UNIVERSIDAD CATÓLICA SANTA MARÍA

FACULTAD DE CIENCIAS E INGENIERÍAS FÍSICAS Y FORMALES

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

ESQUEMA PARA DEFINIR HERRAMIENTAS DE SOFTWARE COMO APOYO A LA CONSTRUCCIÓN DE PROYECTOS DE SOFTWARE EN MYPES UTILIZANDO EL MODELO SISTEMATIZADO DE CALIDAD DE SOFTWARE

Tesis presentado por el Bachiller:

CARLOS EDUARDO SANCHEZ DEL ARROYO

Para optar el Título Profesional de: INGENIERO DE SISTEMAS

AREQUIPA-PERÚ

2015

PRESENTACIÓN

Sra. Directora del Programa Profesional de Ingeniería de Sistemas.

Srs. Miembros del Jurado.

De conformidad con las disposiciones del Reglamento de Grados y Títulos del Programa Profesional de ingeniería de Sistemas, ponemos a vuestra consideración el presente trabajo de investigación titulado: “ESQUEMA PARA DEFINIR HERRAMIENTAS DE SOFTWARE COMO APOYO A LA CONSTRUCCION DE PROYECTOS DE SOFTWARE EN MYPES UTILIZANDO EL MODELO SISTEMATIZADO DE CALIDAD DE SOFTWARE”, el mismo que de ser aprobado me permitirá optar el Título Profesional de Ingeniería de Sistemas.

Carlos Eduardo Sánchez del Arroyo

AGRADECIMIENTOS

A Dios por darme todo lo necesario para alcanzar cada logro y en especial este. Gracias Señor por la paciencia, fuerza, valor, amor y determinación que me diste para hacer las cosas siempre.

A mi Gran Familia (Sánchez del Arroyo y Benegas Ramos) por estar conmigo en todos los momentos de mi vida, por apoyarme y alentarme a alcanzar mis sueños. Gracias por creer en mí y darme un ejemplo de vida tan grande como lo son ustedes.

A Ana Lucia Benegas Ramos por su amor, cariño y apoyo incondicional para cumplir esta meta.

A mis profesores por brindarme sus conocimientos y experiencia que me permitieron crecer. Muchos de ustedes no sólo aportaron en mi vida conocimientos sino que también me enseñaron acerca del mundo y la realidad de la vida.

A todas las personas que de una u otra manera estuvieron a mi lado, que me enseñaron y me dieron ánimos. Gracias a todos

A la memoria de Yowana Del Arroyo Carnero y Tomas Del Arroyo Barrantes, que desde el cielo me guiaron y protegieron. A mi gran familia quienes me apoyaron todo el tiempo y me permitieron llegar a cumplir esta meta tan importante en mi vida.

Carlos Eduardo Sánchez Del Arroyo.

RESUMEN

Para construir productos informáticos se emplean modelos, métodos, metodologías y técnicas que respalden los procesos del desarrollo de software. Estos elementos le insertan características para una buena construcción asegurándonos buenos resultados. Durante la misma se utilizan un conjunto de herramientas que se convierten en una ayuda en cualquiera de las fases de elaboración. Es común que estas herramientas se empleen sin criterios tomándose solo en cuenta su funcionalidad.

El presente trabajo de investigación propone un esquema donde se presentan un conjunto de criterios que permitan seleccionar las herramientas computacionales de acuerdo a sus fases de desarrollo y al tipo de aplicación a desarrollar. Asimismo, permite una retroalimentación para las fases de análisis, diseño, construcción y pruebas de software, a partir del criterio del cliente y de la experiencia de los constructores.

La clasificación de las herramientas, según el ciclo de vida de construcción del producto software, permite mantener una estrecha relación para caracterizar el artefacto a emplear.

Palabras clave: Herramientas de software, modelo de calidad, proyecto de software, patrones de trabajo, criterios de selección, MYPES.

ABSTRACT

Computer products to build models, methods, methodologies and techniques that support software development processes are employed. These elements will insert construction features for ensuring good results. During the same set of tools that become an aid in any stage of development they are used. It is common for these tools are used not only criteria taking into account its functionality.

This paper proposes a scheme where a set of criteria to select the computational tools according to their stages of development and the type of application to be developed are presented. It also allows feedback to the phases of analysis, design, construction and testing of software, from the judgment of the customer and the experience of the builders.

The classification of the tools as the life cycle of building the software product, keeps a close relationship to characterize the device to be used.

Keywords: Software tools, quality model, software project, work patterns, selection criteria, MYPES.

INTRODUCCION

Los artefactos de software desempeñan un importante papel en la construcción de aplicaciones informáticas o sistemas de información. Como consecuencia del avance tecnológico estos han experimentado continuos cambios convirtiéndose en verdaderas soluciones para lograr diseños adecuados.

En la actualidad, las herramientas construidas cuentan con documentación donde evidencian su utilidad y trabajos de investigación que revelan avances en herramientas particulares. Obviamente, existe literatura técnica explicando la escasez de estas e identificando la necesidad de canalizar iniciativas para formular mapas conceptuales que apoyen la compresión de las mismas.

Este trabajo de investigación, formula un conjunto de criterios que identifican los tópicos relevantes para el uso de herramientas y su relación con el ciclo de vida de construcción. Así, en el capítulo 1 se presenta el planteamiento operacional; en el capítulo 2 se muestra los antecedentes de investigación y el marco teórico. El capítulo 3 presenta la descripción de la propuesta en forma detallada para que en el capítulo 4 se valide la propuesta en base a la opinión de expertos; finalmente se presentan las conclusiones y recomendaciones donde se incluyen los trabajos futuros.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

1.1 Título del proyecto

ESQUEMA PARA DEFINIR HERRAMIENTAS DE SOFTWARE COMO APOYO AL DESARROLLO DE PROYECTOS DE SOFTWARE EN MYPES UTILIZANDO EL MODELO SISTEMATIZADO DE CALIDAD DE SOFTWARE.

1.2 Descripción del problema

El desarrollo de software es un tema en el cual muchos investigadores han propuesto una serie de técnicas, métodos y modelos que permitan que tanto el producto como el proceso se puedan desarrollar siguiendo estos alcances. Cuando un desarrollador se enfrenta a este problema, puede escoger cualquier técnica, método o modelo que le ayuda a organizar el proceso de desarrollo permitiendo invertir tiempo y conseguir calidad en el producto. Estos artefactos los encontramos disponibles para todas las fases de construcción de productos de software; por lo que la documentación de la misma se hace engorrosa.

Cuando se pasa por todas las fases de la construcción de productos de software se desea tener un conjunto de herramientas de software que permitan automatizar una serie de actividades del desarrollo de software, así podemos encontrar herramientas de software para el análisis de requisitos, herramientas de software para el diseño, herramientas de software para la

construcción o de apoyo al lenguaje de programación, herramientas de software para la codificación, herramientas de software para las pruebas de software, herramientas de software para el mantenimiento y herramientas de software para la fase de post-producción del software.

El mercado presenta un gran cantidad de herramientas de software que apoyan estas actividades por lo que surge la duda de cuál herramienta de software emplear. El presente trabajo de investigación se encuentra orientado a sistematizar las herramientas de software en función de la complejidad del problema. El esquema propuesto, basado en el Modelo Sistémico de Calidad del Software (MOSCA) permitirá otorgar criterios al momento de seleccionar la herramienta de software ya sea en software libre o propietario de tal manera que a los desarrolladores les aporte calidad al momento de analizar, diseñar e implementar productos de software.

1.3 Delimitaciones

a) Delimitación espacial

El presente trabajo de investigación se lleva a cabo en la ciudad de Arequipa.

b) Delimitación temporal

El trabajo se inicia en setiembre del 2014 y culmina en marzo del 2015.

c) Delimitación social

Está orientado a resolver problemas de construcción de software seleccionando herramientas de software adecuado por lo que su orientación social implica resolver una problemática a los Ingenieros de Sistemas, Ingenieros Informáticos, Ingenieros en Ciencias de la Computación e Ingenieros de Software.

d) Delimitación conceptual

Herramientas computacionales, fases de desarrollo de software, procesos de software, arquitectura de software, construcción de software, procesos de software, calidad en el desarrollo de software.

1.4 Definición del problema

En la elaboración de productos de software nos encontramos con un conjunto de herramientas durante su construcción. Se emplean por la fuerza de la costumbre pero no por razones técnicas; y al momento de terminar el producto nos damos cuenta de las ventajas y desventajas de las mismas.

Por otro lado, aplicado el esquema propuesto basado en el Modelo Sistémico de Calidad del Software (MOSCA) se espera definir herramientas de software que ayuden en las etapas de construcción de un producto de software de tal forma que reduzcan el tiempo de construcción de estos productos en cada una de las fases de desarrollo. La herramienta escogida

puede incrementar la calidad al desarrollar productos de software.

El presente trabajo de investigación permite definir un esquema de trabajo en el tema de la selección de la herramienta de software adecuada para la elaboración de los artefactos en estas fases de construcción. El hecho de elaborar estos artefactos bajo la herramienta de software adecuada puede permitir la reducción del tiempo de construcción del producto final.

1.5 Formulación del problema

1.5.1 Problema principal

Las MYPES dedicadas al desarrollo de software comúnmente cuentan con un número reducido de trabajadores que asumen las tareas propias de la construcción del software; esto conlleva a que empleen metodologías ágiles para concluir con sus productos a tiempo dejando de lado el definir esquemas y artefactos que les sirvan de guía para la construcción de futuros productos. El presente trabajo de investigación implica proponer un esquema de trabajo, basado en el Modelo Sistémico de Calidad del Software (MOSCA), que permita definir las herramientas de software adecuadas para elaborar los artefactos en las fases de construcción de productos de software. Esta selección permitirá reducir los tiempos de construcción permitiendo una solidez en el diseño de los mismos.

1.6 Objetivos de la investigación

1.6.1 Objetivo general

Elaborar un esquema de trabajo, basado en el Modelo Sistémico de Calidad del Software (MOSCA), que permita definir herramientas de

software como apoyo al desarrollo de proyectos de software para pequeñas y medianas empresas dedicadas al desarrollo de software.

1.6.2 Objetivos específicos

- Conocer las características de las herramientas para la construcción de software, para que a partir de ellas se puedan definir aquel conjunto de herramientas de trabajo para la construcción del mismo.
- Definir un conjunto de criterios a partir de análisis del problema que permita especificar el tipo de herramientas a emplear.
- Cumplir con los indicadores propuestos para la variable dependiente.

1.7 Viabilidad de la investigación

1.7.1 Económica

Se cuenta con los recursos económicos necesarios para solventar el presente trabajo de investigación.

1.7.2 Técnica

Se cuenta con la capacidad académica y los conceptos adecuados para resolver el problema.

1.7.3 Operativa

Medios bibliográficos, Internet, bibliotecas y otras universidades con infraestructura, laboratorios, entre otros.

1.8 Justificación e importancia de la investigación

1.8.1 Justificación

Cuando se construye un producto de software, primero se recepciona, y luego se comprende, las necesidades del usuario (educción de requisitos) para luego interpretar el dominio (elicitación de requisitos) y finalmente llevar a cabo una especificación de requisitos que proporcionen buenas rutas para su codificación.

La construcción de buenos interfaces de usuario permite que los mismos entiendan las finalidades del producto de software. Los especialistas de la organización también intervienen el ello, ya sea de manera directa o indirecta con la finalidad de que los constructores no se salgan del dominio del problema a resolver.

Es común que a los desarrolladores les interese más la trazabilidad de los requisitos y que el producto cumpla con las especificaciones técnicas proporcionadas para lograr una calidad adecuada. Cabe mencionar que el análisis de los procesos para la construcción del producto de software le proporciona la estabilidad adecuada.

El emplear herramientas de software durante la construcción del producto permite un ahorro sustancial de tiempo y por efecto del mismo, de dinero. Ello implica que las mismas insertan calidad durante todas las etapas de construcción del producto final.

Finalmente, se debe de hacer hincapié y prestar atención a la evaluación del producto final ya que proporcionará información valiosa para la construcción de futuros productos de software.

1.8.2 Importancia

Permitir a los desarrolladores de software contar con un conjunto de herramientas de software que le proporcionen una ayuda sustancial en todas las etapas de construcción de productos de software. Las herramientas seleccionadas están orientadas a proporcionar artefactos de software que sigan estándares de calidad para la construcción de estos productos.

1.9 Limitaciones de la investigación

El esquema de trabajo producto del presente trabajo de investigación será aplicada por la empresa Smart Reason S.R.L. en la construcción del producto de software denominado CoMoSoft y cuyo cliente es la Cooperativa de Ahorro y Crédito MAXSER de la ciudad de Arequipa.

1.10 Hipótesis de la investigación

1.10.1 Hipótesis general

Es probable que el nuevo esquema propuesto, basado en el Modelo Sistémico de Calidad del Software (MOSCA), proporcione una ayuda en el momento de definir las herramientas para construir productos de software en la MYPES.

1.11 Variables e indicadores

1.11.1 Variable independiente

- Modelo Sistémico de Calidad de Software (MOSCA).
- Herramientas de software.

- Proyecto de software.

1.11.2 Variable dependiente

- El esquema de trabajo propuesto basado en el Modelo Sistémico de Calidad de Software (MOSCA).

1.11.3 Indicadores

- Encontrar las mejores herramientas usadas para la construcción de la aplicación.
- Proporcionar una valoración a los criterios que permitan definir las herramientas a seleccionar.
- Definir el grado de aceptación de las herramientas empleadas según los criterios propuestos en base al modelo conceptual de MOSCA.

1.12 Área, línea, tipo y nivel de la investigación

1.12.1 Área de investigación

El área de investigación es la de Ingeniería de Software.

1.12.2 Línea de investigación

La línea de investigación es la de desarrollo de software.

1.12.3 Tipo de investigación

Aplicada.

1.12.4 Nivel de investigación

Exploratorio porque es necesario obtener un conjunto de datos que permitan obtener indicadores por medio de la percepción experta.

1.13 Método y diseño de la investigación

1.13.1 Método de la investigación

Investigación aplicada, empleando el método lógico deductivo.

1.13.2 Diseño de la investigación

Para la investigación se utilizará las siguientes técnicas, instrumentos y materiales de verificación, como se muestra en la tabla 1:

Tabla 1

Variables y técnicas para la investigación

VARIABLES	TECNICAS	INTRUMENTOS DOCUMENTALES
• Modelo Sistémico de Calidad de Software • Herramientas de software • Proyecto de software	Observación	Ficha de escala de valores
• Esquema de trabajo propuesto.	Observación	Ficha de escala de valores

Fuente: Elaboración propia

1.14 Forma de tratamiento de los datos

Para el proceso de los datos provenientes de la aplicación de los instrumentos, se utilizará:

a) Matriz de Sistematización de Datos

Para consolidar los datos de la aplicación de la técnica, estos se sistematizarán de acuerdo a los rangos personalizados previstos en la definición.

b) Matriz de Tabulación

Con el fin de contabilizar las respuestas a las observaciones hechas al caso de estudio que se aplicará.

c) Cuadros Estadísticos

Elaboración de cuadros estadísticos descriptivos que permitan visualizar las respuestas correspondientes en términos de indicadores.

d) Análisis

Se hará un análisis estadístico descriptivo aplicado a los resultados obtenidos. La descripción de los cuadros de distribución de frecuencias permitirá obtener resúmenes adecuados de la información referente a los datos.

1.15 Cobertura del estudio

1.15.1 Universo

Uno de los mayores problemas que presenta nuestra ciudad es la confección de estadísticas sobre el área de tecnología, a pesar de los esfuerzos que viene realizando el gobierno peruano por medio del Instituto de Estadística e Informática, no se ha podido consolidar esta información por lo que nuestro universo se encuentra conformado por MYPES dedicadas a esta área que se encuentren trabajando actualmente y que empleen estas técnicas.

1.15.2 Muestra

Al no existir estadísticas sobre empresas que emplean este tipo de esquemas basados en MOSCA para la construcción de software basados en proyectos; el estudio se llevará a cabo únicamente en la empresa Smart Reason S.R.L. en la construcción del producto de software denominado CoMoSoft.

1.16 Esquema de trabajo

- Análisis de las herramientas de software que existen en el mercado.
- Análisis de las estadísticas de comportamiento de las herramientas computacionales empleadas en la construcción de productos de software.
- Grado de utilización efectiva de las herramientas seleccionadas por medio del Modelo Sistémico de Calidad del Software (MOSCA)
- Generación de documentación en cada una de las etapas de la construcción del software.

2.1 Antecedentes investigativos

- Pilar Rodríguez Gonzáles (Rodríguez, 2008) menciona que la forma como la industria del software mira actualmente la construcción y administración del software han hecho que se tenga que volver a plantear las bases del desarrollo de software tradicional. Bohem hace hincapié de que actualmente el mercado se encuentra representado por una rápida construcción de aplicativos. En este ambiente, generar competitividad implica incrementar el rendimiento y saciar las exigencias de la persona que necesita el software, exigencias que deben otorgarse en tiempo mínimo permitiendo incrementar el valor del negocio. Bajo este enfoque, es bueno preguntarse si las metodologías convencionales se adecuan a estos hechos. Las investigaciones concuerdan en que estas metodologías convencionales resultan engorrosas cuando se trata de cubrir las expectativas del actual mercado del software. Recientemente las metodologías ágiles han ingresado con un gran impulso como respuesta a lo complicado en la aplicación de las metodologías convencionales. Lo nuevo de las metodologías ágiles es que tienen un alto impacto cuando se trata de construir software de pequeña relevancia y resultan de aplicación crítica con proyecto de alta relevancia. Recientes investigaciones mencionan que tanto la productividad y la calidad de los productos software se intensifican cuando se encuentran asociadas con sus principios. Es por esta razón es que se necesita una mayor cantidad de estudios sobre estas metodologías para comprobar la efectividad de sus ganancias. El principal objetivo de esta investigación es observar

cómo evolucionan los productos de software al emplear los principios de las metodologías ágiles lo cual permite incrementar la calidad e incrementar su lucha dentro de la industria del software.

- Sebastián Bamonde Rodríguez (Bamonde, 2013) lleva a cabo un análisis del progreso de la tecnología informática asociada a una empresa. Este análisis lo hace producto de su experiencia lo cual le permite respaldar algunas teorías de la Ingeniería del Software así como sus soluciones arquitecturales.

- Sebastián Barbieri (Barbieri, 2008) refiere que los marcos de trabajo para mejorar los procesos son impuestos por el tipo de problemas a solucionar y por los objetivos de las instituciones que desean alcanzar. Estos marcos de trabajo consisten en procesos iterativos ordenados que permiten realizar mejoras dentro del proceso de construcción del software. El marco de trabajo que presenta no es la solución definitiva ni presenta los mismos resultados en tiempo y esfuerzo entre distintas instituciones. Este es un producto en el que se pueden trabajar diferentes modelos de referencia.

- Eduardo Diez (Diez, 2003) hace mención que cuando se trata de construir productos de software, se debe de tener en cuenta el problema para poder, de esa manera, asumir la metodología correspondiente. Una forma de darse cuenta es entender el mapa de actividades; la confección de la misma no es una tarea fácil y sencilla, requiere de habilidades y

experiencia de trabajo en el uso de metodologías. En su trabajo de investigación presenta un sistema basado en el conocimiento que apoya a jefe del proyecto de desarrollo de software elaborando el correspondiente mapa de actividades. Este software permite ingresar información relacionada con el proyecto de desarrollo de software y proporcionar el mapa de actividades sobre la metodología Métrica versión 3.

- Toni Granollers y Salvatori (Granollers & Salvatori, 2004) presenta un esquema para elaborar sistemas interactivos que unen los procesos y métodos de la Ingeniería de Software con los fundamentos de la usabilidad y la interacción humano-computador. El marco que propone fue probado experimentalmente en casos reales con el objetivo de entregar una metodología para que los equipos de trabajo puedan construir sistemas usables.

- Francisco Javier Ruíz Bertol (Ruíz, 2011) precisa que en estos momentos que el desarrollo de software requiere mayor calidad tanto en sus productos como procesos, la gestión de estos proyectos requieren formalidades que conlleven una buena planificación y estimaciones. Su trabajo de investigación estriba en el estudio de técnicas de representación en la gestión de proyectos de software desde dos ángulos distintos: a) Mejorando los actuales métodos que permitan visualizar información de los proyectos. b) Analizando técnicas ontológicas que nos

lleven a modelar proyectos de software en el nivel de restricciones así como en el razonamiento de los mismos. Su estudio presenta dos propuesta las mismas que son: a) PARMA (Project Activity Representation MAtrix), modelo que lleva a cabo la representación matricial de la información de los proyectos desarrollando una ontología que obtiene el conocimiento para el dominio de los proyectos, la misma que es empleada para llevar a cabo el razonamiento.

- Rosas Zegarra Jordan (Rosas, 2005) indica que la etapa más complejo en la construcción de aplicativos es el diseño de la base de datos. En su trabajo de investigación describe la construcción de una herramienta orientada al modelo relacional generando sentencias SQL y reconstruyendo una serie de artefactos orientados a la base de datos

- Mendoza Luís & Pérez María (Mendoza, 2005) hace referencia que ahora se están proponiendo modelos para medir la calidad de los Sistemas de Software, los cuales tienen las características para que estos sean productos o servicios de calidad que compitan en el mercado. Los modelos se formulan teniendo como sustento características competitivas y considerando la participación de personas en el proceso de desarrollo de software. Esta investigación propone el MOdelo Sistémico de CAlidad (MOSCA) para evaluar la calidad de los mismos, integrando el modelo de Calidad del Producto y el modelo de Calidad del Proceso de Desarrollo, soportado en los conceptos de la Calidad Total. MOSCA fue probado en

empresas venezolanas, utilizando el Método “Análisis de Características por Caso de Estudio”, indicado por la metodología DESMET.

2.2 Marco conceptual

2.2.1 Proyectos de software

La Gestión de Proyectos ha sufrido grandes avances desde que fue concebido inicialmente hasta nuestros días. Las grandes construcciones antiguas constituyen un amplio ejemplo de planificación y gestión. Es conocido que primero se le ha asociado a un campo específico como es la arquitectura, pero posteriormente se empezó a aplicar a otras áreas del conocimiento. Como por ejemplo, los procesos de fabricación, la realización de experimentos químicos y físicos, la observación y exploración del Universo, la agricultura, la publicación de los primeros periódicos, la logística, entre otros, son algunos de los ejemplos de la necesidad de realizar una coordinación adecuada para conseguir la meta final, y por lo tanto, es necesaria una secuencia adecuada de tareas y control sobre dichas actividades (Bamonde, 2013).

La necesidad de planificar, controlar y dirigir las actividades de un proyecto de desarrollo de software, viene marcado por el desarrollo del estudio formal de la mejora en la Gestión de Proyectos. La disciplina de Gestión de Proyectos incluye áreas de conocimiento como la Gestión de Recursos Humanos y la forma de intercambio de

información e interacción delimitada de las tareas realizadas, la distribución de tareas secuenciadas y planificación de dichas tareas, la representación adecuada de la planificación, el estudio de costes de un proyecto, la gestión de riesgos, o la planificación de las compras (Bamonde, 2013).

Aunque antiguamente la gestión de proyectos ha estado asociada a la arquitectura, en el siglo pasado el desarrollo y expansión del ser humano en diversas áreas ha hecho que la Gestión de Proyectos se extendiera a la mayor parte de las disciplinas, como la medicina, la informática, la política, entre otros. De todas estas disciplinas, la informática es la más reciente, y su exponente más destacable, donde la Gestión de Proyectos se ha convertido en fundamental, es la Ingeniería del Software. La Ingeniería del Software, con apenas cuatro décadas de singladura, constituye uno de los mayores avances de la humanidad, ya que su extensión ha sido exponencial en la población, principalmente debido al fenómeno Internet, y poco a poco se está convirtiendo en uno de los elementos necesarios para el desarrollo de las labores de la vida diaria (Bamonde, 2013).

Sin embargo, aunque la aparición de la Ingeniería del Software como disciplina y el estudio formal en la Gestión de Proyectos fue prácticamente simultáneo en los años 60, aún existen aspectos en los que divergen fuertemente. Por una parte, la Ingeniería del Software

es un área de conocimiento que es compleja, ya que similares productos con similar funcionalidad pueden desarrollarse usando diferentes metodologías, con diversas tecnologías, con interfaces distintas, con procesos totalmente opuestos, incluyendo o excluyendo temas de calidad, gestión de pruebas, validaciones y verificaciones, con ciclos de vida en el desarrollo muy diferentes, etc. Esto hace que la Ingeniería del Software carezca aún de la madurez suficiente para poder afirmar que se trata de un área de conocimiento bien establecida, con métodos formales, con desarrollos predecibles, con metodologías correctamente utilizadas, con requisitos y funcionalidades que se satisfacen, interfaces usables, conjuntos de pruebas que validen la aplicación, entre otros (Bamonde, 2013).

Por otra parte, la Gestión de Proyectos ha tenido un gran apoyo en las aplicaciones software para su desarrollo, tanto a nivel de modelos gráficos, como a nivel de control, por lo que hoy en día puede decirse que es un ´área de conocimiento casi completamente desarrollada. Pero el gran problema que surge es que la Gestión de Proyectos está proyectada hacia un ámbito genérico, en detrimento de áreas de conocimiento específicas, como por ejemplo, la Ingeniería del Software (Ruíz, 2011).

Una de las áreas de conocimiento que no aprovecha adecuadamente las herramientas y técnicas de la gestión de proyectos es

precisamente la Ingeniería del Software, debido no sólo a que los requisitos se definen de una manera vaga, las planificaciones son en muchos casos deficientes, no se realizan las pruebas para verificar los requisitos, o surgen defectos en el fase de desarrollo, sino también porque es un área donde el producto no es físicamente tangible, y depende de la tecnología. Aunque la tangibilidad del software, está siendo solventada en cierta manera por las métricas, aún hay mucho camino que recorrer en el establecimiento de la Ingeniería del Software como un área de conocimiento formalmente establecida (Ruíz, 2011).

Como se ha mencionado anteriormente, productos de similares características y con similar funcionalidad pueden haberse desarrollado de maneras completamente diferentes y con gran diferencia de costes, de tiempo y de personal. Medir la bondad de un producto o servicio software pertenece al ámbito de métricas y calidad. En Ingeniería del Software, aunque se dispone de metodologías, ciclos de vida, y procesos estándar, el control de estos es bastante variable, tanto en el fondo como en la forma. Adicionalmente, al ser un área relativamente nueva, los recursos humanos también forman parte de este descontrol existente (Bamonde, 2013).

No obstante, se puede decir que parte de la culpa proviene de la propia Gestión del Proyecto, y más concretamente de la

representación de proyectos, ya que generalmente se trata de establecer una paridad entre actividades realizadas anteriormente, y no se tiene en cuenta el hecho de que dos productos iguales pueden variar significativamente tanto en costes, tiempo y/o recursos. Uno de los factores más débiles de la representación deriva precisamente de la sencillez con la que se trata de exponer los datos históricos de los proyectos. Esto lleva a que la síntesis también provoca la pérdida de datos significativos de apoyo a la decisión (Bamonde, 2013).

a) Historia

A comienzos del siglo anterior, la Gestión de Proyectos no se consideraba un elemento fundamental en los proyectos, y por lo tanto, esta era nula en prácticamente la gran mayoría de los ámbitos. Y esto se debía a que la ejecución de los proyectos se llevaba realizando de la misma manera durante siglos, y los plazos, necesidades de materiales, necesidades de personal, el rendimiento del personal y las condiciones de trabajo habían sido eficientes, efectivas y completamente calculadas dentro de las organizaciones (Bamonde, 2013).

Además, cualquier retraso era inmediatamente solventado mediante horas extras, hasta extenuar a los trabajadores. Y es que la jornada laboral era lo suficientemente extensa, los proyectos tan sumamente simples, y los pedidos de cliente lo

suficientemente previsibles para que surgiera la necesidad de gestionar algo que muchas empresas habían estado realizando durante toda su vida. La revolución industrial del siglo XIX había asentado bien sus bases, sin riesgos, con la tasa de productividad bien ajustada –prácticamente la totalidad del trabajo era manual–, con unas condiciones laborales que separaban claramente la clase obrera de las demás clases sociales, lo que no dejaba lugar a retrasos (Bamonde, 2013).

Por entonces, los proyectos eran todo uno: se comenzaba la producción, se ejecutaba todo el trabajo a desempeñar y se obtenía el producto resultado. Sin embargo, los tiempos estaban cambiando, hacia una nueva sociedad que exigía cada vez más, con mayor calidad, con la necesidad de planificación, realizando proyectos de una manera más eficiente. Los principales hitos de esta época, históricamente hablando fueron el nacimiento de una nueva sociedad, el establecimiento de Estados Unidos como potencia mundial, y el estallido de la I Guerra Mundial (Bamonde, 2013).

b) El Diagrama Gantt

En la primera década del siglo XX, Frederic Taylor (1856-1915) había comenzado a investigar de una manera científica la manera de mejorar la productividad. La conclusión a la que llegó Taylor es que un proyecto no era una tarea monolítica, sino un conjunto de

actividades manejables relacionadas entre sí, y que para llegar a mejorar la productividad, se debía actuar individualmente sobre estas unidades de trabajo para que ´estas se ejecutaran más eficientemente. La solución que dio Taylor para lograr que las tareas se ejecutaran más eficientemente, era emplear a los trabajadores una mayor cantidad de tiempo extra (Bamonde, 2013).

Durante esa época, Henry L. Gantt, colaborador durante muchos años de Taylor, continuó esta investigación, utilizándola en la empresa Frankford Arsenal para la construcción de barcos de guerra para la I Guerra Mundial. Fue en este momento cuando desarrolló el diagrama de barras o diagrama Gantt, dividiendo los proyectos de construcción de barcos en unidades más pequeñas, denominadas tareas, y estableciendo las primeras estimaciones de duración de estas (Bamonde, 2013).

El diagrama Gantt representa las actividades de un proyecto en una barra horizontal dispuesta sobre un calendario o línea temporal. Las actividades a ejecutar se obtienen de la descomposición que se realice del proyecto, en base al nivel definido de división necesario para que dichas actividades sean manejables. Para obtener esta descomposición en actividades es recomendable utilizar el Work Breakdown Structure (WBS). La otra opción –WBS como tal no se desarrolló hasta los años 60–

fue simplemente dividir el trabajo en tareas, y más concretamente la subdivisión se basaba en las tareas que realizaba cada trabajador (Bamonde, 2013).

Las actividades obtenidas de la descomposición, se agrupaban posteriormente en el diagrama por unidades lógicas del proyecto, como fases o procesos. Para la representación de las actividades en el diagrama, no existe una notación estándar, y cada desarrollador puede establecer una notación diferente, pero siempre siguiendo unas guías determinadas por el diagrama Gantt (Bamonde, 2013).

En esta notación se puede observar que la actividad se define como una barra rectangular, situada en el diagrama Gantt utilizando una escala temporal. En este diagrama también se pueden definir agrupaciones de tareas o actividades, representadas por una barra delimitada por triángulos invertidos. La barras que representan grupos de actividades tienen la extensión correspondiente desde que se inicia la primera tarea en el tiempo de dicho grupo de actividades, hasta que finaliza la última tarea planificada. En los diagramas actuales, cada vez más es necesario realizar una monitorización de las actividades realizadas, en comparación con la planificación realizada. En este caso es cuando se utiliza la barra del estado de las actividades,

que permite predecir el progreso actual de la actividad frente a su planificación, y se representa como una actividad, pero indicando en el interior de dicha barra el progreso alcanzado en dicha actividad (Bamonde, 2013).

Este método ha sido utilizado durante la gran cantidad de proyectos desde entonces, variando muy poco desde aquella primera propuesta. A inicios de los años 90 surgió la necesidad de añadir las dependencias entre tareas, para intentar establecer la planificación óptima de los proyectos. En los últimos años, ha surgido una nueva extensión al modelo Gantt, basado en la necesidad creciente de los proyectos en realizar seguimiento (Bamonde, 2013).

El diagrama Gantt surgió de la necesidad en los comienzos del siglo XX de realizar una división de los trabajos, e intentar actuar sobre la eficiencia de las tareas individuales, para lograr mejores resultados y productos, pero también para prever la duración de los proyectos, y así tener mejor controlados tanto los tiempos de entrega como el coste de los proyectos. Partiendo del hecho de que nadie anteriormente había aplicado una planificación de estas características con éxito, el diagrama Gantt surgió como la herramienta definitiva para la planificación, utilizándose intensivamente aún hoy, donde el efecto gráfico aún sirve de

herramienta de comunicación entre los participantes del proyecto, tanto internos como externos (Bamonde, 2013).

Un diagrama Gantt consiste en una tabla bidimensional donde se listan las tareas a realizar en el proyecto, los hitos, y las dependencias que existen entre las tareas y con los hitos, junto con la representación en forma de barras o rombos escaladas en el tiempo. La organización de las tareas en el diagrama suele ser la siguiente (Bamonde, 2013):

1. Ordenación por fases o grupo de actividades relacionadas.
2. Ordenación por fecha de comienzo de las actividades.

De esta manera, se obtiene un diagrama donde las actividades se disponen en estructura de escalera dentro del diagrama, de tal manera que se pueden observar cuál va a ser el calendario planificado del proyecto, donde se listan todas las actividades a realizar en un calendario ideal (Bamonde, 2013).

Inicialmente, el diagrama Gantt solamente contenía las barras de actividad. Posteriormente, se incluyeron los hitos –proveniente del diagrama de hitos–, las dependencias y el seguimiento. Respecto a las dependencias, surgida a raíz de las necesidades de incremento de la productividad de los años 90, explican la relación

entre actividades que pueden existir entre el comienzo y el fin de actividades (Bamonde, 2013).

La más común de las dependencias que se suelen dar es Finish-to-Start, que define una secuenciación entre las tareas, y suele indicar que el producto resultado de la primera tarea (informe, código fuente, aplicación o entregable) es necesaria para realizar la siguiente tarea. Las demás dependencias de la tabla no se suelen utilizar con tanta asiduidad en los diagramas Gantt, debido a que la definición de estas dependencias implica un mayor control sobre lo que determinan las tareas, los recursos que utilizan, entre otros (Bamonde, 2013).

Por otra parte, se puede determinar el seguimiento sobre un proyecto utilizando líneas de monitorización bajo la planificación realizada, determinando el grado de finalización en un momento dado. De esta manera, el Diagrama Gantt es un sistema de representación temporal de la planificación de un proyecto de forma visual, de tal manera, que actividades y procesos están completamente determinados dentro de esta herramienta de planificación. La disposición en forma de líneas durante el gráfico ayuda a determinar dependencias y relaciones entre tareas o actividades, pudiéndose agrupar estas. Las extensiones han proporcionado las necesidades que fueron surgiendo desde su

creación, como las propias dependencias, la realización del seguimiento, o la inclusión del diagrama de hitos (Bamonde, 2013).

El diagrama Gantt es el método de representación por excelencia en la Gestión de Proyectos, siendo utilizado mayoritariamente en la totalidad de software de gestión de proyectos. Este método de representación es genérico y aplicable a cualquier ámbito, y de hecho constituye la herramienta gráfica de Gestión de Proyectos más práctica y potente de cara al cliente y entre los propios participantes del proyecto (Bamonde, 2013).

c) El diagrama de hitos (Milestone Chart).

El diagrama Gantt que conocemos en la actualidad es una combinación del diagrama Gantt creado por Henry Gantt, y el diagrama de hitos. El diagrama de hitos se basa en eventos planificados en el proyecto, en vez de en la duración de las actividades, y proporciona una medida para decidir si se continúa con el proyecto o la toma de acciones correctivas que permitan finalizar dicho proyecto en tiempo y calidad planificado. De nuevo, no hay un estándar para la notación, aunque sí que han aparecido diversas propuestas para una estandarización de la notación (Bamonde, 2013).

Desafortunadamente, el diagrama de hitos ha quedado prácticamente relegado en la gestión de proyectos debido en gran parte, a que su fusión con el diagrama Gantt, la opción más utilizada en la actualidad, se ha perdido prácticamente la utilización de este diagrama, excepto en entornos militares, donde aún se sigue utilizando, principalmente por el Departamento de Defensa. Aunque de nuevo no existe un estándar en la definición del conjunto de símbolos para la representación de hitos, se ha utilizado el conjunto de símbolos fijados (Bamonde, 2013).

d) Teoría de redes / diagramas de red.

Más adelante, se comenzó a trabajar en la teoría de grafos, desarrollada dos siglos atrás. Concretamente, Leonard Euler resolvió un problema que consistía en encontrar un camino entre los siete puentes de Konigsberg, cruzando cada puente una única vez, y volviendo al punto de inicio. Gracias a este problema, Euler resolvió que era imposible encontrar dicho camino, y lo demostró gracias a su Teoría de Grafos (Bamonde, 2013).

Sobre el año 1920, observando que el gran handicap para las escasas técnicas de planificación existentes era la propia representación de las dependencias y las restricciones, se llegó a la conclusión de que este problema se podía solucionar aplicando la teoría de grafos al problema de representación de estas

restricciones y dependencias. Determinar un paralelismo entre proyecto y la Teoría de Redes fue inicialmente en lo que se estuvo trabajando durante muchos años, desarrollándose los Diagramas de Red ya en los años 50. Tanto la Teoría de Redes, como los Diagramas de Redes han sido masivamente utilizados desde entonces en la Gestión y Representación de Proyectos, tanto como herramienta de análisis y planificación, como artefacto para la representación de los propios proyectos (Bamonde, 2013).

Para comprender los diagramas de red, primero es necesario introducir la notación específicas que permita identificar los distintos tipos de diagramas. Así, una actividad se define como una tarea específica o conjunto de tareas que son necesarias para el desarrollo del proyecto, y que consume recursos y necesita de una cierta cantidad de tiempo para ser llevada a cabo. Por el contrario, un evento es el resultado de completar una o más actividades, cuya duración es cero, y es determinable en un punto del tiempo. Y finalmente, una red es la combinación de todas las actividades y eventos que toman parte en un proyecto y las relaciones que existen entre ellos. De esta manera se pueden tener dos variantes de diagramas de red (Bamonde, 2013):

- Activity–on–Node (AoN). Determina que las actividades están representadas a través de nodos y, consecuentemente, los arcos que unen actividades representan relaciones y/o

dependencias entre actividades. El evento se toma como una actividad de duración cero.

- Activity–on–Arrow (AoA). Es el tipo de diagrama de red donde las actividades se encuentran representadas a través de flechas, y los eventos a través de nodos.

Determinar cuál se utiliza más en la gestión y análisis de proyectos depende de las herramientas utilizadas. Por ejemplo, en la herramienta de análisis PERT se utiliza AoN, mientras que en CPM se utiliza AoA. Sin embargo, ninguna de las dos herramientas se ha impuesto hasta el día de hoy, ya que han existido propuestas para la gestión de proyectos utilizando ambos tipos de diagramas de red (Bamonde, 2013).

Para la construcción del diagrama de red (vamos a utilizar el tipo AoN), se procede a listar las actividades y eventos que van a estar presente en el diagrama y las relaciones de dependencia entre ellas. Una vez determinadas las actividades y dependencias, se determina el orden lógico de las actividades observando sus dependencias. El diagrama se construye de izquierda a derecha y de arriba hacia abajo (en caso de ser necesario) (Bamonde, 2013).

e) Critical path: PERT/CPM.

En el año 1956, E.S. Slagle, por entonces presidente del Instituto de Ingenieros Eléctricos (IEE), publicó un editorial sobre la necesidad de la utilización de distribuciones de probabilidad en la planificación de los proyectos (Bamonde, 2013).

En el año 1958, tratando de optimizar los problemas de planificación, y teniendo en cuenta las líneas apuntadas por Slagle, el U.S. Navy publicó lo que hoy conocemos como Program Evaluation and Review Technique (PERT), un sistema de gestión de control para el desarrollo del programa armamentístico Polaris (Bamonde, 2013).

En el desarrollo del programa Polaris intervinieron 250 contratos de desarrollo, y más de 9.000 subcontratas, con cientos de miles de actividades a realizar. PERT se define como una herramienta de análisis para la optimización del calendario basándose en las estimaciones de tiempo establecido por tarea. PERT se basa en la teoría de grafos, desarrollada durante los años 20, para analizar y optimizar el tiempo estimado de ejecución del proyecto, cuando este tiempo es el factor fundamental del proyecto. La idea de PERT surgió para permitir a los gestores la visualización del proyecto completo, observar las relaciones y dependencias entre

tareas, y para poder reconocer cuándo y dónde los retrasos son aceptables (Bamonde, 2013).

Paralelamente al desarrollo de PERT, surgió Critical Path Method (CPM), siguiendo la misma línea de desarrollo basado en grafos, y para la obtención del camino crítico del proyecto, centrándose en mejorar y visualizar tanto el tiempo como los costes. CPM fue desarrollado por J.E. Kelly de Remington-Rand y M.R. Walter de DuPont para la programación del mantenimiento de las plantas de procesado químico (Bamonde, 2013).

Entre CPM y PERT existen ciertas diferencias, aunque el enfoque conceptual es el mismo. Una de las mayores diferencias es que PERT es un método de análisis orientada a eventos, mientras que CPM es un método orientado a las actividades, que gráficamente se observa en el diagrama de red utilizado. PERT utiliza Activity on Arrow -AoA-, mientras que CPM utiliza Activity on Node -AoN- (Bamonde, 2013).

Para la construcción de un diagrama PERT, todas las tareas a ejecutar deben ser visualizadas de una manera suficientemente clara para que ´estas puedan formar parte de una red que abarque tanto actividades como eventos. Dichas actividades se dispondrán en una secuencia en la red siguiendo unas reglas de permisividad de visualización que permitan detectar rápidamente el camino crítico (Bamonde, 2013).

Cada actividad disponen de dos tipos de referencias temporales para hallar este camino crítico: la duración óptima; y la peor duración. Basándose en estos parámetros, y los tiempos sin actividad, el camino crítico se puede definir como la secuencia de actividades lógica cuya ejecución requiera la máxima duración, comenzando desde la primera actividad hasta finalizar con la última actividad (Bamonde, 2013).

f) Estructura del desglose del trabajo (Work Breakdown Structure).

A inicios de los años 60, la necesidad imperiosa por llevar a cabo cada vez proyectos de mayor envergadura, tanto en los EE.UU. con la Guerra Fría, como en Europa, como en el resurgimiento de Japón como nación puntera en el desarrollo de sistemas de alta tecnología, hacía necesario un replanteamiento de si la división que se había realizado de los proyectos era la correcta (Bamonde, 2013).

Debido a que la complejidad de los proyectos se hizo inmanejable incluso para poder desarrollar las técnicas descritas anteriormente, el gobierno de EE.UU. comenzó a investigar sobre una subdivisión de los proyectos en curso para definir un guía que

permitiera, antes de comenzar un proyecto, definir los recursos y distribución que serían necesarios. Como consecuencia de esta investigación, se desarrolló Work Breakdown Structure (WBS) a finales de los años 60. En la actualidad se sigue utilizando como estándar para programas del sistema de defensa, estando vigente en la actualidad del MIL-HDBK-881ª (Bamonde, 2013).

Fuera de los entornos militares, WBS ha sido utilizado como herramienta para la división del trabajo en proyectos que por su extensión o duración, hacían necesaria una división adecuada de las tareas a realizar (Bamonde, 2013).

El principal objetivo a la hora de estructurar el proyecto, es que los elementos en los que divide el proyecto deben ser:

- Gestionables, donde se puede asignar una autoridad y responsabilidad específica,
- Independientes, con mayor o menor dependencia y/o relación con los demás elementos,
- Integrable, de tal manera que el conjunto forme el proyecto, y mensurable, donde puedan determinarse los costes, los recursos necesarios, y evaluar el progreso.

El Diagrama de Descomposición del Trabajo (Work Breakdown Structure - WBS) es una subdivisión orientado al producto de la

familia de los ´arboles de los elementos físicos, servicios y datos necesarios para obtener el producto final. La principal aportación de WBS no es la mera división en tareas manejables del proyecto, sino que proporciona un marco de trabajo que puede ser posteriormente utilizada en otros entregables del proyecto, como la matriz de responsabilidad, la planificación, el presupuesto, el análisis de riesgos, la estructura de la organización, la coordinación de objetivos, el seguimiento o el control del proyecto (Bamonde, 2013).

g) Redes de petri (Petri Nets).

Los años 60 fue una época en la que Institutos, Departamentos del Gobierno, Universidades y empresas privadas proponían nuevos modelos para la representación de procesos, generalmente centrados en la solución de problemas particulares, y sin posibilidad de extensión a otros ámbitos. Sin embargo, Carl Adam Petri, presentó su tesis: ”Kommunikation mit automaten” (Comunicación con autómatas) en el Instituto para Instrumentos Matemáticos de la Universidad de Bonn, donde presentaba un novedoso modelo del flujo de información en sistemas, denominado Petri-nets (Bamonde, 2013).

Desde la aparición de esta tesis en 1962 hasta la actualidad, las Redes de Petri o Petri–Nets han sido una base para el estudio y

desarrollo de muchos trabajos de investigación relacionados con el modelado de sistemas y la aplicación de estos. Una Red de Petri es un grafo bipartito dirigido que puede tener dos tipos de nodos: los lugares o places (P) representados por círculos, y las transiciones o transitions (T) representadas por líneas. Las conexiones se realiza a través de las funciones de entrada (I) y salida (O), representadas por arcos dirigidos desde los places hasta las transiciones, y desde las transiciones hasta los lugares, respectivamente. Las conexiones entre el mismo tipo de nodo no está permitida. Las marcas de las Petri- Nets son asignaciones de elementos o tokens a los lugares de la red. Las marcas se representan como círculos rellenos dentro de los lugares (Bamonde, 2013).

h) El modelo IDEFØ.

IDEFØ (Integration DEFinition for Function Modeling) es un método basado en la combinación de una representación gráfica de funciones y un texto explicativo para el modelado de decisiones, acciones, y actividades de una organización o sistema. IDEFØ forma parte de un conjunto de métodos definidos para el diseño y modelado del software. Entre los métodos más importantes se encuentran IDEFØ (Modelado de funciones), IDEF1 (Modelado de la información), IDEF1X (Modelado de datos), IDEF3 (Método de Captura de Descripción del Proceso),

IDEF4 (Método de Diseño Orientado a Objetos) e IDEF5 (Método de Captura de Descripción de Ontologías). El modelo IDEFØ se basa en SADT (Structured Analysis and Design Technique), que determina un modelado de las funciones para el análisis y comunicaciones de un sistema (Bamonde, 2013).

El principal objetivo de IDEFØ es, por una parte, el desarrollo de una representación gráfica estructurada de las funciones (actividades, procesos, acciones u operaciones) de un sistema o proyecto que soporte la integración de sistemas, y por otra, la definición de un método de modelado independiente de las herramientas CASE (Computer Aided Software Engineering). IDEFØ se ha diseñado para que se pueda utilizar conjuntamente con herramientas CASE, y para proporcionar un método de modelado que posea las características deseables de ser general, precisa, concisa, y que proporcione la conceptualización y flexibilidad (Bamonde, 2013).

i) Herramientas de software

En el siglo pasado innumerables áreas de Tecnología han tenido progresos considerables, pero una destaca sobre las demás no porque haya dejado de existir o por que se haya convertido en una innovación radical, sino porque ha cambiado tanto que

apenas es reconocible a la situación en la que se encontraba hace 10 años: la Administración de Proyectos (Galeano. 2007).

Aun cuando la experiencia en la administración de proyectos se ha desarrollado considerablemente, la necesidad de poder administrar un número cada vez más grande de proyectos con características variables y disruptivas, que además se encuentran en diferentes fases dentro de su ciclo de vida, presenta nuevos y difíciles retos en las organizaciones. Las tendencias de competencia global, cambios tecnológicos y reingenierías cada vez más rápidas incrementan la importancia de los procesos de administración de proyectos, si consideramos al administrador de proyectos y a su equipo como un agente de cambio, debido a la esencia “temporal” del proyecto (Galeano. 2007).

Hablando del desarrollo de software es posible mencionar que los proyectos de software se encuentran pobremente administrados. Frecuentemente se retrasan o sobrepasan lo presupuestado inicialmente (se estima un factor del 50 al 100%), además de que los clientes o usuarios de la misma manera se muestran insatisfechos con la calidad de los sistemas de software. Es por esto que no es de sorprender que las organizaciones de desarrollo de software busquen activamente nuevas maneras de mejorar su desempeño (Galeano. 2007).

j) Las tecnologías de información y su relación con los cambios en la administración de proyectos.

Tanto ha cambiado la manera de Manejar la Administración de un proyecto que es difícil identificar inclusive las herramientas que están disponibles para apoyar esta actividad. Hoy en día no solo se habla de aplicaciones que emplean los tradicionales Diagramas de Gannt, PERT y capacidades para elaborar reportes rápidamente, sino que se habla de soluciones que las empresas emplean como herramientas de apoyo que no se hacen llamar formalmente como herramientas de Administración de Proyectos. De acuerdo con lo anterior se habla de herramientas de administración de procesos, de administración de portafolios, de administración de conocimiento, portales, e inclusive Groupware y Sistemas Colaborativos integrados con otras aplicaciones completamente distintas, las cuales ahora son consideradas como Herramientas de Administración de Proyectos (Caballero, 2006).

A lo largo de esta sección se discutirán algunos cambios relevantes que han surgido a raíz de la evolución de la tecnología y las necesidades que surgen como consecuencia de las nuevas tendencias para aumentar la competitividad dentro de una organización que desarrolla proyectos de software (Caballero, 2006).

k) Herramientas y técnicas de apoyo a la administración de proyectos.

En esta sección se describen algunas iniciativas, marcos de referencia, y aplicaciones que surgen como consecuencia de los cambios percibidos dentro de la Administración de Proyectos de Software y que además complementan y dan soporte a las actividades que abarca. Dentro de estas herramientas se encuentran algunas iniciativas en disciplinas como la administración de la calidad y la administración del conocimiento. Finalmente, y como una última consideración a estas soluciones e iniciativas, se analiza la posibilidad de adoptar herramientas de software por parte de la organización (Caballero, 2006).

l) Administración de la calidad del proceso de desarrollo.

Tal como se pudo observar anteriormente, el aspecto de calidad del proceso de desarrollo ha tomado mayor importancia dadas las tendencias competitivas actuales. De la misma manera, existen modelos de mejora de calidad como herramientas y marcos de referencia. Entre éstos se encuentra la Mejora del Proceso de Software, el cual dentro de las iniciativas contemporáneas es la más ampliamente utilizada para mejorar el desempeño de las organizaciones desarrolladoras de software (Caballero, 2006).

La mejora del proceso de software (SPI de sus siglas en inglés: Software Process Improvement) es una iniciativa de mejora de la capacidad de una organización para proporcionar servicios de calidad de una manera competitiva, el cual tiene una naturaleza cíclica y evolutiva que involucra las siguientes actividades: Iniciación. Involucra el desarrollar planes, agendas, e infraestructura. Diagnóstico. En esta etapa se evalúa el nivel de madurez actual de la organización para el desarrollo de software. Establecimiento. Diseño de proyectos de mejora en base a los resultados de la actividad anterior. Acción. Implementación de los proyectos de mejora de las prácticas de la organización Aprendizaje. Revisión de las lecciones aprendidas durante el proceso (Caballero, 2006).

m) Aplicaciones de software de apoyo a la administración de proyectos.

El software de administración de proyectos, es un término que cubre muchos tipos de software, incluso programación, asignación de recursos, software de colaboración, comunicación y sistemas de documentación, que están acostumbrados al trato con la complejidad de proyectos grandes (Caballero, 2006).

Las herramientas de software son requeridas para automatizar y facilitar la aplicación de la metodología particular de la

organización para la administración de proyectos. Esta metodología incluye cómo se organiza para manejar sus proyectos, qué prácticas son necesarias para llevar a cabo su administración y sus procesos, y además que requerimientos se tienen en relación con su cultura organizacional (Caballero, 2006).

Una vez que se determinan y documentan estas necesidades, es posible evaluar que aplicaciones de software cumplen con dichos criterios y en base a esta evaluación realizar una selección (Caballero, 2006).

La nueva generación de herramientas de Tecnologías de Información de administración de proyectos combinan las tres S's: scope, scheduling y status. Es decir, herramientas para administrar el alcance, la programación de tareas y el estado en que se encuentran. Por otra parte, y a diferencia de las herramientas desarrolladas anteriormente, estas tienen la característica de incorporar el poder de tecnologías basadas en Internet (Caballero, 2006).

2.2.2 Modelo de calidad MOSCA

El modelo se basa en los modelos de Calidad del Producto de Software de Ortega et. Al y Estructura del Modelo de la Calidad del Software del Proceso de Desarrollo de Software de Pérez et. El

propone la solución del problema enfocando cuatro niveles de solución:

Nivel 0: Dimensiones. Aspectos Internos del proceso, Aspectos Contextuales del proceso, Aspectos Internos del producto y Aspectos Contextuales del producto son las cuatro dimensiones propuestas en el prototipo de modelo. Sólo un balance y una buena interrelación entre ellas garantizan la calidad Sistémica global de una organización

Nivel 1: Categorías. Se contemplan once categorías: seis (6) pertenecientes al producto y cinco al proceso de desarrollo. Esta división no implica un desligamiento entre ellas, simplemente se realiza para identificar a que sector o sub-modelo pertenecen.

Nivel 2: Características. Cada categoría tiene asociado un conjunto de características, las cuales definen las áreas claves a satisfacer para lograr, asegurar y controlar la calidad tanto en el producto como en el proceso. Entre las características asociadas a cada categoría del producto, se proponen una serie de características del proceso. Esto se debe, a que algunas características de la calidad del proceso, impactan directamente en las categorías del producto al igual que ciertas características de la calidad del producto definen categorías del proceso. Esto ayuda a precisar que si una vez medidas las

características asociadas a una categoría en particular del producto, arroja resultados no deseados, se pueden analizar las características de la calidad del proceso asociadas a esa categoría del producto para encontrar las posibles causas.

Nivel 3: Métricas. Para cada característica se propone una serie de métricas utilizadas para medir la calidad sistémica. Dada la cantidad de métricas asociadas a cada una de las características que conforman MOSCA las cuales son 679.

El proceso de aplicación se muestra en la figura 1.

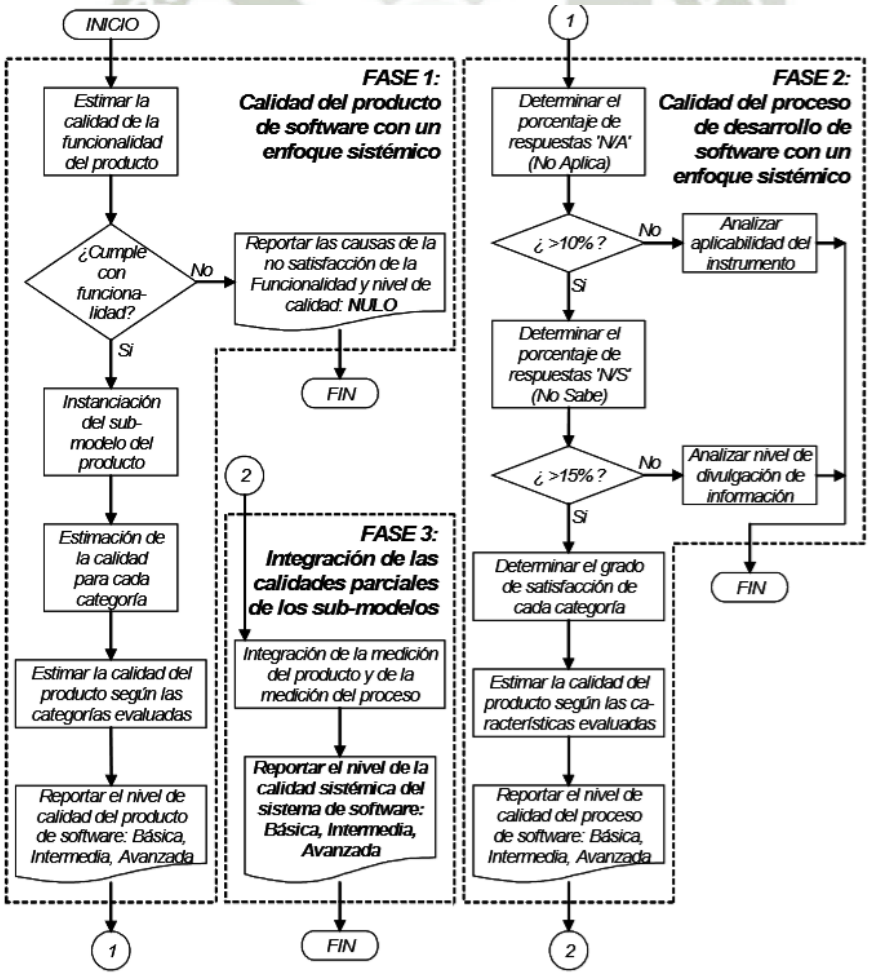

Figura 1. Algoritmo de aplicación de MOSCA
Fuente: [Mendoza, 2005]

2.2.3 Etapas del desarrollo de software

1. Proyectos de software

a. Artefactos de uso común

- Diagrama de Gantt

  Herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

b. Artefactos de uso esporádico

- Diagrama PERT

  Herramienta para representar gráficamente las relaciones entre las tareas del proyecto y calcular los tiempos del mismo de forma ágil y sencilla.

- Diagrama CPM

  Artefacto que hace un seguimiento del desarrollo y control de proyecto. El objetivo principal es determinar la duración de un proyecto, entendiendo éste como una secuencia de actividades relacionadas entre sí, donde cada una de las actividades tiene una duración estimada.

- Diagrama de la cadena crítica

  Artefacto empleado en el cálculo de tiempos y plazos para la planificación de proyectos.

c. Criterios

- Estimación de costos

- Estimación de hitos de control
- Estimación de la planificación
- Seguimientos del proyecto en bases a sus actividades
- Estimación de riesgos asociados
- Estimación del esfuerzo
- Generación de artefactos o documentación correspondiente

2. Análisis de sistemas

a. Artefactos de uso común

- Plantilla de educción de requisitos

  Artefacto empleado para plasmar los criterios principales de la lógica del negocio vista desde el punto de vista del cliente.

- Plantilla de elicitación de requisitos

  Plantilla donde el Ingeniero de Requisitos plasma los criterios de la lógica del negocio desde el ángulo de su propio punto de vista y manteniendo una relación con la plantilla de educción de requisitos para permitir una trazabilidad adecuada.

- Plantilla de especificación de requisitos de software

  Plantilla donde el Ingeniero de Requisitos transforma la lógica del negocio en recursos computacionales para ser implementados por el codificador del producto.

- Plantilla de casos de uso esenciales

  Artefacto donde se plasma toda la información correspondiente a la representación de los elementos primarios del análisis del sistema; en ellos se muestran

características importantes del modelo del negocio a automatizar.

- Diagrama de casos de uso esenciales

  Estos diagramas muestran operaciones que se esperan de una aplicación o sistema y como se relaciona con su entorno, es por ello que se ve desde el punto de vista del usuario. Describen un uso del sistema y como éste interactúa con el usuario.

- Diagrama de flujo de datos

  Muestra el flujo de información producto de entradas y salidas en cada uno de los procesos a ser automatizados o auxiliares al producto de software.

- Diagrama de secuencias

  Muestra una interacción ordenada según la secuencia temporal de eventos y el intercambio de mensajes. Los diagramas de secuencia ponen especial énfasis en el orden y el momento en el que se envían los mensajes a los objetos.

- Plantilla de contratos de software

  Plantilla donde se plasma la relación recurso computacional – código del producto.

- Diagrama de actividades

  Un diagrama de actividades representa un flujo de trabajo paso a paso de negocio y operacionales de los componentes en un sistema.

- Diagrama de bloques

Diagrama donde se puede plasmar información genérica de la construcción de un producto de software.

- Diagrama de clases de análisis

  Tiene como fin describir la estructura de un sistema mostrando sus clases, atributos y relaciones entre ellos.

b. Artefactos de uso esporádico

Artefactos que son empleados de manera esporádica y que son sustitutos de algunas ya empleadas.

- Plantilla de tarjetas CRC

  Plantilla donde se plasma la información recolectada de la Ingeniería de Requisitos; mantiene los criterios de los casos de uso esenciales.

- Plantilla de trazabilidad de requisitos

  Herramienta que permite encadenar todos los requisitos, de cada uno de sus etapas, los objetivos de la construcción y la codificación del mismo.

- Plantilla de métricas de software

  Plantilla donde se plasman los criterios de deducción de indicadores que permitirá deducir las métricas correspondientes al proyecto en construcción.

c. Criterios

- Construcción de diagramas según modelo escogido
- Especificación de prototipos basados en la etapa de requisitos
- Especificación de requisitos en las etapas de educción, elicitación y especificación

- Modificación de requisitos en sus etapas de educción, elicitación y especificación
- Importación de requisitos bajos formatos diferentes al de la herramienta
- Organización de requisitos de acuerdo a diferentes criterios
- Análisis de la trazabilidad de los requisitos
- Importación de objetos
- Exportación de objetos
- Generación de los artefactos correspondientes

3. Diseño de sistemas

a. Artefactos de uso común

- Plantilla de casos de uso reales

  Plantilla donde quedan explicitadas las condicionalidades de la lógica del negocio para ser implementadas en la codificación del producto.

- Diagrama de casos de uso reales

  Esquema que representa las relaciones de casos de uso reales lo que implica la relación entre los recursos computacionales a ser implementados en la codificación.

- Plantilla de reportes

  Plantilla que permite especificar la forma de presentación de los datos y sus diagramas, tanto en video como el papel.

- Diagrama del modelo de navegabilidad

Diagrama que visualiza el secuenciamiento de aparición de los interfaces gráficos de usuario permitiendo una correcta codificación.

- Plantilla de diseño de interfaces de usuario

  Plantilla que permite integrar las definiciones de la forma y el color para representar los interfaces gráficos que tomarán contacto con el usuario.
- Diagrama de bloques

  Artefacto que permite representar esquemas genéricos de la construcción de productos de software.
- Diagrama de colaboración

  Es una forma alternativa al diagrama de secuencias a la hora de mostrar un escenario. Este tipo de diagrama muestra las interacciones que ocurren entre los objetos que participan en una situación determinada. A diferencia del diagrama de secuencia, el diagrama de colaboración se enfoca en la relación entre los objetos y su topología de comunicación.
- Diagrama de clases de diseño

  Diagrama que representa las soluciones de construcción de los productos de software; permite una codificación clara y elegante del producto.
- Diagrama entidad-relación

Artefacto que representa las relaciones y cardinalidades de entidades que conformarán las tablas de las bases de datos; asimismo permite conocer las llaves de conectividad.

b. Artefactos de uso esporádico

- Diagrama de objetos

  Modelan las instancias de las clases del Diagrama de Clases. Este diagrama cuenta con objetos y enlaces y es posible encontrar las clases para tomar como referencia su instanciación.

- Diagrama de estados

  El diagrama de estados engloba todos los mensajes que un objeto puede enviar o recibir, en otras palabras es un escenario que representa un camino dentro de un diagrama.

- Diagrama de componentes

  Normalmente contiene componentes, interfaces y relaciones entre ellos. Los componentes perteneces a un mundo físico, es decir, representan a un bloque de construcción al modelar aspectos físicos de un sistema.

- Diagrama de despliegue

  Este tipo de diagrama se utiliza para modelar el Hardware utilizado en la implementación del sistema y las relaciones entre sus componentes.

c. Criterios

- Construcción de diagramas de diseño según modelo escogido
- Especificación de prototipos basados en la etapa de análisis

- Modificación de requisitos en sus etapas de educción, elicitación y especificación a partir de la especificación del diseño
- Modificación de la información de acuerdo a criterios de análisis y su reinserción en la documentación de análisis
- Análisis de la trazabilidad de los requisitos hasta el diseño
- Generación de código
- Generación de los artefactos o documentos correspondientes

4. Construcción de software

a. Artefactos de uso común

- Plantilla de clases

  Plantilla con especificaciones del recurso computacional a emplear en la codificación del producto.

- Plantilla de interfaces

  Plantilla que permite la visualización de los interfaces de las clases, atributos y métodos permitiendo una codificación clara y segura.

- Plantilla de diseño de reportes

  Plantilla que permite el diseño del tipo de reporte que el usuario desea para su producto. En el diseño se mantiene las características de calidad de la información.

- Plantilla de implementación de bases de datos

Plantilla que permite especificar la forma de construcción de la base de datos que interactuará con el producto de software a construir.

- Plantilla de codificación

  Plantilla que permite plasmar los elementos de codificación bajo esquemas específicos y elementos de calidad en la respectiva codificación.

b. Artefactos de uso esporádico

- Artefactos generadores de código

  Artefactos que permiten generar el correspondiente código bajo la especificación de algún lenguaje de programación.

c. Criterios

- Especificación del control de acceso a los datos incluyendo la especificación de componentes
- Especificación de copias de la codificación del producto
- Actualización de copias producto del versionamiento
- Especificación del control de cambios
- Soporte de repositorios como medio del control de la seguridad
- Integración con otras herramientas de diseño
- Generación de artefactos o documentos correspondientes

5. Pruebas de software

a.1 Artefactos de uso común

- Plantilla de pruebas de caja negra

Plantilla donde se especifican todas las pruebas para los interfaces gráficos con los que interactúa el usuario final. Se relaciona con una plantilla donde se plasman las soluciones correspondientes y sus alternativas de solución.

b.1 Artefactos de uso esporádico

- Plantilla de pruebas de caja blanca

  Plantilla donde se especifican las pruebas inherentes a los módulos objetivo de control.

- Plantilla de pruebas de complejidad ciclomática

  Plantilla donde se especifican las clases de equivalencia que permite una prueba contundente a la codificación del producto.

- Plantilla de pruebas de integración

  Plantilla donde se plasma las pruebas para las relaciones entre módulos producto.

- Plantilla de pruebas unitarias

  Plantilla donde se plasma las condiciones para llevar a cabo las pruebas específicas a aquellos módulos, métodos o rutinas que mantienen la conectividad para el producto.

c.1 Criterios

- Capacidad de asumir el plan de pruebas en formatos diferentes
- Especificación de la diversidad de pruebas que pueden llevar a cabo a partir del plan

- Especificaciones gráficas de la prueba y generación
- Generación de artefactos o documentos correspondientes

3.1 Introducción

El esquema propuesto consiste en cuatro etapas, las mismas que guardan una interrelación con el ciclo de vida de la construcción del software. En la primera etapa se define el proyecto de desarrollo de software logrando concebir el análisis, diseño, construcción y las respectivas pruebas de software.

La segunda etapa permite escoger los criterios de selección de las herramientas a emplear en la construcción del software; en primer lugar se definen los criterios de contexto, interno, conceptuales del producto, conceptuales del proceso y los del cliente. Solo los cuatro primeros criterios retroalimentan al análisis, diseño, construcción y definición de pruebas de software. Los criterios del cliente son tomados como punto de partida para concebir el problema y sus herramientas a emplear.

La tercera etapa consiste en la clasificación de las herramientas de software en función de su licenciamiento guardando una estrecha relación con la validación del esquema desde el punto de vista de la ingeniería y los equipos de trabajo. La figura B.1 muestra gráficamente el ciclo de vida del esquema propuesto. La cuarta etapa presenta los criterios para validar el esquema propuesto.

3.2 Resumen del esquema

3.2.1 Etapa N° 1: Definición de patrones

El esquema comienza con la preparación de proyecto de software, a partir de ello se comienza analizando y luego diseñando las soluciones del producto a construir. Seguidamente se construye y se prueba el producto. Si en alguna etapa se encuentra error alguno, se puede pasar a cualquiera de las etapas anteriores para resolver el problema. Esta operación otorga la suficiente idea para determinar con qué tipo de herramientas de software se debe de contar.

En cada etapa se determinan un conjunto de patrones que implican el motivo de uso de la herramienta de software escogida como solución al problema de construcción del producto.

3.2.2 Etapa N° 2: Criterios de selección

En esta etapa se toman en cuenta los criterios de contexto, internos, contextuales del proceso y los del cliente. Los criterios de contexto permiten obtener una opinión sobre aquellos factores externos donde funcionará el producto, los internos dan la idea de cuáles son las herramientas que servirán de apoyo durante el proceso de desarrollo del producto; los contextuales del proceso permiten definir herramientas de apoyo al proceso de desarrollo y los criterios del cliente darán una visión sobre las fortalezas de las herramientas para resolver las necesidades particulares del cliente.

Se debe tener en cuenta que los criterios pueden ser diferentes para muchas aplicaciones informáticas por lo que se debe de trabajar en la formación de una base de datos que sirve como punto de partida para proyectos futuros.

3.2.3 Etapa N° 3: Clasificación de herramientas de software

Para definir las herramientas de software a emplear también se toman en cuenta el tipo del mismo: libre o propietario. Ello implica un factor directo en el costo del producto ya que se tienen que adquirir herramientas para emplearlas en la construcción de las aplicaciones informáticas.

3.2.4 Etapa N° 4: Validación del esquema

En esta etapa se lleva a cabo la validación del esquema en función de las herramientas de software escogidas. Cada herramienta de software escogida debe tener alguna puntuación para poder determinar cuál ha sido el nivel de servicio del uso de la herramienta de software.

También se lleva a cabo por medio de una validación subjetiva que sirva de punto de partida para lograr una clasificación de herramientas adecuadas para cada una de las etapas de construcción del producto así en la etapa del proceso de desarrollo del producto.

.

3.3 Etapas

Etapa N° 1: Definición de patrones

En la presente etapa se definen los artefactos que comúnmente se emplean en la construcción de productos de software. Aunque los productos a construir sean diferentes como por ejemplo sistemas de información administrativos, sistemas de tiempo real, sistemas distribuidos entre otros, siempre existen un conjunto de artefactos que comúnmente se emplean para construir estos productos. Estos artefactos constituyen patrones a ser empleados durante la construcción de la mayoría de los productos de software. Como elemento referencial se toma en cuenta los aspectos internos del proceso como lo propone MOSCA.

La tabla 2 muestra los artefactos que permiten determinar patrones de trabajo en la primera etapa del esquema. Esto permitirá determinar las herramientas de software necesarias que se usarán en cada una de las etapas previstas en el proceso de desarrollo del proyecto de software. Es necesario especificar que el uso de los artefactos de uso común así como los de uso esporádico va a depender del tipo de software a construir así como la de sus complejidades inherentes del problema a resolver.

Tabla 2

Artefactos empleados en la construcción de productos de software

Área	Artefactos	
	Uso común	**Uso esporádico**
Proyecto de software	Diagrama de Gantt	Diagrama PERT Diagrama CPM Diagrama de cadena crítica
Análisis de sistemas	Plantilla de educción de requisitos Plantilla de elicitación de requisitos Plantilla de especificación de requisitos Plantilla de casos de uso esenciales Diagrama de flujo de datos Diagrama de secuencias Plantilla de contratos de software Diagrama de actividades Diagrama de bloques Diagrama de clases de análisis	Plantilla de tarjetas CRC Plantilla de trazabilidad de requisitos Plantilla de métricas de software
Diseño de sistemas	Plantilla de casos de uso reales Diagrama de casos de uso reales Plantilla de reportes Diagrama del modelo de navegabilidad Plantilla de diseño de interfaces de usuario Diagrama de bloques Diagrama de colaboración Diagrama de clases de diseño Diagrama entidad - relación	Diagrama de objetos Diagrama de estados Diagrama de componentes Diagrama de despliegue
Construcción de software	Plantilla de clases Plantilla de interfaces Plantilla de diseño de reportes Plantilla de implementación de bases de datos Plantilla de codificación	Artefactos generadores de código
Pruebas de software	Plantilla de pruebas de caja negra	Plantilla de pruebas de caja blanca Plantilla de pruebas de complejidad ciclomática Plantilla de pruebas de integración Plantilla de pruebas unitarias

Fuente: Elaboración propia

a) Proyectos de software

a.1 Artefactos de uso común

- Diagrama de Gantt

  Herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

a.2 Artefactos de uso esporádico

- Diagrama PERT

  Herramienta para representar gráficamente las relaciones entre las tareas del proyecto y calcular los tiempos del mismo de forma ágil y sencilla.

- Diagrama CPM

  Artefacto que hace un seguimiento del desarrollo y control de proyecto. El objetivo principal es determinar la duración de un proyecto, entendiendo éste como una secuencia de actividades relacionadas entre sí, donde cada una de las actividades tiene una duración estimada.

- Diagrama de la cadena crítica

  Artefacto empleado en el cálculo de tiempos y plazos para la planificación de proyectos.

a.3 Criterios

- Estimación de costos
- Estimación de hitos de control
- Estimación de la planificación
- Seguimientos del proyecto en bases a sus actividades
- Estimación de riesgos asociados
- Estimación del esfuerzo
- Generación de artefactos o documentación correspondiente

b) Análisis de sistemas

b.1 Artefactos de uso común

- Plantilla de educción de requisitos
- Plantilla de elicitación de requisitos
- Plantilla de especificación de requisitos de software
- Plantilla de casos de uso esenciales
- Diagrama de casos de uso esenciales
- Diagrama de flujo de datos
- Diagrama de secuencias
- Plantilla de contratos de software
- Diagrama de actividades
- Diagrama de bloques
- Diagrama de clases de análisis

b.2 Artefactos de uso esporádico

- Artefactos que son empleados de manera esporádica y que son sustitutos de algunas empeladas.
- Plantilla de tarjetas CRC
- Plantilla de trazabilidad de requisitos
- Plantilla de métricas de software

b.3 Criterios

- Construcción de diagramas según modelo escogido
- Especificación de prototipos basados en la etapa de requisitos
- Especificación de requisitos en las etapas de educción, elicitación y especificación

- Modificación de requisitos en sus etapas de educción, elicitación y especificación
- Importación de requisitos bajos formatos diferentes al de la herramienta
- Organización de requisitos de acuerdo a diferentes criterios
- Análisis de la trazabilidad de los requisitos
- Importación de objetos
- Exportación de objetos
- Generación de los artefactos correspondientes

c) Diseño de sistemas

c.1 Artefactos de uso común

- Plantilla de casos de uso reales
- Diagrama de casos de uso reales
- Plantilla de reportes
- Diagrama del modelo de navegabilidad
- Plantilla de diseño de interfaces de usuario
- Diagrama de bloques
- Diagrama de colaboración
- Diagrama de clases de diseño
- Diagrama entidad-relación

c.2 Artefactos de uso esporádico

- Diagrama de objetos
- Diagrama de estados
- Diagrama de componentes

- Diagrama de despliegue

c.3 Criterios

- Construcción de diagramas de diseño según modelo escogido
- Especificación de prototipos basados en la etapa de análisis
- Modificación de requisitos en sus etapas de educción, elicitación y especificación a partir de la especificación del diseño
- Modificación de la información de acuerdo a criterios de análisis y su reinserción en la documentación de análisis
- Análisis de la trazabilidad de los requisitos hasta el diseño
- Generación de código
- Generación de los artefactos o documentos correspondientes

d) Construcción de software

d.1 Artefactos de uso común

- Plantilla de clases
- Plantilla de interfaces
- Plantilla de diseño de reportes
- Plantilla de implementación de bases de datos
- Plantilla de codificación

d.2 Artefactos de uso esporádico

- Artefactos generadores de código

d.3 Criterios

- Especificación del control de acceso a los datos incluyendo la especificación de componentes
- Especificación de copias de la codificación del producto
- Actualización de copias producto del versionamiento
- Especificación del control de cambios
- Soporte de repositorios como medio del control de la seguridad
- Integración con otras herramientas de diseño
- Generación de artefactos o documentos correspondientes

e) Pruebas de software

e.1 Artefactos de uso común

- Plantilla de pruebas de caja negra

e.2 Artefactos de uso esporádico

- Plantilla de pruebas de caja blanca
- Plantilla de pruebas de complejidad ciclomática
- Plantilla de pruebas de integración
- Plantilla de pruebas unitarias

e.3 Criterios

- Capacidad de asumir el plan de pruebas en formatos diferentes
- Especificación de la diversidad de pruebas que pueden llevar a cabo a partir del plan
- Especificaciones gráficas de la prueba
- Generación de artefactos o documentos correspondientes

Etapa N° 2: Criterios de selección

En esta etapa se toma en cuenta las sugerencias que entrega MOSCA en lo que respecta a los aspectos internos del producto. Estas sugerencias deben tratarse bajo un conjunto de criterios que permitan, posteriormente, encontrar la herramienta adecuada. La figura 2 muestra el proceso de adecuación de criterios.

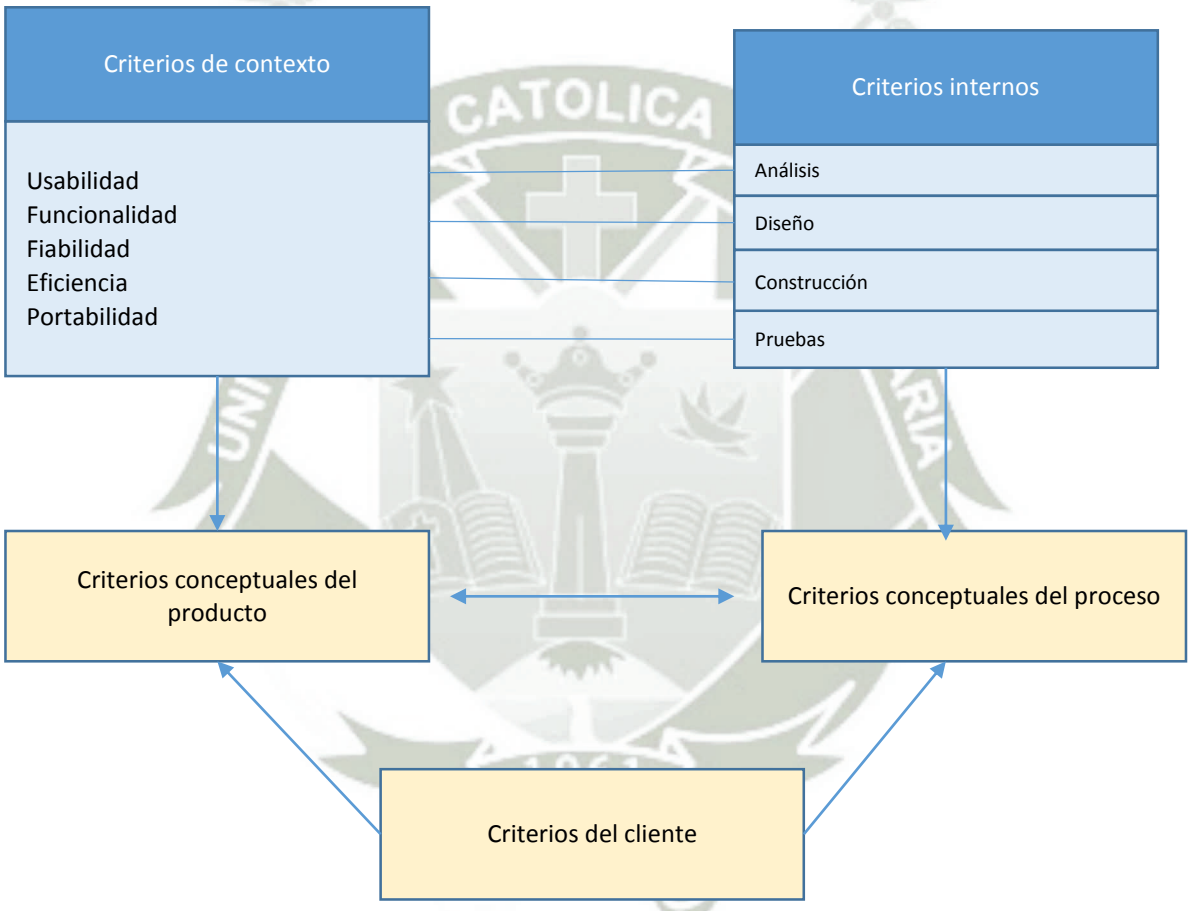

Figura 2. Proceso de adecuación de criterios

Fuente: Elaboración propia

a) Criterios de contexto

Para elegir una herramienta computacional cuando se trata de proyectos de desarrollo de software se debe tener en cuenta aspectos como:

- Software online. Herramientas disponibles desde algún sitio web, estas herramientas, en la actualidad, ya han alcanzado el nivel funcional a las herramientas “tradicionales” más consolidadas.
- Fácil, usable e intuitivo. Este es un aspecto fundamental que permitirá hacer un primer filtro en la evaluación de cualquier aplicación.
- Versión o periodo de prueba. La mejor forma de acertar en la elección es probarlo por uno mismo. Por eso, es importante elegir aplicaciones que cuenten con versión o periodo de prueba para poder hacer una evaluación previa.
- Escalabilidad. Orientada a fundamentar los distintos planes de precios ofertados y tener visión de largo plazo. Es importante conocer si el proveedor ofrece planes más avanzados que el que se haya elegido en un primer momento, que permitan adaptarse a las posibles necesidades futuras.
- Funciones básicas. Una herramienta debe incluir, al menos, algunas funciones mínimas elementales como son la gestión de múltiples proyectos, la delegación de tareas a usuarios, la agenda o los informes de tiempo.
- Funciones avanzadas. En base a las necesidades, se puede necesitar algunas funcionalidades extra como son los diagramas de Gantt (y otras herramientas de planificación), la gestión de archivos, la gestión de gastos, la opción de exportar a Excel, un sistema de roles y permisos o informes de rentabilidad.

- Distintos perfiles, distintas necesidades. En función del tipo de organización y de tipo de proyecto a realizar, es necesario que el software cubra las necesidades de distintos miembros implicados en el proyecto, más centrados en la coordinación, o los miembros del equipo con mayor foco en la comunicación y reflejo del trabajo diario.
- Otras necesidades más particulares. En ocasiones, por el sector de actividad en el que se desarrolla el proyecto y/o la exigencia de una metodología concreta a seguir, se puede necesitar una herramienta más concreta. En ese caso, busca aquella que mejor se ajuste a las necesidades teniendo en cuenta los pros y contras de elegir una aplicación de uso menos generalizado.
- Excelencia en la atención al cliente. Buscar en la página web los canales de contacto que el proveedor del software pone a disposición. Se debe poner a prueba el servicio de atención al cliente enviando aquellas consultas, dudas o sugerencias que hayan surgido durante el proceso de evaluación. De esta forma, se podrá conocer de antemano la calidad del trato, la rapidez de respuesta y la cercanía de la empresa con los usuarios.
- Referencias. Buscar referencias a través de las redes sociales para ver si alguien ha probado el software que se está evaluando y qué opinión le merece la aplicación en su conjunto o aquella funcionalidad que más interesa.

b) Criterios internos

Los criterios internos a tomar en cuenta, en la presente técnica, se mencionan a continuación:

- Conectividad: Este criterio implica tomar en cuenta los motores de bases de datos.
- Medición: Se debe tomar en cuenta la cantidad de motores de bases de datos que el producto soporta.
- Aprendizaje: Tomar en cuenta la cantidad de tiempo invertido en el aprendizaje.
- Artefactos disponibles: La disponibilidad de artefactos de la herramienta permiten mantener la confianza en su uso.
- Escalabilidad: Se debe conocer la factibilidad de agregar nuevas características.
- Estabilidad: Se debe conocer la cantidad de fallos que han sido tolerados.
- Flexibilidad: La herramienta debe tener la capacidad de incrementar sus funcionalidades.
- Portabilidad: Capacidad de la herramienta computacional para trabajar en plataformas diferentes.
- Robustez (solidez): Capacidad de la herramienta para ejecutar diversos procesos sin bloquearse.
- Rendimiento: Capacidad que presenta la herramienta por atender a trabajos con la mayor cantidad de información en el mínimo tiempo.
- Tipo de licencia: Tipo de licencia de la herramienta empleada.

c) Criterios conceptuales del producto.

- Estimación de costos e hitos: Estimación de costos e hitos, incluye su forma gráfica por medio de diagramas.
- Planificación: Entrada y análisis de datos de planificación del proyecto, incluyendo la asignación de tareas, responsabilidades y recursos.
- Seguimiento de proyectos: Entradas de datos sobre actividades del proyecto, incluyendo la gestión de incidentes.
- Ítems de riesgos: Riesgos asociados con un elemento de un modelo, considerando prioridades, niveles de importancia.
- Ítems de esfuerzos: Detalles sobre el esfuerzo requerido para ciertas actividades, asociado con un elemento de un modelo.
- Generación de diagramas: Entrada y edición de diagramas en los lenguajes de modelado que utilice la organización.
- Generación de prototipos: Generación de prototipos sobre la base de requerimientos e información de diseño.
- Gestión de requisitos: Entrada y edición de especificaciones de requisitos y chequeo de su consistencia y completitud en relación con constructos y reglas permitidas.
- Modificación de requisitos: Ajustes en los datos registrados para cada requerimiento durante el desarrollo del proyecto.
- Importación de requisitos: Importación de requisitos de otros formatos y/o fuentes y poder manejarlos como parte de los que han sido creados mediante la herramienta.
- Organización de requisitos: Organización de requisitos de acuerdo con diferentes criterios que faciliten su análisis y su gestión.

- Trazabilidad de requisitos: Análisis de trazabilidad, llevando información desde la especificación de requisitos hasta el diseño.
- Control de acceso: Control de acceso a elementos de datos. Puede incluir especificación de componentes de solo lectura, restricciones de acceso, entre otros.
- Actualización de copias locales: Actualización de copias locales, haciéndolas disponibles para el resto del equipo del proyecto.
- Control de modificaciones: Modificación específica a un documento bajo control de versiones.
- Comparación de versiones: Comparación de versiones, también llamadas revisiones, de un mismo archivo en un proyecto.
- Detección de conflictos de versiones: Identificación de conflictos (entre archivos, en números de líneas de un mismo archivo, etc.).
- Soporte a repositorios basados en SMBD: Soporte a repositorios basados en diferentes Sistemas Manejadores de Bases de Datos SMBD, o en el que la organización requiere
- Reportes basados en plantillas: Generación de reportes en forma de plantillas, que faciliten la posterior edición y reutilización de los mismos.
- Reportes en diferentes formatos: Generación de reportes en diferentes formatos, tales como archivos pdf, txt, entre otros.
- Facilidades para copiar y pegar: Copias e inserciones en otros documentos a partir de texto y gráficos generados.
- Acoplamiento de ventanas: Facilidades para organizar ventanas en el espacio de trabajo de la herramienta.

- Personalización de ventanas: Facilidades para personalizar la organización de ventanas en el espacio de trabajo de la herramienta, guardando esta configuración para un determinado usuario.
- Personalización de barras de herramientas y menús: Facilidades para organizar opciones establecidas por defecto en barras de herramientas y menús.
- Enlaces entre modelos: Enlaces entre elementos que hacen parte de diferentes modelos, tales como clases.
- Facilidades de búsqueda: Opción de hacer búsqueda de elementos de un modelo dentro de un proyecto.
- Opciones de color: Facilidades para cambiar la apariencia de colores de la interfaz.
- Diagramas en varios formatos: Generación de reportes en formatos que pueden incluir PDF, y JPEG para los modelos, entre otros.
- Amigabilidad: Habilidad para ser integrada en las actividades del usuario, considerando los conceptos y procedimientos que son parte del dominio y cultura del mismo.
- Orientación al usuario: Habilidad para proveer al usuario conocimiento sobre estatus de la operación de la herramienta.
- Homogeneidad: Consistencia lógica entre las aplicaciones dentro de una aplicación o entre aplicaciones.
- Adaptabilidad: Habilidad de la interface para adaptarse a varios requerimientos de tareas, estrategias, hábitos y modos de cultura.
- Concisión: Reducción de pasos requeridos para identificar y memorizar, y con los cuales se incrementa la eficiencia del diálogo.

- Facilidad de aprendizaje: La capacidad para habilitar al usuario el aprendizaje de la aplicación.
- Modificación remota de requerimientos: Realización de modificaciones en los requerimientos, desde distintas locaciones y con respectivo control de acceso.
- Servicios de notificación a usuarios: Habilidad para informar a miembros del equipo sobre cambios, asignación de responsabilidades, entre otros, vía correo electrónico u otros mecanismos.
- Crear vistas de requerimientos para acceso remoto: Generación de vistas para que miembros del equipo con diferentes roles puedan acceder de forma remota, con respectivo control de acceso.
- Reportes en HTML: Generación de reportes publicables en web, en formato HTML, y por tanto accesibles para los miembros del equipo.
- Edición colaborativa de artefactos: Edición colaborativa documentos generados durante el proceso de desarrollo.
- Actualizar y descargar documentos: Actualización de un repositorio de documentos, con respectivo control de acceso.

d) Criterios conceptuales del proceso

- Selección flexible de actividades y diagramas: Personalización de actividades del proceso, seleccionando y adaptando los entregables a elaborar según las necesidades.
- Orientación y seguimiento en proceso de desarrollo: Orientación y seguimiento a los usuarios sobre su proceso de desarrollo.

En lo referente a las bases de datos, se tiene lo siguiente:

- Dimensionamiento de la BD

  El SGBD deberá garantizar que es capaz de manejar el volumen de datos requerido. Para ello, deberá comprobarse que es adecuado en cada uno de los siguientes puntos:

  - Número total de bases de datos que se van a crear.
  - Número total de tablas por base de datos.
  - Número máximo de filas por tabla.
  - Longitud máxima de fila.
  - Número máximo de índices por tabla.
  - Número máximo de campos por índice.

- Rendimiento transaccional exigible

  Si se va a utilizar el SGBD en un entorno transaccional, se deberá conocer cuál es la carga (en transacciones por segundo o por minuto) que deberá soportar el sistema y también, se debe indicar cuál es el tiempo de respuesta aceptable (máximo y medio).

- Plataforma/s sobre la/s que debe funcionar

  Se deberá especificar la plataforma o plataformas, físicas y lógicas, sobre las que debe funcionar el SGBD. Para cada una se deberá especificar, al menos, el fabricante, modelo y sistema operativo (especificando el número de versión).

- Tipo de información que se va a tratar

  Todos los productos incluyen soporte para una serie de datos básicos: alfanuméricos, numéricos (enteros y decimales), empaquetados, lógicos y fecha. Según las necesidades específicas de cada caso se deberá exigir el soporte de tipos de datos especiales tales como gráficos, información textual, etc.

- Acceso a los datos

  Para el acceso a los datos debería evaluarse, además del acceso desde el lenguaje propio del SGBD (si existe), la existencia de herramientas de usuario final tales como generadores de informes, formularios de entrada de datos, etc. También, la posibilidad de acceder desde los lenguajes que se estén utilizando previamente como COBOL, PL/I, etc., que suelen estar soportados por medio de precompiladores.

  Se debe tener en cuenta que el lenguaje SQL, es el único modo estandarizado de acceso a los datos en BDs relacionales, por lo que es especialmente deseable que esté soportada la versión normalizada de dicho lenguaje.

- Integración en el entorno existente

  Si la buena integración de cualquier producto informático con el resto de la instalación es siempre una materia de gran importancia, en el caso de un SGBD es absolutamente imprescindible dadas sus características como almacén integrado de datos.

Aunque todo SGBD suele tener alguna herramienta de desarrollo propia, en las instalaciones con desarrollos previos es importante examinar si se podrá acceder a la base de datos con las antiguas herramientas y a qué coste. Este punto determinará, en gran medida, si es posible integrar las aplicaciones antiguas con el SGBD. También es necesario preparar la migración de los datos del soporte antiguo a la BD y evaluar el coste en recursos y de personal que supondrá. Otro factor importante es la integración del SGBD con el sistema operativo en que se vaya a instalar, por ejemplo, su integración con las herramientas estándar de seguridad del sistema operativo o su capacidad para aprovechar las facilidades de los monitores de teleproceso instalados.

Si se tiene intención de exportar o importar periódicamente los datos a otras plataformas o productos, se debe asegurar que el SGBD soporta directamente esta facilidad, por ejemplo, creando ficheros ASCII para tratarlos desde PC. No debe olvidarse que es cada vez más habitual que una organización tenga distintos modelos de SGBD y/o un SGBD en varias plataformas distintas. Por tanto, se debe estudiar la capacidad de comunicarse entre sí de todos ellos. Para esto, es fundamental que el SGBD seleccionado soporte los estándares del mercado.

- Herramientas de administración

  El número de las herramientas a disposición del administrador de la base de datos suele ser muy variable según el SGBD. Se deberá tener en cuenta que es frecuente que parte de ellas se comercialicen como

opciones separadas. Típicamente existirán herramientas para crear la base de datos, definir la estructura física, cargar los datos a partir de ficheros secuenciales externos y viceversa, copias de seguridad, utilidades para reorganizar la base de datos para mejorar su eficiencia, aumentar o reducir su tamaño, etc. También se debe valorar la existencia de herramientas para gestionar la seguridad de monitorización y de obtención de estadísticas (de utilización, consumo, rendimiento entre otros).

- Características de multiproceso

  En entornos donde se requiere un alto rendimiento, puede ser interesante que el SGBD soporte multiproceso. Esto permite que una BD sea accedida por varios procesos que están ejecutándose a la vez, en distintos procesadores y, por tanto, evita las contenciones debidas a sobrecarga del procesador.

- Conectividad y comunicaciones

  Cuando se necesite poder acceder a BDs situadas en varias máquinas, habrá que asegurarse que el SGBD es capaz de hacerlo o que se incluye el producto adecuado que lo permite. Igualmente, se debe comprobar que funciona con el protocolo de comunicación bajo el que se desea trabajar.

e) Criterios del cliente

- Criterios económicos. Por desgracia, éste suele ser uno de los criterios principales para la elección. Cuando no existe mucha “cultura informática”, éste es prácticamente el único criterio que se tiene en

cuenta. Si a eso sumamos la difícil situación económica de crisis actualmente, este se convierte en un factor clave.

- Criterios de afinidad y/o cercanía. Se conoce al proveedor actual muchos años y en algunos casos se dan relaciones de amistad, lo que condiciona a una continuidad y evolución con su software. Pensemos que los humanos somos “per se” reacios a los cambios, de ahí la expresión “Más vale malo conocido que bueno por conocer”, por lo que nos sentimos mejor dentro de nuestra zona de confort, salirnos de ella implica desasosiego y temor.
- Reputación del software. En ocasiones nos guiamos por la marca o notoriedad del software, ya sea porque conocemos a alguien que ha sufrido una implantación o simplemente porque nos suena por su publicidad. Este es uno de los criterios más delicados porque a pesar que un software se haya implementado con éxito en una empresa, esto no garantiza nada con respecto a la implantación en la nuestra. Cada empresa es única y cada implementación también.
- Especialización/sectorización del software. La opinión particular es que éste es el criterio más importante. Si mi negocio está orientado a la fabricación de artículos de descanso por ejemplo, deberemos buscar el software que mejor se adapte a nuestra fabricación, y en el peor de los casos en los que no exista nada similar en el mercado, buscaríamos un software lo más similar posible: fabricación de tapizados, fabricación de muebles, etc. La elección de un software no especializado conducirá inevitablemente a la tensión entre cliente y proveedor debido a la

detección de innumerables carencias relacionadas con la sectorización.

- Especialización de la consultoría. Imprescindible la experiencia de los consultores en el sector y ámbito de actuación de la empresa. Deberá elegirse éstos con sobrada experiencia en estas empresas y con tamaño similares con la finalidad de garantizar el éxito de la implantación.
- Demostrada eficacia en empresas del sector. Un sondeo sobre las empresas de la competencia con la intención de realizar una pre-evaluación del software es necesaria si queremos asegurarnos de que el software cumple las especificaciones y posee las adaptaciones propias de nuestro sector.
- Software cerrado. El proveedor nos vende un paquete totalmente cerrado, sin posibilidad de cambios, o en su defecto, adaptaciones particularizadas que nos llevarán fuera de su versión estándar. Esto no es un problema inicialmente, pero sí a medio plazo, pues nos condenará a un sistema dependiente del proveedor y no evolutivo. Cualquier evolución pasará por una actualización de esos desarrollos, la mayoría de veces, penalizado con nuevos pagos.
- Software a medida. Es el caso contrario al anterior, el proveedor nos ofrece no solo un producto, sino todas las mejoras que se nos ocurran. Esto podría estar justificado si nuestra empresa perteneciese a un nicho muy especializado para el que no se encuentra un software que cumpla las especificaciones, y por tanto, hay que hacer grandes desarrollos sobre uno ya preexistente para su adaptación. En cualquier

otro caso, estamos condenados a la obsolescencia del producto debido a la dificultad por parte del proveedor de mantener software personalizado para cada empresa.

- Conectividad. El software debe poder ofrecer conectividad con otras herramientas, sobretodo de ofimática. Ello nos liberará de cierta dependencia del proveedor para algunos informes, correos electrónicos, entre otros.
- Soporte, atención post-venta y servicios de mantenimiento. Éste ha de ser el resultado y continuación de una consultoría especializada. La implantación de un software tiene un principio, pero no es fácil establecer un fin, por lo que una vez puesto en marcha el software, habrá un tiempo de indudable necesidad de soporte y ayuda. La realización de un mantenimiento no solo correctivo, sino también preventivo, tanto en base de datos como en el propio software nos asegurará una mejora en la eficiencia de la herramienta.
- Adaptación a la normativa vigente. Puede parecer una obviedad, pero la realidad es que algunos software no están localizados de forma suficiente, bien porque vienen de otros países con otras leyes y otras problemáticas, o porque no son lo bastante ágiles para incorporar las nuevas normativas legales o bancarias en un mercado como el actual muy cambiante.
- Plataforma y evolución del software. Es conveniente dedicar un poco de tiempo a analizar las plataformas sobre las que está desarrollado el software: base de datos, lenguaje, etc. Conocerlos puede darnos una señal de su modernidad u obsolescencia. En cuanto a la evolución del

propio software, analizar la periodicidad de las versiones, si éstas suelen ser correcciones o nuevas prestaciones, el número de éstas, etc. Esto nos dará una muestra bastante fiable de lo "vivo" que se encuentra el software.

- Migración de datos. En ocasiones es necesario trasladar parte de la información del sistema anterior al actual. Esto podrá hacerse de forma manual o mediante la asistencia del propio proveedor. Éste deberá además asesorarnos sobre la conveniencia o no de esas migraciones en función de la compatibilidad de los datos.
- Seguridad de la información. Siempre hay información sensible que ha de ser protegida, tanto para agentes externos, como hacia determinados usuarios. Un buen sistema de configuración de menús, niveles de acceso, roles de usuario, así como de auditoría de datos, aportarán un plus de confianza adicional hacia el nuevo sistema.
- Modularización de la herramienta. Es decir, las necesidades de cada empresa no son las mismas, por lo tanto, debe permitir hacer la inversión en función de las necesidades de cada momento y de forma incremental. De esta forma, nos aseguramos la puesta en marcha de los módulos comprados y en un futuro la implementación según necesidades y nuevos requerimientos. Asimismo, evitaremos la aparición de costes ocultos e imprevistos.

Etapa N° 3: Clasificación de las herramientas de software

Para esta etapa también se toman en cuenta los aspectos contextuales tanto del producto como del proceso, de la misma forma como los define MOSCA.

a) Software libre

Software que respeta la libertad de los usuarios y la comunidad. En grandes líneas, significa que los usuarios tienen la libertad para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Es decir, el «software libre» es una cuestión de libertad, no de precio. Con estas libertades, los usuarios (tanto individualmente como en forma colectiva) controlan el programa y lo que este hace. Cuando los usuarios no controlan el programa, decimos que dicho programa «no es libre», o que es «privativo». Un programa que no es libre controla a los usuarios, y el programador controla el programa, con lo cual el programa resulta ser un instrumento de poder injusto.

Un programa es software libre si los usuarios tienen las cuatro libertades esenciales:

- La libertad de ejecutar el programa como se desea, con cualquier propósito.
- La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que se desea. El acceso al código fuente es una condición necesaria para ello.
- La libertad de redistribuir copias para ayudar a su prójimo.
- La libertad de distribuir copias de sus versiones modificadas a terceros. Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello.

b) Software propietario

Aquel en el que un usuario tiene limitadas sus posibilidades de usarlo, modificarlo o redistribuirlo, y a menudo su licencia tiene un coste. Se le llama software propietario, no libre, privado o privativo al tipo de programas informáticas o aplicaciones en el que el usuario no puede acceder al código fuente o tiene un acceso restringido y, por tanto, se ve limitado en sus posibilidades de uso, modificación y redistribución. Este tipo de software se opone al más recientemente popularizado software libre, que permite que cualquiera lo modifique y lo redistribuya.

El software propietario es el más común, ya que implica que para acceder al mismo el usuario debe pagar por una licencia y sólo puede hacer uso del mismo en un contexto restringido, es decir, que para que puedan usarlo distintas computadoras deben abonarse otras licencias. Además, este software no puede modificarse ni perfeccionarse en su funcionamiento, como así tampoco redistribuirse a otros destinatarios.

Los softwares propietarios son con frecuencia desarrollados por corporaciones, como ocurre con aquel producido y distribuido por Microsoft. Estas compañías poseen los derechos de autor sobre el software y, por tanto, los usuarios no pueden acceder al código fuente, distribuir copias, mejorarlo o hacer públicas las mejoras.

Etapa N° 4: Validación desde la ingeniería y en equipos de trabajo

Para esta etapa, la figura 3 muestra el proceso de validación de las herramientas que se emplearán en el proyecto de desarrollo de software.

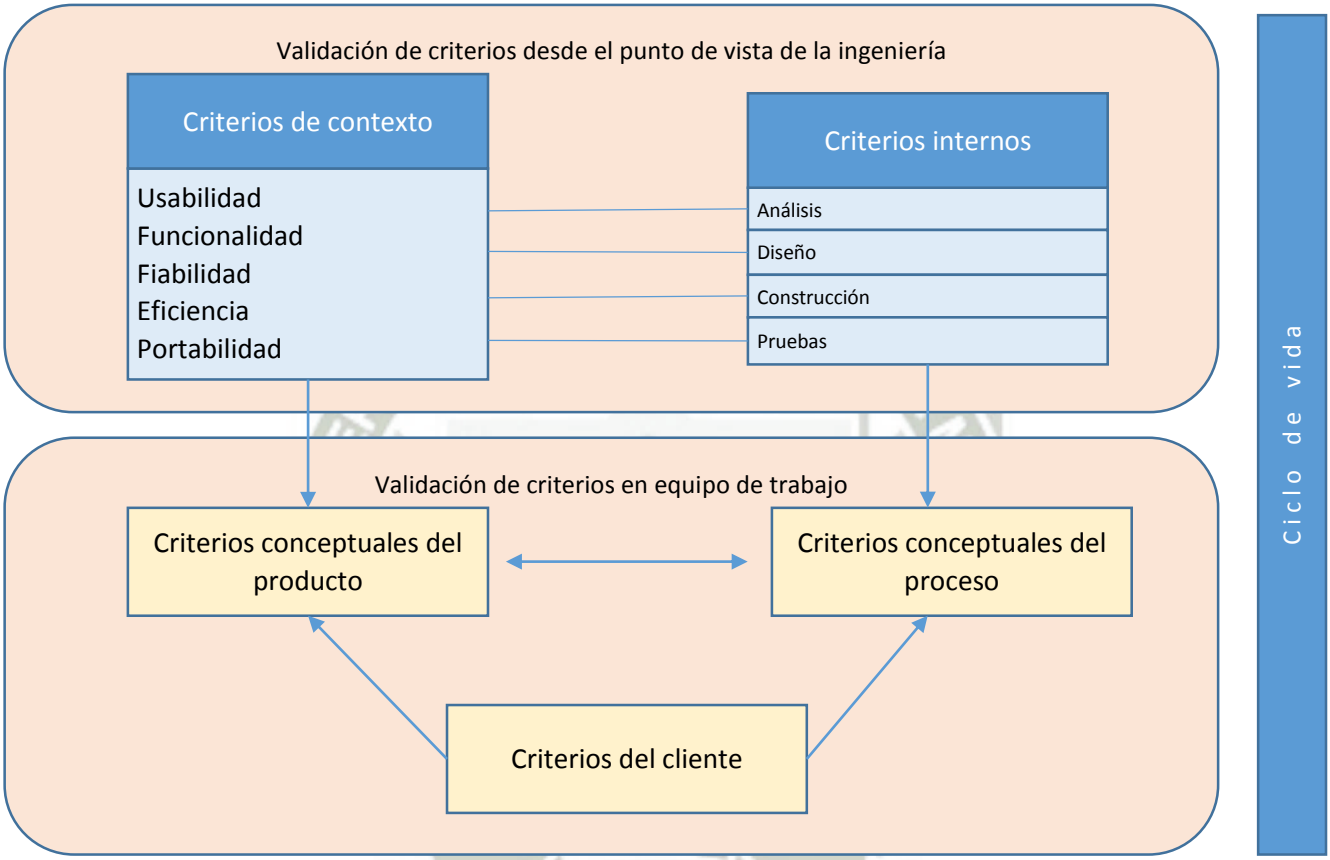

Figura 3. Proceso de validación del esquema
Fuente: elaboración propia

a) Instanciación de herramientas

a.1 Aspectos contextuales

Tomando como punto de partida el ámbito de la construcción del software, se desglosan un conjunto de diagramas que representan los aspectos de construcción del producto como por ejemplo: Diagrama de casos de uso esenciales, diagrama de casos de uso reales, diagrama de clases, diagrama de actividad, diagrama de despliegue, diagrama de colaboración, diagrama de interacción, diagrama de flujo de datos, entre otros.

Estos diagramas presentan relaciones sustanciales que permiten encaminar la construcción del producto. Por ejemplo la plantilla de educción de requisitos presenta una relación con la plantilla de elicitación de requisitos, el mismo que se encuentra relacionado con la plantilla de especificación de requisitos de software. El diagrama de secuencias permite observar el nivel de mensajes que se deben de programar cuando se crean las clases en función del diagrama de clase.

Estas relaciones deben ser tomadas en cuenta cuando se analice el tipo de metodología, técnica o modelo que se emplee para la construcción del producto. El análisis de estos aspectos es importantes porque nos entregan el derrotero definitivo para saber la casuística de construcción y por consecuencia tomar conciencia del tipo de herramienta con la cual se debe de trabajar.

En estos momentos muchas de las herramientas construidas contemplan una serie de etapas de la construcción del producto por lo que una buena herramienta, por ejemplo las herramientas CASE son suficientes para resolver el problema, por otro lado estas mismas no contemplan todas las necesidades de construcción debido a la buena especificación de requisitos.

a.2 Aspectos de funcionalidad

Para resolver los aspectos de funcionalidad, es preciso que se deban de tomar en cuenta los siguientes criterios:

- Analizar el submodelo inherente en la construcción del producto. Esto permite lograr un mayor detalle de preparación en la construcción del producto. Por ejemplo puede que en un módulo se emplee el prototipado y en otro el modelo de cascada. El problema radica en la utilización de un modelo híbrido basados en modelos bastante probados.

- Se debe de seleccionar la dimensión o efecto mayor de uno de los submodelos para lograr una definición precisa de la herramienta a emplear. Por ejemplo si el modelo del prototipado tiene una mayor presencia en la construcción entonces la herramienta a escoger debe de estar en función de este modelo pero analizando los efectos colaterales que surgen cuando la misma no cuenta con soluciones para el otro modelo. Para este efecto se debe de

esperar hasta la elicitación de requisitos para tomar una decisión bastante aceptable.

- Selección en función de la categoría el mismo que implica un análisis de los algoritmos internos del producto que induzcan la solución en el uso de la herramienta. Especial interés se debe de tener con las herramientas ya que los mismos deben de resaltar las características apropiadas para la construcción del producto de software.

b) Características internas del producto

Los productos que apoyan a la construcción del software deben de mantener las siguientes características:

- Eficiencia
- Mantenibilidad
- Confiabilidad
- Portabilidad
- Usabilidad

c) Preparación de medidas

Se deben de tomar en cuenta herramientas que permitan generar medidas de uso y permitan un historial de detalles que permitirán, en el futuro, lograr objetivos concretos.

d) Criterios de funcionalidad

Los criterios de funcionalidad, tomados en cuenta, de las herramientas de software que apoyan en al modelamiento y construcción del producto son los siguientes:

- Funcionalidad
  - Ajuste a los propósitos de construcción de productos de software
  - Precisión de la herramienta en apoyo a la construcción del producto de software.
  - Interoperabilidad de la herramienta en plataformas diferentes.
  - Seguridad de obtener resultados correctos. Por ejemplo herramientas que proporcionen como salida código fuente.
- Usabilidad
  - Usabilidad como capacidad de la herramienta para lanzar ayudas adecuadas en su utilización.
  - Capacidad de aprendizaje apoyado por manuales físicos o virtuales sustentando sus fortalezas.
  - Interfaz de usuario fácil de entender y entregar derroteros que permitan conocer la herramienta en el menor tiempo posible.
  - Su operabilidad debe ayudar a conocer la herramienta de trabajo en el menor tiempo posible.

- Fiabilidad
  - Tolerancia a fallos cuando se insertan modelos mal conceptualizados.
  - Recuperación de los errores suministrados por los usuarios o por el mal concepto de los insumos de entrada al producto.
- Eficiencia
  - Comportamiento del tiempo.
  - Empleo de recursos.

3.4 Asignación de valores

La calificación de cada ítem se encuentra dado bajo el criterio de asignación de valores personalizados. Los valores propuestos son dados como punto de partida para que en trabajos futuros se puedan asignar valores más cercanos a la realidad:

- Si no posee la característica: 0
- Si tiene la característica: 1
- Si presenta relación con terceros: 2

Para obtener el puntaje final se calcula el promedio simple para finalmente hacer la comparación entre productos. Aquel ítem que obtenga el máximo promedio implica que es la herramienta que debe ser asignada para llevar a cabo la tarea correspondiente. El resumen de los criterios, conjuntamente con sus nemónicos se muestra en la tabla 3.

3.5 Cálculo de las métricas

- Métrica para la fase de proyectos de software

$IP(PS) = Max (P_1,\ldots, P_j)$

IP: Índice de productividad

PS: Proyecto de software

P_i: Promedio general de cada herramienta de software

- Métrica para la fase de análisis de sistemas

  $IP(AS) = Max (P_1,\ldots, P_j) * \sigma(P_{11},\ldots,P_{1j})$

  IP: Índice de productividad

  AS: Análisis de sistemas

  P_i: Promedio general de cada herramienta de software

  σ: Desviación de los promedios internos de la herramienta a medir

- Métrica para el diseño de sistema

  $IP(DS) = Max (P_1,\ldots, P_j) * \sigma(P_{11},\ldots,P_{1j})$

  IP: Índice de productividad

  DS: Diseño de sistemas

  P_i: Promedio general de cada herramienta de software

  σ: Desviación de los promedios internos de la herramienta a medir

- Métrica para la construcción de software

  $IP(CS) = Max (P_1,\ldots, P_j) * K(P_{11},\ldots,P_{1j})$

  IP: Índice de productividad

  CS: Construcción de software

  P_i: Promedio general de cada herramienta de software

  K: Cantidad de plantillas aceptadas por la herramienta de software.

- Métrica para la prueba de software

  $IP(PS) = Max (P_1,\ldots, P_j) * R(T_i)$

IP: Índice de productividad

PS: Prueba de software

P_i: Promedio general de cada herramienta de software

R: Valor del tipo de prueba de software

T_i: Tipo de prueba de software

Prueba de caja negra	: 1
Prueba de caja blanca	: 2
Prueba de complejidad ciclomática	: 3
Prueba de integración	: 4
Pruebas unitarias	: 5

Tabla 3
Resumen de criterios

ETAPA	FASE	CRITERIO	NEMONICO
Definición de patrones	Proyectos de software	Estimación de costos	C1
		Estimación de hitos de control	C2
		Estimación de la planificación	C3
		Seguimientos del proyecto en bases a sus actividades	C4
		Estimación de riesgos asociados	C5
		Estimación del esfuerzo	C6
		Generación de artefactos o documentación correspondiente	C7
	Análisis de sistemas	Construcción de diagramas según modelo escogido	C1
		Especificación de prototipos basados en la etapa de requisitos	C2
		Especificación de requisitos en las etapas de educción, elicitación y especificación	C3
		Modificación de requisitos en sus etapas de educción, elicitación y especificación	C4
		Importación de requisitos bajos formatos diferentes al de la herramienta	C5
		Organización de requisitos de acuerdo a diferentes criterios	C6
		Análisis de la trazabilidad de los requisitos	C7
		Importación de objetos	C8
		Exportación de objetos	C9
		Generación de los artefactos correspondientes	C10
	Diseño de sistemas	Construcción de diagramas de diseño según modelo escogido	C1
		Especificación de prototipos basados en la etapa de análisis	C2
		Modificación de requisitos en sus etapas de educción, elicitación y especificación a partir de la especificación del diseño	C3
		Modificación de la información de acuerdo a criterios de análisis y su reinserción en la documentación de análisis	C4
		Análisis de la trazabilidad de los requisitos hasta el diseño	C5
		Generación de código	C6
		Generación de los artefactos o documentos correspondientes	C7

Fuente: Elaboración propia

Tabla 3 continuación

Resumen de criterios

ETAPA	FASE	CRITERIO	NEMONICO
	Construcción de software	Especificación del control de acceso a los datos incluyendo la especificación de componentes	C1
		Especificación de copias de la codificación del producto	C2
		Actualización de copias producto del versionamiento	C3
		Especificación del control de cambios	C4
		Soporte de repositorios como medio del control de la seguridad	C5
		Integración con otras herramientas de diseño	C6
		Generación de artefactos o documentos correspondientes	C7
	Pruebas de software	Capacidad de asumir el plan de pruebas en formatos diferentes	C1
		Especificación de la diversidad de pruebas que pueden llevar a cabo a partir del plan	C2
		Especificaciones gráficas de la prueba	C3
		Generación de artefactos o documentos correspondientes	C4
Criterios de selección	Criterios de contexto	Software online	C1
		Fácil, usable e intuitivo	C2
		Versión o periodo de prueba	C3
		Escalabilidad	C4
		Funciones básicas	C5
		Funciones avanzadas	C6
		Perfiles	C7
		Necesidades particulares	C8
		Atención al cliente	C9
		Referencias	C10

Fuente: Elaboración propia

Tabla 3 continuación

Resumen de criterios

ETAPA	FASE	CRITERIO	NEMONICO
		Conectividad	C1
		Medición	C2
		Curva de aprendizaje	C3
		Documentación disponible	C4
		Escalabilidad	C5
		Estabilidad	C6
	Criterios internos	Flexibilidad	C7
		Portabilidad	C8
		Robustez	C9
		Rendimiento	C10
		Respaldo	C11
		Tipo de licencia	C12
		Economía	C1
		Afinidad	C2
Criterios de selección		Reputación del software	C3
		Especialización del software	C4
		Especialización de la consultoría	C5
		Eficacia en empresas	C6
		Software cerrado	C7
	Criterios del cliente y clasificación	Software a medida	C8
		Conectividad	C9
		Soporte	C10
		Adaptación a la norma	C11
		Evolución del software	C12
		Migración de datos	C13
		Seguridad de la información	C14
		Modularización de la herramienta	C15
		Software libre	C16

Fuente: Elaboración propia

Capítulo 4: Aplicación del esquema propuesto al caso de estudio

4.1 Introducción

La empresa Smart Reason S.R.L. proporcionó la anuencia de aplicar el esquema en el producto de software que iban a empezar a construir, independientemente del esquema de procesos propio de la empresa. El módulo a construir se denomina Control de Morosidad bajo las siglas CoMoSoft.

El producto ha sido solicitado por la Cooperativa de Consumo MAXSER y construido empleando criterios de Ingeniería de Software.

4.2 Etapa 1: Definición de patrones

La búsqueda de patrones consiste en buscar coincidencias en los aspectos de análisis, diseño y construcción en el producto de software a construir. Para este caso de estudio, los patrones estuvieron enmarcados en el control de la morosidad y la forma como este problema debería de atacarse logrando determinarse los siguientes aspectos:

- Modelos de casos de uso
  - Audiencia: Analistas funcionales
  - Ámbito: Modelo de casos de uso: describe los casos de uso de realización más importantes.
  - Artefacto Relacionado: Diagrama de Casos de Uso.
- Modelo de análisis
  - Audiencia: Analistas funcionales
  - Ámbito: Requerimientos: Detalla la realización de los Casos de Uso más importantes.

  - Artefacto Relacionado: Clase de Análisis, Realización de Casos de Uso-Análisis

- Modelo de diseño
  - Audiencia: Diseñadores
  - Ámbito: Modelo de análisis: crea una entrada apropiada y un punto de partida para actividades de implementación subsecuentes capturando los requisitos ó subsistemas individuales, interfaces y clases.
  - Artefacto Relacionado: Modelo de diseño, Clase del diseño, Realización del caso de uso-diseño.
- Modelos de despliegue
  - Audiencia: Administradores de despliegue.
  - Ámbito: Topología: describe el mapeo del software en el hardware y muestra los diferentes aspectos distribuidos.
  - Artefacto Relacionado: Modelo del despliegue.
- Vista Lógica
  - Audiencia: Diseñadores.
  - Ámbito: Requerimientos Funcionales: describe el modelo de objetos del diseño. También describe los casos de uso de realización más importantes.
  - Artefacto Relacionado: Modelo del diseño

Finalmente se determinó que el software que se pudiera emplear eran los siguientes: Gantt PV, Microsoft Project y Gantt Project. La tabla 4 muestra los resultados de la fase proyectos de software de la etapa definición de patrones.

Tabla 4

Resultados de la fase proyectos de software, etapa definición de patrones

ETAPA 1: DEFINICION DE PATRONES										
FASE: PROYECTOS DE SOFTWARE										
HERRAMIANTA	SUB-ETAPAS	CRITERIOS							VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7		
Gantt PV	Diagrama gantt	0	1	1	1	0	1	1	0.71	0.62
	Diagrama PERT	0	1	1	1	0	0	1	0.57	
	Diagrama CPM	0	1	1	1	0	0	1	0.57	
Microsoft Project	Diagrama gantt	1	1	1	1	1	1	1	1.00	0.81
	Diagrama PERT	0	1	1	1	0	0	1	0.57	
	Diagrama CPM	1	1	1	1	1	0	1	0.86	
Gantt Project	Diagrama gantt	1	1	1	1	1	1	1	1.00	0.71
	Diagrama PERT	0	1	1	1	0	0	1	0.57	
	Diagrama CPM	0	1	1	1	0	0	1	0.57	

Fuente: Elaboración propia

La tabla 4 nos indica que el producto que debe de emplearse para la fase de elaboración de proyectos de software es Microsoft Project ya que se ajusta a las consideraciones de construcción del producto y cuyo índice de productividad es la mayor de las herramientas analizadas.

De igual manera para la fase de análisis, diseño, construcción y pruebas de software del producto a construir se determinó que se puedan utilizar las herramientas expuestas en las tablas correspondientes.

La tabla 5 muestra los resultados obtenidos para la fase de análisis del sistema de la etapa definición de patrones.

Tabla 5

Resultados de la fase análisis de sistemas, etapa definición de patrones

ETAPA 1: DEFINICION DE PATRONES
FASE: ANALISIS DE SISTEMAS

PRODUCTOS	SUB-ETAPAS	CRITERIOS										VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10		
REM	Plantilla de educción de requisitos	1	1	1	1	1	1	1	1	1	1	1.00	1.36
	Plantilla de elicitación de requisitos	1	1	1	1	1	1	1	1	1	1	1.00	
	Plantilla de especificación de requisitos de software	1	1	1	1	1	1	1	1	1	1	1.00	
	Plantilla de casos de uso esenciales	1	1	1	1	1	1	1	1	1	1	1.00	
	Diagrama de casos de uso esenciales	2	2	2	2	2	2	2	2	2	2	2.00	
	Diagrama de flujo de datos	2	2	2	2	2	2	2	2	2	2	2.00	
	Diagrama de secuencia	2	2	2	2	2	2	2	2	2	2	2.00	
	Plantilla de contratos de software	1	1	1	1	1	1	1	1	1	1	1.00	
	Diagrama de actividades	2	2	2	2	2	2	2	2	2	2	2.00	
	Diagrama de bloques	2	2	2	2	2	2	2	2	2	2	2.00	
	Diagrama de clases de análisis	2	2	2	2	2	2	2	2	2	2	2.00	
	Plantilla de tarjetas CRC	1	1	1	1	1	1	1	1	1	1	1.00	
	Plantilla de trazabilidad de requisitos	1	1	1	1	1	1	1	1	1	1	1.00	
	Plantilla de métricas de software	0	0	0	0	0	0	0	0	0	0	0.00	
StarUML	Plantilla de educción de requisitos	0	0	0	0	0	0	0			0	0.00	0.64
	Plantilla de elicitación de requisitos	0	0	0	0	0	0	0	0	0	0	0.00	
	Plantilla de especificación de requisitos de software	0	0	0	0	0	0	0	0	0	0	0.00	
	Plantilla de casos de uso esenciales	1	1	1	1	1	1	1	1	1	1	1.00	
	Diagrama de casos de uso esenciales	1	1	1	1	1	1	1	1	1	1	1.00	
	Diagrama de flujo de datos	1	1	1	1	1	1	1	1	1	1	1.00	
	Diagrama de secuencia	1	1	1	1	1	1	1	1	1	1	1.00	
	Plantilla de contratos de software	2	2	2	2	2	2	2	2	2	2	2.00	
	Diagrama de actividades	1	1	1	1	1	1	1	1	1	1	1.00	
	Diagrama de bloques	1	1	1	1	1	1	1	1	1	1	1.00	
	Diagrama de clases de análisis	1	1	1	1	1	1	1	1	1	1	1.00	
	Plantilla de tarjetas CRC	0	0	0	0	0	0	0	0	0	0	0.00	
	Plantilla de trazabilidad de requisitos	0	0	0	0	0	0	0	0	0	0	0.00	
	Plantilla de métricas de software	0	0	0	0	0	0	0	0	0	0	0.00	

Fuente: Elaboración propia

Tabla 5 Continuación

Resultados de la fase análisis de sistemas, etapa definición de patrones

ETAPA 1: DEFINICION DE PATRONES

FASE: ANALISIS DE SISTEMAS													
PRODUCTOS	SUB-ETAPAS	CRITERIOS										VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10		
Poseidon	Plantilla de educción de requisitos	0	0	0	0	0	0	0			0	0.00	0.62
	Plantilla de elicitación de requisitos	0	0	0	0	0	0	0	0	0	0	0.00	
	Plantilla de especificación de requisitos de software	0	0	0	0	0	0	0	0	0	0	0.00	
	Plantilla de casos de uso esenciales	1	0	1	1	1	1	1	1	1	1	0.90	
	Diagrama de casos de uso esenciales	1	1	1	1	1	1	1	0	1	1	0.90	
	Diagrama de flujo de datos	1	1	1	1	1	1	1	1	1	1	1.00	
	Diagrama de secuencia	1	1	1	1	0	1	1	1	1	1	0.90	
	Plantilla de contratos de software	2	2	2	2	2	2	2	2	2	2	2.00	
	Diagrama de actividades	1	1	1	1	1	0	1	1	1	1	0.90	
	Diagrama de bloques	1	1	0	1	0	1	0	1	1	1	0.70	
	Diagrama de clases de análisis	1	1	1	1	1	1	1	1	1	1	1.00	
	Plantilla de tarjetas CRC	0	0	1	0	0	0	1	0	0	0	0.20	
	Plantilla de trazabilidad de requisitos	0	0	0	0	0	0	0	0	0	0	0.00	
	Plantilla de métricas de software	0	0	1	0	0	0	0	0	1	0	0.20	
Requisite Pro	Plantilla de educción de requisitos	1	1	1	0	1	1	1	1	1	1	0.90	0.84
	Plantilla de elicitación de requisitos	1	1	1	1	1	1	0	1	1	1	0.90	
	Plantilla de especificación de requisitos de software	1	0	1	1	1	1	1	1	1	1	0.90	
	Plantilla de casos de uso esenciales	1	0	1	1	1	0	1	0	1	1	0.70	
	Diagrama de casos de uso esenciales	1	0	2	0	2	2	0	2	2	1	1.20	
	Diagrama de flujo de datos	0	0	1	0	1	0	1	0	1	0	0.40	
	Diagrama de secuencia	2	0	1	2	1	2	2	2	1	2	1.50	
	Plantilla de contratos de software	1	1	1	1	1	1	1	1	0	0	0.80	
	Diagrama de actividades	1	1	1	1	1	0	0	1	1	1	0.80	
	Diagrama de bloques	1	0	1	1	1	1	1	1	1	1	0.90	
	Diagrama de clases de análisis	1	1	1	1	1	1	1	1	1	1	1.00	
	Plantilla de tarjetas CRC	1	1	1	0	1	0	1	1	1	0	0.70	
	Plantilla de trazabilidad de requisitos	1	1	1	1	0	0	1	1	1	1	0.80	
	Plantilla de métricas de software	0	0	1	0	0	1	0	0	0	1	0.30	

Fuente: Elaboración propia

Para esta fase, y específicamente la Ingeniería de Requisitos, la herramienta a emplear es REM porque mantiene muchos criterios de los estándares además de su capacidad de integración hacia otras herramientas en la construcción del producto final. Su índice de productividad es elevada con respecto a las de otras herramientas expuestas.

La tabla 6 muestra los resultados de la fase de diseño del sistema de la etapa definición de patrones.

Tabla 6

Resultados de la fase diseño del sistema, etapa definición de patrones

ETAPA 1: DEFINICION DE PATRONES										
FASE: DISEÑO DEL SISTEMA										
PRODUCTOS	**SUB-ETAPAS**	**CRITERIOS**							**VALOR ITEM**	**VALOR PRODUCTO**
		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**		
QT	Plantilla de casos de uso reales	1	1	1	1	1	0	1	0.86	1.00
	Diagrama de casos de uso reales	1	1	1	0	1	0	1	0.71	
	Plantilla de reportes	0	2	1	1	1	0	1	0.86	
	Diagrama del modelo de navegabilidad	2	1	2	1	1	1	1	1.29	
	Plantilla de diseño de interfaces	1	0	1	1	1	1	1	0.86	
	Diagrama de bloques	2	1	1	1	1	1	1	1.14	
	Diagrama de colaboración	2	1	1	0	1	1	1	1.00	
	Diagrama de clases de diseño	2	0	1	1	1	1	1	1.00	
	Diagrama entidad - relación	2	1	1	1	1	1	1	1.14	
	Diagrama de objetos	2	1	1	1	1	1	1	1.14	
	Diagrama de estados	2	1	1	1	1	1	1	1.14	
	Diagrama de componentes	2	0	0	1	1	1	1	0.86	
	Diagrama de despliegue	2	1	0	1	1	1	1	1.00	

Fuente: Elaboración propia

Tabla 6 Continuación

Resultados de la fase diseño del sistema, etapa definición de patrones

ETAPA 1: DEFINICION DE PATRONES											
FASE: DISEÑO DEL SISTEMA											
PRODUCTOS	SUB-ETAPAS	CRITERIOS							VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7		
ArgosUML	Plantilla de casos de uso reales	1	1	1	0	1	0	1	0.71	0.89
	Diagrama de casos de uso reales	1	1	1	0	1	0	1	0.71	
	Plantilla de reportes	2	0	0	1	1	0	1	0.71	
	Diagrama del modelo de navegabilidad	1	2	1	1	1	0	1	1.00	
	Plantilla de diseño de interfaces	1	2	2	1	1	0	1	1.14	
	Diagrama de bloques	1	2	2	1	1	0	1	1.14	
	Diagrama de colaboración	1	1	1	1	1	0	1	0.86	
	Diagrama de clases de diseño	1	1	1	0	1	0	1	0.71	
	Diagrama entidad - relación	1	0	2	0	1	2	1	1.00	
	Diagrama de objetos	1	0	1	1	1	0	1	0.71	
	Diagrama de estados	1	0	0	1	1	2	1	0.86	
	Diagrama de componentes	1	1	0	1	1	2	1	1.00	
	Diagrama de despliegue	1	0	1	1	1	2	1	1.00	
iReport Designer	Plantilla de casos de uso reales	0	1	1	0	1	0	1	0.57	0.87
	Diagrama de casos de uso reales	0	1	1	1	1	0	1	0.71	
	Plantilla de reportes	0	1	2	0	1	0	1	0.71	
	Diagrama del modelo de navegabilidad	2	1	1	1	1	1	1	1.14	
	Plantilla de diseño de interfaces	2	2	0	1	1	0	1	1.00	
	Diagrama de bloques	2	2	0	1	1	0	1	1.00	
	Diagrama de colaboración	2	0	0	0	1	1	1	0.71	
	Diagrama de clases de diseño	2	0	2	1	1	0	1	1.00	
	Diagrama entidad - relación	2	1	1	0	1	1	1	1.00	
	Diagrama de objetos	2	1	0	1	0	1	1	0.86	
	Diagrama de estados	2	1	1	0	1	1	1	1.00	
	Diagrama de componentes	2	1	1	1	1	0	1	1.00	
	Diagrama de despliegue	2	0	0	1	0	0	1	0.57	

Fuente: Elaboración propia

Tabla 6 Continuación

Resultados de la fase diseño del sistema, etapa definición de patrones

ETAPA 1: DEFINICION DE PATRONES										
FASE: DISEÑO DEL SISTEMA										
PRODUCTOS	SUB-ETAPAS	CRITERIOS							VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7		
Poseidon	Plantilla de casos de uso reales	1	1	1	0	1	1	1	0.86	0.91
	Diagrama de casos de uso reales	1	1	0	1	1	1	1	0.86	
	Plantilla de reportes	2	2	1	1	1	1	1	1.29	
	Diagrama del modelo de navegabilidad	2	2	0	1	1	1	1	1.14	
	Plantilla de diseño de interfaces	1	2	1	1	1	1	1	1.14	
	Diagrama de bloques	1	2	0	1	1	1	1	1.00	
	Diagrama de colaboración	1	1	2	1	1	0	1	1.00	
	Diagrama de clases de diseño	1	1	0	1	0	0	1	0.57	
	Diagrama entidad - relación	1	0	1	0	0	1	1	0.57	
	Diagrama de objetos	1	0	0	0	1	1	1	0.57	
	Diagrama de estados	1	1	1	1	1	1	1	1.00	
	Diagrama de componentes	1	1	0	1	1	1	1	0.86	
	Diagrama de despliegue	1	1	2	0	1	1	1	1.00	

Fuente: Elaboración propia

La tabla 6 indica que para la fase de diseño del sistema, la herramienta adecuada para construir el producto es QT, ello debido a su flexibilidad y porque en la representación de interfaces se respeta las reglas de la interacción humano-computador.

La tabla 7 muestra los resultados de la fase construcción de software de la etapa definición de patrones, en ella se lleva a cabo la puntuación correspondiente a los criterios definidos en la tabla 3 y con respecto a los productos de software Netbeans, Eclipse y Visual.Net.

Tabla 7
Resultados de la fase construcción de software, etapa definición de patrones

ETAPA 1: DEFINICION DE PATRONES										
FASE: CONSTRUCCION DE SOFTWARE										
PRODUCTOS	SUB-ETAPAS	CRITERIOS							VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7		
NetBeans	Plantilla de clases	1	1	1	1	0	1	1	0.86	0.88
	Plantilla de interfaces	1	1	1	1	0	1	1	0.86	
	Plantilla de diseño de reportes	1	1	1	1	1	1	1	1.00	
	Plantilla de implementación de bases de datos	1	1	0	1	1	1	1	0.86	
	Plantilla de codificación	1	1	0	1	0	1	1	0.71	
	Generación de código	1	1	1	1	1	1	1	1.00	
Eclipse	Plantilla de clases	1	1	0	1	0	1	1	0.71	0.90
	Plantilla de interfaces	1	1	1	1	1	1	1	1.00	
	Plantilla de diseño de reportes	2	1	1	1	0	1	1	1.00	
	Plantilla de implementación de bases de datos	1	1	1	1	1	1	1	1.00	
	Plantilla de codificación	1	1	0	1	0	1	1	0.71	
	Generación de código	1	1	1	1	1	1	1	1.00	
Visual Studio.Net	Plantilla de clases	1	1	1	1	0	1	1	0.86	0.98
	Plantilla de interfaces	2	1	1	1	0	1	1	1.00	
	Plantilla de diseño de reportes	2	1	0	1	1	1	1	1.00	
	Plantilla de implementación de bases de datos	2	1	1	1	0	1	1	1.00	
	Plantilla de codificación	1	1	1	1	1	1	1	1.00	
	Generación de código	1	1	1	1	1	1	1	1.00	

Fuente: Elaboración propia

En esta tabla se demuestra que la herramienta que se debe de emplear para esta etapa es Visual studio.Net. Esto se debe a las características que deben ser consideradas por el producto de software a construir.

La tabla 8 muestra los resultados de la fase pruebas de software de la etapa definición de patrones.

Tabla 8
Resultados de la fase pruebas de software, etapa definición de patrones

ETAPA 1: DEFINICION DE PATRONES							
FASE: PRUEBAS DE SOFTWARE							
PRODUCTOS	SUB-ETAPAS	CRITERIOS				VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4		
JUnit	Plantilla de pruebas de caja negra	1	1	0	1	0.75	0.75
	Plantilla de pruebas de caja blanca	1	0	0	1	0.50	
	Plantilla de pruebas de complejidad ciclomática	1	1	1	1	1.00	
	Plantilla de pruebas de integración	1	1	0	1	0.75	
	Plantilla de pruebas unitarias	1	1	0	1	0.75	
Ant	Plantilla de pruebas de caja negra	1	0	0	1	0.50	0.75
	Plantilla de pruebas de caja blanca	1	1	1	1	1.00	
	Plantilla de pruebas de complejidad ciclomática	1	1	0	1	0.75	
	Plantilla de pruebas de integración	1	0	0	1	0.50	
	Plantilla de pruebas unitarias	1	1	1	1	1.00	
Mock Objects	Plantilla de pruebas de caja negra	1	1	0	1	0.75	0.75
	Plantilla de pruebas de caja blanca	1	1	0	1	0.75	
	Plantilla de pruebas de complejidad ciclomática	1	0	0	1	0.50	
	Plantilla de pruebas de integración	1	1	0	1	0.75	
	Plantilla de pruebas unitarias	1	1	1	1	1.00	

Fuente: Elaboración propia

La tabla 8 nos indica que la herramienta a emplear en esta fase puede ser JUnit, Ant o Mock Objects. Se escoge, por criterio personal, la herramienta JUnit porque los probadores se encontraban más acostumbrados a la herramienta, además de que su índice de productividad es elevado.

4.3 Etapa 2: Criterios de selección

Para esta etapa se emplean los criterios de selección definidos en el esquema propuesto.

La tabla 9 muestra los resultados obtenidos para la fase de criterios de contexto de la etapa criterios de selección. Esta tabla indica que de acuerdo a los criterios de contexto tomados en cuenta, las herramientas Gantt Project, Poseidón, StarUML y ArgoUML son las adecuadas para la construcción del producto de software en sus diferentes fases.

Asimismo, la tabla 10 muestra los resultados de la fase criterios internos de la etapa criterios de selección. Esta tabla indica que las herramientas Gantt Project, Poseidón o QT y de acuerdo a estos criterios, son los elegidos para la construcción del producto de software.

Por otro lado la tabla 11 muestra los resultados obtenidos para la fase criterios del cliente y selección de la etapa criterios de selección. Esta tabla muestra que las herramientas escogidas para la construcción de producto de software son: Microsoft Project, Netbeans o Visual Studio.Net.

Tabla 9

Resultados de la fase criterios de contexto, etapa criterios de selección.

ETAPA 2: CRITERIOS DE SELECCIÓN																
FASE: CRITERIOS DE CONTEXTO																
CRITERIOS	**PRODUCTOS**															
	Gantt PV	**Microsoft Project**	**Gantt Project**	**REM**	**StarUML**	**Poseidon**	**Requisite Pro**	**QT**	**ArgoUML**	**iReport Designer**	**Netbeans**	**Eclipse**	**Visual Studio.Net**	**Junit**	**Ant**	**Mock Objects**
Software online	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Fácil, usable e intuitivo	1	0	1	1	1	1	0	1	1	0	0	0	0	1	1	1
Versión o periodo de prueba	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Escalabilidad	1	0	1	1	1	1	0	1	1	0	1	1	0	1	1	1
Funciones básicas	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Funciones avanzadas	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Perfiles	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Necesidades particulares	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Atención al cliente	0	1	1	0	1	1	1	0	1	1	1	1	1	0	0	0
Referencias	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
VALOR	0.90	0.80	1.00	0.90	1.00	1.00	0.80	0.90	1.00	0.80	0.90	0.90	0.80	0.90	0.90	0.90

Fuente: Elaboración propia

Tabla 10

Resultados de la fase criterios internos, etapa criterios de selección.

ETAPA 2: CRITERIOS DE SELECCIÓN																
FASE: CRITERIOS INTERNOS																
CRITERIOS	PRODUCTOS															
	Gantt PV	Microsoft Project	Gantt Project	REM	StarUML	Poseidon	Requisite Pro	QT	ArgoUML	iReport Designer	Netbeans	Eclipse	Visual Studio.Net	Junit	Ant	Mock Objects
Conectividad	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0
Medición	1	1	1	0	0	0	1	0	0	0	0	0	0	1	1	1
Curva de aprendizaje	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1
Documentación disponible	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Escalabilidad	0	0	1	0	1	1	1	1	1	1	1	1	1	0	0	0
Estabilidad	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	0
Flexibilidad	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	1
Portabilidad	0	0	1	0	1	1	0	1	1	0	1	1	1	0	0	0
Robustez	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Rendimiento	1	1	1	0	1	1	1	1	1	0	1	1	1	0	0	1
Respaldo	0	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1
Tipo de licencia	0	1	0	0	0	1	1	1	0	1	0	0	1	0	0	1
VALOR	0.58	0.67	0.83	0.33	0.75	0.83	0.75	0.83	0.75	0.67	0.67	0.67	0.75	0.50	0.50	0.67

Fuente: Elaboración propia

Tabla 11

Resultados de la fase criterios del cliente y clasificación, etapa criterios de selección.

ETAPA 2 y 3: CRITERIOS DE SELECCIÓN Y CLASIFICACION																
FASE: CRITERIOS DEL CLIENTE Y CLASIFICACION																
CRITERIOS	**PRODUCTOS**															
	Gantt PV	**Microsoft Project**	**Gantt Project**	**REM**	**StarUML**	**Poseidon**	**Requisite Pro**	**QT**	**ArgoUML**	**iReport Designer**	**Netbeans**	**Eclipse**	**Visual Studio.Net**	**Junit**	**Ant**	**Mock Objects**
Economía	0	1	0	0	0	1	1	1	0	1	0	0	1	0	0	1
Afinidad	1	1	1	1	1	0	1	0	1	1	1	0	1	1	0	1
Reputación del software	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Especialización del software	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
Especialización de la consultoría	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Eficacia en empresas	0	1	1	0	1	1	0	0	0	1	1	1	1	0	0	0
Software cerrado	0	1	0	0	0	1	1	1	1	1	0	0	1	0	1	1
Software a medida	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Conectividad	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0
Soporte	0	1	1	0	1	1	1	1	1	0	1	1	1	1	0	1
Adaptación a la norma	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Evolución del software	0	1	1	0	1	1	0	1	1	1	1	1	1	0	1	1
Migración de datos	0	1	0	1	0	0	1	0	0	0	1	1	0	0	0	0
Seguridad de la información	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Modularización de la herramienta	0	1	1	0	1	0	1	0	1	0	1	1	1	1	0	0
Software libre	1	0	1	1	1	0	0	0	1	0	1	1	0	1	0	1
VALOR	0.50	0.88	0.75	0.56	0.75	0.69	0.75	0.69	0.75	0.69	0.88	0.81	0.88	0.63	0.50	0.75

Fuente: Elaboración propia

4.4 Etapa 3: Clasificación de herramientas de software

Para el caso de estudio, cada vez que un conjunto de herramientas presentaban los mismos valores entonces se escogía una de ellas aplicando los siguientes criterios:

- Conocimiento de la funcionalidad de la herramienta
- Tiempo de uso de la herramienta por parte de la persona encargada.
- Costo generado por el uso de la herramienta.

4.5 Etapa 4: Validación del esquema

Para validar el esquema, las tablas de sistematización fueron llenadas con los puntajes sugeridos en el esquema propuesto. La interpretación llevada a cabo de cada una de ellas muestra que el hecho de poder encontrar y seleccionar una herramienta de software implica que la misma apoya en la construcción del producto en la mayor cantidad de tiempo posible.

Esta aseveración hace posible entender que el hecho de que se indique la herramienta a usar, permita definir, bajo un conjunto de criterios, la herramienta que más valor agregado le otorgará a la construcción del producto de software.

La tabla 12 muestra los resultados de haber aplicado la evaluación a las herramientas definidas por el esquema propuesto y las herramientas tradicionales que la empresa Smart Reason S.R.L. usa para construir los productos de software solicitados.

Tabla 12
Resultados de la validación.

Etapa	Herramienta sugerida	Herramienta tradicional	Índice productividad		Diferencia
			Herramienta Sugerida	Herramienta Tradicional	
Proyecto de software	Microsoft project	Gantt project	0.81	0.63	0.18
Análisis del sistema	REM	Heler	2.34	1.95	0.39
Diseño del sistema	QT	ArgosUML	1.20	0.83	0.37
Construcción de software	Visual Studio.net	Eclipse	5.88	4.12	1.76
Pruebas de software	JUnit	JUnit	3.75	3.08	0.67
PROMEDIO			2.80	2.12	0.68

Fuente: Elaboración propia

Se aprecia que en cada una de las etapas, la herramienta sugerida por el esquema propuesta ha tenido mejor efectividad que la herramienta tradicional, lo que significa que el grado de utilización es mayor. En el caso de las pruebas de software coinciden las herramientas. En términos generales, las herramientas sugeridas por el esquema propuesto presentan una mayor efectividad de uso que las herramientas tradicionales que la empresa usa en el desarrollo de productos de software.

Conclusiones

1. Se elaboró un esquema de trabajo que permite seleccionar un conjunto de herramientas para las diferentes etapas de la construcción del producto de software la misma que se encuentra enmarcada en la conveniencia y necesidad de la organización tomando en cuenta los procesos bien definidos para su construcción exigiendo aplicaciones con calidad, seguridad y estabilidad.

2. Los criterios seleccionados propuestos, muestran en conjunto, que es importante y obligatorio tomar en cuenta las características del producto de software a construir para que a partir de ella se definan las herramientas e emplear en su construcción.

3. Las herramientas para construir los productos de software se encuentran disponibles en el mercado y su selección depende no solo del resultado de la evaluación de sus características especiales, sino también de las necesidades específicas del desarrollo de software recopiladas en los necesidades del cliente e incluso, de las inclinaciones personales del desarrollador encargado del proyecto o de la gerencia del mismo.

4. Se realizó el estudio y análisis comparativo de los aspectos más relevantes de las herramientas, estos son importantes para la buena construcción del producto de software. Como resultado de la evaluación de los criterios propuestos también es posible seleccionar la mejor herramienta o conjunto de ellas para satisfacer las necesidades de la construcción.

Recomendaciones

1. Se recomienda tomar en cuenta todas las características del modelo sistematizado de calidad de software (MOSCA) para concebir criterios que permitan un mejor refinamiento en la decisión de elegir una herramienta computacional para el desarrollo de productos de software.

2. Se recomienda buscar otros modelos de calidad para ser contrastados con los criterios sugeridos permitiendo, de esta manera, expandir los criterios en la localización y selección de herramientas para la construcción de productos de software por pequeñas y medianas empresas.

3. Se recomienda hacer un análisis comparativo de los modelos encontrados y sugerir un modelo híbrido que resuma las características para un modelo que se ajuste a la construcción de software que llevan a cabo las MYPES.

4. Se recomienda llevar a cabo un trabajo de investigación que permita encontrar una metodología de trabajo que conduzca a elaborar metamodelos que permitan definir herramientas de trabajo en la construcción de productos de software.

5. El esquema de trabajo propuesto mantiene una flexibilidad que implica que la empresa desarrolladora de software pueda añadir criterios de selección, no incluidas en la presente investigación, producto de la experiencia profesional. Para estos nuevos criterios se deben de añadir ponderaciones que mantengan la valoración propuesta.

6. Para eliminar tendencias en lo que respecta a preferencia en marcas y/o productos o a la adulteración de valores en la matriz de sistematización, se recomienda que la empresa desarrolladora de software nomine a personal que no conforme el proyecto para que elabore la escala de valores correspondiente.

BIBLIOGRAFIA

[BAMONDE, 2013] Bamonde Rodríguez Sebastián. Evolución Tecnológica e Ingeniería de Software (Una visión desde la perspectiva de la empresa). Tesis para optar el grado de doctor. Departamento de Computación. Universidad La Coruña. 2013.

[BARBIERI, 2008] Barbieri Sebastián. Framework de mejora de procesos de desarrollo de software. Tesis para optar el grado de Magister en Ingeniería de Software. Facultad de Informática. Universidad de la Plata. Argentina. 2008.

[CABALLERO, 2006] Caballero Cervantes Omar. Tecnologías de información y herramientas para la administración de proyectos de software. Revista Digital Universitaria. Instituto Tecnológico de Estudios Superiores de Monterrey. México. Volumen 7 Número 6. ISSN: 1067-6079. 2006

[CHIRINOS & LOPEZ, 2008] Chirinos Gómez Juan Carlos, López Escobedo Tania Cindy. Herramientas para un lenguaje de modelado de sistemas domóticas independiente de la tecnología de implementación. Tesis para optar el título profesional de Ingeniero de Sistemas. Universidad Católica Santa María. 2008.

[DE CORDOVA, 2007] De Córdova López del Solar Adolfo Arturo. Herramienta para el intercambio de plantillas de datos basadas en estándares a través de web services.

Tesis para optar el título profesional de Ingeniero de Sistemas. Universidad Católica Santa María. 2007.

[DIEZ, 2003] Diez Eduardo. Generador del Mapa de Actividades de un Proyecto de Desarrollo de Software. Tesis para optar el grado de Magister en Ingeniería de Software. Universidad Politécnica de Madrid. 2003.

[FUENTES, 2010] Fuentes Revilla José Andrés. Sistema prototipo para firmar digitalmente documentos haciendo uso de criptografía asimétrica y el estándar de certificados digitales x.509 para la Universidad Católica de Santa María. Tesis para optar el título profesional de Ingeniero de Sistemas. Universidad Católica Santa María. 2010.

[GALEANO, 2007] Galeano Carlos. Herramientas de software con licencia pública general para el modelado por elementos finitos. Universidad Nacional de Colombia. 2007.

[GAMA & VIGO, 2006] Gama Contreras Alfredo Berly, Vigo Luna Karen Pamela. Evaluación de las tecnologías de desarrollo web en la construcción de un prototipo de publicación de documentos para el programa profesional de ingeniería de sistemas. Tesis para optar el título profesional de Ingeniero de Sistemas. Universidad Católica Santa María. 2006.

[GRANOLLERS, 2004] Granollers y Saltiveri Toni. MPIu+a: Una metodología que integra la ingeniería del software, la interacción persona-ordenador y la accesibilidad en el contexto de equipos de desarrollo multidisciplinares. Tesis para optar el grado de Doctor en Informática. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Lleida. 2004.

[MENDOZA, 2005] Mendoza Luis y Pérez María. Prototipo de Modelo Sistémico de Calidad (MOSCA) del Software. Computación y Sistemas Vol. 8 Núm. 3, pp. 196-217 © 2005, CIC-IPN, ISSN 1405-5546, Impreso en México. Universidad Simón Bolívar. Caracas. Venezuela.

[RODRIGUEZ, 2008] Rodríguez Gonzáles Pilar. Estudio de la aplicación de metodologías ágiles para la evolución de productos software. Tesis para optar el grado de Magister en Tecnologías de la Información. Facultad de Informática. Universidad Politécnica de Madrid. 2008.

[ROSAS, 2005] Rosas Zegarra Jordan Stuart. Diseño de una herramienta para el modelamiento, implementación y mantenimiento de una base de datos en MYSQL. Tesis para optar el título profesional de Ingeniero de Sistemas. Programa Profesional de Ingeniería de Sistemas. Universidad Católica Santa María. 2005.

[RUIZ, 2011] Ruíz Bertol Francisco Javier. Técnicas Conceptuales en la Gestión de Proyectos Software. Tesis para optar el grado de Doctor en Informática. Facultad de Informática. Universidad del País Vasco. 2011.

ANEXO A

FICHA DE ESCALA DE VALORES

ETAPA 1: DEFINICION DE PATRONES										
FASE: PROYECTOS DE SOFTWARE										
PRODUCTOS	SUB-ETAPAS	CRITERIOS							VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7		

Tabla 13 *Escala de valores para la fase de proyectos de software*
Fuente: Elaboración propia

ETAPA 1: DEFINICION DE PATRONES												
FASE: ANALISIS DE SISTEMAS												
PRODUCTOS	SUB-ETAPAS	CRITERIOS									VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	

Tabla 14 *Escala de valores para la fase de análisis de sistemas*
Fuente: Elaboración propia

ETAPA 1: DEFINICION DE PATRONES										
FASE: DISEÑO DEL SISTEMA										
PRODUCTOS	SUB-ETAPAS	CRITERIOS							VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7		

Tabla 15 *Escala de valores para la fase de diseño del sistema*
Fuente: Elaboración propia

ETAPA 1: DEFINICION DE PATRONES										
FASE: CONSTRUCCION DE SOFTWARE										
PRODUCTOS	SUB-ETAPAS	CRITERIOS							VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4	C5	C6	C7		

Tabla 16 *Escala de valores para la fase de construcción de software*
Fuente: Elaboración propia

ETAPA 1: DEFINICION DE PATRONES							
FASE: PRUEBAS DE SOFTWARE							
PRODUCTOS	SUB-ETAPAS	CRITERIOS				VALOR ITEM	VALOR PRODUCTO
		C1	C2	C3	C4		

Tabla 17 *Escala de valores para la fase de pruebas de software*
Fuente: Elaboración propia

ETAPA 2: CRITERIOS DE SELECCIÓN									
FASE: CRITERIOS DE CONTEXTO									
CRITERIOS	PRODUCTOS								
VALOR									

Tabla 18 *Escala de valores para la fase de criterios de contexto*
Fuente: Elaboración propia

ETAPA 2: CRITERIOS DE SELECCIÓN									
FASE: CRITERIOS INTERNOS									
CRITERIOS	PRODUCTOS								
VALOR									

Tabla 19 *Escala de valores para la fase de criterios internos*
Fuente: Elaboración propia

ETAPA 2 y 3: CRITERIOS DE SELECCIÓN Y CLASIFICACION							
FASE: CRITERIOS DEL CLIENTE Y CLASIFICACION							
CRITERIOS	PRODUCTOS						
VALOR							

Tabla 20 *Escala de valores para la fase de criterios del cliente y clasificación*
Fuente: Elaboración propia

ANEXO B

DIAGRAMA DEL ESQUEMA PROPUESTO

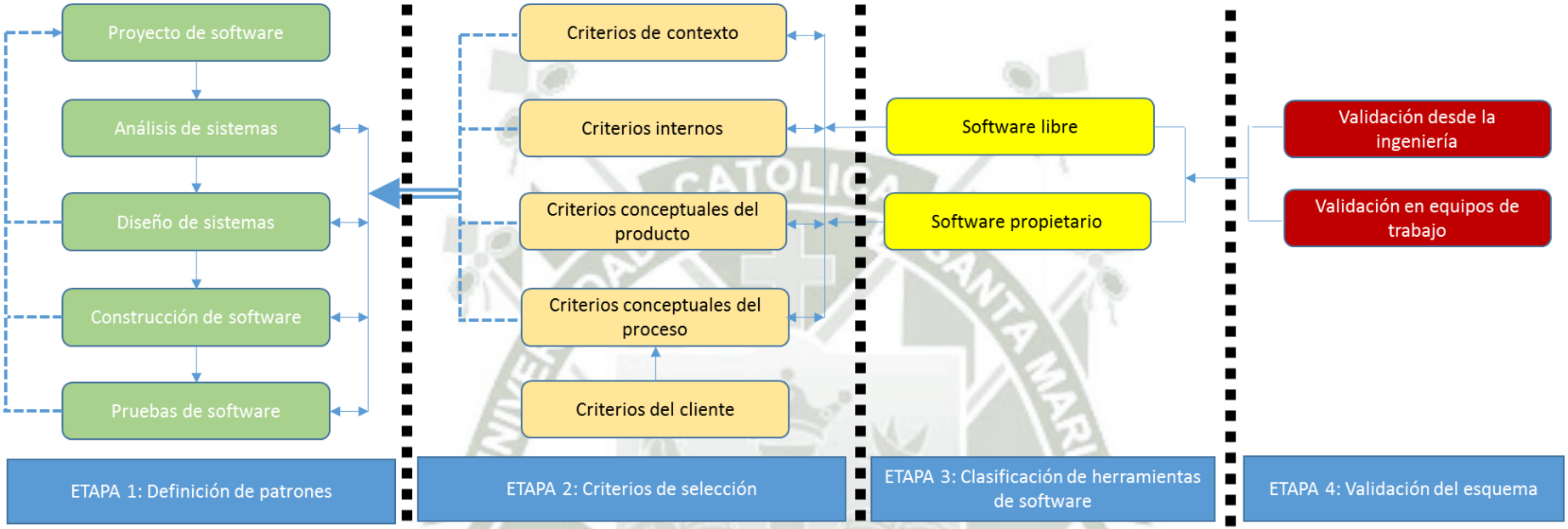

Figura B.1. Diagrama del esquema propuesto

Fuente: Elaboración propia

ANEXO C

GLOSARIO DE TERMINOS

ACTIVITY–ON–ARROW (AOA)	Es el tipo de diagrama de red donde las actividades se encuentran representadas a través de flechas, y los eventos a través de nodos.
ACTIVITY–ON–NODE (AON)	Determina que las actividades están representadas a través de nodos y, consecuentemente, los arcos que unen actividades representan relaciones y/o dependencias entre actividades. El evento se toma como una actividad de duración cero.
ARTEFACTO DE SOFTWARE	Un artefacto es un producto tangible resultante del proceso de desarrollo de software. Algunos artefactos como los casos de uso, diagrama de clases u otros modelos UML ayudan a la descripción de la función, la arquitectura o el diseño del software. Otros se enfocan en el proceso de desarrollo en sí mismo, como plan de desarrollo.
BD	Sigla que significa Base de Datos Se le llama base de datos a los bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.
CALIDAD TOTAL	La Gestión de la Calidad Total (abreviada TQM, del inglés Total Quality Management) es una estrategia de gestión desarrollada en las décadas de 1950 y 1960 por las industrias

	japonesas, a partir de las prácticas promovidas por el experto en materia de control de calidad W. Edwards Deming, impulsor en Japón de los círculos de calidad, también conocidos, en ese país, como “Círculos de Deming”.
COBOL	El lenguaje COBOL (acrónimo de COmmon Business-Oriented Language, Lenguaje Común Orientado a Negocios) fue creado en el año 1959 con el objetivo de crear un lenguaje de programación universal que pudiera ser usado en cualquier ordenador (ya que en los años 1960 existían numerosos modelos de ordenadores incompatibles entre sí), y que estuviera orientado principalmente a los negocios, es decir, a la llamada informática de gestión.
CoMoSoft	Sigla que significa Software para el Control de la Morosidad.
CONFIABILIDAD	Grado en el que un programa se espera que realice su función con una precisión requerida.
CPM	El método CPM (Crítical Path Method), el segundo origen del método actual, fue desarrollado también en 1957 en los Estados Unidos de América, por un centro de investigación de operaciones para la firma Dupont y Remington Rand, buscando el control y la optimización de los costos de

	operación mediante la planeación adecuada de las actividades componentes del proyecto.
DESMET	El método DESMET propone un conjunto de criterios prácticos y técnicos para la selección de un método para evaluar métodos.
DIAGRAMA GANTT	El diagrama de Gantt es una útil herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. A pesar de esto, el Diagrama de Gantt no indica las relaciones existentes entre actividades.
EDUCCION DE REQUISITOS	Actividad que entrega el artefacto donde se plasma los requisitos o necesidades del cliente.
EFICIENCIA	El conjunto de recursos informáticos y de código necesarios para que un programa realice su función
ELICITACION DE REQUISITOS	Actividad que entrega el artefacto donde se plasma los requisitos o necesidades del analista.
ESPECIFICACION DE REQUISITOS	Actividad que entrega el artefacto donde se plasma los requisitos o necesidades del analista.

FINISH-START	Que define una secuenciación entre las tareas, y suele indicar que el producto resultado de la primera tarea (informe, código fuente, aplicación o entregable) es necesaria para realizar la siguiente tarea.
DIAGRAMA DE HITOS	El diagrama de hitos se basa en eventos planificados en el proyecto, en vez de en la duración de las actividades, y proporciona una medida para decidir si se continúa con el proyecto o la toma de acciones correctivas que permitan finalizar dicho proyecto en tiempo y calidad planificado
GESTION DE PROYECTOS	La gestión de proyectos es la disciplina del planeamiento, la organización, la motivación, y el control de los recursos con el propósito de alcanzar uno o varios objetivos. Un [proyecto] es un emprendimiento temporario diseñado a producir un único producto, servicio o resultado1 con un principio y un final definidos (normalmente limitado en tiempo, en costos y/o entregables), que es emprendido para alcanzar objetivos únicos2 y que dará lugar a un cambio positivo o agregará valor.
HERRAMIENTA CASE	Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas o programas informáticos destinadas a aumentar la productividad en el desarrollo de

	software reduciendo el costo de las mismas en términos de tiempo y de dinero.
HERRAMIENTAS DE SOFTWARE	Una herramienta de desarrollo de software es un programa informático que usa un programador para crear, depurar, gestionar o mantener un programa.
IDEF0	Integration DEFinition for Function Modeling) es un método basado en la combinación de una representación gráfica de funciones y un texto explicativo para el modelado de decisiones, acciones, y actividades de una organización o sistema
IEEE	El Instituto de Ingeniería Eléctrica y Electrónica —abreviado como IEEE, leído i-triple-e en Hispanoamérica o i-e-cubo en España; en inglés Institute of Electrical and Electronics Engineers— es una asociación mundial de ingenieros dedicada a la estandarización y el desarrollo en áreas técnicas.
MANTENIBILIDAD	Capacidad de un producto software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en el entorno, en los requerimientos o en las especificaciones funcionales.

MAXSER	Cooperativa que otorga préstamos a sus asociados, es el acrónimo de MAXimo SERvicio.
METRICA	Medida, estructura y combinación de los versos de una determinada composición poética, de un escritor, de una época o de un lugar.
MIL-HDBK-881ª	MIL-HDBK-881A, DEPARTMENT OF DEFENSE HANDBOOK: WORK BREAKDOWN STRUCTURES FOR DEFENSE MATERIEL ITEMS (30-JUL-2005) [S/S BY MIL-STD-881C]
MOSCA	Modelo que analiza las relaciones sistémicas presentes entre las categorías del proceso de desarrollo de software y las características de calidad correspondientes al producto, establecen que las metas para la calidad del producto de software van a determinar los objetivos del proceso para su desarrollo.
MYPES	Abreviatura de micro y pequeñas empresas.
PARMA	Project Activity Representation MAtrix), modelo que lleva a cabo la representación matricial de la información de los proyectos desarrollando una ontología que obtiene el conocimiento para el dominio de los proyectos, la misma que es empleada para llevar a cabo el razonamiento.

PERT	El método PERT es una técnica que le permite dirigir la programación de su proyecto. El método PERT consiste en la representación gráfica de una red de tareas, que, cuando se colocan en una cadena, permiten alcanzar los objetivos de un proyecto.
PL/I PL/1	Acrónimo de Programming Language 1 (Lenguaje de Programación 1), fue propuesto por IBM hacia 1970 para responder simultáneamente a las necesidades de las aplicaciones científicas y comerciales, disponible en las novedosas plataformas de utilidad general IBM 360 y más adelante IBM 370. Este lenguaje tenía muchas de las características que más adelante adoptaría el lenguaje C y algunas de C++
PORTABILIDAD	El esfuerzo necesario para trasladar el programa de un entorno de sistema hardware y/o software a otro.
PROYECTO DE SOFTWARE	La definición de un proyecto de software se hace identificando los requerimientos del software a partir de las necesidades que se deben satisfacer del negocio. La definición cubre aspectos funcionales del proyecto y los atributos de calidad que debe tener en términos de desempeño, facilidad de uso, confiabilidad, seguridad, y facilidad de mantenerlo a lo largo del tiempo.

PRUEBA DE SOFTWARE	Las pruebas de software (en inglés software testing) son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o stakeholder. Es una actividad más en el proceso de control de calidad. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo.
PUENTES DE KONIGSBERG	El problema de los puentes de Königsberg, también llamado más específicamente problema de los siete puentes de Königsberg, es un célebre problema matemático, resuelto por Leonhard Euler en 1736 y cuya resolución dio origen a la teoría de grafos. Su nombre se debe a Königsberg, la ciudad de Prusia Oriental y luego de Alemania que desde 1945 se convertiría en la ciudad rusa de Kaliningrado.
REDES DE PETRI	Una Red de Petri es una representación matemática o gráfica de un sistema a eventos discretos en el cual se puede describir la topología de un sistema distribuido, paralelo o concurrente. La red de Petri esencial fue definida en la década de los años 1960 por Carl Adam Petri. Son una

	generalización de la teoría de autómatas que permite expresar un sistema a eventos concurrentes.
SCHEDULING	El planificador (en inglés scheduler) es un componente funcional muy importante de los sistemas operativos multitarea y multiproceso, y es esencial en los sistemas operativos de tiempo real. Su función consiste en repartir el tiempo disponible de un microprocesador entre todos los procesos que están disponibles para su ejecución.
SGBD	Un sistema gestor de base de datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. Los usuarios pueden acceder a la información usando herramientas específicas de interrogación y de generación de informes, o bien mediante aplicaciones al efecto.
SOFTWARE LIBRE	El término software libre refiere el conjunto de software que por elección manifiesta de su autor, puede ser copiado, estudiado, modificado, utilizado libremente con cualquier fin y redistribuido con o sin cambios o mejoras. Su definición está asociada al nacimiento del movimiento de software libre, encabezado por Richard Stallman y la

	consecuente fundación en 1985 de la Free Software Foundation, que coloca la libertad del usuario informático como propósito ético4 fundamental.
SOFTWARE PROPIETARIO	El término ha sido creado para designar al antónimo del concepto de software libre, por lo cual en diversos sectores se le han asignado implicaciones políticas relativas al mismo. Para la Fundación para el Software Libre (FSF), este concepto se aplica a cualquier programa informático que no es libre o que sólo lo es parcialmente (semilibre), sea porque su uso, redistribución o modificación está prohibida, o sea porque requiere permiso expreso del titular del software.
SQL	SQL (por sus siglas en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas.
TARJETA CRC	Las tarjetas CRC (Clase-Responsabilidad-Colaboración) son una herramienta de brainstorming, usado como metodología para el diseño de software orientado a

	objetos creada por Kent Beck y Ward Cunningham.
TEORIA DE GRAFOS	La teoría de grafos (también llamada teoría de las gráficas) es un campo de estudio de las matemáticas y las ciencias de la computación, que estudia las propiedades de los grafos (también llamadas gráficas, que no se debe confundir con las gráficas que tienen una acepción muy amplia) estructuras que constan de dos partes, el conjunto de vértices, nodos o puntos; y el conjunto de aristas, líneas o lados que pueden ser orientados o no. Por ello, también se conoce como análisis de redes.
TEORIA DE REDES	El análisis de redes es el área encargada de analizar las redes mediante la teoría de redes (conocida más genéricamente como teoría de grafos). Las redes pueden ser de diversos tipos: social, transporte, eléctrica, biológica, internet, información, epidemiología
USABILIDAD	El esfuerzo necesario para aprender, operar, y preparar datos de entrada e interpretar las salidas (resultados) de un programa.
WBS	Una Estructura de Descomposición del Trabajo o EDT, también conocida por su nombre en inglés Work Breakdown Structure o WBS, es en gestión de proyectos una descomposición jerárquica orientada al entregable, del trabajo a ser ejecutado por el

equipo de proyecto, para cumplir con los objetivos de éste y crear los entregables requeridos, con cada nivel descendente de la EDT representando una definición con un detalle incrementado del trabajo del proyecto. La EDT es una herramienta fundamental en la gestión de proyectos.