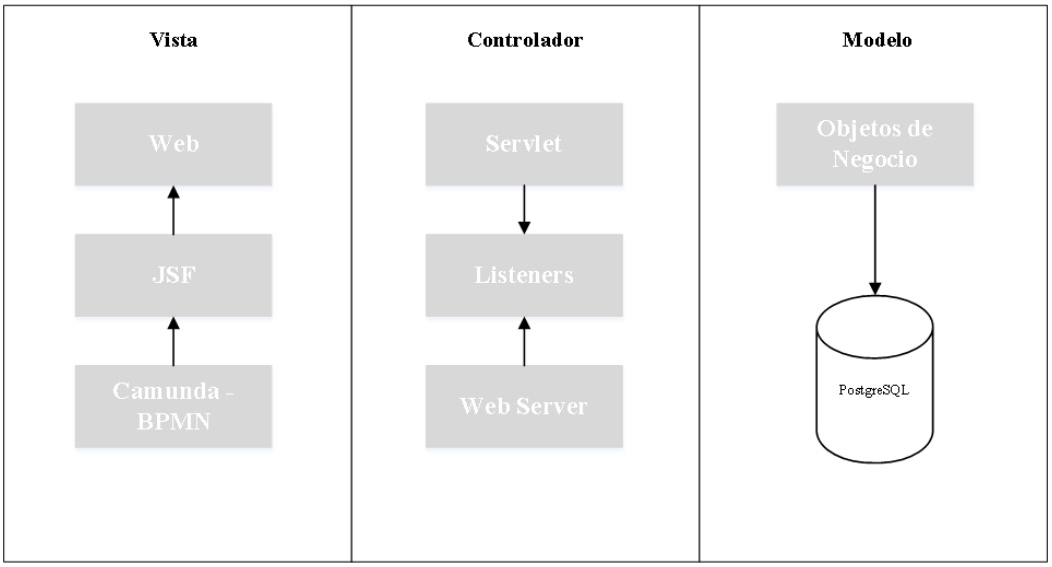

Figura 20 - Patrón MVC asociado a la tecnología Web para el proyecto

Planeación del proyecto – ProductBacklog

Como menciona Guzmán et al. (2014), el productbacklog es una lista de los requerimientos del proyecto para realizar un producto de software completo y funcional. La lista tiende a cambiar con el tiempo de acuerdo a las condiciones de la empresa. La metodología de desarrollo Scrum ayudará a manejar estos cambios en los requerimientos de manera mucho más transparente.

Las estimaciones mencionadas en la Tabla 7 corresponden a las proyecciones en horas brindadas por experiencia y análisis del desarrollador usando los casos de uso, cada sprint corresponde a una semana de labores con una estimación de 40 horas por desarrollador. Otras columnas que se pueden observar en la siguiente tabla son el código el cual hace referencia al requerimiento, la prioridad permitirá al desarrollador gestionar sus horas de acuerdo a las posibilidades.

Código	Descripción	Prioridad	Estimación	Asignación	Sprint
RF01	Administración de Procesos	Alta	80.00	Desarrollador 1	1,2
RF02	Administración de Subprocesos	Alta	40.00	Desarrollador 2	1
RF03	Administración de Actividades	Alta	20.00	Desarrollador 1	3
RF04	Administración de Recursos	Alta	20.00	Desarrollador 1	3
RF05	Administración de Responsables	Alta	20.00	Desarrollador 2	2

RF06	Administración de Empresas	Alta	20.00	Desarrollador 2	2
RF07	Administración de Departamentos	Alta	20.00	Desarrollador 1	4
RF08	Administración de Funciones	Alta	20.00	Desarrollador 1	4
RF09	Administración de Líneas de Producción	Alta	30.00	Desarrollador 2	3
RF10	Administración de Usuarios de la plataforma	Alta	30.00	Desarrollador 2	3,4
RF11	Administración de Instancias de Actividades	Alta	20.00	Desarrollador 2	4
RF12	Cálculos TDABC	Alta	200.00	Desarrollador 1	5,6,7,8,9
RF13	Exportación de Reportes de Resultados TDABC	Alta	20.00	Desarrollador 2	5
RF14	Representación de procesos gráficamente TDABC	Alta	40.00	Desarrollador 2	5,6
RF15	Simulación de procesos	Alta	60.00	Desarrollador 2	7,8
RF16	Optimización de procesos	Alta	40.00	Desarrollador 2	9
RF17	Visualización de resultados de Optimización de Procesos	Alta	25.00	Desarrollador 1	10
RF18	Configuración de parámetros de TDABC	Alta	15.00	Desarrollador 1	10
RF19	Plataforma de Conexión de datos con BD externas	Media	45.00	Desarrollador 2	10,11
RF20	Visualización de Resultados del Análisis TDABC	Alta	35.00	Desarrollador 2	11
RF21	Autenticación de usuarios	Alta	10.00	Desarrollador 1	11
RF22	Seguridad, auditoria y manejo de sesiones de usuario	Alta	70.00	Desarrollador 1	11,12
RF23	Pruebas de implementación en servidor Web	Alta	25.00	Desarrollador 2	12
RF24	Módulo de idiomas	Media	15.00	Desarrollador 2	12
RF25	Configuraciones generales	Media	15.00	Desarrollador 1	13
RF26	Opciones de ayuda al usuario	Media	15.00	Desarrollador 1	13
RF27	Acceso al manual de la plataforma	Media	10.00	Desarrollador 1	13
RF28	Asignación de permisos a usuarios	Alta	10.00	Desarrollador 2	13
RF29	Módulo de Protección contra inyección de SQL	Alta	30.00	Desarrollador 2	13
RF30	Versionamiento de procesos	Alta	55.00	Desarrollador 1	14,15
RF31	Registro de tiempos observados en el desarrollo de una actividad	Alta	25.00	Desarrollador 1	15
RF32	Registro de un tiempo de referencia en el desarrollo de actividades	Alta	20.00	Desarrollador 2	14
RF33	Registro de fechas de ingresos de datos de procesos	Alta	20.00	Desarrollador 2	14
RF34	Perfiles de Usuario	Alta	20.00	Desarrollador 2	15
RF34	Permisos sobre procesos	Alta	20.00	Desarrollador 2	15
RF35	Personalización de medidas de tendencia central para cálculos TDABC	Media	70.00	Desarrollador 1	16,17
RF36	Comparación de versiones de procesos	Media	50.00	Desarrollador 1	17,18
RF37	Almacenamiento de esquema de procesos estándar	Alta	40.00	Desarrollador 2	16
RF38	Definir plantilla de diseño de la plataforma	Media	80.00	Desarrollador 2	17,18

RF39	Definición de los perfiles de usuario de plataforma	Alta	50.00	Desarrollador 1	19,20
RF40	Despliegue en servidor de aplicaciones web	Alta	30.00	Desarrollador 2	19
Total Horas					1480.00

Tabla 7 - Product Backlog para el desarrollo de la aplicación

Como se puede observar en el *Product backlog,* el desarrollo se realizará en un periodo de 20 semanas con el uso de dos desarrolladores sin tomar en cuenta las pruebas finales y aprobaciones que corresponden a 80 horas de trabajo. Si tomamos en cuenta el uso de un solo desarrollador incrementará el periodo de desarrollo de entre 20 a 24 semanas ya que no se podrá realizar ninguno de los requerimientos de manera paralela.

Definición y seguimiento de los Sprints del proyecto

Para la definición y seguimiento de los Sprints del proyecto se ha planteado una herramienta colaborativa Web denominada Jira la cual permite compatibilidad con cualquier metodología ágil de gestión de proyectos de desarrollo de software en nuestro caso específico Scrum.

Herramientas de desarrollo recomendadas

Luego de los análisis presentados en capítulos anteriores, en donde se hacía referencia a herramientas de desarrollo se ha planteado un ambiente de programación basado en las ventajas que aportaron cada una de estas herramientas. Para el caso de del lenguaje de programación se escogió Java ya que las observaciones y comparativo planteado por (Dávila, M. R. 2017) propone este lenguaje, así como el uso de Javascript para las llamadas en el lado del cliente. El manejador de base de datos propuesto como sistema de almacenamiento es PostgreSQL luego de la revisión comparativa proporcionada por (Aránega Hernández, A. 2015) como se había mencionado anteriormente en el capítulo 5.

El Framework de desarrollo Java propuesto para el proyecto corresponde a JSF tomando en cuenta la conclusión del análisis realizado por Pantoja (Pantoja, L., & Pardo, C. 2016) en la que se hace referencia a la evaluación global de los Frameworks.

Dentro del enfoque de *User Interface(*UI), se ha planteado el uso de Bootstrap que según (Arias, M. A. G. 2013) es un enfoque de diseño web destinado a la elaboración de sitios para proporcionar una visualización óptima para una experiencia de navegación fácil y con un mínimo de cambio de tamaño, paneo, y desplazamiento a través de una amplia gama de dispositivos o plataformas. Un sitio desarrollado con Bootstrap adapta su diseño a las condiciones de visualización mediante un uso correcto de las proporciones basadas en cuadriculas, imágenes flexibles y CSS3.

De acuerdo al análisis presentado por (Quijada, F., & Ignacio, G. (2015) la mejor opción en herramienta BPMN para el uso de lenguaje de programación Java es Camunda el cual está escrito en Javascript y tiene la ventaja de poseer la totalidad de los elementos BPMN y una alta usabilidad en la interfaz con el usuario. Un resumen de las herramientas a utilizar en el ambiente de desarrollo de la aplicación se puede ver en la Tabla 8.

IDE	Netbeans
Testing	Junit
Framework	JSF version 2.2(estable)
Java	Version 7 o superior
Diseño	Bootstrap v3.3.7
Base de Datos	PostgreSQL 9.1
BPMN	Camunda

Tabla 8 - Resumen de herramientas a utilizar en la aplicación

Arquitectura del software recomendada

La arquitectura de software resultante que se presenta en la Figura 18, se encuentra basada en el patrón de arquitecturas de software MVC. El bloque de Fuentes de Datos representa los recursos que proporcionan datos de la empresa a la plataforma. En este bloque se destacan los valores de Tiempo por Actividad y Costo de Recurso para el cálculo TDABC, así como la metodología y sistema de calidad utilizados en cada uno de los procesos y que proveerá información al software. Se incluyen también otras fuentes como archivos, sistemas transaccionales y herramientas de evaluación de calidad. Todos estos módulos, sistemas y recursos serán conectados mediante el uso de herramientas informáticas como Web Service, herramientas intermedias como Excel, y la importación de archivos planos.

El segundo bloque presenta las bases de datos, la estructura de modelos de la implementación MVC y repositorios que almacenarán datos de los procesos de la empresa y los procesos de análisis de riesgo, optimización y simulación. Con estos últimos datos y mediante el uso del software @Risk se podrá simular diferentes tipos de escenarios, como por ejemplo la variación del flujo de caja. Como se puede observar en la Figura 21 existe la posibilidad de manejar diferentes tipos de repositorios para un acceso más rápido del historial de datos dentro del repositorio de cálculos. Todo el esquema de estas bases de datos se manejará en la sección de modelos proporcionando una capa dentro del sistema el cual permitirá un mantenimiento más sencillo.

El bloque de Procesamiento encierra la lógica de la aplicación, controladores, programación de flujo de datos y los módulos encargados del control de cálculo de la metodología los cuales permiten manipular los datos almacenados y realizar los cálculos del sistema TDABC. Finalmente, el bloque de Presentación describe la manera en que se mostrarán los datos al usuario mediante el uso de las vistas de MVC, incluyendo los flujos de proceso y los costos obtenidos.

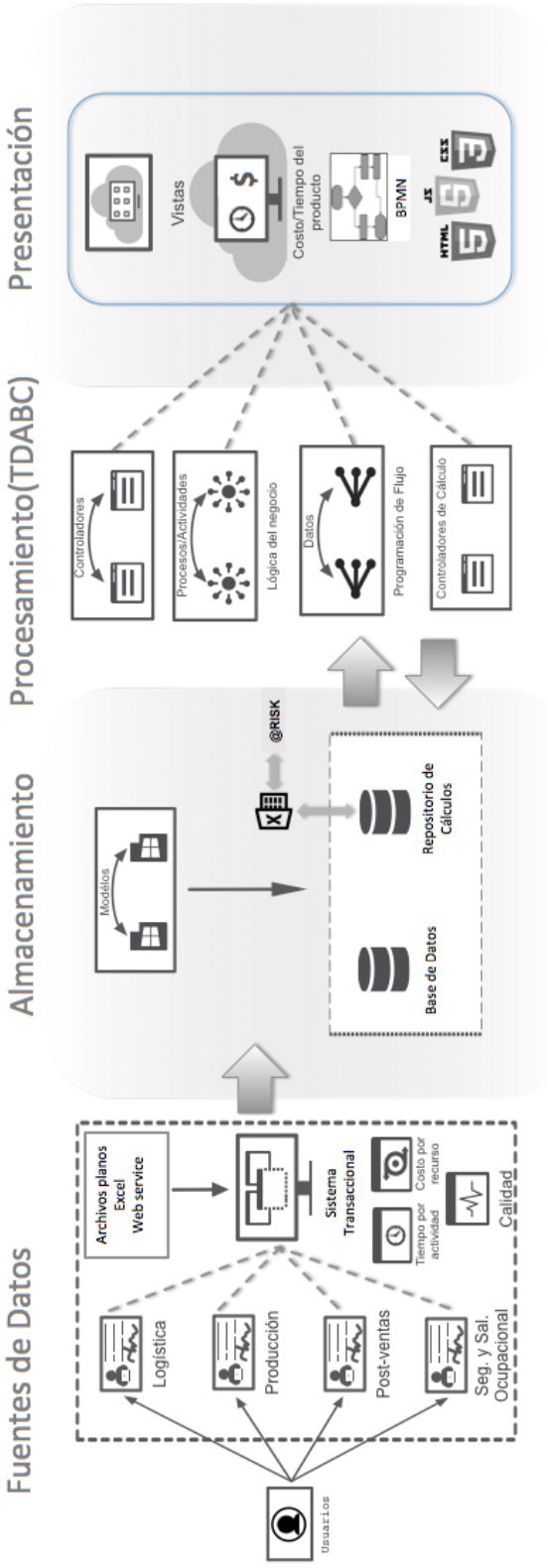

Figura 21 - Arquitectura de Software recomendada en software TDABC

Proceso de evaluación y satisfacción del usuario

Para los procesos de evaluación y satisfacción del usuario se plantearon cuatro secciones siguiendo el concepto de prototipado evolutivo: *bosquejo* de las pantallas de la aplicación, guía de estilos en formato Web para la aplicación en cada uno de los componentes, prototipo funcional de cada una de las pantallas de la aplicación y por último historias de usuario usando Scrum.

Para el caso de los bosquejos de cada una de las pantallas se utilizó un software de maquetado en el que se graficó cada uno de los componentes de la pantalla con un diseño básico. La Figura 19 corresponde a la forma en la que se generarán los listados dentro de la plataforma. La Figura 20 corresponde a la forma en la que se crearán los procesos y subprocesos, así como su distribución. La Figura 21 visualizará la forma de modificar los procesos. Cada uno de los procesos podrá ser generado en versiones con el objetivo de comparar procesos y tomar decisiones gerenciales, esto se puede observar en la Figura 22.

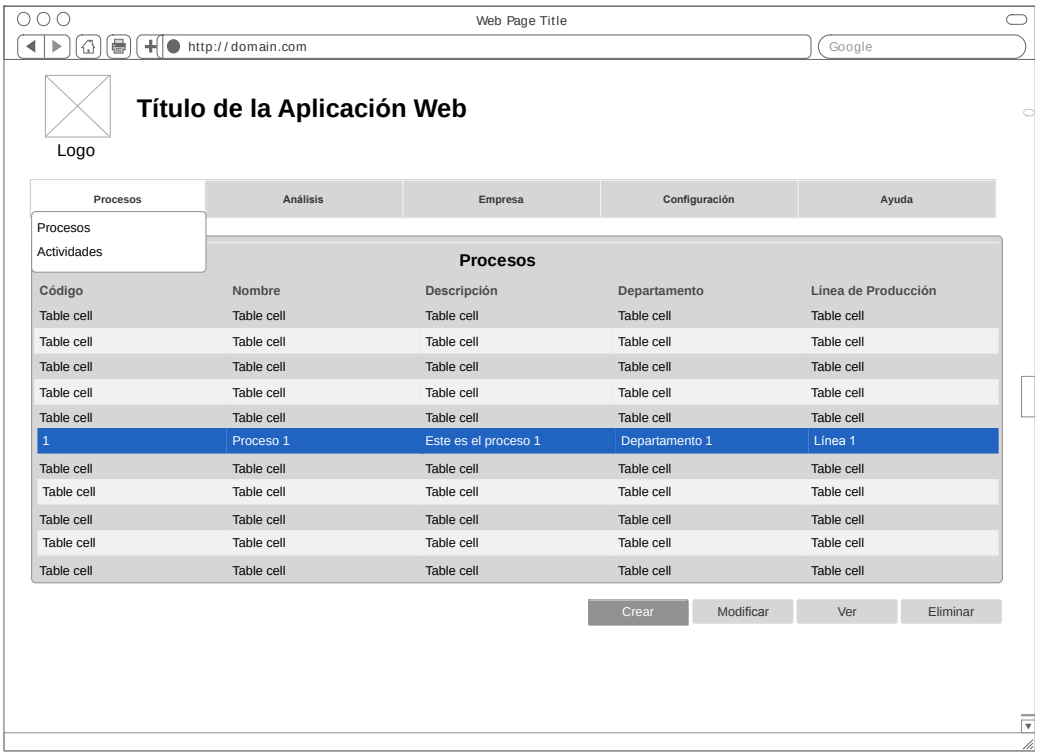

Figura 22 - Bosquejo de listado de procesos

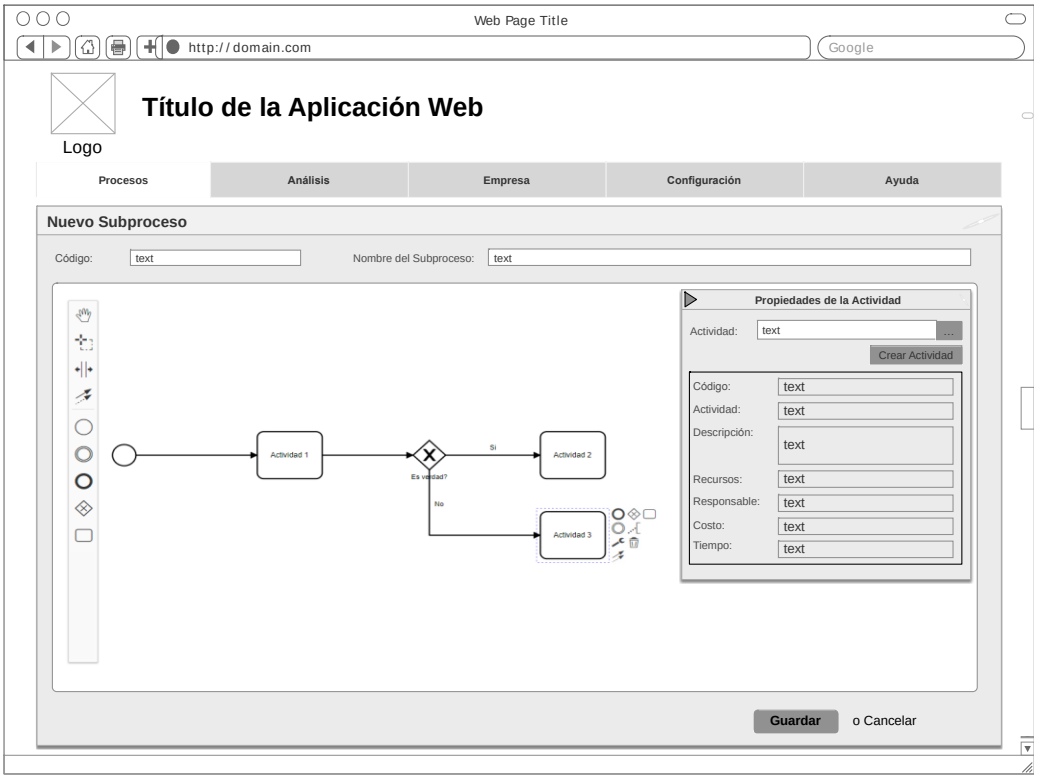

Figura 23 - Bosquejo de generación de procesos y subprocesos

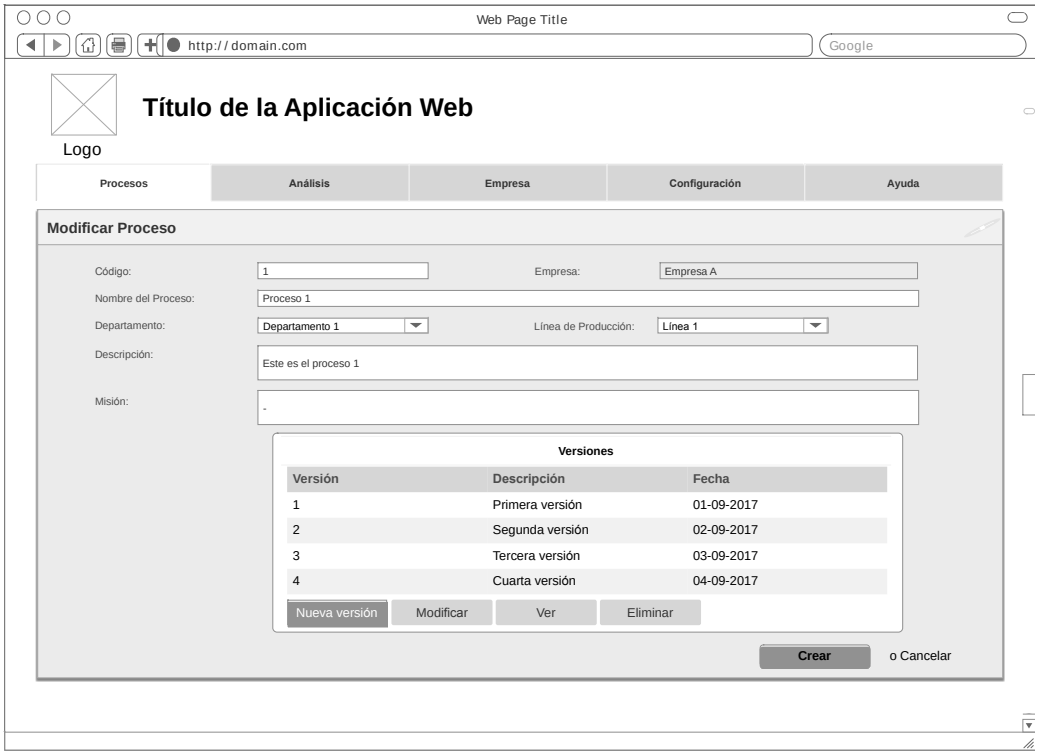

Figura 24 - Bosquejo de modificación de procesos

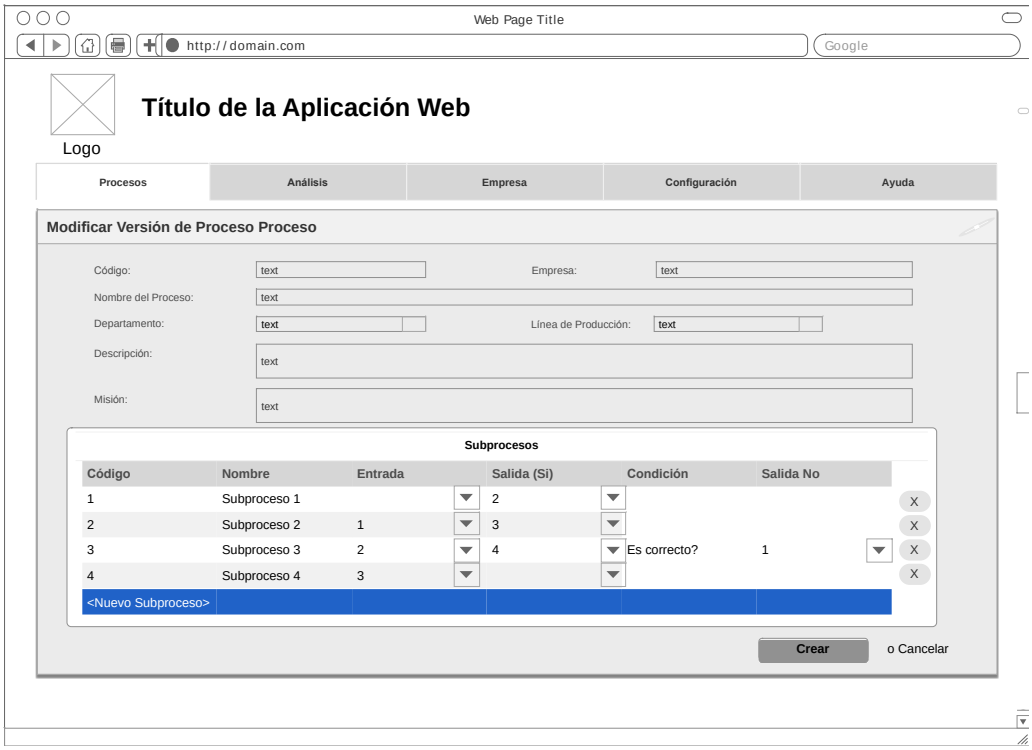

Figura 25 - Bosquejo de modificación de versiones de proceso

En base al diseño de interfaz de usuario (UI) correspondiente a Bootstrap el cual es un Framework creado por Twitter de esta forma se planteó la guía de estilos correspondientes a la aplicación. Estos archivos fueron generados en formato HTML + CSS utilizados por Boostrap.

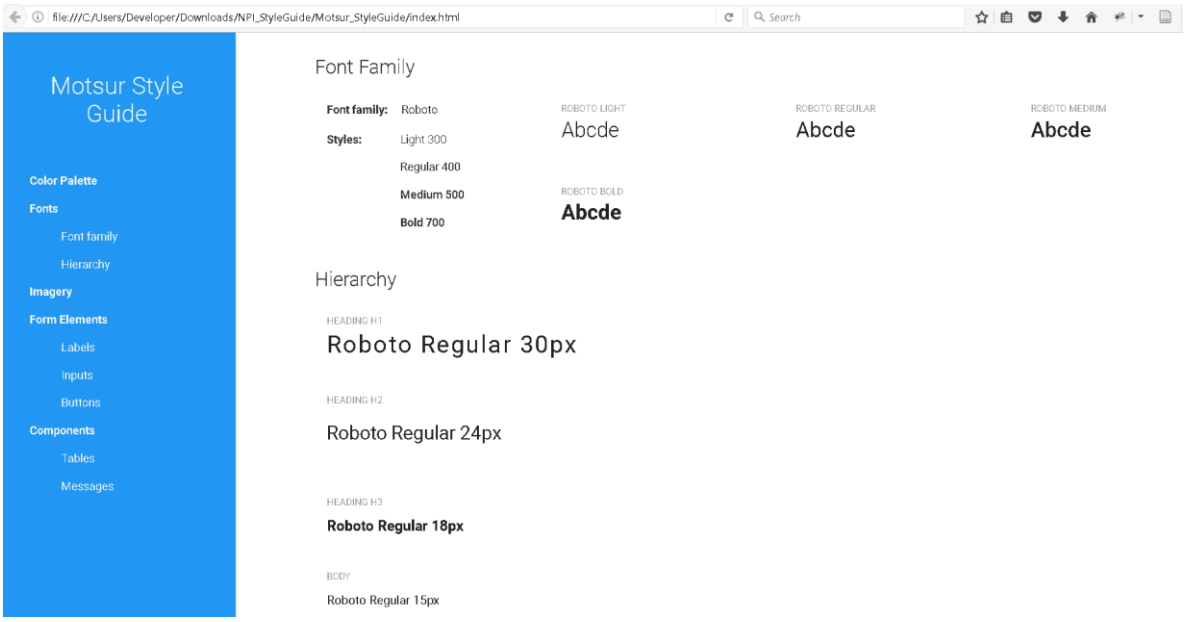

Figura 26 - Ejemplo de Guía de Estilos Booststrap

El prototipo funcional será realizado a partir de la capa Vistas del aplicativo, en donde se tendrá una combinación de HTML y CSS con funcionalidad básica y navegación dentro de la plataforma.

Las historias de usuario de la aplicación ayudarán a desarrollar y realizar las pruebas correspondientes a cada uno de los componentes de la aplicación. Para esto se ha definido una plantilla que ayudará con este objetivo.

Desarrollo aplicación Motsur: Historias de usuario y criterios de aceptación
Elaborado por: Product Owner

	Enunciado de la historia				Criterios de aceptación			
Identificador (ID) de la historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Criterio de aceptación (Título)	Contexto	Evento	Resultado / Comportamiento esperado
XX-XXXX-XXXX	Como un [Rol]	Necesito [Descripción de la funcionalidad]	Con la finalidad de [Descripción razón o resultado]	1	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]
				2	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]
				3	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]
				4	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]
XX-XXXX-XXXX	Como un [Rol]	Necesito [Descripción de la funcionalidad]	Con la finalidad de [Descripción razón o resultado]	1	[Titulo del	En caso que [contexto] y/ o	cuando [evento]	el sistema [resultado /
				2	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]
				3	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]
				4	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]
XX-XXXX-XXXX	Como un [Rol]	Necesito [Descripción de la funcionalidad]	Con la finalidad de [Descripción razón o resultado]	1	[Titulo del	En caso que [contexto] y/ o	cuando [evento]	el sistema [resultado /
				2	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]
				3	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]
				4	[Titulo del escenario]	En caso que [contexto] y/ o [contexto]	cuando [evento]	el sistema [resultado / comportamiento]

Nota:
- Reemplazar las etiquetas [Rol], [Descripción de la funcionalidad], [Descripción de razón o resultado] por el contenido del enunciado de la historia.
- Reemplazar las etiquetas [Título del escenario], [contexto], [evento] y [resultado / comportamiento] por el contenido de los criterios de aceptación.

Figura 27 - Plantilla de Historias de Usuario para empresas de producción

Historias de usuario y criterios de aceptacion: Ejemplo Productos

Elaborado por: Product Owner

Enunciado de la historia				Criterios de aceptación			
Identificador (ID) de la historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Criterio de aceptación (Título)	Contexto	Evento	Resultado / Comportamiento esperado
CU-XXXX-XXXX	Como un cliente.	Necesito ver un listado de categorías de productos y poder seleccionar una categoría.	Con la finalidad de realizar busquedas de productos por categorías.	1	Categoría con al menos un producto.	En caso que una categoría tenga al menos un producto asociado.	Cuando se despliegue el listado de categorías a seleccionar.	A continuación del nombre de la categoría, se mostrará entre paréntesis el número de productos asociados.
				2	Categoría sin productos.	En caso que una categoría no tenga productos asociados.	Cuando se despliegue el listado de categorías a seleccionar.	A continuación del nombre de la categoría, se mostrará entre paréntesis el siguiente texto "Sin Productos asociados".
				3	Ordenamiento de las categorías	N/A	Cuando se despliegue el listado de categorías a seleccionar.	El sistema mostrará las categorías en orden alfabético.

Figura 28 - Ejemplo de Historia de Usuario para la Sección Productos

Gestión de Riesgos

De acuerdo a lo presentado por Cordero et al. (2013), para el conjunto de riesgos definidos dentro del desarrollo de software se plantearon algunos de los riesgos más importantes en lo que corresponde a este proyecto, los cuales se encuentran nombrados en la Tabla 9.

Nombre del Riesgo	Valor de dominio	Tipo
Experiencia en la plataforma	Baja	Ordinal
Nivel de conocimiento de las herramientas	Media	
Estabilidad de los requerimientos	Estables	
Información que se maneja	Confidencial	Literal
Cantidad de funcionalidades medias	121	Numérico
Cantidad de funcionalidades complejas	48	

Tabla 9 - Conjunto de rasgos de riesgos definidos para la aplicación en base a (Cordero Morales, D et al., 2013)

Para el primer riesgo experiencia en la plataforma se ha colocado un valor de dominio bajo ya que es la primera vez que se plantea un desarrollo TDABC para empresas en el sector productivo. En el segundo riesgo al ser el aplicativo un desarrollo mediante el uso de varios tipos de herramientas no se posee un nivel completo de la interacción entre cada una de ellas. La estabilidad de los requerimientos siempre puede llegar a ser un riesgo al interactuar con los usuarios ya que muchos de ellos especifican con más detalle al momento del uso de los prototipos, además de intentar incrementar funcionalidad. La información manejada en algunas secciones es de carácter confidencial por esta razón podría existir un acceso restringido de los datos. De acuerdo al detalle proporcionado por los casos de uso, de un total de 226 funcionalidades, la mayor cantidad son medias y complejas lo que provoca la dificultad del desarrollo de la aplicación.

Conclusiones

En la propuesta de análisis y diseño del sistema planteado en este capítulo, se pretende realizar un resumen de todas las partes involucradas para el desarrollo del software para el manejo de TDABC. Por esta razón se parte de la metodología escogida con la que desarrollara todo el proyecto, cada uno de los componentes de análisis y diseño recomendados. Una vez claros los componentes de análisis y diseño es posible adaptarlos a la metodología ágil seleccionada lo que correspondería a la planificación. Los siguientes puntos permitirán conocer las herramientas de desarrollo y la arquitectura que va a poseer el aplicativo una vez desarrollado. Para finalizar una correcta retroalimentación por parte del usuario, así como un manejo correcto de la gestión de riesgos adecuadamente documentados nos presentaran menos sorpresas al momento de la implementación.

Capítulo 8: Conclusiones

Introducción

Este capítulo final pretende dar una visión global de todo el análisis, proporcionando la experiencia de análisis y diseño de la plataforma, realizar una alineación de las ventajas del TDABC con las ventajas del uso de la plataforma y de igual manera realizar un comparativo de las limitaciones con las bondades del aplicativo para verificar si algunas de estas fueron eliminadas o minimizadas provocando en conjunto una versión mejorada de la metodología TDABC.

Ventajas y alineación de la metodología TDABC con el uso de la aplicación

Tomando como base las ventajas del TDABC que se mencionó en el capítulo 2 se realiza un resumen del incremento de las ventajas del uso de la plataforma propuesta en el trabajo para manejo de TDABC. Simplicidad: El uso de una herramienta de software para crear diagramas de procesos, asignándoles tiempos y costos por actividad, hace que la construcción de modelos de costeo por procesos sea una tarea más simple e intuitiva, lo que simplifica aún más la construcción de modelos de costo. Reducción de complejidad en operaciones: Esta reducción de complejidad se acentúa mediante la automatización de los cálculos de costos provista por la plataforma. El usuario del software solamente requerirá asignar responsables y recursos utilizados a cada actividad de un proceso y el software se encargará de realizar la construcción de las ecuaciones de tiempo y calcular los resultados del análisis TDABC por procesos. Desagregación de costos de procesos: Los resultados del análisis TDABC utilizando la plataforma informática permitirán visualizar y detallar los costos de las diferentes situaciones o flujos que pueden darse en el desarrollo de un proceso. Visualización de la capacidad de utilización: La plataforma aprovecha esta ventaja de TDABC debido a que se pueden realizar varias consultas de información por departamento, proceso, subproceso o actividad, para identificar como se están utilizando los recursos materiales y humanos, con la finalidad de realizar una mejor planificación de su asignación para el desarrollo de las actividades.

Versatilidad del sistema: Dado que, el software propuesto en este trabajo tiene como objetivo su uso en las empresas del ámbito de producción, se validará esta ventaja de TDABC al aplicarlo en tres empresas de caso de estudio diferentes. Modularidad y versatilidad de modelos de costeo: El modularidad que provee TDABC al análisis de costos es una ventaja que se resalta aún más con la implementación de la plataforma. Esto se debe a que, la actualización de información sobre el desarrollo de actividades requerirá solamente el ingreso de datos a la plataforma o la creación de una nueva versión de un proceso. Capacidad de simulación de procesos: La plataforma aprovecha esta capacidad al momento de crear y visualizar diferentes flujos de procesos. Esto sumado al uso de BPMN con su detalle de costos, permite, al usuario del sistema informático, identificar las posibilidades de mejora (ej. procesos, actividades, recursos y tiempo). Se puede potenciar este beneficio aplicando herramientas de simulación y optimización automática sobre los datos obtenidos de la plataforma mediante la generación de modelos adaptados a las necesidades específicas de la empresa.

Limitaciones vs Bondades de la aplicación del software

Dentro de las limitaciones que presenta (Namazi, M. 2016) en su artículo podemos encontrar las enumeradas a continuación: Como primera limitación tenemos que es posible ejecutar TDABC en varias industrias, pero su aplicación se limita a situaciones en las que el "tiempo" puede ser ejercido como el único factor de costeo; esta limitación puede ser disminuida con el uso de la plataforma ya que al ser posible las configuraciones y manejo de datos es posible realizar cálculos de distintas maneras logrando ejecutar comparativos de entre varios resultados, además de interactuar con sistemas que permitan determinar el tiempo de manera más exacta. Esto podría hacerse principalmente en los procesos en los que intervengan equipos que ayuden en la toma de tiempos. Los demás procesos que requieren una observación o encuestas todavía tendrán ese error en la medición. La segunda limitación corresponde a la falta de identificación de la actividad en la primera etapa, desvía al TDABC significativamente de los mayores y principales fundamentos del ABC. Si las "Actividades" no se identifican claramente al principio, y se calcula una sola tasa de costos holística para todo el departamento, equivale a volver a los sistemas tradicionales de contabilidad de costos basados en el volumen como lo cita (Namazi, M. 2009); de igual manera como se menciona en la primera limitación el uso del sistema permitirá mantener los datos disponibles durante cualquier etapa y de esta forma resultados y comparativos en cualquiera de las mismas. La tercera limitación corresponde a que a pesar de que aparentemente TDABC aborda la simplicidad, la determinación exacta de los costos de capacidad de manera práctica, la tasa de capacidad y la absorción de una tasa de costo de capacidad uniforme para todas las actividades del departamento han surgido como nuevos obstáculos, así como lo menciona el mismo autor en otro de sus artículos (Namazi, M. 2009); dentro de este punto es todavía difícil determinar los costos de cada una de las actividades de manera independiente ya que el TDABC facilita la determinación de los costos en departamentos en los que existe una sola actividad, si bien es cierto que el sistema nos ayudara con la organización de los datos en este punto, la determinación de los costos directos e indirectos de los recursos en cada una de las actividades es difícil colarlos en las proporciones adecuadas aun con el sistema.

La cuarta limitación nos menciona que, aunque parece que la construcción de un modelo TDABC es más fácil que la creación de un modelo CABC, que no siempre es el caso porque "ABC cronometrado requiere tanto la recopilación de datos como el modelo ABC tradicional. Cada vez que se actualiza y recalcula un modelo, se deben actualizar los controladores de duración. "; con el uso del sistema se organizará la actualización de datos y la interacción con el usuario, pero todavía no es posible la realización de una actualización de los controladores de duración de manera automática, requiriendo tiempo hombre para el cumplimiento de esta actividad, sin embargo, existe la posibilidad de importación de archivos de texto o conexiones a bases de datos de sistemas propios de la empresa como sistemas ERP disminuyendo un poco esta limitación. La quinta limitación nos dice que el TDABC añade un paso más que podría considerarse innecesario para el proceso de implementación de tiempo del ABC porque requiere que el administrador se involucre en el proceso de la estimación del tiempo. Este proceso no sólo aumenta los costos de recolección de la información requerida, sino que también hace que consuma mucho tiempo y crea disimetría

de información; al requerir la interacción humana para la estimación del tiempo el sistema no proporcionara ayuda alguna en este sentido más que al momento del ingreso de los datos. La sexta limitación se fija en que la metodología TDABC al calcular el costo de la capacidad no utilizada, no considera la capacidad de los recursos y el comportamiento de los costos completamente; al hacer uso del sistema es posible interactuar y hacer uso de datos de otros módulos dentro del sistema para obtener un mayor detalle en lo que se refiere a la capacidad de los recursos y el comportamiento de los costos.

La séptima limitación nos dice que la toma de decisiones con el uso de la metodología TDABC es restringida por las siguientes razones:

a) TDABC asume que la relación entre las actividades y los recursos consumidos es lineal, absoluta y determinada
b) TDABC ignora las restricciones sobre los recursos de la actividad y los cuellos de botella.
c) Principalmente, las decisiones gerenciales deben seguir el concepto de "información relevante", que no es equivalente al concepto de la información sobre costes de absorción
d) La información de TDABC es útil sólo cuando el tipo de decisión se define sin ambigüedad

Mediante el uso del sistema, datos históricos, así como las herramientas de optimización y análisis de datos que se pretende usar se podría mejorar los puntos b) donde se podría registra los tiempos muertos de manera histórica y en el punto c) al poseer resultados en base a un modelo de predicciones las decisiones gerenciales podrían ser más exactas en base a modelos probabilísticos basados en la simulación Montecarlo. La capacidad de realizar evaluaciones comparativas de procesos es otra oportunidad que ya se aprovechó en la implementación del prototipo del módulo TDABC para bibliotecas (Cabrera, Ordoñez, 2012). Esto mediante la implementación de una funcionalidad de comparación de procesos dentro de la empresa o entre empresas del mismo grupo empresarial, la cual se ha considerado como requerimiento para la plataforma propuesta. Dentro de estos puntos algo que se ha considerado para futuras versiones son la integración de la plataforma TDABC con una herramienta de Balance Scorecard (BSC) para proveer información de los indicadores de gestión. y la integración de TDABC con sistemas de calidad, tales como Total Quality Management (TQM) queda pendiente para un análisis posterior.

La última limitación considera que la metodología TDABC, al derivar modelos de ecuación de tiempo de "actividad", puede no necesariamente generar información de costo más precisa, porque:

a) Los modelos de ecuaciones de tiempo se basan en la hipótesis de linealidad y certeza. Estos supuestos, por supuesto, son muy restrictivos.
b) Las ecuaciones de tiempo no reducirán las complejidades del proceso (Tse, M., & Gong, M. 2009) y corren el riesgo de una estimación de la transparencia de costos.

c) Las ecuaciones de tiempo no consideran los efectos de un "proceso" designado en la estimación de costos,
d) Se desarrollan modelos de ecuaciones de tiempo para cada actividad individualmente, y se ignoran los efectos de interacción de las actividades

Mediante el uso de BPMN es posible ser más específico en los efectos y los tiempos que se tienen al momento de la interacción de las actividades manejando los datos e incluyéndoles en el modelo.

Recomendaciones

Las recomendaciones principales se enfocarán al diseño de la plataforma en donde se deberá tener en cuenta la calidad de la información para la creación de los prototipos necesarios y retroalimentación por parte del usuario, para esto es necesario usar la información que la empresa dispone actualmente y mediante la serie de iteraciones que generara la metodología Scrum mejorar la calidad de manera secuencial para las pruebas respectivas. Otra recomendación enfocada a la información hace referencia a la posibilidad de definir de manera más clara los costos exactos y aproximados, es decir los costos que tan exactos son y los aproximados que tan aproximados son. La recomendación relacionada con las características específicas de las empresas en el sector productivo hace que varios de los aspectos del diseño no se apliquen completamente a otras compañías de diferentes sectores, pero permitirá iniciar el proceso para un planteamiento enfocado a otras áreas para un futuro comparativo y análisis.

Una vez culminado todo el análisis se propone como recomendaciones adicionales realizar investigaciones para la integración con *Balance Scored Card*(BSC)para la obtención de indicadores a partir de la estrategia de negocios en base a los resultados obtenidos por la plataforma. Otra recomendación adicional podría ser la implementación de sistemas de calidad sobre la plataforma denominado Total Quality Management (TQM) el cual permitirá el crecimiento continuo en todas las áreas de la organización.

Comentarios Finales

El análisis de costos, procesos y optimización de datos resulta importante e innovador en la industria de la producción y de forma específica en la de ensamblaje en el Ecuador. Por este motivo y para aprovechar las experiencias y resultados obtenidos en investigaciones previas, se ha propuesto una arquitectura para el desarrollo de un software de soporte del modelo de gestión de procesos, costos que automatice y mejore el análisis TDABC en industrias de ensamblaje dentro del sector productivo. Este estudio trata de actualizar y adaptar experiencias previas, para aplicarla a los procesos de empresas de producción y por ende al de ensamblaje. La implementación de una plataforma de gestión de TDABC permitirá obtener resultados de costos de forma más precisa, modelando los procesos por actividades y facilitando la visualización de los recursos sub-utilizados y sobre utilizados; además de la obtención de modelos de optimización y análisis de datos. La plataforma ayudará a los

administradores a tomar decisiones más acertadas intentando llegar al equilibrio de las capacidades de cada una de las etapas productivas.

La arquitectura presentada es el resultado de una evaluación de las herramientas y metodologías para el desarrollo de software que mejor se ajustan a los datos y funcionalidades que serán provistas por la plataforma propuesta. La arquitectura, el esquema de procesos y la plataforma resultantes son validadas por medio de su aplicación en el análisis de los procesos de tres líneas de ensamblaje de una empresa de caso de estudio. Así también, para proceder con la implementación se seguirán procesos iterativos y graduales que permitirán la inserción de cada uno de los componentes de forma natural. **Finalmente, se espera que esta plataforma informática pueda ser reutilizada por otras empresas del ámbito nacional, con la finalidad de evaluar su rendimiento y contribuir al mejoramiento de los procesos de ensamblaje y, por lo tanto, contribuir al mejoramiento de la industria productiva del país o servir como punto de partida para la generación de una solución aplicable a los diferentes tipos de empresas**.

El sistema informático brinda la posibilidad de potenciar las ventajas que posee TDABC, así como también, superar algunos desafíos presentados por este sistema de costeo. Sin embargo, el diseño actual no está exento de limitaciones, que deberán ser analizadas en futuras versiones de la plataforma.

Bibliografía

Abdelhady, I. A. (2015). A comparative approach to map BIM workflow in US mid-size firms using BPMN and IDEF methods. of Architectural Research, 509.

Acevedo Suárez, J. A., Urquiaga Rodríguez, A. J., & Gómez Acosta, M. (2001). Gestión de la cadena de suministro. Centro de estudio Tecnología de Avanzada (CETA) y laboratorio de Logística y Gestión de la producción (LOGESPRO). Ciudad de La Habana.

Ai-Min, D. E. N. G., Hong, L. I., & Hao, T. I. A. N. (2016). Based on the Cloud ERP and TDABC for the SMEs' Logistics Cost Accounting. DEStech Transactions on Engineering and Technology Research, (sste).

Alfonso Salinas, "Sistemas de Costeo," 24-Oct-2011. [Online]. Available: http://www.loscostos.info/sistemas.html.

Andradre, E. y Elizalde, B. (2017). LEVANTAMIENTO DE PROCESOS DE ENSAMBLAJE DE TELEVISORES PARA LA EMPRESA SURAMERICANA DE MOTORES MOTSUR CIA. LTDA (Documento de Trabajo). Cuenca: Universidad de Cuenca.

Aránega Hernández, A. (2015). Almacenes de datos: Análisis de ventas de una compañía (Master's thesis, Universitat Oberta de Catalunya).

Arango Serna, M. D., Campuzano Zapata, L. F., & Zapata Cortes, J. A. (2015). Mejoramiento de procesos de manufactura utilizando Kanban. Revista Ingenierías Universidad de Medellín, 14(27), 221-233.

Arias Chaves, M. (2005). La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. InterSedes: Revista de las Sedes Regionales, 6(10).

Arias, M. A. G. (2013). Responsive Design. Desarrolla webs sensitivas con Bootstrap. IT Campus Academy.

Aristegui, J. L. (2010). Los Casos de Prueba en la Prueba del Software. Lámpsakos, (3), 27-34.

Asensio, J. J., Mora, A. M., Garcıa-Sánchez, P., & Merelo, J. J. (2013). Code Reimagined: Gamificación a través de la visualización de código. In Actas del Primer Simposio Español de Entretenimiento Digital (p. 72).

Ávila-Gutiérrez, M. J., Aguayo-González, F., Marcos-Bárcena, M., Lama-Ruiz, J. R., & Peralta-Álvarez, M. E. (2017). Reference holonic architecture for sustainable manufacturing enterprises distributed. DYNA, 84(200), 160-168.

Avison, D., & Fitzgerald, G. (2003). Information systems development: methodologies, techniques and tools. McGraw Hill.

Aycart Pérez, D., Gibert Ginestà, M., & Hernández Matías, M. (2007). Ingeniería del software en entornos del software libre, febrero 2007.

Aycart, D., Ginestá, M. y Hernández, M. (2007). Ingeniería de Software en Entornos de SL.Barcelona: Universitat Oberta de Catalunya.

BARRET, R. (2005). Time-Driven Costing: the bottom line on the new ABC. Business Performance Management, 11, 35-39.

Benedetto, M. G., Carabio, A. L. R., Alvez, C. E., Fernández, M., Etchart, G., Cabrera, S. A., ... & Benítez, H. D. (2015, May). Selección de lenguajes orientados a objetos para un estudio comparativo y análisis de rendimiento. In XVII Workshop de Investigadores en Ciencias de la Computación (Salta, 2015).

Cabrera Encalada, P., & Ordoñez Parra, C. (2012). Tesis. Recuperado a partir de http://dspace.ucuenca.edu.ec/handle/123456789/646

Cadavid, A. N., Martínez, J. D. F., & Vélez, J. M. (2013). Revisión de metodologías ágiles para el desarrollo de software. Prospectiva, 11(2), 30-39.

Camarena Sagredo, J. G., Trueba Espinosa, A., Martínez Reyes, M., & López García, M. D. L. (2016). Redalyc. Automatización de la codificación del patrón modelo vista controlador (MVC) en proyectos orientados a la Web. Ciencia Ergo Sum, 19(3), 239-250.

Capilla Rafael, Jansen Anton, Tang Antony, Avgeriou Paris, Ali Babar Muhammad. (2016). 10 years of software architecture knowledge management: Practice and future. ScienceDirect, 116, 191-205.

Cataldi, Z. (2000). Una metodología para el diseño, desarrollo y evaluación de software educativo (Doctoral dissertation, Facultad de Informática).

Catalinas, E. Q. (2002). Sistemas operativos y lenguajes de programación. Editorial Paraninfo.

CERT, Software Engineering Institute, Carneige Mellon University, Octave Risk Analisys Methodology, http://www.cert.org/resilience/products-services/octave/, Fecha de última consulta: 18 de noviembre de 2015.

Coad, P., Luca, J. D., & Lefebvre, E. (1999). Java modeling color with UML: Enterprise components and process with Cdrom. Prentice Hall PTR.

Cockburn, A. (2002). Agile software development (Vol. 177). Boston: Addison-Wesley.

Contreras, H., & McCawley, A. (2006). Implementación de un modelo de costos ABC en una empresa vitivinícola. Economía Agraria, 10, 25-36.

Cooper, R. (1989). The Rise of Activity-based Costing: How Many Cost Drivers Do You Need, and how Do You Select Them?.

Cooper, R., & Kaplan, R. S. (1988). Measure costs right: make the right decisions. Harvard business review, 66(5), 96-103.

Cordero Morales, D., Ruiz Constanten, Y., & Torres Rubio, Y. (2013). Sistema de Razonamiento Basado en Casos para la identificación de riesgos de software. Revista Cubana de Ciencias Informáticas, 7(2), 222-239.

Córdova Espinoza, R. F., & Cuzco Sarango, B. E. (2013). Análisis comparativo entre bases de datos relacionales con bases de datos no relacionales(Bachelor's thesis).

Croce, L. D., & Salinas, J. A. (2016). Flexibilidad en bases de datos NoSQL sobre ambientes web mining (Doctoral dissertation, Facultad de Informática).

Dávila, M. R. (2017). Investigación en Progreso: Estudio Comparativo de la Incidencia de los Lenguajes de Programación en la Productividad Informática. Revista Latinoamericana de Ingenieria de Software, 4(6), 255-258.

de Arbulo López, P. R., & Santos, J. F. (2011). Innovación en gestion de costes: del abc al tdabc. Dirección y organización, (43), 16-26.

Dejnega, O. (2011). Method time driven activity based costing–Literature review. Journal of Applied Economic Sciences (JAES). (1 (15)), 9–15

Díaz Piraquive, F. N. (2008). Gestión de procesos de negocio BPM (Business Process Management), TICs y crecimiento empresarial.¿ Qué es BPM y cómo se articula con el crecimiento empresarial?. Universidad & Empresa, 7(15).

Everaert, P., Bruggeman, W., Sarens, G., Anderson, S. R., & Levant, Y. (2008). Cost modeling in logistics using time-driven ABC: Experiences from a wholesaler. International Journal of Physical Distribution & Logistics Management, 38(3), 172-191.

Fiannaca, A. J., and Huang, J. (2015). Benchmarking of Relational and NoSQL Databases to Determine Constraints for Querying Robot Execution Logs. Computer Science & Engineering, University of Washington, USA, 1-8.

García Molina, J., Ortín, M. J., & Moros, B. (2007). De los Procesos del Negocio a los Casos de Uso.

García Rodríguez, M. J. (2015). Estudio comparativo entre las metodologías ágiles y las metodologías tradicionales para la gestión de proyectos software.

Gervais, M., Levant, Y., & Ducrocq, C. (2010). Time-driven activity-based costing (TDABC): An initial appraisal through a longitudinal case study. Journal of Applied Management Accounting Research, 8(2), 1.

Gervais, Michel; Levant, Yves; Ducrocq, Charles. (2010). Journal of Applied Management Accounting Research. Scholarly Journals, 8, 20. 2017, De ProQuest Central Base de datos.

Gómez Niño, O. (2011). Los costos y procesos de producción, opción estratégica de productividad y competitividad en la industria de confecciones infantiles de Bucaramanga. Revista EAN, (70), 167-180.

Gómez Valdés, J. (2015). Costes basados en el tiempo invertido por actividad (TDABC): una aplicación práctica= Time-driven activity-based costing (TDABC): a practical application.

Gutiérrez, J. J. (2014). ¿ Qué es un framework web?. Available in: http://www. lsi. us. es/~ javierj/investigacion_ficheros/Framework. pdf Accessed May, 12.

Guzmán, D. I., Islas, U. C., Corona, C. P., & Méndez, B. E. P. (2014). Metodología ágil Scrumban en el proceso de desarrollo y mantenimiento de software de la norma MoProSoft. Research in Computing Science, 79, 97-107.

Highsmith, J. (2013). Adaptive software development: a collaborative approach to managing complex systems. Addison-Wesley.

International Journal of Computer Science and Mobile Computing, 5, 801 – 815.

Karaman, E., & Kurt, M. (2015). Comparison of project management methodologies: prince 2 versus PMBOK for it projects. Int. Journal of Applied Sciences and Engineering Research, 4(5), 657-664.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas. (2001). Manifiesto Ágile. 07/22/2017, de agilemanifesto.org Sitio web: http://agilemanifesto.org

Laurentiis Gianni Renato. (2005). BPMS, Tecnología para la Integración y Orquestación de Procesos, Sistemas y Organización. 2017, de rrhhmagazine.com Sitio web: http://www.rrhhmagazine.com/

Letelier, P. (2006). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).

López Supelano, K. (2015). Modelo de automatización de procesos para un sistema de gestión a partir de un esquema de documentación basado en Business Process Management (bpm). Universidad & Empresa, 17(29).

Meléndez Reyes, H. (2004). Gestión de Producción. Bucaramanga: Universidad SantoTomás.

Mendes Calo, K., Estévez, E. C., & Fillottrani, P. R. (2010). Evaluación de metodologías ágiles para desarrollo de software. In XII Workshop de Investigadores en Ciencias de la Computación.

Mesías, J. L. (2006). La Comprensión de los Costos en las Organizaciones desde la Perspectiva Cualitativa (Master's thesis, Universidad EAFIT).

Mohammad Imran, Dr. Abdulrahman A. Alghamdi, Bilal Ahmad. (March 2016). SOFTWARE ENGINEERING: ARCHITECTURE, DESIGN AND FRAMEWORKS.

Namazi, M. (2009). Performance-focused ABC: A third generation of activity-based costing system. Cost management, 23(5), 34.

Namazi, M. (2016). Time-driven activity-based costing: Theory, applications and limitations. Iranian Journal of Management Studies, 9(3), 457.

Nitin Upadhyay. (2016). Systematic Decision-making Framework for Evaluation of Software Architecture. ScienceDirect, 91, 599-608.

Palisade . (2000). Simulación Monte Carlo. 18 de Julio de 2017, de Palisade Corp Sitio web: http://www.palisade-lta.com/risk/simulacion_monte_carlo.asp

Pantoja, L., & Pardo, C. (2016). Evaluando la Facilidad de Aprendizaje de Frameworks mvc en el Desarrollo de Aplicaciones Web. Publicaciones e Investigación, 10, 129-142.

Parra Castrillón, E. (2011). Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje-MESOVA. Revista Virtual Universidad Católica del Norte, (34).

Parra, V. M., Syed, A., Mohammad, A., & Halgamuge, M. N. (2016). Pentaho and Jaspersoft: A Comparative Study of Business Intelligence Open Source Tools Processing Big Data to Evaluate Performances. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 7(10), 20-29.

Poppendieck, M., & Poppendieck, T. (2003). Lean Software Development: An Agile Toolkit: An Agile Toolkit. Addison-Wesley.

Pressman, R. S. (2010). Ingeniería del Software UIn enfoque práctico, Septima edición ed.

Pressman, R. S., & Troya, J. M. (1988). Ingeniería del software.

Pressman, Roger. S. (2010). Ingeniería de Software un Enfoque Práctico. México, D.F: Mc Graw Hill.

Project Management Institute. (2013). A guide to the project management body of knowledge (PMBOK guide). Newtown Square, Pa: Project Management Institute.

Puello, O. (2013). Modelo de verificación y Validación basado en CMMI. Investigación e Innovación en Ingenierías, 1(1).

Quijada, F., & Ignacio, G. (2015). Implementación de un prototipo de la Extensión dqBP en BPMN.

Quispe, H. G. M., Capcha, R. O. T., Morales, P. A. G., & Quintana, C. M. (2017). Modelado BPMN (Business process managment notation) para la gestión de procesos. CIENCIA & DESARROLLO, (18).

Ramirez Padilla, D. N. (2008). Contabilidad Administrativa. México, D.F.: Mc Graw Hill.

Reynoso, R. N., & Esteban, F. C. L. (2010, October). Propuesta de una Metodología de BPM para el Modelado AS IS y TO BE de Procesos de Negocio de Bioseguridad (Terrorismo Alimentario), dentro del Contexto de la Cadena de Suministro. Aplicación en la Industria Mexicana Alimentaria. In 4th International Conference On Industrial Engineering and Industrial Management (pp. 258-267).

Salgado, C. H., Peralta, M., Riesco, D. E., & Montejano, G. A. (2016). Un método para la evaluación de modelos conceptuales de procesos de negocio basado en lógica difusa. In XVIII Workshop de Investigadores en Ciencias de la Computación (WICC 2016, Entre Ríos, Argentina).

Sánchez, J. (2004). Principios sobre bases de datos relacionales. Creative Commons, 1ra ED, Estados Unidos.

Sanchis, R., Poler, R., & Ortiz, Á. (2009). Técnicas para el Modelado de Procesos de Negocio en Cadenas de Suministro. Información tecnológica, 20(2), 29-40.

Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum (Vol. 1). Upper Saddle River: Prentice Hall.

Schwaber, Ken and Sutherland, Jeff, The Scrum Guide, Scrum.org, 2013

Siguenza Guzman L., Cattrysse, D. (sup.), Verhaaren, H. (cosup.) (2015). Optimal Resource Allocation and Budgeting in Libraries, 258 pp.

Sigüenza Guzmán, L., Van den Abbeele, A., & Cattrysse, D. (2014). Time-driven activity-based costing systems for cataloguing processes: a case study.

Siguenza Guzman, L., Van den Abbeele, A., Vandewalle, J., Verhaaren, H., & Cattrysse, D. (2013). Recent evolutions in costing systems: A literature review of Time-Driven Activity-Based Costing. Review of Business and Economic Literature, 58(1), 34-64.

Siguenza-Guzman, L., Van den Abbeele, A., Vandewalle, J., Verhaaren, H., & Cattrysse, D. (2014). Using Time-Driven Activity-Based Costing to support library management decisions: A case study for lending and returning processes. The Library Quarterly, 84(1), 76-98.

Stapleton, J. (1997). DSDM, dynamic systems development method: the method in practice. Cambridge University Press.

Tafur, J. C., & Osorio, J. A. (2016). Costeo basado en actividades ABC: gestión basada en actividades ABM. Ecoe Ediciones.

Tafur, J. C., & Osorio, J. A. (2016). Costeo basado en actividades ABC: gestión basada en actividades ABM. Ecoe Ediciones.

Tse, M., & Gong, M. (2009). Recognition of idle resources in time-driven activity-based costing and resource consumption accounting models. Journal of applied management accounting research, 7(2), 41-54.

Vidal Duarte, E., Morales, E., & Leger, P. (2014). Usando BPMN para modelar procesos en el área de Ingeniería y Proyectos de una empresa minera del Perú.