UCI

UNIVERSIDAD DE LAS CIENCIAS

INFORMÁTICAS

Proceso de desarrollo para portales Web

Trabajo de diploma para optar por el título de:

Máster en Gestión de Proyectos Informáticos

Autor: Ing. William Santana Méndez

Tutora: Msc. Yaimí Trujillo Casañola

Ciudad de la Habana, julio de 2010

Declaración de autoría

Declaro por este medio que yo William Santana Méndez soy el autor principal de la tesis de maestría Proceso de desarrollo para portales Web desarrollada como parte de la Maestría en Gestión de Proyectos Informáticos y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio así como los derechos patrimoniales de la misma con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Nombre del autor Firma

Dedicatoria

A todos los que de una forma u otra me han apoyado durante toda mi vida, alentándome a seguir alcanzando mis metas paso a paso.

Agradecimientos

A mi familia, a mis amistades y a todas las personas que han contribuido a la realización de esta investigación.

Resumen

La presente investigación expone la implantación de un modelo de factoría de software en la fábrica de portales Web de la Facultad #10 de la Universidad de las Ciencias Informáticas. La implantación del modelo tiene como principal objetivo aumentar la eficiencia en el desarrollo de portales y para ello propone un modelo funcional basado en la reutilización de componentes y conocimiento. Además se propone un proceso de desarrollo conformado para lograr la estandarización en el desarrollo de portales y modelar con calidad un producto de este tipo. Durante la investigación se analizaron los principales modelos de producción de software existentes en el mundo. Se identificaron las principales características de los sistemas Web y finalmente se realizó un análisis de una amplia gama de metodologías que se utilizan para desarrollar sistemas Web. Como resultado se identificaron un conjunto de buenas prácticas, que finalmente se utilizaron para conformar el proceso propuesto.

Abstract

This research describes the implementation of a software factory model in the Web portals factory's of the Faculty #10 of the University of Informatics Sciences. The implementation of the model's main objective is to increase efficiency in the development of portals and this suggests a functional model based on component reuse and knowledge. It also proposes a development process made to achieve standardization in the development of portals and a product quality model of this type. During the study analyzed the main models of software production in the world. We identified the main characteristics of Web systems and finally an analysis of a wide range of methodologies used to develop Web systems. As a result we identified a set of best practices, which were finally used to form the proposed process.

Índice

Introducción

En la actualidad una de las industrias con mayor crecimiento a nivel mundial, es la industria del software, donde cada año se reportan cifras millonarias alrededor de la misma. Por tal motivo el número de empresas dedicadas a esta rama crece constantemente y a su vez, aumenta la competencia. Producto de tales circunstancias, una gran parte de dichas empresas se ven obligadas a mejorar su capacidad de proporcionar soluciones para sus clientes, mejorando para ello, sus niveles de producción y elevando la calidad de sus productos y servicios.

En Cuba, uno de los centros más destacados en el desarrollo de sistemas y servicios informáticos es la **Universidad de las Ciencias Informáticas (UCI)**. Su misión es producir, a partir de la vinculación estudio – trabajo como modelo de formación. La producción en la UCI se basa en la integración de los procesos de formación, investigación y producción en torno a una temática para convertirla en una rama productiva. Esta integración garantiza la innovación continua que genera y aporta valor a los productos y servicios, promueve la gestión del conocimiento garantizando un mayor rendimiento, logra una mejor utilización y aprovechamiento de los recursos humanos y materiales, generando alta especialización y colaboración.

En la UCI existe una gran cantidad de proyectos productivos en los que se desarrolla un considerable número de productos de software. Varios de ellos, se encuentran enmarcados en el ámbito de los sistemas Web y una de las entidades dentro de la UCI, que se dedica a su producción, es la fábrica de portales de la Facultad #10. Dicha área presenta dos grandes contratiempos, por una parte, posee un enorme potencial para la reutilización de componentes y sin embargo no lo puede aprovechar, ya que no se rige por un modelo de producción que lo propicie. Como consecuencia los proyectos demoran mucho más de lo que durarían si el desarrollo estuviese basado en el aprovechamiento de componentes reutilizables. Por otra parte, se utiliza como proceso de desarrollo de software la metodología *Rational Unified Process* (RUP), sin adaptarla al contexto del

proyecto. Se parte de la idea de que el ciclo de vida es el mismo para todos los tipos de sistemas (los tradicionales y los sistemas Web). Se definen un conjunto de artefactos que se deben generar y perfeccionar por cada iteración que se ejecute. La única diferencia que se aplica entre los sistemas tradicionales y los sistemas Web, son los estereotipos de los artefactos construidos a través del Lenguaje Unificado de Modelado (*Unified Modeling Language*, en inglés, abreviado UML) que utiliza RUP. Como consecuencia, no se logra obtener un correcto entendimiento y modelación de un sistema Web; ya que siguiendo dicha idea no se consideran en profundidad aspectos como la estructura de navegación, donde el usuario debe entender dónde puede ir y cómo llegar al lugar deseado y la interfaz de la aplicación Web, donde no sólo se necesita especificar cuáles son los objetos de la interfaz que deberían ser implementados, sino también la manera en la cual estos objetos interactuarán con el resto de la aplicación.

Teniendo en cuenta la **problemática** presentada con anterioridad, se enuncia el siguiente **problema de investigación**: ¿Cómo mejorar la eficiencia del desarrollo de portales Web en la fábrica de la Facultad #10 de la UCI?

Como respuesta tentativa al problema planteado se formula la siguiente **hipótesis**: si se implantara un modelo de factoría de software en la fábrica de portales Web de la Facultad #10 mejoraría su eficiencia en la producción.

De ahí que el **objeto de estudio** sea el proceso de desarrollo de software y el **campo de acción** el proceso de desarrollo de los proyectos de portales Web.

La investigación estará encaminada a cumplir el **objetivo general** de mejorar la eficiencia del desarrollo de portales Web en la fábrica de la Facultad #10 de la UCI. A su vez dicho objetivo estará desglozado en los siguientes **objetivos específicos:**

- ✓ Proponer, implantar y evaluar un modelo de producción de software orientado a la reutilización de componentes y la industrialización.

- ✓ Conformar un proceso estandarizado para el desarrollo de proyectos de portales Web.

Como vía para alcanzar los objetivos planteados, se definieron las siguientes **tareas de investigación**:

- ✓ Sistematización teórica de los antecedentes y estado actual de la problemática.
- ✓ Conformación de un proceso para el desarrollo de los proyectos de portales Web de la Facultad #10 de la UCI.
- ✓ Selección y puesta en práctica de un método para la evaluación técnica del proceso propuesto.
- ✓ Aplicación y evaluación del proceso propuesto en un proyecto piloto.
- ✓ Implantación y evaluación del modelo de producción propuesto.

La **población** que se estudiará estará definida por todos los proyectos de portales Web desarrollados en la fábrica de portales de la Facultad #10 de la UCI, constituyendo cada portal, la **unidad de estudio** y tomando como **muestra** todos los existentes, lo cual constituye el 100 % de la población.

La estrategia a seguir será la **investigación descriptiva**; empleando **métodos teóricos** tales como: el **histórico** y el **sistémico** y dentro de los **métodos empíricos**: la **encuesta** y la **entrevista no estructurada**.

El documento estará estructurado en cuatro capítulos, conformados con la información que a continuación se presenta.

Capítulo 1: se realiza la fundamentación teórica del tema, donde se exponen los criterios y opiniones respecto al tema de investigación, expuesto en la bibliografía consultada por los autores a nivel nacional e internacional. Se exponen los principales modelos de producción de software existentes en el mundo, abordando principalmente el modelo de factoría de software. Se analizan los principales modelos de factoría y los resultados de varias entidades que han adoptado dicho enfoque. Finalmente se realiza un estudio y análisis crítico de varias metodologías de desarrollo que se utilizan para modelar sistemas Web.

Capítulo 2: se detalla el entorno, se describen los métodos utilizados y finalmente se exponen los resultados de las encuestas y entrevistas realizadas.

Capítulo 3: se describe la propuesta del proceso de desarrollo para portales Web. Para ello se detallan las fases, actividades, tareas, artefactos de entrada/salida y roles con sus respectivas responsabilidades en cada fase.

Capítulo 4: se valida la propuesta con especialistas en el tema, se exponen los resultados de la aplicación del proceso de desarrollo propuesto en un proyecto piloto y por último se muestran los efectos de la implantación del modelo Aplicando Inteligencia en la fábrica de portales de la Facultad #10 de la UCI.

Capítulo 1. Fundamentación teórica

En el presente capítulo se exponen los principales modelos de producción de software existentes en el mundo, abordando principalmente el modelo de factoría de software. Finalmente se realiza un análisis crítico de varias metodologías que se utilizan para el desarrollo de sistemas Web, destacando sus principales características.

1.1. Modelos de producción de software

Los modelos de producción juegan un papel fundamental en el desarrollo de software. Promueven la reutilización de los diferentes elementos del software y facilitan la labor de los diferentes roles que participan del proceso. La reutilización es una de las estrategias que se considera promisoria para que esta industria pueda enfrentar el reto de desarrollar productos con niveles de calidad y productividad adecuados en un contexto de negocio altamente complejo y dinámico y con acelerados cambios tecnológicos. El uso de plantillas, componentes de granularidad gruesa, patrones de diseño, arquitecturas de referencia, frameworks, entre otros, son mecanismos cada vez más utilizados por los desarrolladores de software. El objetivo de dichas prácticas es lograr que la reutilización se integre de forma sistémica en las diferentes etapas del desarrollo, de tal manera que su impacto en los diferentes artefactos resultantes del proceso de desarrollo sea efectivo y, en lo posible medible (JACOBSON, 1997).

Paradigmas de modelado de sistemas tradicionales

Los paradigmas de modelado de sistemas tradicionales se basan en lenguajes de programación, plataformas de objetos, componentes de software y la generación de documentos. Estos últimos resultan muy difíciles de mantener en sincronía con los cambios realizados en el código. Esto puede provocar que si ocurren cambios dentro de las especificaciones iníciales, se corre riesgo de que el modelo no se corresponda con el sistema desarrollado. Una de las causas principales de este problema es que dichos métodos de modelado y los ambientes de desarrollo

existentes no son muy buenos en separar los detalles arquitectónicos y tecnológicos del espacio conceptual del problema. Como consecuencia los sistemas producidos son extremadamente sensibles a los cambios en el ambiente de desarrollo, y al mismo tiempo, se vuelve muy difícil, reflejar de forma directa en la implementación del sistema los cambios producidos en el dominio. Los paradigmas tradicionales dependen en gran medida de la infraestructura tecnológica y requieren un alto nivel de experiencia y conocimiento por parte de los equipos de desarrollo; lo que incrementa los costos de contratación de personal o entrenamiento del mismo (STEPHEN J. MELLOR, 2003).

Actualmente la Comunidad Internacional de Ingeniería de Software presta un especial interés a dos enfoques para el desarrollo de software. Por una parte las Factorías de Software promovidas por *Microsoft* y por otra parte, el Desarrollo Dirigido por Modelos, propuesto por el *Object Management Group* (OMG).

Factoría de Software

El modelo de Factoría de Software se centra en el desarrollo de sistemas similares promoviendo la reutilización de arquitecturas, componentes de software y conocimiento. En la factoría, tanto la arquitectura como el *framework* de soporte, hacen uso de patrones que encapsulan conocimiento sobre el dominio. Según Greenfield (2004*) "Una factoría de software puede definirse como una línea de productos de software que configura herramientas extensibles, procesos y contenido, con el objetivo de automatizar el desarrollo y mantenimiento de variantes de un producto".* Para ello cuenta con lenguajes específicos del dominio, una arquitectura común y la posibilidad de llevar a cabo la adaptación, ensamblaje y configuración de componentes basados en un *framework* de soporte. Como vía para alcanzar este objetivo, las factorías de software utilizan técnicas tales como:

- ✓ Construcción de familias de software similar. Se desarrolla una arquitectura común para varios sistemas y un *framework* que soporte dicha arquitectura.
- ✓ Ensamblado de componentes. Para la construcción de todos los nuevos sistemas reutilizan los componentes proporcionados por el *framework*, ya sea mediante el ensamblado y/o configuración de los mismos.

- ✓ Desarrollo de lenguajes de modelado y herramientas específicas para el dominio. Se utilizan lenguajes de modelado específicos para describir los requisitos del sistema y a partir de estas especificaciones se genera automáticamente el código.
- ✓ Uso de una planificación basada en restricciones. El proceso de desarrollo debe estar bien definido y adaptado al contexto del sistema.

En resumen, las factorías de software utilizan métodos específicos para cada tipo de aplicación. Con dichos métodos y lenguajes de modelado, los desarrolladores definen cada sistema. A partir de las especificaciones realizadas durante la descripción del sistema se genera automáticamente el código que desarrolla o configura el *framework* que sigue la arquitectura, la cual será compartida por todos los sistemas desarrollados en la factoría.

Desarrollo Dirigido por Modelos

El Desarrollo Dirigido por Modelos, busca elevar el nivel de abstracción en el desarrollo de software y para ello utiliza los modelos como el recurso principal durante el proceso de desarrollo. Convierte los modelos y las transformaciones entre ellos en los principales artefactos de todas las fases del proceso; lo cual permite generar directamente los sistemas de software a partir de modelos de dicho sistema. (GIACHETTI G, 2008).

Entre las propuestas de paradigmas basados en el Desarrollo Dirigido por Modelos se encuentra *Domain Driven Modeling* (MDD), que como su nombre lo indica es una aproximación al desarrollo de software basada en el modelo de negocio o dominio. Según Koutsoukos (2002), *"MDD considera que la complejidad de un software depende en gran medida de que tan complejo sea el contexto en el cual se encuentra enmarcado"*. Es por ello que propone que se dedique el tiempo necesario para comprender los detalles esenciales del dominio, antes que a las especificaciones del software en sí, lo cual le permite a los desarrolladores lograr un nivel de abstracción con respecto a las plataformas de desarrollo, lenguajes de programación, etc. Para materializar esta idea MDD propone el uso de un modelo

independiente de la plataforma de desarrollo *Platform Independent Model* (PIM), que represente los conceptos principales del dominio y las reglas del mismo y otro modelo que represente los detalles tecnológicos del sistema a construir *Platform Specific Model* (PSM) (G. KOUTSOUKOS, 2002).

MDD proporciona una estrategia general a seguir en el desarrollo de software; pero no define técnicas a utilizar, ni fases del proceso, ni ningún tipo de guía metodológica.

Por otra parte la propuesta *Model Driven Architecture* (MDA), desarrollada por *Object Managment Group* (OMG) es una arquitectura que defiende el modelo MDD y posee el valor añadido de proporcionar lenguajes con los que definir métodos que sigan esta aproximación (UML, CWM, MOF, QVT). Sin embargo MDA no es por si mismo un método que define técnicas, etapas, artefactos, etc., solamente proporciona la infraestructura tecnológica y conceptual con la que construir estos métodos MDD (PIRES, 2004).

Como vía para alcanzar este objetivo, MDA utiliza técnicas tales como:

- ✓ Uso de técnicas de modelado que permitan especificar el sistema de manera independiente de la plataforma. La especificación MDA sugiere *Unified Modelling Language* (UML) para la creación de modelos y *Meta Object Facility* (MOF) para la creación de metamodelos.
- ✓ Uso de estrategias que permitan transformar los modelos independientes de la plataforma (PIM) en modelos dependientes de la plataforma (PSM). OMG sugiere para la especificación de las transformaciones, el estándar Query View Transformation (QVT). Luego, a partir de los modelos, se genera el código de la aplicación.
- ✓ Uso de herramientas de modelado que soporten técnicas de modelado propuestas por MDA para construir los modelos.

En resumen los sistemas realizados mediante MDA utilizan la abstracción de sus modelos para desarrollar aplicaciones independientes de la plataforma de desarrollo y de los lenguajes de programación. Este enfoque permite lograr un alto grado de reutilización y flexibilidad en el desarrollo de los sistemas, ya que, si se

necesita cambiar la plataforma o el lenguaje de programación, puede llevarse a cabo sin demasiadas complicaciones.

1.1.1. Consideraciones sobre los modelos de producción de software

Tanto las factorías de software como MDA, hacen énfasis en determinados aspectos del proceso de desarrollo de software. Esto les concede ciertas ventajas respecto al otro en aspectos específicos. La utilización de un enfoque u otro, depende de cuáles sean las premisas a seguir en el proceso de desarrollo. Por un lado el enfoque MDA es conveniente cuando se necesita un alto nivel de interoperabilidad con distintas herramientas que sigan los estándares de OMG. Por otra parte el enfoque de las Factorías de Software es conveniente cuando existe la intención de construir una serie de sistemas similares y/o se va trabajar dentro de un dominio determinado. En el anexo 2, se muestra para cada modelo; sus principales características, ventajas y desventajas.

Teniendo en cuenta este análisis, se decide aplicar el enfoque de factoría de software, para el desarrollo de portales Web en la fábrica de la Facultad #10 de la UCI. Dicha decisión tiene como basamento que este enfoque se centra en el desarrollo de sistemas similares, como es el caso de los portales que se desarrollan en la Facultad #10, donde se promueve la reutilización del patrón de diseño, Modelo – Vista – Controlador como arquitectura de software y el Sistema de Gestión de Contenido Drupal, como *framework*, que soporta dicha arquitectura y sobre el cual se desarrollan todos los portales de la fábrica.

Otra característica del modelo de factoría que puede ser muy bien aprovechada, es la reutilización; ya que en la fábrica de portales existen grandes posibilidades de reutilizar tanto el conocimiento de los miembros del proyecto como los componentes de software, por ejemplo los módulos que utiliza Drupal.

Por tanto se considera que la implantación de este enfoque, favorecería la creación de componentes reutilizables, que serían utilizados para automatizar el desarrollo y mantenimiento de variantes de los productos, mediante la adaptación, ensamblaje y configuración de dichos componentes, lo cual permite dividir el trabajo propiciando un mayor grado de especialización en los miembros del

proyecto, logrando así una considerable reducción de los costos y el tiempo de desarrollo, contribuyendo al aumento de la productividad y la calidad.

1.2. Orígenes y evolución de las factorías de software

El término de Factoría de software fue usado por primera vez entre 1950 y 1960 en los Estados Unidos y Europa con el objetivo de mejorar la productividad en el desarrollo de software, a través de la reutilización de componentes y la estandarización de herramientas (GESTIÓN, 2008). Existen tres épocas fundamentales en la evolución de la estrategia que ha guiado a las factorías de software (CAPRIO, 2005). En el anexo 1 se muestra la evolución de la estrategia en las factorías de software desde su surgimiento hasta la actualidad.

En el mundo existe un gran número de expresas con una amplia experiencia en temas de factoría de software, las cuales son reconocidas por sus destacados resultados en la producción de software. Según el Grupo Calidad del Software Ingeniería y Gestión (2008), entre los principales patrocinadores de factorías de software en la actualidad se destacan empresas como:

Ingeniería de Software Avanzado (INSA)

Es una compañía de IBM, fundada el 31 de Octubre de 1991 como un proyecto empresarial entre IBM España y Catalana Occidental. Su trabajo consiste en el diseño, desarrollo, gestión y explotación de sistemas de información basados en la tecnología de la información. Es considerada como el primer proveedor mundial de servicios de Desarrollo y Gestión de Aplicaciones y cuenta con más de 30 Centros Factoría en 27 países. Además cuenta con varios Centros de Innovación Tecnológica y con un fuerte crecimiento de negocio (INSA, 2008).

VECTOR SW FACTORY

Es una compañía de desarrollo de componentes y construcción de soluciones software avanzadas basadas en componentes reutilizables, siempre en entornos de nuevas tecnologías. Fue fundada en el ano 2000 con el objetivo de desarrollar una Red de Centros de Producción de Componentes Software orientada al desarrollo y mantenimiento de componentes y soluciones complejas para empresas y organizaciones de diversos sectores. *VECTOR SW FACTORY*

proporciona a sus clientes soluciones flexibles y adaptables a las necesidades específicas de cada proceso de negocio. Se basa en la implantación de metodologías para optimizar la producción de componentes en volumen y asegurar altos niveles de calidad con resultados óptimos en rendimiento y plazos (FACTORY, 2008).

En Cuba existen algunas empresas que han adoptado el enfoque de factoría de software, como vía para mejorar su capacidad de proporcionar soluciones de software para sus clientes, adaptadas a los cambios y avances tecnológicos. Entre ellas, una de las más prestigiosas es la empresa Softel, la cual se dedica al desarrollo de soluciones informáticas para el Sistema de Salud. Según García (2010), jefe de la factoría de Softel, dicha empresa ha hecho un gran esfuerzo para implantar este enfoque, creando una estructura organizativa en su personal de trabajo que le permite organizar y controlar el desarrollo de los proyectos de forma satisfactoria. No obstante, en Softel no existe una investigación teórica sobre los distintos modelos de factoría de software existentes en el mundo, lo cual los limita de adoptar las características de un modelo acorde al contexto de su empresa y del tipo de producto que desarrolla. Por ejemplo una de las bases de una factoría la constituye la reutilización de componentes y conocimiento y en el caso de Softel no posee un mecanismo que le permita aprovechar las ventajas de la reutilización.

1.3. Modelos de Factoría de software

Para definir el modelo de factoría de software a implantar en la fábrica de portales de la Facultad #10 de la UCI, se abordarán 6 modelos que según Trujillo (2007) constituyen los más representativos:

Modelo propuesto por Basili

El modelo propuesto por Basili, propone dividir el área de producción en dos sub-áreas: la de producción de software y la de producción de componentes. Donde la sub-área de producción de software realiza las solicitudes de productos (componentes para la producción del software), de datos (estadísticas para la

estimación de costos y plazos) y de planos (modelos, métodos para el análisis y diseño del software) a la sub-área de producción de componentes.

La sub-área de producción de componentes posee una base de componentes reutilizables, los cuales pueden ser también estadísticas y datos históricos de los cuales se apoya para dar respuesta a las solicitudes hechas por la sub-área de producción de software (BASILI, 1992).

Modelo Eureka

La idea del modelo Eureka consiste, en dividir el proyecto en pequeñas factorías durante el proceso de desarrollo, con el objetivo de que cada una se especialice en uno o varios componentes. La unión de los componentes elaborados conforma el producto final y se realiza de acuerdo a reglas establecidas por las personas involucradas en el ambiente de desarrollo. Dichas reglas constituyen patrones a seguir, algoritmos, métodos de desarrollo de software y sirven para evaluar los productos que se van obteniendo (ROCKWELL, 2003).

Modelo clasificatorio

El aporte fundamental del modelo clasificatorio es que ofrece una idea de cual debe ser el ciclo de vida de un proyecto, o parte de el. Para ello, propone clasificar la factoría de acuerdo al alcance o fases de desarrollo definidas en el proceso. Los flujos de trabajo para cada una de las clasificaciones de las factorías, se definen en dependencia de su alcance, permitiendo hacer una mejor planificación de tiempo y selección del personal necesario para el desarrollo del proyecto (FERNÁNDEZ, 2004).

Modelo basado en la norma ISO 9001 y CMMI

La característica más relevante de este modelo es que el proceso es guiado por el estándar de calidad CMMI y los requisitos de calidad para la organización de la factoría son definidos por la norma ISO 9001. En el modelo, CMMI constituye un marco de referencia para mejorar los procesos de desarrollo, adquisición, y mantenimiento de productos y servicios. La norma ISO 9001, la toma como estándar cuyos principios básicos de la gestión de la calidad le permiten mejorar la

marcha y funcionamiento de sus relaciones internas, lo cual contribuye a mejorar la calidad final del producto. (LI, 2001)

Modelo replicable

De manera general este modelo reúne las características más relevantes de los modelos mencionados anteriormente y es considerado sumamente adaptable; ya sea al entorno, necesidades o recursos del área donde se aplique. Ofrece un alto grado de detalle, describiendo las técnicas y herramientas a utilizar, definiendo el flujo de los procesos, las actividades, las responsabilidades y los roles a asignar aunque no los organiza estructuralmente. Tiene como desventajas que no se aplica a todo tipo de producto y no enmarca su proceso de producción en el uso de alguna de las metodologías estandarizadas y en cierta medida descuida un poco la gestión del proyecto (Marante, 2005).

Modelo de factoría de software aplicando inteligencia

Este modelo concibe la misma idea del Modelo Clasificatorio de clasificar las factorías de acuerdo a su alcance y al igual que Modelo basado en la norma ISO 9001 y CMMI, utiliza CMMI para el desarrollo y evaluación del proceso de y la gestión de proyectos.

Un aspecto sumamente importante a destacar en este modelo es que propone los principales elementos a definir para establecer un proceso de desarrollo en base a una metodología; lo que permite seleccionar una, acorde a la magnitud del proyecto y al personal con que cuenta el mismo.

Otro aspecto de interés en el modelo Aplicando Inteligencia es la definición de entidades que propone para la factoría. Dicha definición permite una mejor estructuración, organización y distribución del trabajo. Por ejemplo: define una entidad llamada “Bases Tecnológicas” que se encarga de controlar las técnicas, mecanismos y herramientas y las demandas tecnológicas para su uso. Esto alivia en gran medida la carga de trabajo del Grupo de Gestores, al no tener que controlar directamente dichas cuestiones, sino que se dedican específicamente a la gestión y organización del proceso, ya que estas últimas responsabilidades pertenecen a la entidad “Gestión de Proyecto”. Además se incorpora la entidad

“Inteligencia”, que tiene como principales responsabilidades la orientación estratégica y la gestión del conocimiento, liberando también de estas responsabilidades a la entidad de “Gestión de Proyecto”.

Un último aspecto a considerar en el modelo, es que propone un método para codificar los procesos, actividades y tareas, que tiene como objetivo, lograr un mayor entendimiento del flujo del proceso y de datos (Trujillo, 2007).

1.3.1. Consideraciones sobre los modelos de factoría de software

Todos los modelos de factoría de software analizados poseen características significativas a tener en cuenta a la hora de definir un modelo de desarrollo de software. No obstante se considera que el modelo “Aplicando Inteligencia” es el más integral y eficiente; ya que fue concebido a partir del análisis de sus antecesores, tomando las mejores prácticas de cada uno. Entre las ideas que adopta de sus predecesores, podemos mencionar las siguientes:

Al igual que el modelo clasificatorio sigue la idea de clasificar la factoría de acuerdo al alcance o fases de desarrollo definidas en el proceso, logrando así una mejor planificación de tiempo y selección del personal necesario para el desarrollo del proyecto.

Del modelo basado en la norma ISO 9001 y CMMI, adopta la idea de guiar el proceso por el estándar de calidad CMMI y definir los requisitos de calidad para la organización de la factoría en base a la norma ISO 9001.

También existe relación entre el modelo Aplicando Inteligencia y el modelo propuesto por Basili. Dicha afirmación se basa, en que una vez que la factoría haya ganado en madurez, se podría adoptar la idea del modelo propuesto por Basili y dividir el área de producción en dos sub-áreas: la de producción de software y la de producción de componentes. Donde la sub-área de producción de componentes, sería la encargada de desarrollar componentes reutilizables para abastecer a la sub-área de producción de software. Incluso a medida que aumente la madurez de la factoría, podría pensarse en un proyecto mucho más ambicioso, que consiste en elevar los niveles de variedad y producción de la sub-área de producción de componentes. De esta forma no solo podría abastecer a la sub-área

de producción de software, sino que también sería capaz de suministrar componentes reutilizables a clientes externos.

Además de reunir las características mencionadas anteriormente, el modelo "Aplicando Inteligencia", aporta una serie de ideas que favorecen el funcionamiento y rendimiento de la factoría. Entre ellas resulta relevante mencionar las siguientes:

- ✓ Exhibe una eficiente definición de sus entidades, permitiendo una mejor estructuración, organización y distribución del trabajo. Por ejemplo: define una entidad llamada "Bases Tecnológicas" que se encarga de controlar las técnicas, mecanismos y herramientas y las demandas tecnológicas para su uso. Esto alivia en gran medida la carga de trabajo del Grupo de Gestores, al no tener que controlar directamente dichas cuestiones, sino que se dedican específicamente a la gestión y organización del proceso, ya que estas últimas responsabilidades pertenecen a la entidad "Gestión de Proyecto". Por otra parte se incorpora la entidad "Inteligencia", que tiene como principales responsabilidades la orientación estratégica y la gestión del conocimiento, liberando también de estas responsabilidades a la entidad de "Gestión de Proyecto".
- ✓ Propone un método para codificar los procesos, actividades y tareas, que tiene como objetivo, lograr un mayor entendimiento del flujo del proceso y de datos.

1.4. Modelo funcional Aplicando Inteligencia

Según Trujillo (2007), el modelo "Aplicando Inteligencia" consta de 6 entidades y sus relaciones, las cuales se describen a continuación:

Gestión de Proyecto: *comprende todas las áreas de la gestión de proyecto. Presenta las unidades de gestión, organización del proceso, gestión del capital humano y gestión de la calidad.*

Proceso: *comprende el conjunto de actividades que conforman el flujo de trabajo, el cual depende de la metodología que se utilice para guiar el desarrollo del proyecto.*

Personas: *comprende el capital humano involucrado con el proceso de desarrollo de software, la estructura organizativa y los roles que ocupan, está dividida en dos unidades: Gestores de la Factoría y Grupo de desarrollo.*

Inteligencia: *comprende los métodos que permitan la orientación estratégica de la factoría con el uso de herramientas de Vigilancia Tecnológica, Inteligencia Empresarial, Prospectiva. Presenta dos unidades: la interna de inteligencia organizacional y la externa de inteligencia empresarial.*

Bases tecnológicas: *Comprende el contexto de las bases tecnológicas y herramientas, las técnicas y mecanismos para construir, soportar y gestionar el proceso de desarrollo.*

Repositorio de componentes: *comprende el almacenamiento y gestión de los activos del proceso y componentes de código. Entiéndase como activos del proceso formularios, documentos, patrones, algoritmos utilizados como artefactos en el proceso. Los activos del proceso también pueden ser denominados como componentes de infraestructura, componentes de valor en el proceso.*

En el Anexo 3 se muestra la arquitectura del modelo. Según Trujillo (2007)*, dicho modelo* [*se basa en que la entrada de un proyecto son los requerimientos y el resultado final es un producto, que toma forma durante su desarrollo gracias a la intervención de las personas representadas por la entidad* ***Personas****, utilizando PSP y TSP para la planificación personal y en equipo. El equipo de desarrollo lo forman las personas involucradas directamente en el proceso, el de gestores comprende el equipo de dirección de la misma, encargados del control y gestión del grupo de desarrollo. Los cuales son quienes ejecutan las actividades o flujos de trabajo, a su vez son guiados por el proceso de desarrollo de software, representado en el modelo mediante la entidad* ***Proceso*** *que utiliza el Modelo de Madurez de Capacidades Integrado (CMMI), modelo de calidad integrado para la industria del software que provee áreas y practicas importantes para el desarrollo y evaluación del proceso de desarrollo y la gestión de proyectos.*

El proceso es automatizado y soportado por diversas tecnologías y herramientas, técnicas y mecanismos representados en la entidad ***Bases tecnológicas****. La reutilización tiene efectos muy positivos en el desarrollo de software, entre estos*

efectos están el aumento en la productividad y calidad así como la reducción del tiempo de desarrollo, para dar soporte al proceso en este sentido la factoría cuenta con una base de componentes reutilizables, representada en la entidad ***Repositorio de Componentes****. Todo esto es gestionado desde la entidad* ***Gestión de Proyecto*** *la que tiene la responsabilidad de definir el proceso, gestionar la calidad, el capital humano y el proyecto. Esta entidad recibe la orientación estratégica de la entidad* ***Inteligencia,*** *que posee las unidades interna y externa]".*

Modelo de Madurez de Capacidades Integrado (CMMI)

El CMMI es un modelo de calidad resultado de la evolución de tres modelos: El Modelo de Capacidad de Madurez para Software (*Capability Maturity Model for Software-CMM SW,* en inglés), el Modelo de Capacidad de Ingeniería de Sistema (*Systems Engineering Capability Model,* en inglés, abreviado SECM) y el Modelo de Capacidad de Madurez Integrada en el Desarrollo de Producto (*Integrated Product Development Capability Maturity Model*, en inglés, abreviado IPD CMM).

CMMI fue concebido para determinar y mejorar la capacidad de los procesos de las organizaciones, al punto de que desarrollen productos de calidad de manera consistente y predecible (TEAM, 2006).

Este modelo establece un conjunto de prácticas o procesos clave agrupados en Áreas Clave de Proceso, donde cada área de proceso presenta un conjunto de buenas prácticas que ayudan a establecer un proceso de mejora continua en indicadores como calidad, productividad, planificación, costo y satisfacción de los clientes.

El proceso personal de software (PSP)

PSP es un proceso de auto mejoramiento diseñado para ayudar a controlar, administrar y mejorar la forma en que se trabaja individualmente. Está estructurado por formularios, guías y procedimientos para desarrollar software. Si es usado debidamente, es capaz de brindar datos históricos que permiten lograr que numerosos elementos del trabajo sean más predecibles y eficientes.

PSP muestra cómo aplicar métodos avanzados de ingeniería a las tareas diarias de cada individuo, proporciona métodos detallados de planificación y estimación,

muestra a los implicados en el proceso como controlar su rendimiento y explica como los procesos definidos guían su trabajo (INSTITUTE, 2007).

Proceso de Software en Equipos (TSP)

TSP es un conjunto de procesos estructurados que indican qué hacer en cada fase del desarrollo del proyecto. Muestra cómo conectar cada fase para construir un producto completo y cómo aplicar prácticas de Ingeniería de Software conocidas, en un ambiente de trabajo en equipo (HUMPHREY, 2003).

TSP persigue los siguientes objetivos:

- ✓ Integrar equipos independientes de alto rendimiento que planeen y registren su trabajo, establezcan metas, y sean dueños de sus procesos y planes
- ✓ Mostrar a los gerentes como monitorear y motivar a sus equipos de trabajo y como ayudarlos a alcanzar su máxima productividad
- ✓ Acelerar la mejora continua de procesos
- ✓ Proveer de una guía para el mejoramiento en organizaciones maduras.

Una vez expuestas las entidades del modelo Aplicando Inteligencia, sus interrelaciones y las técnicas y mecanismos que utilizan, es posible enfocar la investigación en la entidad “Proceso” de dicho modelo. Dicha entidad, constituye el eje de la presente, ya que comprende el conjunto de actividades que conforman el proceso de desarrollo de software.

Proceso de desarrollo del software

Existen varias definiciones de proceso de desarrollo de software, pues son muchos los autores que han estudiado el tema y han emitido su criterio; pero de manera general podemos definir un proceso de desarrollo de software como el conjunto de actividades necesarias para convertir los requerimientos del usuario en un grupo de artefactos que conforman el producto de software. (PRESSMAN, 2005)

Específicamente los procesos de desarrollo utilizados para sistemas Web, de una forma u otra, centran su atención en producir aplicaciones donde el usuario pueda aprovechar el potencial del paradigma de la navegación Web, mientras realiza acciones sobre bases de datos. Esta tarea no resuta sencilla ya que para lograr

una estructura de navegación robusta se debe logar que el usuario entienda dónde puede ir y cómo llegar al lugar deseado. También resulta complejo construir la interfaz de una aplicación Web, ya que no sólo se necesita especificar cuáles son los objetos de la interfaz que deberían ser implementados, sino también la manera en la cual estos objetos interactuarán con el resto de la aplicación. En este sentido existen dos principios que garantizan un buen desarrollo de un sistema Web. Por un lado, la navegación y el comportamiento funcional de la aplicación deben integrarse. Por otro lado, durante el proceso de diseño se deben separar las decisiones relacionadas con la estructura navegacional de la aplicación, de aquellas relacionadas con el dominio de la aplicación.

A continuación se realiza un análisis crítico de un conjunto de metodologías de desarrollo hipermedia que suelen ser utilizadas para el desarrollo de sistemas Web, con el objetivo de valorar si alguna resulta adecuada para ser utilizada durante el proceso de desarrollo de portales en la fábrica de portales Web de la Facultad #10 de la UCI. Para ello se tendrán en cuenta dos aspectos fundamentalmente: la estructura de navegación y la interfaz de la aplicación.

- ✓ Estructura de navegación: si el usuario entiende dónde puede ir y cómo llegar al lugar deseado, es una buena señal de que la aplicación ha sido bien diseñada.
- ✓ Interfaz de la aplicación: no sólo se necesita especificar cuáles son los objetos de la interfaz que deberían ser implementados, sino también la manera en la cual estos objetos interactuarán con el resto de la aplicación.

1.5. Metodologías utilizadas para el desarrollo de sistemas Web

El objetivo de las metodologías Web es producir aplicaciones en las cuales el usuario pueda aprovechar el potencial del paradigma de la navegación de sitios Web, mientras ejecuta transacciones sobre bases de información. Para lograr este objetivo se deben tener en cuenta fundamentalmente la estructura de navegación y la interfaz de la aplicación.

- ✓ Estructura de navegación: si el usuario entiende dónde puede ir y cómo llegar al lugar deseado, es una buena señal de que la aplicación ha sido

bien diseñada.

- ✓ Interfaz de la aplicación: no sólo se necesita especificar cuáles son los objetos de la interfaz que deberían ser implementados, sino también la manera en la cual estos objetos interactuarán con el resto de la aplicación.

Existen varias metodologías para el desarrollo de aplicaciones Web, a continuación se hace referencia a algunas de ellas:

***A Model-based Approach to Hypertext Application Design* (HDM)**

HDM fue el primer modelo para el desarrollo de Web. Este modelo propone el uso de perspectivas, entidades, componentes, unidades y enlaces como elementos que ayudan a conformar la aplicación (BRAMBILLA, 2007), donde:

- ✓ Las entidades son vistas como una jerarquía de componentes, en la que dichos componentes heredan las propiedades de la entidad.
- ✓ Las perspectivas son el mecanismo para representar la multiplicidad de las presentaciones de un mismo contenido de información.
- ✓ Una unidad constituye un fragmento del contenido de una entidad expuesta bajo una perspectiva en particular.
- ✓ Un componente representa una abstracción que se utiliza para diseñar todas las unidades que constituyen las perspectivas de un mismo contenido de información.

Para este modelo, los enlaces tienen un doble rol; el de representar la semántica del dominio y el de permitir la navegación. Es por ello que HDM define tres tipos de enlaces:

1. Enlaces estructurales entre componentes de una misma entidad.
2. Enlaces de perspectiva entre unidades de un mismo componente.
3. Enlaces de aplicación entre entidades o componentes.

Al utilizar la clasificación propuesta por este modelo para los enlaces, se derivan las siguientes ventajas:

- ✓ Utilización más consistente de los enlaces.

- ✓ Esquemas de navegación definidos de antemano.
- ✓ Se obtiene una semántica de navegación por defecto.
- ✓ La puesta en marcha, desde la fase de desarrollo, de mecanismos como la derivación de enlaces.

HDM es solo un modelo, no obstante, los elementos que propone, tales como entidades, componentes, perspectivas, unidades y enlaces; constituyen un lenguaje de especificación que brinda la posibilidad de obtener un modelo de dominio enriquecido, a partir del cual se puede definir con mayor facilidad y eficiencia la estructura navegacional de la aplicación.

Relationship Management Methodology **(RMM)**

La metodología RMM se basa en el modelo HDM, utilizando el modelo *Entity-Relationship (E-R)* y partiendo de ellos define un nuevo modelo, el modelo *Relationship Management Data Model* (RMDM), que propone un lenguaje que permite describir los objetos del dominio de aplicación, sus interrelaciones y los elementos de navegación hipermedia de la aplicación. Sin embargo, el método no permite al diseñador definir elementos hipermedia propios que tengan capacidades específicas ya que impone precisiones sobre todas las etapas a realizar, desde la concepción del esquema del dominio hasta el desarrollo de la aplicación hipermedia (*ESCALONA*, 2007). A diferencia de HDM este modelo si es considerado una metodología pues propone un ciclo de vida completo para el desarrollo de aplicaciones. Este ciclo está compuesto por siete fases:

1. Realizar el modelo E-R: el proceso de desarrollo da inicio al comenzar la elaboración del modelo E-R, obviando los detalles de navegación e interfaz.
2. Realizar el modelo de *slices*: realiza el diseño de *slices*, donde un *slice* puede ser considerado como una vista del sistema.
3. Diseñar la navegación: define el sistema de navegación, el cual a su vez enriquece el diseño de *slices* y con ambos se genera el modelo RMDM.
4. Definir el protocolo de conversión: precisa el protocolo de conversión que va a describir el proceso a seguir para pasar del modelo RMDM a la plataforma de desarrollo concreta.

5. Diseñar la interfaz: realiza el diseño de la interfaz en el que se define la estructura abstracta que tendrá la interfaz del sistema, sin tener en cuenta el diseño gráfico, dígase colores, tipos de letra, etc.
6. Implementar la aplicación: construye la aplicación en el lenguaje específico previamente seleccionado.
7. Probar la aplicación: realiza las pruebas necesarias para comprobar que la aplicación cumple con los requerimientos deseados.

***Object Oriented Hypermedia Design Methodology* (OOHDM)**

OOHDM es una metodología que sigue la filosofía orientada a objetos y se basa en gran medida en el modelo HDM, sobre todo en los aspectos de navegación (GARY WILLS, 2009). No obstante OOHDM supera en gran medida a HDM, ya que no es simplemente un lenguaje de modelado, sino que define pautas de trabajo, apoyándose principalmente en el diseño, para desarrollar aplicaciones de forma metodológica. Esta metodología propone modelar lo conceptual, lo navegacional y la interfaz abstracta de manera independiente, donde la separación del diseño navegacional y de la interfaz de usuario permite dividir las tareas del desarrollo, así como tener diferentes interfaces para un mismo modelo navegacional. OOHDM consta de cuatro fases:

1. Diseño conceptual: especifica el dominio de aplicación mediante un esquema conceptual en términos de clases y sus relaciones. Para ello, el método propone el modelo objeto de OMT.
2. Diseño de navegación: define las clases navegacionales tales como nodos, enlaces e índices, donde los enlaces derivan de relaciones y los nodos representan las vistas sobre las clases conceptuales. Posteriormente se reorganiza la información del modelo estructural y se describe la estructura navegacional en términos de contextos navegacionales, o sea establece los caminos de acceso a la información y sus permisos de visibilidad.
3. Diseño de interfaz abstracta: describe los objetos de interfaz y los asocia con objetos de navegación. En OOHDM, se utilizan las Vistas de Datos Abstractas (ADVs), para describir la interfaz de usuario. Las ADVs son objetos que tienen un estado y una interfaz, donde la interfaz puede ser

ejercida a través de mensajes (en particular, eventos externos generados por el usuario).

4. Aplicación: construcción del software a partir de los artefactos generados en las etapas previas.

***Web Site Design Modeling* (WSDM)**

Es una metodología totalmente orientada a diseñar la aplicación en base a los grupos de usuarios que visitaran el sitio. Según estas futuras visitas, y la forma en que estos usuarios recorrerán el sitio, se establecen los parámetros de diseño (O.M.F. DE TROYER, 2005). Sin embargo, no comenta nada sobre aspectos que pueden resultar importantes partiendo de esta idea. Por ejemplo, puede ser que la misma información se presente a dos usuarios de forma diferente y en muchos casos puede resultar complejo para el programador detectar que se trata de la misma información. Además no trabaja aspectos como la funcionalidad y la seguridad. WSDM propone un proceso de desarrollo compuesto por cuatro fases:

1. Modelo de usuario: de las cuatro fases el modelo de usuario la más importante, ya que es donde se definen los perfiles de usuarios para los cuales se construye la aplicación. El resto de las fases se desarrollan en base a la documentación sobre los usuarios que se realiza en esta primera etapa. En esta fase se realiza la clasificación de usuarios y la descripción de los grupos de usuarios.

Clasificación de usuarios: se deben identificar y clasificar los usuarios que van a hacer uso del sistema. Para ello, WSDM propone que se describan las actividades que desarrollan los usuarios y las relaciones entre ellos, tomando como base el entorno de la organización donde se vaya a implantar el sistema y los procesos a automatizar. Para la representación gráfica de estas relaciones WSDM propone una especie de mapas de conceptos, de roles y actividades.

Descripción de los grupos de usuarios: se describen más detalladamente los grupos de usuarios definidos en la etapa de clasificación de usuarios. Para ello, WSDM propone que se redacte un documento en el que definan para cada grupo de usuarios los requisitos funcionales, de almacenamiento de información y seguridad.

2. Diseño conceptual: en esta fase se realiza el modelado orientado a objetos y el diseño navegacional.

Modelado orientado a objetos: cada uno de los usuarios definidos en la fase anterior tienen su propia vista de la información y la usabilidad, donde dicha vista es el modelo de objetos del usuario. Este modelo no incluye la totalidad de las clases que modelan una organización, sino sólo las más representativas para cada usuario.

Diseño navegacional: por cada vista de usuario definida en el modelado orientado a objetos, se define un modelo navegacional. Estos modelos se caracterizan por presentar una estructura jerárquica, donde el nodo raíz de un modelo navegacional está determinado por los tipos de usuarios definidos. De esta forma la información que necesita cada tipo de usuario, es accesible desde el nodo raíz.

3. Diseño de la implementación: especifica los requisitos y restricciones del diseño gráfico del sitio, según lo definido en el diseño conceptual.
4. Implementación: selecciona el ambiente de desarrollo y se construye la aplicación.

UML-Based Methodology for Hypermedia Design

Es una metodología basada en otras que le anteceden, por lo que se le considera bastante elaborada. Contiene bien definidas y detalladas las secuencias de pasos y guías para obtener los modelos que se van generando; los cuales son basados totalmente en UML (NORA KOCH, 2006). Sin embargo, no indica cómo conseguir que los casos de uso capturen los conceptos necesarios para definir la navegación o la interfaz. Su proceso de desarrollo consta de cuatro fases:

1. Desarrollo de los casos de uso y el modelo conceptual: define el modelo conceptual del sistema, partiendo de una definición de casos de uso, para la cual se sigue el mismo proceso que propone UML y el Proceso Unificado. El modelo conceptual a obtener contiene clases, con sus atributos, las operaciones más relevantes, asociaciones, herencia, dependencias, las interfaces y restricciones del sistema.

2. Modelo de espacio de la navegación: obtiene como resultado un modelo de clases navegacionales, en el cual están presentes dichas clases y las asociaciones entre ellas, que representan la posibilidad de navegación entre las mismas.
3. Modelo de estructura de navegación: define la estructura de navegación representada por un grafo dirigido en el que los nodos son índices, rutas, selecciones, o menús.
4. Construcción del modelo de presentación: se define la estructura abstracta que tendrá la interfaz del sistema.

***Scenario-based object-oriented Hypermedia design Methodology* (SOHDM)**

Esta metodología tiene prevista una nomenclatura para representar los posibles elementos que se pueden encontrar en una página: botones, imágenes, etc. Por una parte resulta beneficioso ya que constituye una vía para estandarizar el trabajo de los desarrolladores (AROUNA WOUKEU, 2007), sin embargo en ocasiones puede ser una nomenclatura muy cerrada. El proceso de desarrollo, que propone SOHDM consta de seis fases:

1. Análisis del dominio: se estudian las necesidades de la aplicación, el entorno de trabajo y los actores. La finalidad principal de esta fase es definir los requisitos del sistema a partir de la realización de un diagrama de contexto tal y como se propone en los diagramas de flujos de datos (DFD) de Yourdon (1989). En este diagrama se identifican las entidades externas que se comunican con el sistema, así como los eventos que provocan esa comunicación. Con la lista de eventos se conforma una tabla que debe indicar en qué eventos puede participar cada entidad. Por cada evento diferente, SOHDM propone elaborar un escenario, los cuales son representados gráficamente mediante los denominados *Scenario Activity Chart* (SACs). Cada escenario describe el proceso de interacción entre el usuario y el sistema cuando se produce un evento determinado; especificando el flujo de actividades, los objetos involucrados y las transacciones realizadas.

2. Modelado orientado a objetos: los escenarios representados mediante SACs son usados para modelar los objetos del sistema, transformándolos según la propuesta de los *Class Responsability Collaboration* (CRC) *cards*, tal y como se expone en la propuesta *"Software design using CRC cards"* (HALBLEIB, 1999).
3. Diseño de las vistas: elimina las clases, atributos y relaciones que no son visibles y se crean las unidades navegacionales.
4. Diseño navegacional: define los enlaces o hiperenlaces entre las diferentes vistas.
5. Diseño de la implementación: crea los esbozos de las páginas de la aplicación, para luego realizar el diseño de la interfaz de usuario.
6. Construcción: se desarrolla la aplicación en función de los artefactos generados en las fases anteriores.

Hypermedia Flexible Process Modeling Strategy **(HFPM)**

HFPM es una metodología muy completa en el sentido que comprende todas las fases del proceso de desarrollo, para un total de trece fases, en cada una de las cuales ofrece las pautas a seguir (AROUNA WOUKEU, 2007). Sin embargo, no ofrece una guía detallada, o sea, indica el qué se debe hacer, pero no el cómo se debe hacer.

1. Modelado de los requisitos del software: obtiene los requisitos funcionales y no funcionales a partir de una definición de casos de uso. A diferencia de SOHDM, HFPM no impone ninguna norma a seguir para realizar estas tareas, sino que deja que la elección corra por parte de los desarrolladores.
2. Planificación: en esta fase se diseña el plan de trabajo con el objetivo distribuir con eficiencia los recursos del proyecto para cumplir los requisitos del sistema.
3. Modelado conceptual: se define un modelo de clases que representa al sistema sin entrar en aspectos de hipermedia. Para ello se realiza un análisis del dominio del problema y se define un diagrama de clases que modele el sistema.

4. Modelado navegacional: se realiza el modelo navegacional, aplicando el método propuesto en OOHDM.
5. Modelado de la interfaz abstracta: se define la estructura abstracta que tendrá la interfaz del sistema.
6. Diseño del entorno: tiene dos objetivos, por una parte definir la arquitectura del sistema y por otra, mejorar los modelos obtenidos mediante el uso de patrones de diseño.
7. Captura y edición de los elementos multimedia: se plantean los medios que debe contener la aplicación, así como los sistemas de almacenamiento que se emplearán en los mismos.
8. Implementación: se desarrolla la aplicación en función de los artefactos generados en las fases anteriores.
9. Verificación y validación: se analiza si la aplicación construida satisface los requisitos definidos en la primera fase.
10. Evaluación del entorno: se evalúa si el resultado es aplicable al entorno donde se debe establecer.
11. Evaluación de la calidad: se determina la calidad del producto final.
12. Mantenimiento: comprende el mantenimiento correctivo, aumentativo y adaptativo del sistema.
13. Documentación: se genera la documentación del sistema, la cual contiene la información de cada uno de los modelos generados en cada fase, los resultados de las pruebas y las valoraciones realizadas en las fases 9, 10 y 11 y por último el manual de usuario.

1.5.1. Consideraciones sobre las metodologías utilizadas para el desarrollo de sistemas Web

Todas las metodologías mencionadas anteriormente presentan elementos a considerar.

De una forma u otra, cada metodología posee un flujo de trabajo, el cual está constituido por determinadas fases propias; estas fases se llevan a cabo a través de actividades, y cada una de estas actividades genera uno o varios artefactos.

En las propuestas analizadas es posible destacar los siguientes aspectos:

1. Son comunes las actividades para modelar lo conceptual, lo navegacional y las interfaces abstractas, aunque cada una lo hace de manera particular. Por ejemplo: para el caso particular de OOHDM, lo más relevante es que propone modelar estos tres aspectos de forma independiente, mientras que WSDM propone diseñar la aplicación en general en base a los grupos de usuarios que visitaran el sitio; y por otro lado, SOHDM propone el uso de escenarios para obtener el modelo conceptual del sistema.
2. Una gran parte de estas metodologías indican el qué se debe hacer, pero no el cómo se debe hacer. Por ejemplo: UML-Based Methodology for Hypermedia Design, no indica cómo conseguir que los casos de uso capturen los conceptos necesarios para definir la navegación o la interfaz.
3. Algunas como WSDM no trabaja aspectos como la funcionalidad, ni la seguridad; centrándose solo en el contenido del sitio y el cómo presentar este contenido al usuario.
4. La gran mayoría son muy superficiales en cuanto a las fases de implementación, prueba y despliegue; algunas ni siquiera contemplan estas fases.
5. No existe un estándar entre ellas para la clasificación de los requisitos del sistema.
6. Muy pocas se basan en la reutilización de componentes por lo que no contemplan ninguna actividad encaminada a la reutilización de componentes para el desarrollo de aplicaciones.

Es posible afirmar entonces que ninguna de las metodologías presentadas define de forma satisfactoria las etapas que componen un proceso completo y genérico de cualquier aplicación hipermedia; por el contrario, cada metodología propone su propio método para modelar los conceptos del dominio de la aplicación y los del dominio hipermedia.

Capítulo 2. Caracterización del entorno y métodos utilizados

En el presente capítulo se describe la Universidad de las Ciencias Informáticas, los métodos, procedimientos y técnicas utilizadas para llevar acabo la investigación.

2.1. Universidad de las Ciencias Informáticas (UCI)

La Universidad de las Ciencias Informáticas (UCI) es la primera universidad surgida en la Batalla de Ideas, sobre la base del nuevo concepto de universidad productiva. Sus principales objetivos son:

- ✓ Formar ingenieros en Ciencias Informáticas comprometidos con la patria.
- ✓ Producir software y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación.

En la UCI los estudiantes y profesores se deben vincular a la producción participando en proyectos de alto valor tanto para el mercado nacional, como internacional, sus resultados contribuyen al desarrollo económico, político y social del país y se plasma la concepción de que la docencia se realice desde la producción.

La estructura docente de la UCI, está compuesta por 10 facultades, donde reciben docencia más de 10 mil estudiantes de todas las provincias del país. Las facultades se vinculan a proyectos productivos y se especializan en segundos perfiles asociados a determinadas líneas de producción.

Como medio para organizar la producción existe una Infraestructura Productiva (IP) que dirige metodológicamente los proyectos, la cual se subordina a la vice-rectoría primera. La IP presenta un conjunto de direcciones vinculadas a diferentes líneas de producción y áreas para la gestión de la producción.

Aunque la fuerza de trabajo fundamentalmente son los profesores y estudiantes de la UCI, también participan especialistas de empresas, Centros de Investigación, y otras universidades del país.

Entre las áreas productivas que existen en la UCI, se encuentra la fábrica de portales de la Facultad #10. La misma tiene como objetivo desarrollar productos de software enmarcados en el ámbito de los sistemas Web. Posee una plantilla de 65 personas de ellos 52 son estudiantes y 13 profesores; distribuidos en distintos proyectos. Entre los recursos materiales con los que cuenta, se destacan 30 estaciones de trabajo, cuyo horario de trabajo oscila entre las 8:00am y las 2:00am, para un total de 18 horas de trabajo al día.

2.2. Población, unidad de estudio y decisión muestral

Población

La población está compuesta por todos los proyectos desarrollados en la fábrica de portales en el período comprendido entre el mes de octubre de 2008 y diciembre de 2009, para un total de 7.

Muestra

La muestra se compone por 7 proyectos, representando el 100% de la población.

2.3. Estrategia de investigación

La estrategia es la base para la planificación y organización de la investigación, por lo que se hace necesario seleccionar la misma de acuerdo a las condiciones específicas predominantes, atendiendo a la trayectoria del problema y el conocimiento acumulado sobre el mismo, así como a los fines propuestos (LEÓN, 2002).

Para llevar a cabo la presente investigación se toma como estrategia a seguir la investigación descriptiva; haciendo énfasis en reflejar lo esencial y más significativo del fenómeno en cuestión, mediante una profunda investigación teórica del planteamiento investigativo.

2.4. Métodos, procedimientos y técnicas utilizados

El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones (LEÓN, 2002).

Métodos teóricos

La reproducción teórica de un objeto en el pensamiento significa comprenderlo en su desarrollo, en su historia y en su lógica. (LEÓN, 2002)

Para la realización de esta investigación se emplean métodos teóricos tales como: el histórico y el sistémico.

Método histórico: se utilizó para realizar un análisis de las tendencias y evolución de los modelos de producción, factorías y procesos de desarrollo de software, valorando su impacto en las TIC y en la producción.

Método sistémico: se analiza el proceso de desarrollo de software, definiendo para el proceso en especifico su descripción, las diferentes etapas, definiendo para cada una de ellas los artefactos de entrada, salida, así como las actividades a realizar y el rol que la ejecuta.

Métodos empíricos

Los métodos empíricos permiten describir y explicar las características fenomenológicas del objeto en base a la experiencia. (LEÓN, 2002)

Dentro de los métodos empíricos empleados, están la entrevista y la encuesta. Estos fueron vitales para el diagnóstico de la organización, permitiendo avalar los conceptos que se manejan en la investigación, medir el alcance y la importancia que tiene la temática, captar la información cualitativa y cuantitativa del fenómeno y conocer los criterios sobre la forma en que se organiza y se lleva a cabo la producción de portales. Como vía para obtener toda esta información se entrevistaron y encuestaron personas involucradas en la producción de portales en la fábrica de la Facultad #10.

Encuesta: se utiliza un cuestionario previamente elaborado, el cual permite obtener toda la información necesaria para la caracterización de los proyectos de portales en cuanto al proceso de desarrollo; permitiendo medir los conceptos que se evalúan a través de preguntas elaboradas de la forma más concreta posible. La muestra fue seleccionada por el método no probabilístico intencional, para poder obtener la mayor representatividad e información posible, de acuerdo con los intereses de la investigación que fue entrevistar a los líderes de proyecto y desarrolladores.

Se combinaron varios tipos de preguntas. La mayoría fueron semicerradas pues se tiene el interés de conocer la información cuantitativa pero también de saber la opinión del tema, así como involucrar y motivar a los encuestados en la solución. Se utilizaron además preguntas cerradas, directas e indirectas y de control.

Encuesta #1

Se realizó con el objetivo de identificar potencialidades y deficiencias en el proceso productivo de la fábrica de portales. En el Anexo 5 se puede ver el modelo de la encuesta aplicada.

Entrevista individual - no estructurada: se basa en los conocimientos y experiencias personales de los entrevistados. Para la entrevista se planifica una conversación entre el investigador y el entrevistado, obteniendo parte de la información que se desea. Para la confección de la entrevista, primeramente se analizan las características del entrevistado y su posición o conocimientos sobre el tema a tratar. Se confeccionan una serie de preguntas que sirven de guía y que permitieran complementar el proceso de obtención de información.

La selección se realizó con la técnica de muestreo no probabilística, muestreo intencional para poder obtener la mayor representatividad e información posible, de acuerdo con los intereses de la investigación que fue entrevistar personas que tienen experiencia en la producción y que han tenido que enfrentar problemáticas que le han permitido ir madurando.

Entrevista # 1

Se realiza con el objetivo de evaluar el nivel de conocimiento que existe referente al tema de modelos de producción de software, específicamente el modelo factorías de software, basándose principalmente en los beneficios directos de la aplicación de este enfoque en la producción. Para esta investigación la entrevista fue dirigida específicamente a los líderes de proyecto, ya que los mismos son los que poseen una visión completa de todo el proceso de desarrollo, perteneciente al mismo. En el Anexo 4 se puede ver el modelo de la entrevista aplicada.

2.5. Análisis de los resultados de la aplicación de las encuestas y entrevistas.

Se encuestaron 31 programadores, 10 líderes de proyecto, 14 analistas y un planificador, ya que solo existe uno en el proyecto, para un total de 56 personas, siendo un 86 % del total de integrantes de la fábrica de portales de la Facultad #10. En la figura 2.1 se muestra el grafico de pastel que representa la composición de los encuestados.

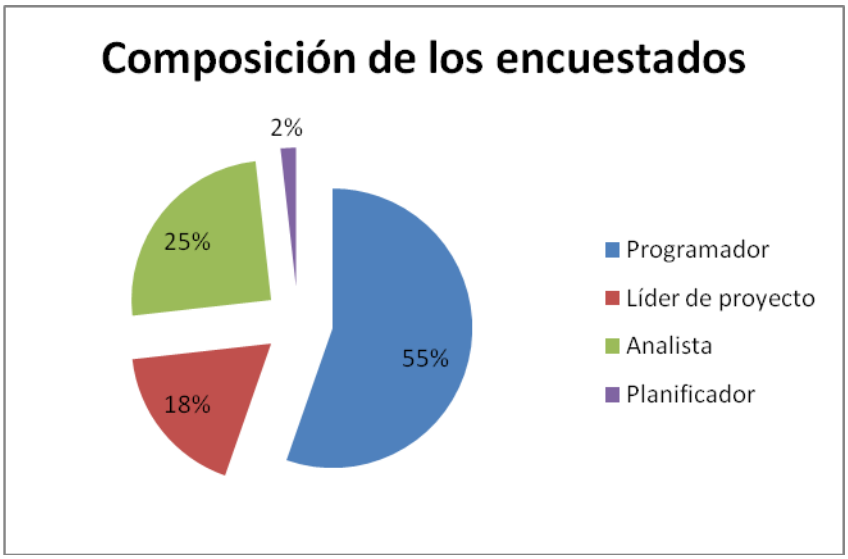


Figura 2.1 Composición de los encuestados

Resultados de la encuesta realizada

El 46% no conoce las responsabilidades de su rol como miembro de su proyecto, por lo que se observa un alto grado de desinformación y baja capacitación de los mismos; lo cual afecta la productividad y las competencias de los recursos humanos del proyecto.

Los resultados arrojaron que en la fábrica de portales solo existe un planificador y es esa la razón de que solo aparezca un planificador entre los encuestados. Otro dato que se obtuvo fue que ninguno de los desarrolladores aplican PSP, ni ninguna otra técnica de planificación individual. Tampoco los líderes de proyecto llevan una planificación a nivel de equipo, lo cual repercute negativamente en la planificación del mismo.

El 43% desconoce una parte representativa de las bases tecnológicas establecidas por el proyecto, lo cual trae como consecuencia que en muchos casos se trabaje con herramientas privativas, u otras poco eficientes.

La reutilización de componentes es mínima, solo el 27% de los encuestados alega que siempre reutiliza componentes durante el desarrollo de los portales, mientras que el 73 % opina que solo a veces y en ningún caso existe la opinión de que no existe reutilización (Ver figura 2.2). La causa principal es que no existe un repositorio en el cual se puedan almacenar y organizar dichos componentes, para su posterior utilización. Por tanto es evidente que se puede trabajar mucho más en base a la reutilización de componentes, incentivando a la creación de los mismos y su almacenamiento y organización dentro de un repositorio de forma tal que facilite su almacenamiento, búsqueda y obtención.

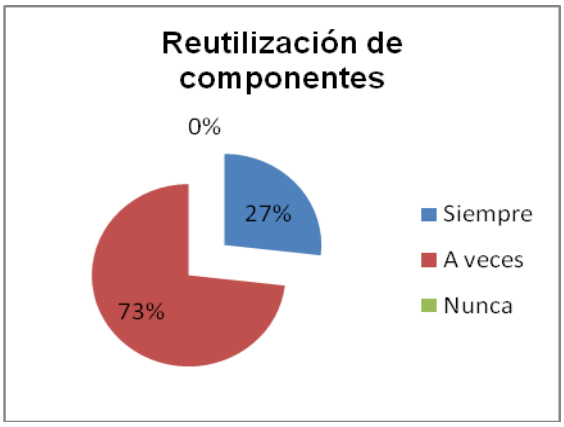


Figura 2.2 Reutilización de componentes

El 66% no conoce los estándares de la Ingeniería de Software definidos por la infraestructura productiva de la universidad y esto provoca que existan dificultades para conformar el expediente de proyecto y que no exista uniformidad entre la documentación de los proyectos.
Como metodología de desarrollo de software, la fábrica de portales utiliza RUP y el 59% plantea que gran parte de la documentación que genera RUP no resulta representativa para desarrollar los portales, lo cual atenta contra el tiempo de desarrollo y por consiguiente los costos del proyecto.

Resultados de la entrevista realizada

Se entrevistó según lo planificado a un total de 5 personas, todos líderes de proyecto. La entrevista arrojó que se tiene conciencia de la situación de la producción y que hay gran interés para la búsqueda de soluciones. Consideran que se puede trabajar mucho más en la planificación y la reutilización de componentes.

Luego de explicarles las características del enfoque de factoría, todos los entrevistados consideran que la implantación de una factoría de software puede mejorar la producción en gran medida. Entre las razones por la cuales alegan que están de acuerdo con la implantación de este modelo, se encuentran: la creación de un repositorio de componentes, y las posibilidades que brinda la factoría para elevar el grado de especialización en el personal de trabajo.

A partir de los resultados se realizó un diagrama Causa-Efecto o de Ishikawa, el cual se muestra en la siguiente figura:

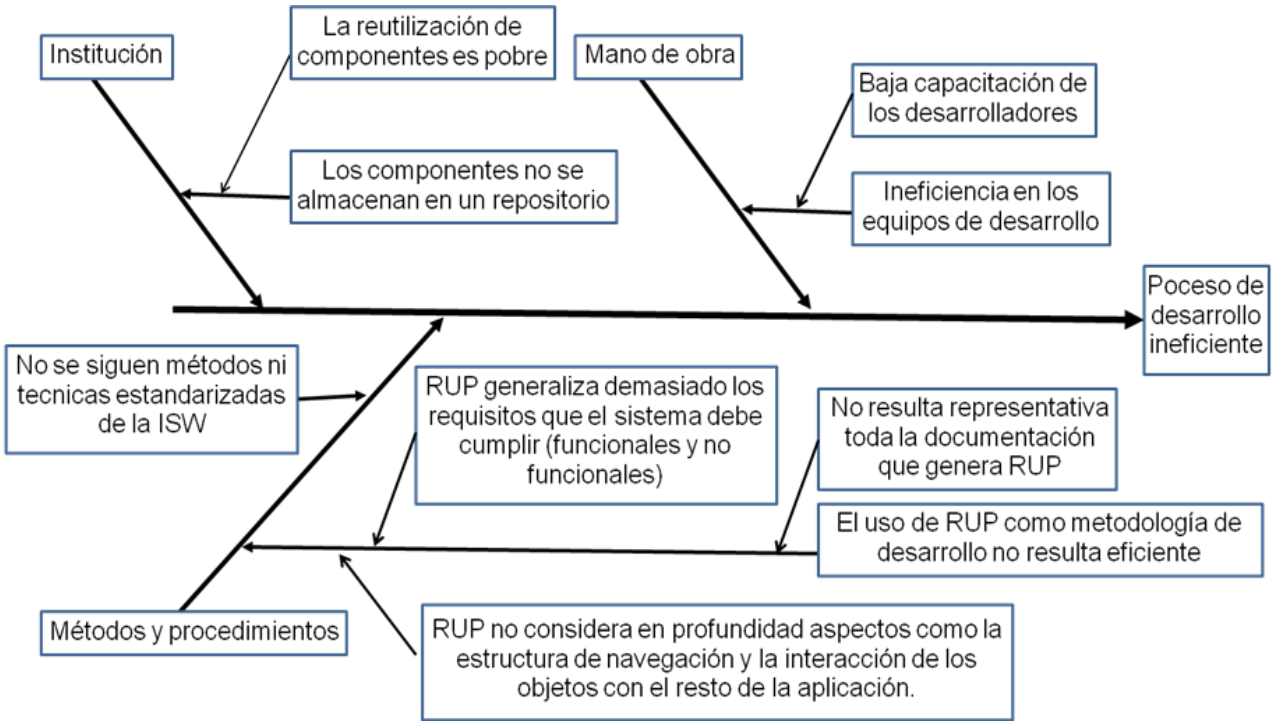


Figura 2.3 Diagrama Causa-Efecto o de Ishikawa

Capítulo 3. Entidad Proceso del Modelo de Factoría Aplicando Inteligencia

Tomando como base al análisis realizado en los capítulos anteriores referente a los modelos de producción, modelos de factoría y metodologías de desarrollo Web; en el presente capítulo se define la Entidad Proceso del Modelo de Factoría Aplicando Inteligencia.

3.1. Propuesta de Proceso de Desarrollo para Portales Web

Para definir un proceso dentro de un modelo de factoría de software, se debe tener en cuenta que el mismo debe ser repetible y mejorable continuamente, adaptable al contexto del proyecto y acorde al alcance del mismo. Estas y otras características están presentes en la presente propuesta, la cual constituye la entidad **proceso** del Modelo de Factoría de Software Aplicando Inteligencia.

Se decidió utilizar como modelo de proceso de software para la propuesta, el modelo basado en componentes, debido a que permite basar el proceso de desarrollo para portales en la creación y reutilización de componentes y el empleo de una biblioteca o repositorio donde estos queden almacenados de forma organizada, facilitando así la accesibilidad a los mismos. Según Pressman (2005), *"El modelo de proceso se selecciona según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse y los controles y entregas que se requieren".*

La arquitectura del proceso de desarrollo propuesto se muestra en la figura 3.1. El mismo, consta de cinco fases para conformar el modelo funcional, las cuales se describen a continuación:

Fase de análisis de requerimientos: comprende dos actividades principales; el levantamiento de la información y la captura de los requisitos que el sistema debe cumplir para su correcto funcionamiento.

Fase de diseño del sistema: en esta fase se modela el sistema a través de casos de uso.

Fase de diseño detallado: en ella se define la arquitectura de software, la navegación y el diseño gráfico del portal.

Fase de implementación: en ella se lleva a cabo una cualificación de los componentes que se encuentran en el repositorio, con vistas a realizar una selección candidata de los componentes más indicados para ser utilizados en la implementación del portal. Una vez concluida la cualificación se procede a la construcción del portal a partir de dichos componentes y los artefactos generados en las fases previas.

Para probar el sistema se propone más que una fase un proceso de prueba dinámico basado en el Modelo V (ver Anexo 8). La esencia de dicho modelo consiste en diseñar las pruebas al software desde el principio del proyecto, o sea bien antes de la codificación (Tolba, 2009). Como beneficio se logra un considerable ahorro del tiempo de desarrollo del proyecto y por ende una disminución de los costos.

Fase de despliegue e instalación: es la última fase del proceso de desarrollo y consiste en hacer que el producto de software esté disponible para el usuario final. Para ello, se desarrollan los materiales necesarios, dígase materiales de formación para el usuario, el material gráfico del producto y los artefactos de instalación. Una vez desarrollados los materiales, se realiza la instalación y evaluación del portal y posteriormente se imparten cursos de administración y uso del mismo.

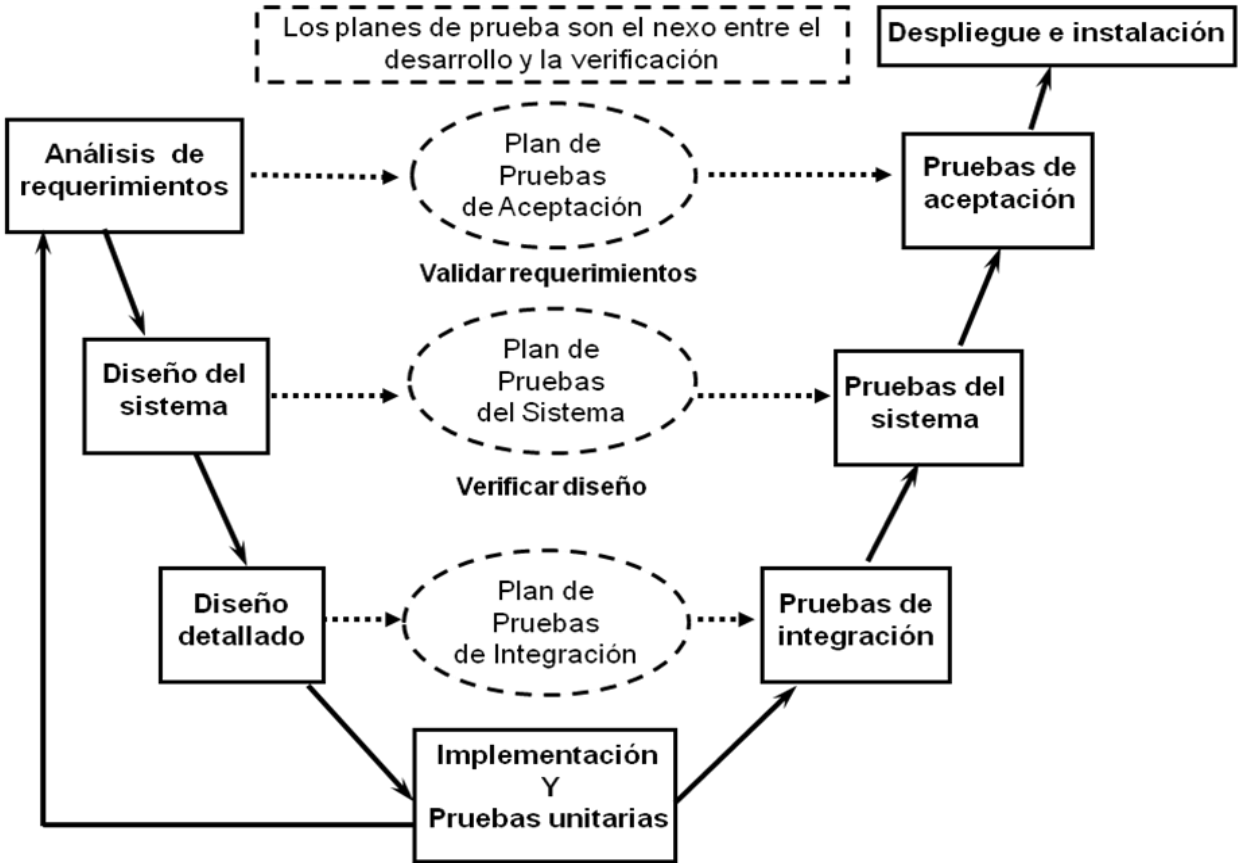


Figura 3.1 Proceso de Desarrollo de Software

Nota aclaratoria:

Para las plantillas a utilizar durante el proceso de desarrollo se hace alusión a las que actualmente tiene definidas la infraestructura productiva de la UCI para el expediente de proyecto.

Fase – Análisis de requerimientos

El propósito general de esta fase es dirigir el proceso de desarrollo hacia el sistema correcto obteniendo los requerimientos del sistema. En la figura 3.2 se muestra el mapa de proceso de la fase de análisis de requerimientos.

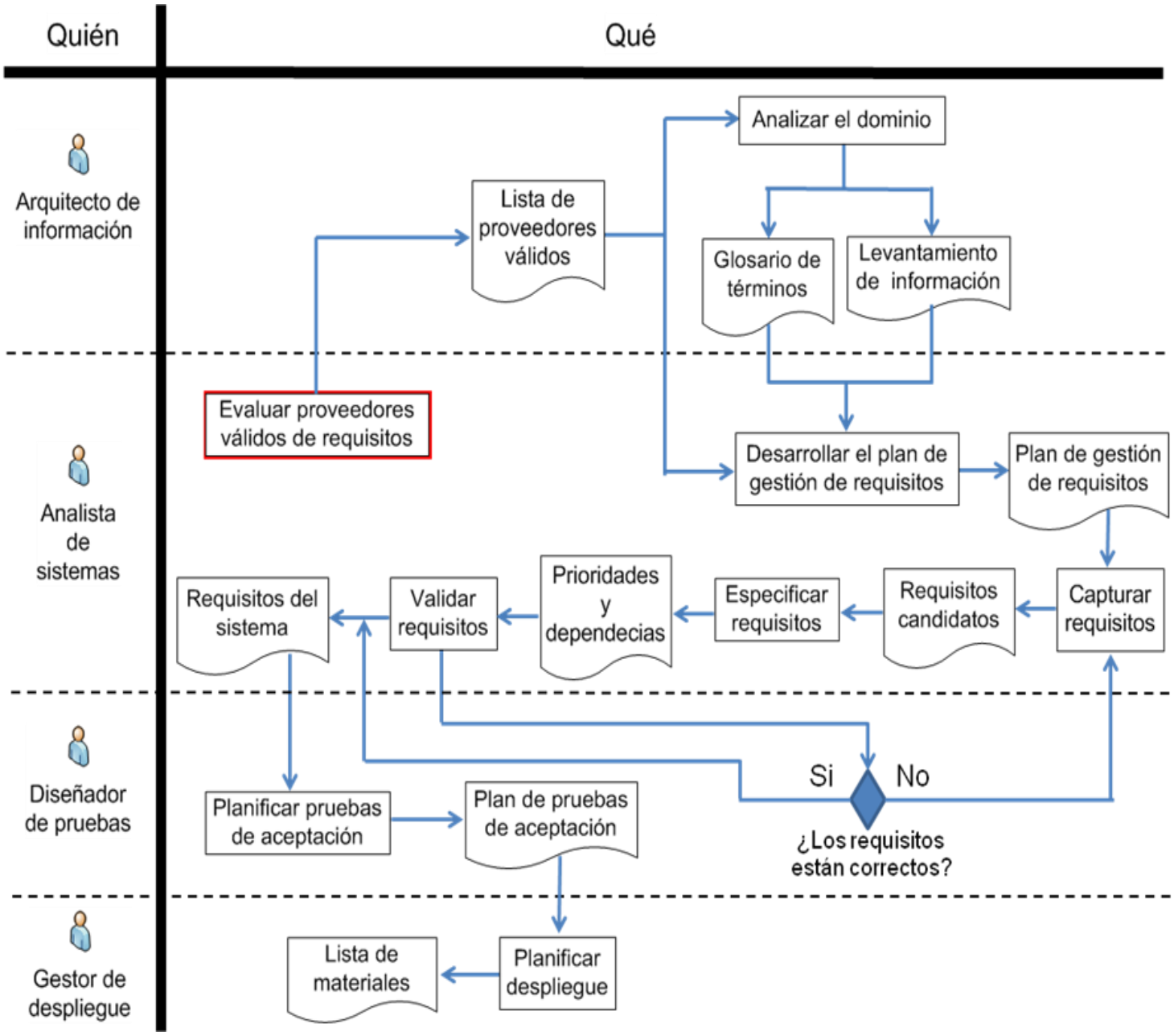


Figura 3.2 Mapa de proceso de la fase análisis de requerimientos

Artefactos

- Lista de proveedores válidos: este artefacto contiene el personal encargado de suministrar toda la información necesaria para definir los requisitos del sistema.
- Levantamiento de información: este artefacto define el conocimiento obtenido por el especialista en información durante las reuniones con el cliente. Dicho conocimiento comprende 3 niveles de información. Un primer

nivel que puntualiza las metas, objetivos y justificación de la aplicación. Un segundo nivel que describe la audiencia, sus necesidades de información, así como una caracterización de la misma, según su cultura informática. Por último un tercer nivel que contiene un esbozo de la estructura y una descripción textual de los elementos que la componen.

- Glosario de términos: este artefacto define términos importantes que se utilizan en el proyecto.
- Requisitos candidatos: este artefacto define una descripción informal de los requisitos del sistema y su clasificación.
- Prioridades y dependencias: este artefacto define una descripción formal de los requisitos del sistema, donde se precisan sus prioridades y dependencias.
- Plan de gestión de requisitos: este artefacto describe los artefactos de requisitos, tipos de requisitos y sus respectivos atributos de requisitos, especificando la información que se debe recopilar y los mecanismos de control que deben utilizarse para medir, informar y controlar los cambios de los requisitos del producto.
- Requisitos del sistema: este artefacto define una descripción formal y validada por el cliente de los requisitos del sistema.
- Plan de pruebas de aceptación: este artefacto define el objetivo de las pruebas de aceptación, los elementos de destino, el enfoque que se adopta, los recursos necesarios y los entregables que se deben generar.
- Lista de materiales: este artefacto lista los componentes que constituyen una versión determinada de un producto y dónde se pueden encontrar los componentes físicos. Describe los cambios que se efectúan en la versión y hace referencia a cómo se puede instalar el producto.

Roles

Tabla 3.1 Roles de la fase – Análisis de requerimientos

Rol	Responsabilidades
Analista de sistemas	- Participa con el cliente y el usuario final recogiendo las entradas de los involucrados relevantes. - Captura los requisitos y definir las prioridades. - Realiza la especificación de requisitos.

	- Realiza el seguimiento de los requisitos durante todo el desarrollo del proyecto. - Determina los proveedores válidos de requisitos - Crea y actualiza Matriz de Trazabilidad
Arquitecto de información	- Identifica la visión, misión y objetivos del producto, equilibrando las necesidades de la organización patrocinadora y la de su público. - Realiza el estudio de homólogos para conocer el estado del arte del producto que se quiere desarrollar. - Realiza auditoría de información identificando las entidades de recursos de información conociéndose como: servicios, fuentes, sistema, contenidos.
Diseñador de pruebas	- Participa en la confección de la estrategia y el plan de pruebas. - Identifica los métodos, las técnicas, herramientas y directrices apropiadas para implementar las pruebas necesarias. - Establece en ambiente de comprobación
Gestor de despliegue	- Lidera la planificación de la transición del producto a la comunidad de usuarios, garantizando que estos planes se llevan a cabo adecuadamente, gestionando los problemas y supervisando el progreso.

Actividades y tareas por actividad

Para la sub-fase de la visión a los requisitos se toma la propuesta confeccionada por Calderón (2007), que detalla un procedimiento a seguir para el levantamiento de requisitos en los sistemas de portales.

- ✓ Evaluar proveedores válidos de requisitos: se define el personal encargado de suministrar toda la información necesaria para definir los requisitos del sistema.

Tareas:

- Confección del catálogo de proveedores de requisitos: el cliente debe facilitar un catálogo de proveedores de requisitos. Para la confección del catálogo se debe tener en cuenta que entre los proveedores se encuentre personal que pertenezca a departamentos o grupos importantes en la definición de los requisitos, como: clientes, usuarios finales, políticas, normativas, grupos de sistemas, grupos de calidad.
- Selección de criterios de validación: a partir del catálogo de proveedores de requisitos proporcionado por el cliente, el analista de sistemas debe evaluar quiénes son adecuados para participar en la captura de

requisitos. Para ello el analista debe definir cuáles van a ser los criterios de validación a tener en cuenta para su selección.

- Validación de proveedores: el analista de sistemas evalúa los proveedores contra los criterios de validación. Luego selecciona los proveedores válidos y se crea una lista con estos proveedores. Posteriormente se selecciona el proveedor responsable y se coloca en la primera fila de la lista de proveedores válidos. Por último se comprueba si el cliente está de acuerdo con la selección tanto de los proveedores como del responsable.

✓ Analizar el dominio: en esta actividad se realiza el Levantamiento de Información para la Arquitectura de Información (ver Anexo 9). Con él se obtiene el conocimiento necesario sobre la entidad y el sistema a desarrollar. Estos conocimientos son obtenidos mediante entrevistas con el cliente y permiten definir la estructura del portal y los elementos que la componen. Además se adquiere una base para definir el alcance, las metas, objetivos y justificación del proyecto.

Tareas:

- Formulación del proyecto: se identifiquen las metas, objetivos y justificación de la aplicación.
- Estudio de la audiencia: se identifica la audiencia, sus necesidades de información y se realiza una caracterización de la misma, según su cultura informática. (se segmenta la audiencia según necesidades, características, experiencias o a partir de los objetivos que se quieren alcanzar).
- Estudio de los homólogos: se realiza un estado del arte del producto, en el cual se investigan los sistemas similares u otros que deban trabajar con el proyecto en cuestión, se identifica quien puede ser una competencia y que elementos me pueden diferenciar de los demás.
- Especificación de los contenidos: en función de los objetivos del producto y las necesidades de información de la audiencia a la que se

dirige; se precisa los contenidos (imágenes, texto, video, etc.) que se desean publicar en el portal y sus interrelaciones.

- Especificación de la taxonomía: se agrupan y clasifican los contenidos estableciéndose relaciones entre las clases creadas.
- Análisis de procesos y servicios: se analizan los procesos y servicios que deben estar presentes en el portal, identificando su funcionamiento y necesidades. Se describen las relaciones entre usuarios y actividades que estos realizan. Para la representación gráfica de estas relaciones se puede emplear una especie de mapas de conceptos de roles y actividades.
- Elaboración del glosario: se confecciona un glosario de términos comunes para la descripción de usuarios procesos y actividades que realizan los usuarios.

✓ Desarrollar el plan de gestión de requisitos: en esta actividad se confecciona el Plan de gestión de requisitos (ver Anexo 11). En él se describe cómo documentar los requisitos, sus atributos y directrices para la rastreabilidad y gestión de los requisitos del producto.

Tareas:

- Planificación del seguimiento: para cada elemento se establecen un grupo de reglas o guías para aplicar la traceabilidad.
- Especificación de atributos de los requisitos: para cada elemento se identifican la lista de los atributos que serán utilizados para caracterizarlo.
- Planificación de los entregables del proyecto: se crea una lista tabular de los artefactos que serán creados durante el proyecto, incluso las fechas de entrega designadas.
- Gestión de cambios a los requisitos: se define cómo debe llevarse a cabo la gestión de los cambios en los requisitos partiendo de los procesos de Gestión de Configuración establecidos en la organización.

✓ Capturar requisitos: es la actividad mediante la cual el analista extrae, todas las necesidades del cliente. Todos los elementos de información deben ser

almacenados e identificados. Pueden provenir de conversaciones con el cliente o estar recogidos en ficheros de datos, gráficos, normativas legales, etc.

Tareas:

- Análisis del entorno: se realiza un estudio detallado sobre el entorno en el que funcionará el sistema, las condiciones bajo las cuales se ejecutará el mismo, el alcance, los objetivos y lograr un entendimiento y/o familiarización por parte del equipo de desarrollo en cuanto a estos aspectos.
- Especificación de usuarios finales: identificar los usuarios potenciales del portal, determinando sus características principales con el objetivo de presentarle la información lo más acorde posible a sus necesidades.
- Especificación de contenidos del portal: determinar los contenidos de información que desea mostrar el cliente y agruparlos según criterios específicos, que se deben tener en cuenta para la visualización de la información. Refiérase a posicionamiento de la información, accesibilidad y usabilidad de la misma, etc.
- Especificación de funcionalidades del portal: identificar las prestaciones que tendrá el portal en cuestión. La definición de los objetivos puede contribuir con la determinación de las funcionalidades del mismo.
- Especificación y clasificación de requerimientos: lograr una especificación detallada de los requerimientos del portal, teniendo en cuenta la clasificación de los requisitos que se plantea a continuación:
  a) Requisitos de datos.
  b) Requisitos de interfaz.
  c) Requisitos navegacionales.
  d) Requisitos de personalización.
  e) Requisitos transaccionales o funcionales internos.
  f) Requisitos no funcionales.

✓ Especificar requisitos: en esta actividad se deben definir los requisitos y negociarlos con el cliente, para lograr obtener el documento de

Especificación de Requerimientos, aplicándole técnicas de revisión al mismo, garantizando que se obtenga una especificación de requisitos lo más certera posible, evitando al máximo los problemas que puede traer consigo una mala definición de requisitos del portal.

Tareas:

- Marcaje de requerimientos: se debe analizar la definición inicial o informal de los requisitos para poder establecer las dependencias que existen entre ellos; asignar un identificador único que permita rastrearlo en todo momento y establecer niveles de prioridad en función de las peticiones del cliente.
- Especificación de requerimientos según su clasificación: se elabora una descripción formal de los requisitos, clasificándolos según los tipos de requisitos vistos con anterioridad. Realizarle revisiones al documento en conjunto con el equipo de desarrollo y el cliente, en caso de ser necesario, para obtener una especificación formal.

✓ Validar requisitos: una vez que han sido definidos los requisitos del portal, se debe realizar un proceso de validación de los mismos, para garantizar que la definición planteada de cada uno de ellos es la más adecuada. Para ello se deben hacer revisiones a los modelos obtenidos verificando el cumplimiento de cada uno de los requisitos especificados en etapas anteriores. Finalmente se confecciona el Documento de especificación de requisitos de software (ver Anexo 12), donde quedan definidos todos los requerimientos de sistema.

Tareas:

- Verificación de la calidad en la especificación de los requerimientos: a través de la aplicación de técnicas realizar revisiones constantes a la definición de los requisitos, para garantizar que estos estén escritos de la forma correcta, sin ambigüedades, verificables, consistentes, entendible por el cliente, trazables en todos los sentidos, organizados y de manera concisa.

- Realización de la trazabilidad de los requerimientos: llevar la matriz de trazabilidad para comprobar que los requerimientos obtenidos inicialmente se transformaron en alguna de las funcionalidades del portal. Se puede evaluar a través de la revisión del cumplimiento de los objetivos del portal.
- Comprobación del ajuste a peticiones del cliente: se tiene que realizar una revisión de los requerimientos definidos con el cliente para comprobar que esos son los que él realmente desea que sean implementados en el sistema.

✓ Planificar pruebas de aceptación: en esta actividad se describe la estrategia, recursos y planificación de las pruebas de aceptación de usuario. Como resultado se conforma el Plan de pruebas (ver Anexo 15).

Tareas:

- Especificación de criterios de aceptación: se definen los criterios de la aceptación del portal, dígase requisitos de funcionamiento, requisitos de calidad de la interfaz y requisitos de calidad totales del software.
- Descripción de la estrategia de prueba: se describe el flujo de trabajo que se utilizará para la ejecución de las pruebas.
- Estimación de requisitos para el esfuerzo de la prueba: se negocia el uso más eficaz de los recursos de pruebas para cada iteración y se toman acuerdos en cuanto a los objetivos de la prueba y se definen cuales van a ser los entregables alcanzables para la iteración.
- Especificación de la evaluación para las pruebas: se describen los criterios de evaluación para las pruebas, como clasificación de las No Conformidades y Pedidos de cambios.

✓ Planificar el despliegue: se planifican los esfuerzos del despliegue para producir el material auxiliar necesario para un despliegue eficiente.

Tareas:

- Especificación de la lista de materiales: se describe la lista de materiales, un inventario del software y de los materiales que deben entregarse como parte del producto global.

Fase – Diseño del sistema

El propósito general de esta fase es modelar el sistema a través de casos de uso.

En la figura 3.3 se muestra el mapa de proceso de la fase de diseño del sistema.

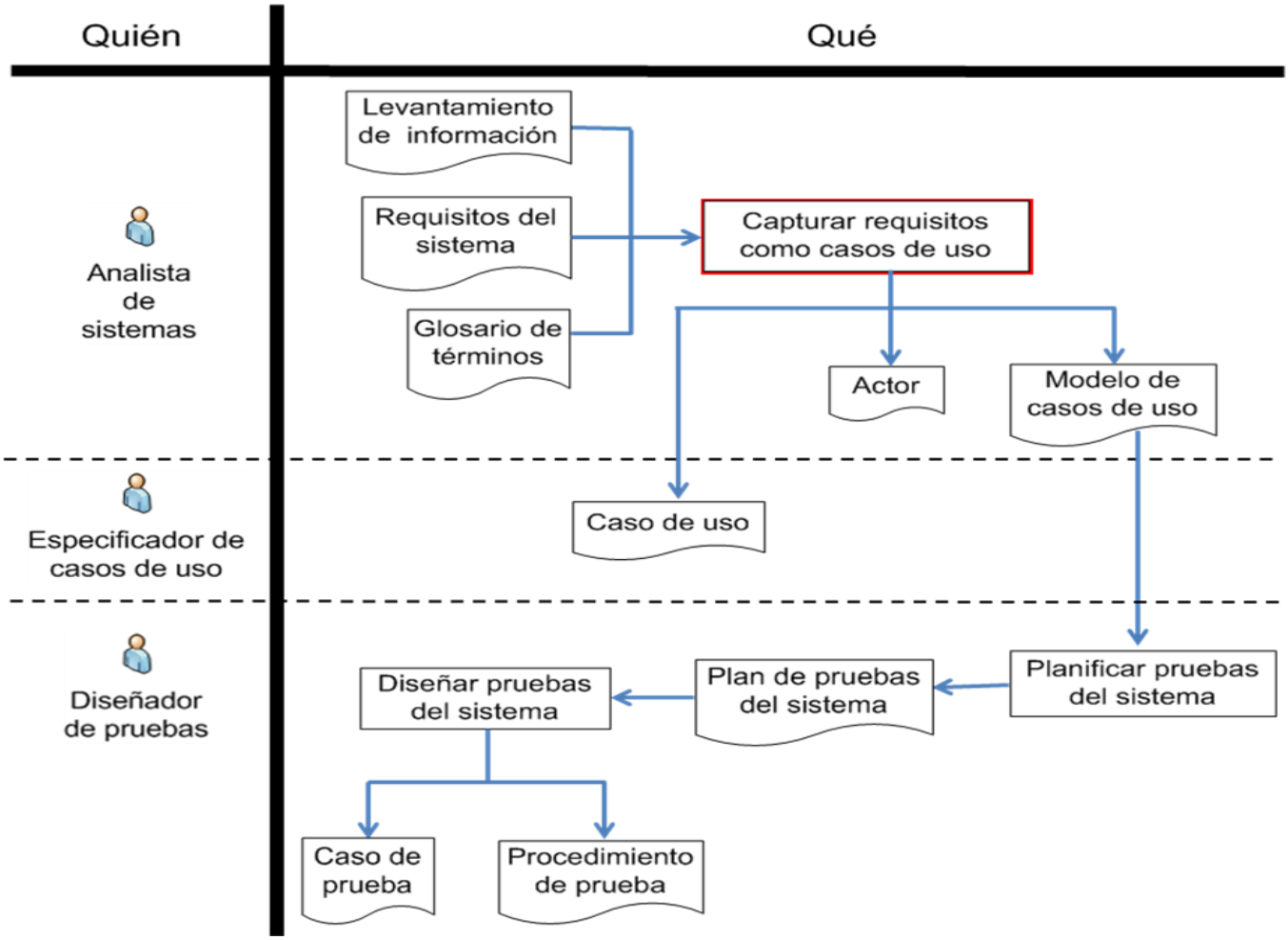


Figura 3.3 Mapa de proceso de la fase diseño del sistema

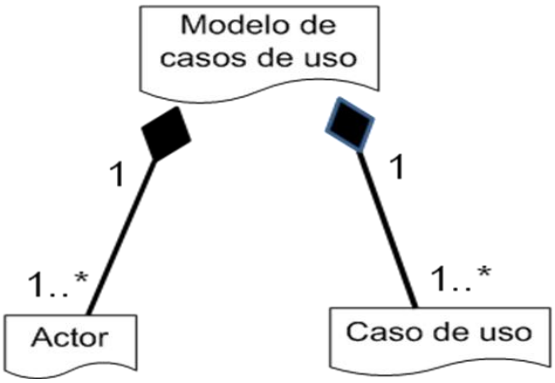


Figura 3.4 Modelo de casos de uso

Artefactos

- Actor: este artefacto define un conjunto coherente de roles que los usuarios del sistema pueden desempeñar cuando interactúan con este. Una instancia de este artefacto se puede llevar a cabo en un sistema individual o externo.
- Modelo de caso de uso: este artefacto es un modelo de las funciones deseadas para el sistema y su entorno, y sirve como contrato entre el cliente y los desarrolladores. Se utiliza como entrada esencial para las fases de diseño, implementación y prueba.
- Caso de uso: este artefacto define un conjunto de instancias, donde cada instancia es una secuencia de acciones que lleva a cabo un sistema que producen un resultado observable de valor para un actor concreto.
- Plan de pruebas del sistema: este artefacto define el objetivo de las pruebas del sistema, los elementos de destino, el enfoque que se adopta, los recursos necesarios y los entregables que se deben generar.
- Caso de prueba: este artefacto define un conjunto de entradas de prueba, condiciones de ejecución, y resultados esperados, identificados con el objetivo de evaluar algunos aspectos particulares de un elemento del destino de la prueba.
- Procedimiento de prueba: especifica cómo se deben realizar uno o varios casos de prueba o parte de estos.

Roles

Tabla 3.2 Roles de la fase – Diseño del sistema

Rol	Responsabilidades
Analista de sistemas	- Dirige y coordina la adquisición de requisitos y casos de uso, esquematizando la funcionalidad del sistema y delimitándolo.
Especificador de casos de uso	- Asume las responsabilidades de las descripciones detalladas de uno o más casos de uso.
Diseñador de pruebas	- Participa en la confección de la estrategia y el plan de pruebas. - Identifica los métodos, las técnicas, herramientas y directrices apropiadas para implementar las pruebas necesarias. - Establece en ambiente de comprobación - Encargado de diseñar casos de pruebas para el sistema - Diseña las pruebas. - Genera los casos y datos de prueba.

Actividades y tareas por actividad

- ✓ Capturar requisitos como casos de uso: en esta actividad se identifican los actores y los casos de uso que dan soporte a los requisitos que se están implementando. La identificación explícita de los actores y los casos de uso debe quedar reflejada en el modelo del sistema (ver Anexo 13).

Tareas:

- Especificación de actores: se identifican los actores para el mantenimiento y operación del sistema y los actores que representan sistemas externos. Luego debe ser descrito el papel de cada actor y para que utiliza el sistema.
- Especificación de casos de uso: para encontrar los casos de uso el analista debe tener reuniones con el cliente y el usuario e ir repasando los actores uno por uno e ir proporcionando los casos de uso para cada actor considerando qué necesita cada actor del sistema.
- Descripción de caso de uso: se debe describir paso a paso lo que el sistema necesita hacer cuando interactúa con sus actores.
- Descripción del modelo de casos de uso: describe a través de un diagrama cómo interactúan los actores y los casos de uso y cómo se relacionan entre si los casos de uso.
- Priorización de casos de uso: se determina cuáles de los casos de uso deben ser priorizados a la hora de comenzar a desarrollar el sistema y cuáles pueden realizarse posteriormente.

- ✓ Planificar pruebas del sistema: en esta actividad describe la estrategia, recursos y planificación de las pruebas funcionales y no funcionales del sistema. Como resultado se conforma el Plan de pruebas (ver Anexo 15).

Pruebas funcionales: estas pruebas se encargan de probar las funcionalidades que comprende cada caso de prueba.

Pruebas no funcionales: estas pruebas son las llamadas pruebas de rendimiento, dígase pruebas de carga, estrés, estabilidad y picos.

Tareas:

- Planificación de requerimientos a probar: se define el listado de requerimientos a probar.
- Descripción de la estrategia de prueba: se describe el flujo de trabajo que se utilizará para la ejecución de las pruebas.
- Estimación de requisitos para el esfuerzo de la prueba: se negocia el uso más eficaz de los recursos de pruebas para cada iteración y se toman acuerdos en cuanto a los objetivos de la prueba y se definen cuales van a ser los entregables alcanzables para la iteración.
- Planificación de la evaluación para las pruebas: se describen los criterios de evaluación para las pruebas.

✓ Diseñar pruebas del sistema: se identifican y se describen los casos y procedimientos de prueba, especificando como realizar los casos de prueba y los prerrequisitos imprescindibles para ejecutarlo (ver Anexo 16).

Tareas:

- Descripción de la funcionalidad: se describe la funcionalidad a probar.
- Descripción del flujo central del caso de prueba: se describen los pasos a desarrollar para probar la funcionalidad que se indicó.
- Descripción de condiciones de ejecución: se definen los prerrequisitos imprescindibles para que se pueda ejecutar correctamente el caso de prueba.

Fase – Diseño detallado

La fase de diseño explica cómo transformar los productos de trabajo de los requisitos en los productos de trabajo que especifiquen el diseño del software que el proyecto va a desarrollar. En la figura 3.5 se muestra el mapa de proceso de la fase de diseño detallado.

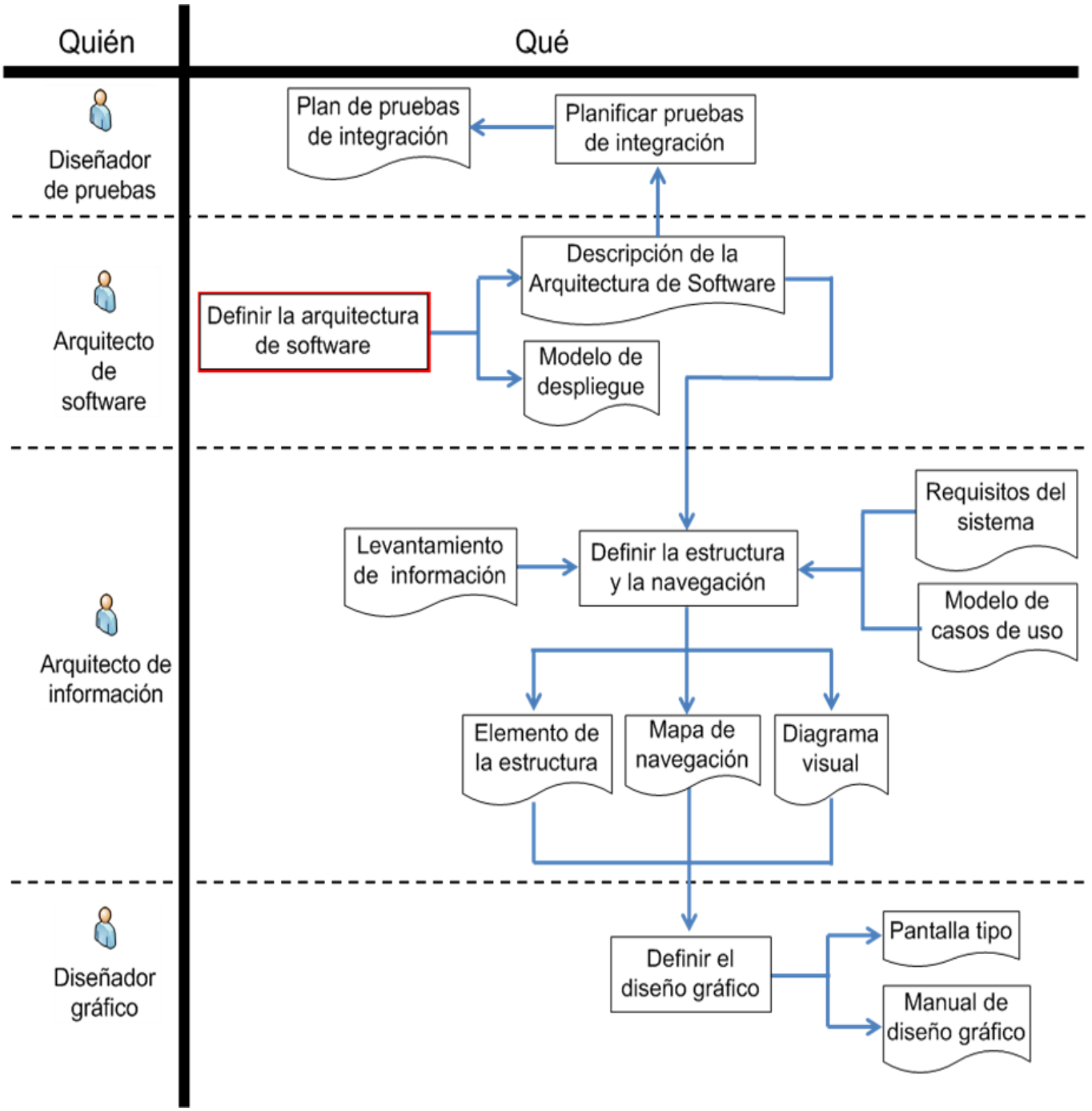


Figura 3.5 Mapa de proceso de la fase de diseño detallado

Artefactos

- Descripción de la arquitectura de software: este producto de trabajo proporciona una visión general arquitectónica completa del sistema,

mediante una serie de vistas arquitectónicas diferentes para representar diferentes aspectos del sistema.

- Modelo de despliegue: este producto de trabajo muestra la configuración de los nodos de proceso en el tiempo de ejecución, los enlaces de comunicación entre ellos, y las instancias de componente y los objetos que residen en ellas.
- Elemento de la estructura: este artefacto define un elemento que forma parte de una página del portal.
- Diagrama visual: este artefacto define la representación lineal de cada uno de los elementos de la estructura, con el objetivo de verificar su ubicación.
- Mapa de navegación: este artefacto describe el sentido y los niveles de navegación que presenta el portal. Su representación suele ser en forma de árbol, con secciones, niveles y contenidos relacionados.
- Pantalla tipo: este artefacto define la apariencia de una página del portal y de los elementos que la componen.
- Manual de diseño gráfico: este artefacto define las pautas de diseño gráfico para todas las páginas y los elementos que las componen.
- Plan de pruebas de integración: este artefacto define el objetivo de las pruebas de integración, los elementos de destino, el enfoque que se adopta, los recursos necesarios y los entregables que se deben generar.

Roles

Tabla 3.3 Roles de la Fase – Diseño detallado

Rol	Responsabilidades
Diseñador gráfico	- Encargado de realizar todo el diseño gráfico que requiera el sistema. - Interviene en la creación del prototipo. - Define las pautas para el diseño de la interfaz.
Arquitecto de software	- Define todos los elementos bases de la arquitectura del proyecto. - Identifica todos los posibles escenarios de despliegue de la aplicación - Determina de conjunto con los diseñadores las interfaces de integración tanto internas como externas. - Elabora el documento de arquitectura de software. - Define las herramientas, bibliotecas, componentes, *Frameworks* y otros componentes que permitan acelerar y mejorar el trabajo del proyecto. - Define de conjunto con el jefe de proyecto el flujo de desarrollo basado en las herramientas identificadas. - Vela por el cumplimiento de los requisitos de hardware. - Responsable de la integración de los componentes del sistema.

Arquitecto de información	- Realiza la organización y representación de los contenidos a través del diseño del sistema de navegación y diseño del sistema de etiquetado para el sistema de navegación.
Diseñador de pruebas	- Participa en la confección de la estrategia y el plan de pruebas. - Identifica los métodos, las técnicas, herramientas y directrices apropiadas para implementar las pruebas necesarias. - Establece en ambiente de comprobación

Actividades y tareas por actividad

El flujo de diseño consta de las siguientes actividades:

- ✓ Definir la arquitectura de software: en esta actividad se utilizan una serie de patrones y abstracciones que proporcionan el marco de referencia necesario para guiar la construcción del software. En el anexo 14 se muestra la plantilla a utilizar para la descripción de la arquitectura de software.

Tareas:

- Descripción de la arquitectura de software: en el caso de los proyectos de portales de la Facultad #10 de la UCI, se utiliza como arquitectura de software el modelo vista controlador (MVC), el cual divide la aplicación en tres áreas: procesamiento, salida y entrada. Para lo cual, utiliza las siguientes abstracciones:

Modelo (*Model*): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

Vista (*View*): Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

Controlador (*Controller*): Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio (*"service requests"*) para el modelo o la vista.

- ✓ Planificar pruebas de integración: en esta actividad describe la estrategia, recursos y planificación de las pruebas de integración. Como resultado se conforma el Plan de pruebas (ver Anexo 15).

Tareas:

- Selección de requerimientos a probar: se define el listado de requerimientos a probar.
- Descripción de estrategia de prueba: se describe el flujo de trabajo que se utilizará para la ejecución de las pruebas.
- Estimación de requisitos para el esfuerzo de la prueba: se negocia el uso más eficaz de los recursos de pruebas para cada iteración y se toman acuerdos en cuanto a los objetivos de la prueba y se definen cuales van a ser los entregables alcanzables para la iteración.
- Planificación de la evaluación para las pruebas: se describen los criterios de evaluación para las pruebas.

✓ Definir la estructura y la navegación: se define la estructura de los diagramas visuales y la navegación del portal. Con ello y la información recogida en el levantamiento de la información durante la fase de requisitos se confecciona el documento Arquitectura de la información (ver Anexo 10). Dicho documento junto con los requisitos del sistema, la arquitectura de software y el modelo de casos de uso del sistema constituyen los pilares para construir la aplicación durante la fase de implementación.

Tareas:

- Especificación de sistemas de navegación: se conceptualizan los diferentes elementos del sistema de navegación que guiaran el proceso de interacción entre el usuario y el portal, elementos que los conformaran y niveles donde estarán presentes (global, local o específico).
- Especificación de sistemas de etiquetas: se seleccionan los conceptos o símbolos que representaran los contenidos agrupados en las clases de los sistemas de taxonomía, pueden ser definidas etiquetas de navegación, enlace, encabezamiento o título y de indización de los productos.

- Diseño de diagramas visuales: se realizan los diferentes prototipos de bajo nivel que definen la representación lineal de cada uno de los elementos de la estructura.

✓ Elaboración del diseño gráfico: en esta actividad se confecciona el diseño gráfico del sitio y su manual correspondiente. Para ello el diseñador gráfico toma como base la arquitectura de la información elaborada previamente. En ella se encuentran detallados todos los elementos que el diseñador necesita para realizar su trabajo.

Tareas:

- Diagramación: consiste en representar los contenidos que tendrá el portal y las relaciones entre dichos contenidos.
- Confección de pantallas tipo: se confeccionan las pantallas tipo, que a diferencia de los diagramas visuales realizados por el especialista en información en el documento de la arquitectura de la información, contienen todos los elementos del diseño gráfico confeccionados por el diseñador. Por tanto cada pantalla tipo constituye una vista exacta de cómo debe verse la correspondiente página para su construcción durante la fase de implementación.
- Corte de las piezas del diseño gráfico: se cortan las piezas del diseño gráfico necesarias para que el programador desarrolle el tema de la aplicación durante la fase de implementación.
- Confección del manual: se confecciona el manual de diseño gráfico que contiene las pautas del diseño realizado. Dichas pautas en la fase de implementación, servirán al programador para conocer todos los detalles que necesita para la construcción del sitio.

Fase – Implementación

En la fase de implementación se realiza la construcción del portal a partir de los artefactos generados en las etapas previas. En la figura 3.6 se muestra el mapa de proceso de la fase de implementación.

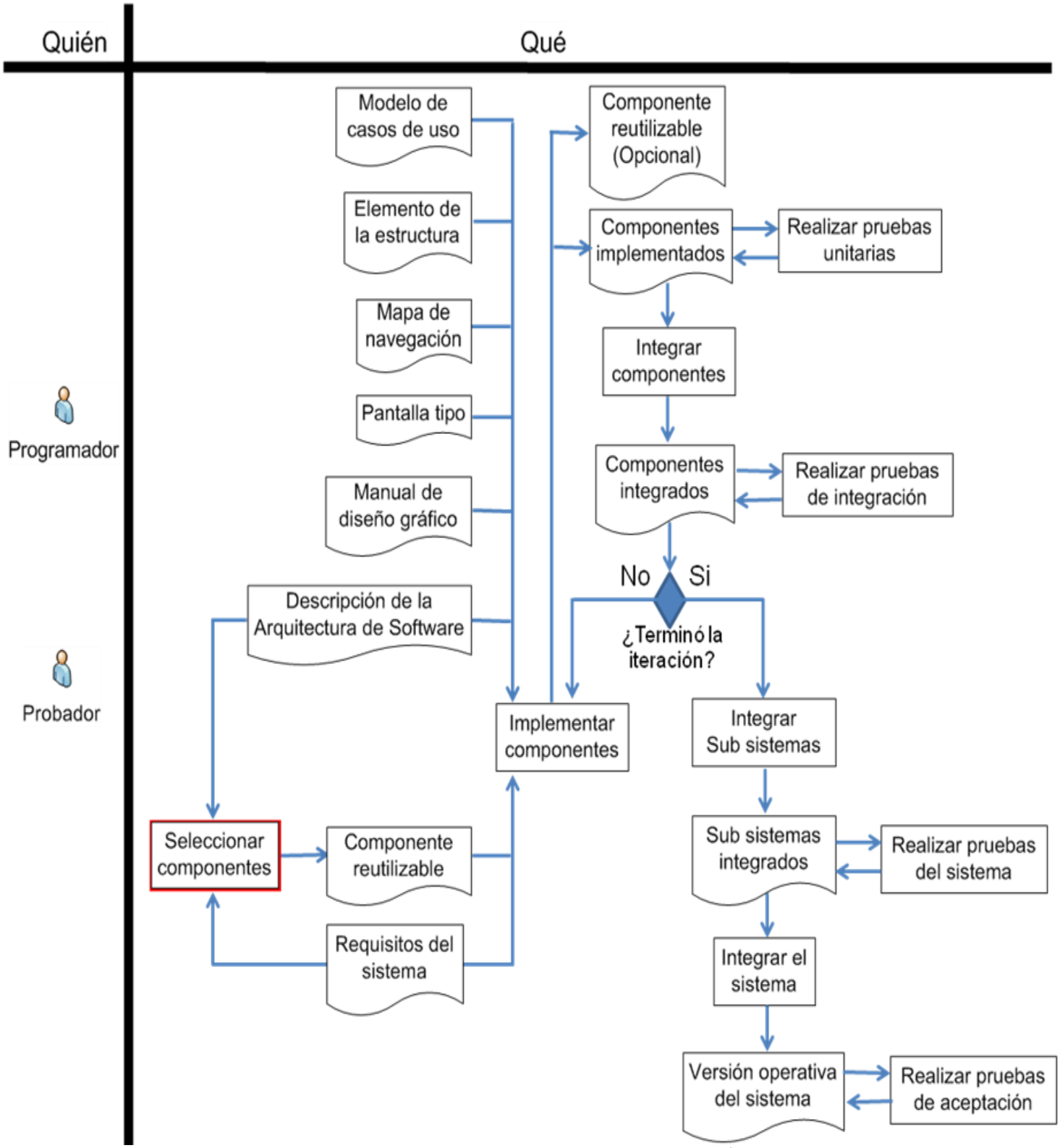


Figura 3.6 Mapa de proceso de la fase de implementación

Artefactos

- Interfaz: este producto de trabajo es el resultado de la implementación de la pantalla tipo definida durante la fase de diseño.
- Componente reutilizable: este artefacto se encuentra dividido en dos grandes grupos. Por un lado los componentes de código, como pueden ser: procedimientos, funciones, módulos, aplicaciones, etc., y por otro lado los activos del proceso, como pueden ser: patrones de diseño, algoritmos, esquema de base de datos, manuales, documentación, modelos, etc.
- Componente: son los componentes físicos que forman una implementación (contenidos, módulos, temas, etc).
- Subsistema de implementación: este artefacto consta de un conjunto de elementos de implementación. Estructura el modelo de implementación dividiéndolo en componentes más pequeños que se pueden integrar y probar separadamente.
- Versión operativa del sistema: este artefacto demuestra un subconjunto de las posibilidades que se proporcionan en el producto final.

Roles

Tabla 3.4 Roles de la Fase – Implementación

Rol	Responsabilidades
Programador	- Determina los componentes candidatos para llevar a cabo las funcionalidades necesarias; y los acopla adecuadamente a la arquitectura especificada para el sistema. - Convierte la especificación del sistema en código fuente ejecutable. - Desarrolla el diseño teniendo en cuenta la arquitectura. - Realiza las pruebas de unidad. - Registra los resultados de las pruebas. - Analiza los resultados de las pruebas realizadas. - Desarrolla el prototipo de la interfaz de usuario. - Integra los componentes que forman parte de la solución.
Probador	- Encargado de seguir los planes de pruebas. - Ejecuta las pruebas de integración y del sistema. - Registra los resultados de las pruebas. - Analiza los resultados de las pruebas realizadas.

Actividades y tareas por actividad

El flujo de implementación consta de las siguientes actividades:

- ✓ Seleccionar componentes: consiste en seleccionar del repositorio de componentes, aquellos que se van a utilizar para la construcción del portal, garantizando que cada componente candidato realizará la función necesaria, se ajustará adecuadamente en la arquitectura empleada para el sistema y poseerá la calidad necesaria, en cuanto a rendimiento, fiabilidad y usabilidad.

Tareas:

- Cualificación de componentes: se identifican los componentes candidatos que pueden servir para la construcción del portal. Todos los componentes seleccionados deben acoplarse adecuadamente a la arquitectura especificada para el sistema. La selección se realiza analizando el modelo de diseño para determinar aquellos elementos del modelo que indican algunos componentes reutilizables ya existentes en el repositorio. Además del modelo de diseño, se puede extraer información a partir de la especificación de requisitos a través de una correspondencia de especificaciones. Se explora el repositorio en un intento de hacer coincidir el requisito con la descripción del componente almacenado en el repositorio.

Una vez seleccionados los componentes, lo ideal sería que estos se pudieran integrar fácilmente a la arquitectura de la aplicación; pero la realidad ha demostrado que es posible exhibir conflictos en una o más áreas. En estos casos el equipo de desarrollo debe determinar si es factible adaptar el componente, o si por el contrario se debería realizar un componente personalizado (diseñado para eliminar los conflictos que se encuentren). Una vez obtenida la selección candidata se revisan nuevamente los requisitos con el objetivo de comprobar si los componentes que constituyen la selección candidata satisfacen eficientemente todos los requisitos definidos. En caso de no ser así, se debe volver a realizar la cualificación de los componentes con todo lo que esto implica.

- Ajuste de componente: se adapta un componente para cumplir las necesidades del sistema o de la arquitectura de software. (Opcional)
- Actualización de repositorio: se actualiza la biblioteca o repositorio de componentes (Opcional).

✓ Implementar componentes: en esta actividad se desarrollan los componentes que forman una parte de la implementación, quedando listos para la integración.

Tareas:

- Planificación de la integración del subsistema: en esta tarea se planifica el orden en se deben implementar los componentes que pertenecen a un subsistema en aras de lograr una correcta integración.
- Implementación de los elementos de diseño: en esta tarea se produce una implementación para una parte del diseño (por ejemplo, un contenido, un caso de uso o una entidad de base de datos), o para solucionar uno o varios defectos. Como resultado de esta tarea se obtiene archivos nuevos o modificados de datos y de código fuente.

✓ Realizar pruebas unitarias: se ejecuta la prueba y se recopilan los resultados.

Tareas:

- Ejecución de pruebas unitarias: se realiza el análisis del código escrito con la intención de eliminar errores y lograr el mayor grado de eficiencia en el mismo. Generalmente se realizan pruebas de caja blanca.
- Evaluación de resultados: se comparan los resultados de las pruebas con los resultados esperados y se investigan los resultados de las pruebas que no coinciden con los esperados.
- Registro de revisión y defectos: se documentan los resultados de la revisión y los defectos identificados en el documento de Registro de pruebas unitarias (ver Anexo 17).

✓ Integrar componentes: en esta actividad se integran los componentes que forman parte de un subsistema de implementación.

Tareas:

- Integración de componentes: en esta tarea se integran los elementos que forman parte de un subsistema.

✓ Realizar pruebas de integración: se ejecutan las pruebas y se recopilan los resultados.

Tareas:

- Ejecución de pruebas de integración: se verifica que los componentes que forman parte de un subsistema se integran correctamente entre sí y con la arquitectura de software. Pueden detectarse averías en interfaces y en la interacción entre los componentes integrados. Generalmente se realizan pruebas de caja negra, pues el código no se comprueba directamente para saber si hay errores. Se realizan usando el diseño para las pruebas de integración elaborado luego de definir la arquitectura de software.
- Evaluación de resultados: se comparan los resultados de las pruebas con los resultados esperados y se investigan los resultados de las pruebas que no coinciden con los esperados.
- Registro de revisión y defectos: se documentan los resultados de la revisión y los defectos identificados en el documento de Registro de pruebas de integración (ver Anexo 17).

✓ Integrar subsistemas: en esta actividad se integran los subsistemas que forman parte del sistema de implementación.

Tareas:

- Selección de subsistemas: se seleccionan los subsistemas que se desean integrar.
- Integración de subsistemas: en esta tarea se integran los subsistemas que forman parte del sistema.

✓ Realizar pruebas del sistema: se ejecutan las pruebas y se recopilan los resultados.

Tareas:

- Ejecución de pruebas del sistema: se realizan las pruebas, ejecutando los procedimientos de prueba para cada caso de prueba. Durante las pruebas del sistema se comparan las especificaciones que se definieron para el software contra el sistema real.
- Evaluación de resultados: se comparan los resultados de las pruebas con los resultados esperados y se investigan los resultados de las pruebas que no coinciden con los esperados.
- Registro de revisión y defectos: se documentan los resultados de la revisión y los defectos identificados para cada caso de prueba.

✓ Integrar el sistema: esta actividad integra los subsistemas de implementación para crear una nueva versión coherente del sistema global.

Tareas:

- Integración del sistema: en esta tarea se integran las partes de los subsistemas de implementación en una versión operativa del sistema.

✓ Realizar pruebas de aceptación: se ejecutan las pruebas y se recopilan los resultados.

Tareas:

- Ejecución de pruebas de aceptación: se verifica el sistema o los cambios según las necesidades originales.
- Evaluación de resultados: se comparan los resultados de las pruebas con los resultados esperados y se investigan los resultados de las pruebas que no coinciden con los esperados.
- Registro de revisión y defectos: se documentan los resultados de la revisión y los defectos identificados en el documento de No conformidades (ver Anexo 18).

Fase – Despliegue e Instalación

El despliegue consiste en hacer que el producto de software esté disponible para el usuario final y es la culminación del esfuerzo de desarrollo de software. En la figura 3.7 se muestra el mapa de proceso de la fase de despliegue e instalación.

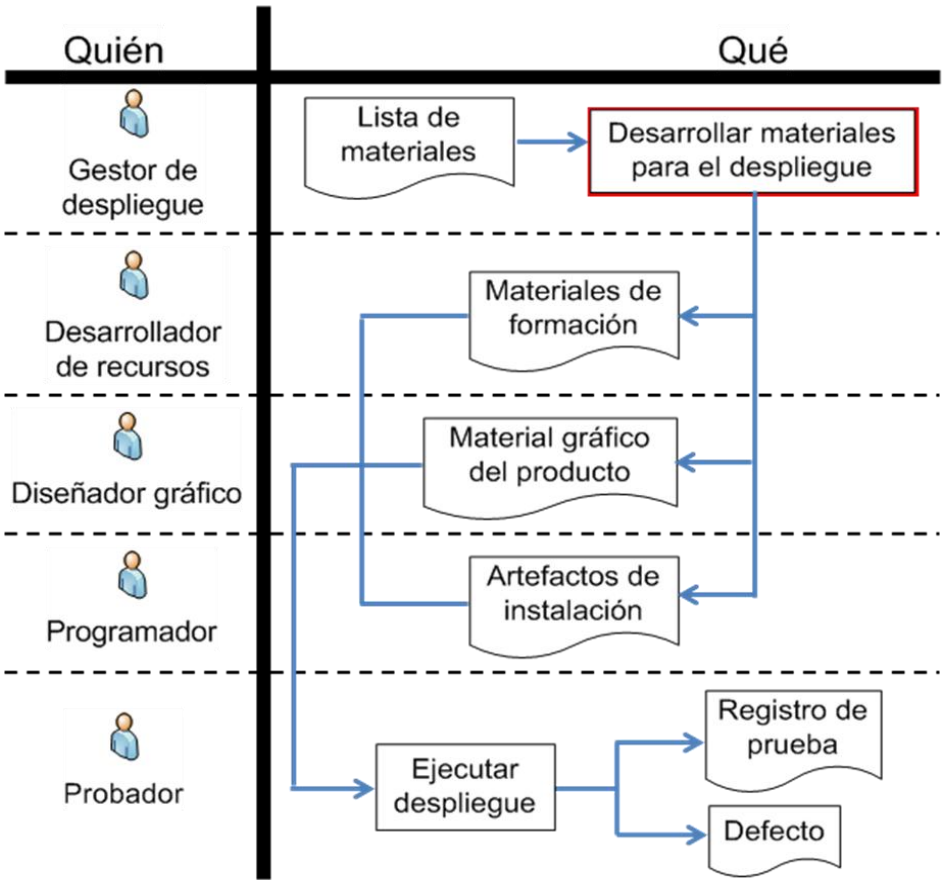


Figura 3.7 Mapa de proceso de la fase de despliegue e instalación

Artefactos

- Materiales de formación: este artefacto consta de los materiales que se utilizan en los programas de formación o cursos para asistir a los usuarios finales con la utilización del producto, operación y/o mantenimiento.
- Material gráfico del producto: este artefacto define el conjunto de materiales que conforman y describen el material gráfico del portal.
- Artefactos de instalación: este artefacto consta del software y de las instrucciones necesarias para instalar el producto.

Roles

Tabla 3.5 Roles de la Fase – Despliegue

Rol	Responsabilidades
Gestor de despliegue	- Lidera la planificación de la transición del producto a la comunidad de usuarios, garantizando que estos planes se llevan a cabo adecuadamente, gestionando los problemas y supervisando el progreso.
Desarrollador de recursos	- Desarrolla los materiales de formación para preparar a los usuarios para utilizar el producto.

Probador	- Realiza el despliegue y registra los resultados.
Programador	- Desarrolla los componentes de software y efectúa las pruebas de desarrollador.
Diseñador gráfico	- Describe la creación del material gráfico del producto.

Actividades y tareas por actividad

El flujo de despliegue e instalación consta de las siguientes actividades:

- ✓ Desarrollar materiales para el despliegue: esta actividad produce el material auxiliar necesario para un despliegue eficiente para sus usuarios.

Tareas:

- Desarrollo de materiales de formación: se producen los materiales necesarios para formar a los usuarios del producto.
- Desarrollo del material gráfico del producto: se describe la creación del material gráfico del producto.
- Desarrollo de productos de trabajo e instalación: se produce todo el software necesario para instalar y desinstalar el producto de forma rápida, sencilla y segura, sin afectar a las demás aplicaciones o características del sistema.

- ✓ Ejecutar el despliegue: se lleva a cabo la instalación y evaluación del portal y posteriormente se imparten cursos de administración y uso del mismo.

Tareas:

- Ejecución del montaje: se realiza la instalación en las unidades del usuario final.
- Ejecución de prueba piloto: se realiza un grupo de pruebas y se registran los resultados, junto con las discrepancias que puedan existir entre los resultados esperados y los reales. Esto permite garantizar que el producto desarrollado cumple con los criterios de aceptación.
- Impartición de cursos de administración y uso del portal: se imparten cursos para preparar a los usuarios finales con el uso del producto, funcionamiento y / o mantenimiento.

3.2. Consideraciones generales sobre el proceso propuesto

La presente propuesta formula un proceso de desarrollo para portales Web. En la misma se describen las fases, actividades, tareas, artefactos de entrada/salida y roles que deben estar presentes durante la ejecución de dicho proceso. Esta propuesta reune toda una serie de buenas prácticas para el desarrollo de un sistema Web y aporta ciertos elementos que contribuyen a aligerar en gran medida el proceso de desarrollo de los portales y a facilitar su modelación. Entre las buenas prácticas y aportes más relevantes que propone se encuentran:

- ✓ Modelación de sistemas donde el usuario pueda aprovechar el potencial del paradigma de sistemas Web, mientras realiza operaciones sobre bases de datos.
- ✓ Utilización de un procedimiento para la captura de requisitos en proyectos de portales Web, desarrollado por Calderón (2007).
- ✓ Aligeramiento de la modelación del sistema. La modelación consiste en obtener solamente los requerimientos del sistema, el modelo de casos de uso del sistema, la arquitectura de software, la arquitectura de la información y el diseño gráfico. De esta forma se logra disminuir el tiempo de modelación del portal.
- ✓ Implementación ágil. Durante la implementación se utiliza el modelo de proceso basado en componentes, el cual propicia que la misma se base en la reutilización de los componentes almacenados en el repositorio. De esta forma se logra disminuir el tiempo de implementación del portal.
- ✓ Permite planificar las pruebas desde etapas tempranas, previendo así gastos mayores producto de errores encontrados en etapas avanzadas del proyecto.

Capítulo 4. Validación de la propuesta

En el presente capítulo tiene como objetivo validar la propuesta. Primeramente se describe el método para la evaluación técnica y los resultados de dicha evaluación. Posteriormente se analizan los efectos de su aplicación en un proyecto piloto y finalmente se emite una valoración final sobre la misma.

4.1. Método para la evaluación técnica del proceso de desrrollo propuesto

Para el proceso de evaluación se seleccionó el método multicriterio basado en la cuantificación de aspectos cualitativos. El cual permite realizar un estudio de expertos que permita tomar decisiones para aceptar o no el proyecto de acuerdo con los criterios técnicos sobre el mismo. (LEÓN, 2005).

Puede calcularse el número de expertos necesarios, utilizando un método probabilístico y asumiendo una ley de probabilidad binomial mediante la siguiente expresión:

$$n = \frac{p * (1 - p) * k}{i^2}$$

Donde:

i- Nivel de precisión deseado.

p- Proporción estimada de errores de los expertos.

k- Constante asociada al nivel de confianza elegido.

Luego se realiza un proceso de selección de los expertos. La calidad de los expertos influye decisivamente en la exactitud y fiabilidad de los resultados y en ello interviene la calificación técnica, los conocimientos específicos sobre el objeto a evaluar y la posibilidad de decisión entre otros.

Para la aplicación de este método se siguieron los siguientes pasos:

Paso 1: Se definieron los objetivos del proyecto

Paso2: Se seleccionaron los especialistas que evaluaran el proyecto. Teniendo en cuenta que la calidad de los especialistas seleccionados, influye decisivamente en la exactitud y fiabilidad de los resultados. Por tal motivo para llevar a cabo el proceso de selección se tuvo en cuenta su especialidad, grado científico, currículo y los conocimientos específicos sobre el objeto a evaluar.

Paso 3: Se definieron los criterios de evaluación y se clasificaron en los cuatro grupos siguientes:

Criterios de méritos científicos

Criterios de implantación

Criterios de generalización

Criterios de impacto

Grupo No. 1: Criterios de méritos científicos

- Calidad de la investigación.
- Novedad científica.
- Valor científico de la propuesta.
- Aporte científico.

Grupo No. 2: Criterios de implantación

- Satisfacción de las necesidades de la producción.
- Garantía de principios básicos de la Ingeniería de Software.
- Uso de estándares de calidad.
- Necesidad del empleo del modelo.

Grupo No.3: Criterios de generalización

- Atractividad para su uso.
- Ventajas del producto, proceso o servicios que se desarrollan
- Adaptabilidad a diferentes entornos de producción de software.

Grupo No.4: Criterios de impacto

- Estandarización en la producción.
- Reutilización de componentes.
- Organización del proceso de producción.
- Ventajas competitivas.
- Posibilidades de aplicación.

Determinación del peso relativo de cada grupo

Paso 4: La comisión evaluadora asigna el porciento que representa cada grupo del total, de acuerdo con las características del proyecto y sus intereses:

Grupo No.1…… 20 Grupo no.3……. 20

Grupo No.2…… 25 Grupo No.4…….35

Paso 5: Se selecciona un grupo de expertos no menor de siete y se les entrega el proyecto a evaluar, los objetivos que se quieren lograr con el mismo y dos modelos. Uno para que valore el peso relativo de cada criterio (Ver Anexo 19) y otro para realizar una evaluación cuantitativa de cada criterio con una escala de 1-5 (Ver Anexo 21). Con la información recibida, los especialistas dispondrán de un tiempo determinado para realizar una evaluación cuantitativa de cada criterio, lo que acompañado de su apreciación cualitativa les permitirá realizar una clasificación final del proyecto con la categoría excelente, bueno, aceptable, cuestionable y malo. Los especialistas también harán su valoración final del proyecto, emitiendo todas aquellas consideraciones que permitan valorar la calidad del mismo, así como los momentos críticos durante su ejecución donde se puede decidir su continuación. Después de recibir los valores del peso relativo de cada criterio se construye la tabla que se muestra en el Anexo 19. Luego es necesario verificar la consistencia en el trabajo de expertos, para lo que se utiliza el coeficiente de concordancia de Kendall y el estadígrafo Chi cuadrado (X2); para lo cual se sigue el procedimiento siguiente:

Sea C el número de criterios que van a evaluarse y E el número de expertos que realizan la evaluación. Para cada criterio se determina ∑E que representa la sumatoria del peso dado por cada experto. Se calcula el peso medio de cada criterio (M∑E) y se determina la desviación de la media de cada criterio (ΔC), donde la misma se calcula como: ΔC = ∑E - ∑ (∑E / C). Posteriormente se eleva ΔC al cuadrado para obtener la dispersión (S) por la expresión: S = ∑ ΔC2. Conociendo la dispersión se puede calcular el coeficiente de concordancia de Kendall (W). W = S / E2 (C3 - C) / 12. El coeficiente de concordancia de Kendall permite calcular el Chi cuadrado real X2 = E (C - 1) W.

El Chi cuadrado calculado se compara con el obtenido de las tablas estadísticas. Si se cumple: X2real < X2 (α, c-1) Existe concordancia en el trabajo de expertos. Para ello se construye la tabla que se muestra en el Anexo 20. Si existe consistencia en el trabajo de expertos el peso de cada criterio se calcula promediando lo que cada uno de ellos le asignó a cada criterio entre 100, pero si no existe concordancia se hace necesario repetir el trabajo de expertos o eliminar a los expertos que afectan la concordancia.

Conociendo el peso de cada criterio y la calificación dada por los evaluadores en una escala de 1-5 se puede construir la tabla de calificación de cada criterio (Ver Anexo 21) y se determina el índice de aceptación del proyecto.

IA = P × C / 5

IA: Índice de Aceptación.

P: Peso de los criterios.

C: Criterio promedio concedido por los expertos.

Si:

IA > 0,7 Existe alta probabilidad de éxito

0,7 > IA > 0,5 Existe probabilidad media de éxito

0,5 > IA > 0,3 Probabilidad de éxito baja

0,3 > IA Fracaso seguro

4.2. Análisis de los resultados de evaluación técnica de la propuesta.

Para organizar la evaluación por el comité de expertos se calculó la cantidad de evaluadores utilizando la expresión

$$n = \frac{p * (1 - p) * k}{i^2}$$

Para los valores de i=0.10, p=0.01, K=6.6564, el número de expertos a utilizar es 7. Posteriormente los expertos emitieron su juicio para darles peso a los indicadores. Se elaboró la tabla de los valores del peso relativo de cada criterio.

G	C/E	E1	E2	E3	E4	E5	E6	E7	∑E	Ep
	C1	7	6	6	8	6	7	8	48	6,857142857
	C2	3	4	4	2	2	2	3	20	2,857142857
	C3	6	6	7	5	7	6	6	43	6,142857143
20	**C4**	4	4	3	5	5	5	3	29	4,142857143
	C5	7	7	9	8	6	6	9	52	7,428571429
	C6	6	6	4	5	6	8	5	40	5,714285714
	C7	8	9	9	8	7	7	9	57	8,142857143
25	**C8**	4	3	3	4	6	4	2	26	3,714285714
	C9	5	4	4	4	2	4	3	26	3,714285714
	C10	6	8	8	7	9	7	8	53	7,571428571
20	**C11**	9	8	8	9	9	9	9	61	8,714285714
	C12	9	9	9	8	7	7	8	57	8,142857143
	C13	9	9	8	8	9	9	8	60	8,571428571
	C14	9	9	8	8	9	9	8	60	8,571428571
	C15	3	2	5	6	6	3	4	29	4,142857143
35	**C16**	5	6	5	5	4	7	7	39	5,571428571
T		**100**	**100**	**100**	**100**	**100**	**100**	**100**	**700**	**100**

Tabla 3.1 Valores del peso relativos a cada criterio.

Una vez obtenido el peso relativo de cada criterio se pasa a verificar la consistencia en el trabajo de expertos, utilizando el coeficiente de concordancia de Kendall y el estadígrafo X2.

G	C/E	E1	E2	E3	E4	E5	E6	E7	∑E	Ep	ΔC	ΔC^2
	C1	7	6	6	8	6	7	8	48	6,857142857	4,25	18,0625
	C2	3	4	4	2	2	2	3	20	2,857142857	23,75	564,0625
	C3	6	6	7	5	7	6	6	43	6,142857143	0,75	0,5625
20	C4	4	4	3	5	5	5	3	29	4,142857143	14,75	217,5625
	C5	7	7	9	8	6	6	9	52	7,428571429	8,25	68,0625
	C6	6	6	4	5	6	8	5	40	5,714285714	3,75	14,0625
	C7	8	9	9	8	7	7	9	57	8,142857143	13,25	175,5625
25	C8	4	3	3	4	6	4	2	26	3,714285714	17,75	315,0625
	C9	5	4	4	4	2	4	3	26	3,714285714	17,75	315,0625
	C10	6	8	8	7	9	7	8	53	7,571428571	9,25	85,5625
20	C11	9	8	8	9	9	9	9	61	8,714285714	17,25	297,5625
	C12	9	9	9	8	7	7	8	57	8,142857143	13,25	175,5625
	C13	9	9	8	8	9	9	8	60	8,571428571	16,25	264,0625
	C14	9	9	8	8	9	9	8	60	8,571428571	16,25	264,0625
	C15	3	2	5	6	6	3	4	29	4,142857143	14,75	217,5625
35	C16	5	6	5	5	4	7	7	39	5,571428571	4,75	22,5625
T		100	100	100	100	100	100	100	700	100		3015

Tabla 3.2 Cálculo de concordancia entre los expertos.

De donde se deduce que M∑E = ∑ΔE / C = 700 / 16 = 43,75 y S = ∑ ΔC2 = 3015 dando como resultado para W = S / E2 (C3 – C) / 12 = 3015 / 72 (163 - 16) / 12 = 0,180972, lo cual resultó en un Chi χ2 real = E (C – 1) W = 7 (16 - 1) 0,180972= 19,00206. Por otra parte el Chi cuadrado de las tablas de Siegel, para un coeficiente de confianza (K) de un 90 % resultó: χ2 (α, c-1) = χ2 (0.10, 15) = 22,3071 el cual es mayor que el real por lo que se cumple la concordancia entre el trabajo de los expertos. Posteriormente se procede a mostrar el peso correspondiente para cada criterio, este resulta de la división por 100 del peso promedio de la puntuación de cada criterio (Ep).

C/E	Peso
Peso de C1	0,068571429
Peso de C2	0,028571429
Peso de C3	0,061428571
Peso de C4	0,041428571
Peso de C5	0,074285714
Peso de C6	0,057142857
Peso de C7	0,081428571
Peso de C8	0,037142857
Peso de C9	0,037142857
Peso de C10	0,075714286
Peso de C11	0,087142857
Peso de C12	0,081428571
Peso de C13	0,085714286
Peso de C14	0,085714286
Peso de C15	0,041428571
Peso de C16	0,055714286

Tabla 3.3 Peso de cada criterio

Conociendo el peso de cada criterio y la calificación de los evaluadores en una escala de 1 a 5 se puede construir la siguiente tabla, para obtener el índice de aceptación.

Calificación	Calificación (c)					P	P x C
	1	2	3	4	5		
C1				x		0,068571429	0,274285716
C2	x					0,028571429	0,028571429
C3	x					0,061428571	0,061428571
C4	x					0,041428571	0,041428571
C5					x	0,074285714	0,37142857
C6					x	0,057142857	0,285714285
C7			x			0,081428571	0,244285713
C8					x	0,037142857	0,185714285
C9				x		0,037142857	0,148571428
C10					x	0,075714286	0,37857143
C11	x					0,087142857	0,087142857
C12				x		0,081428571	0,244285713
C13					x	0,085714286	0,42857143
C14					x	0,085714286	0,42857143
C15				x		0,041428571	0,165714284
C16			x			0,055714286	0,167142858
TOTAL							**3,54142857**

Tabla 3.4 Calificación de los evaluadores

Índice de aceptación: (IA= Σ C X P / 5) =0,708285714, por lo que se puede concluir que existe una alta probabilidad de éxito. La mayoría de los encuestados consideran factible la aplicación de este proceso, opinan que el mismo puede garantizar la eficiencia en el desarrollo de portales, ya que contribuye a organizar y estructurar la producción, lo cual favorece el cumplimiento de los plazos, presupuestos y exigencias de calidad.

4.3. Resultados del proceso de desarrollo en un proyecto piloto

Para evaluar el proceso de desarrollo propuesto, se decidió aplicarlo en un piloto y con este fin fue seleccionada la segunda versión del Portal de Nova, proyecto que surge con el objetivo de satisfacer las expectativas de información, comunicación y servicios de la comunidad de usuarios de la distribución cubana de GNU/Linux, que lleva por nombre Nova.

La propuesta fue aplicada al 100% y el portal de Nova fue liberado por Calisoft sin detectar ningún error en su construcción y funcionamiento. Los resultados de las pruebas realizadas arrojaron que se obtuvo un producto de calidad, donde la misma está asegurada al garantizar la calidad del proceso, ya que los componentes que se reutilizan ya han sido correctamente evaluados antes de ser almacenados en el repositorio. Además la calidad del expediente de proyecto fue excelente lo cual evidencia una mejora considerable en la preparación y el conocimiento de los desarrolladores y el líder del proyecto.

Finalmente durante el despliegue se evidenció que los artefactos generados durante esta fase resultaron satisfactorios, tanto para el cliente como para el personal del proyecto encargado de ejecutar el mismo.

Además por su importancia, cabe destacar que durante la puesta en práctica, se observaron los siguientes resultados:

- ✓ Se produjo una mejora considerable en la calidad de la captura de los requisitos logrando un mejor entendimiento y modelación del sistema.
- ✓ Se logró reducir el tiempo de diseño del sistema, ya que la cantidad y complejidad de actividades y artefactos que genera la presente propuesta resulta mucho menor que lo propuesto por RUP.
- ✓ Se logró implementar satisfactoriamente con los artefactos generados durante las fases previas a la implementación. Esto demuestra que es posible construir un portal partiendo solamente de la definición de los requisitos del sistema, el modelo de casos de uso del sistema, la arquitectura de software y el documento de arquitectura de la información.

Todo lo anteriormente expresado, unido a la gran cantidad de componentes reutilizables empleados durante la construcción del portal, contribuyó a que el tiempo

de desarrollo del mismo fuese de cuatro meses. En base a este resultado se realizó un primer análisis con respecto al tiempo promedio del resto de los proyectos desarrollados en la fábrica de portales con el anterior modelo, que fue de seis meses y medio, lo cual arroja una diferencia de dos meses y medio con respecto al portal de Nova. Dichos proyectos fueron: Intranet de la Facultad #10, La papeleta, Intranet de las Residencias de Protocolo del Consejo de Estado, Sitio del Instituto Cubano de Amistad con los Pueblos (ICAP), Oficinas de Historia del Consejo de Estado (OHCE), Sitio para el registro de proveedores para el Centro para la Promoción del Comercio Exterior de Cuba (CEPEC).

En un segundo análisis se tomaron de los proyectos mencionados anteriormente, aquellos cuyo tamaño, complejidad y equipo de desarrollo fuesen similares al portal de Nova y se realizó una comparación entre ellos en cuanto al tiempo de desarrollo de cada uno. Esto permite comparar varios portales con características similares, pero usando procesos de desarrollo diferentes, evaluando así con mayor precisión el impacto del proceso de desarrollo propuesto. Como resultado se obtuvo la gráfica que se muestra en la figura 4.1, la cual evidencia una notable diferencia a favor del portal de Nova, desarrollado con la presente propuesta.

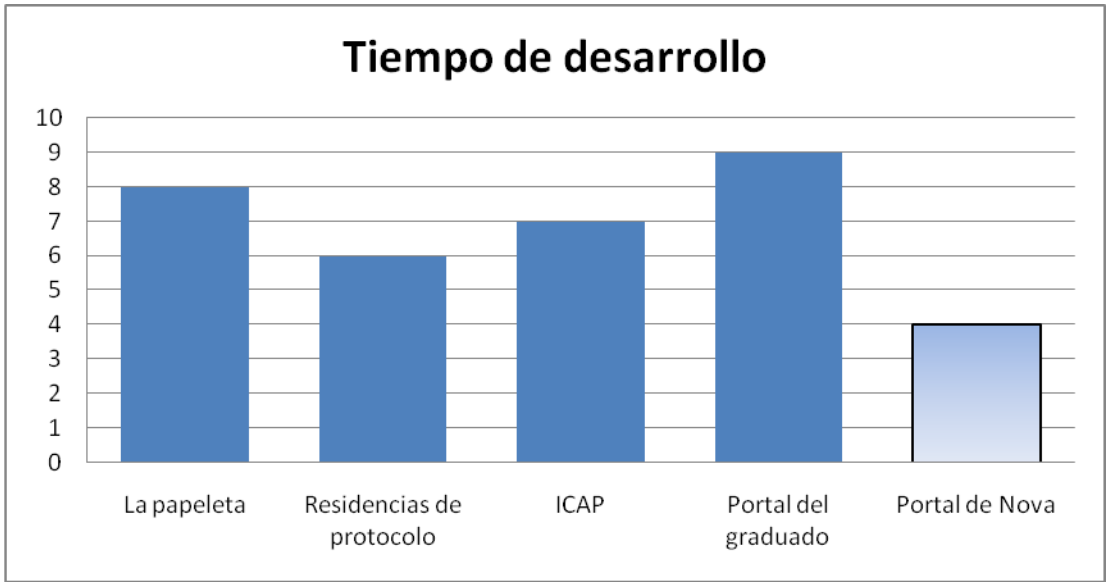


Figura 4.1 Tiempo de desarrollo de los proyectos de portales Web

4.4. Resultados de la implantación del modelo Aplicando Inteligencia en la fábrica de portales.

Para la implantación del modelo Aplicando Inteligencia en la fábrica de portales de la Facultad #10 de la UCI se utilizó la estrategia de implantación propuesta por Santana (2010). Dicha estrategia fue conformada específicamente para la implantación de este modelo en dicha área. En base a esta estrategia se determinó que es posible implantar, todas las entidades de este modelo, excepto la entidad "Inteligencia", según Trujillo (2007), autora de este modelo, recomienda que la entidad Inteligencia" debe implantarse luego que la factoría tenga al menos un año de experiencia, en aras de ganar en organización, madurez y experiencia entre las entidades y miembros de la factoría.

Resultados luego de la implantación:

Se pudo constatar que luego de implantar el modelo, se hizo notable un considerable aumento de un 32%, con respecto al conocimiento de las responsabilidades del Rol de cada miembro del proyecto.

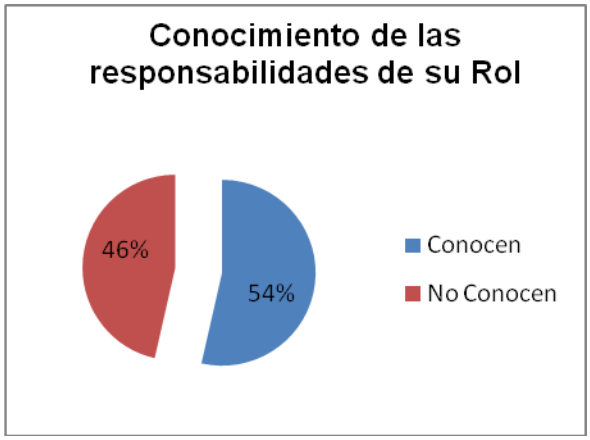


Figura 4.2 Conocimiento de las responsabilidades de su Rol (Antes)

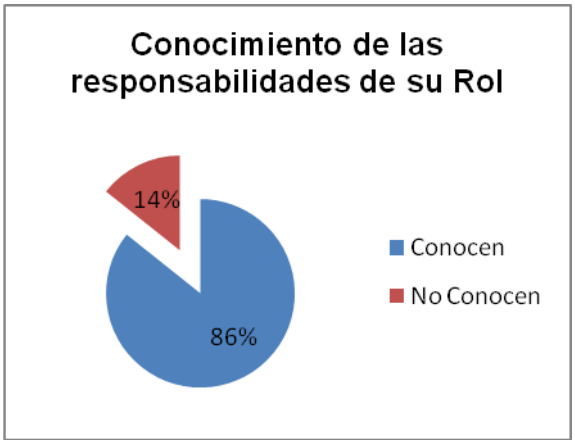


Figura 4.3 Conocimiento de las responsabilidades de su Rol (Después)

El conocimiento por parte de los integrantes del proyecto con respecto a los estándares definidos por el mismo aumentó en un 36%, mejorando así la legibilidad del trabajo y la comunicación.

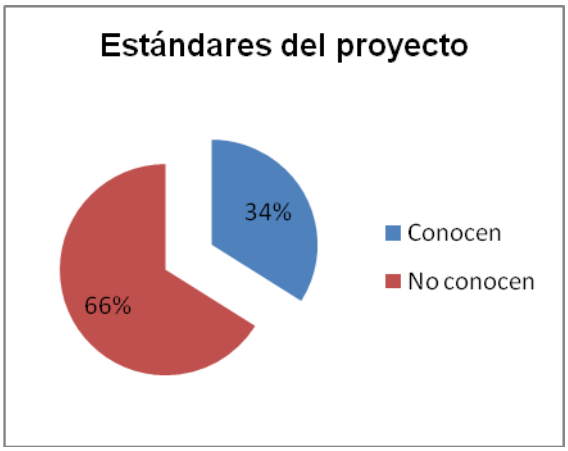


Figura 4.4 Conocimientos de estándares (Antes)

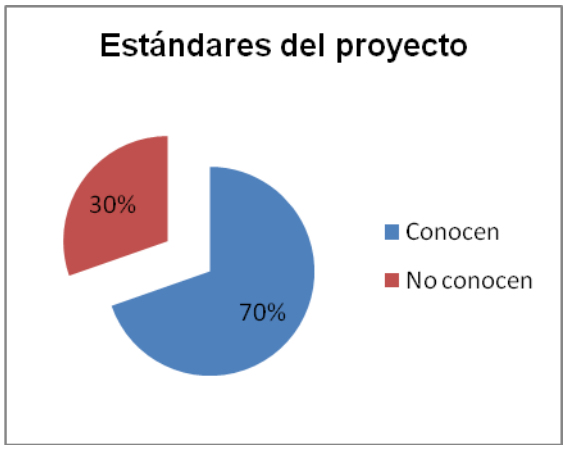


Figura 4.5 Conocimientos de estándares (Después)

Con respecto a la utilización de TSP y PSP por parte de los miembros del proyecto se registró un aumento del 89%. Este crecimiento permitió controlar y planificar mejor el trabajo, además de brindar la posibilidad de que todo el trabajo sea evaluado por los líderes de tarea productiva y por el líder del proyecto.

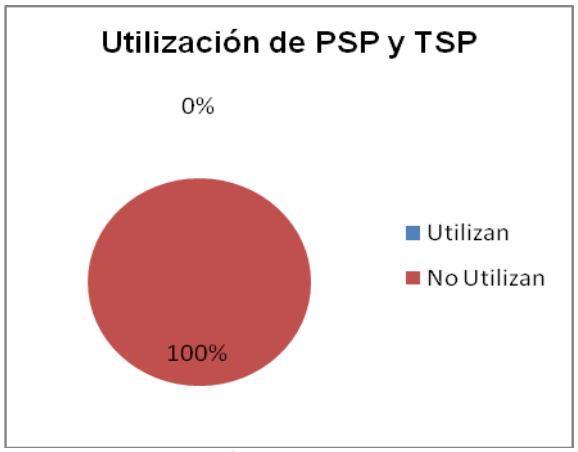


Figura 4.6 Utilización de PSP y TSP (Antes)

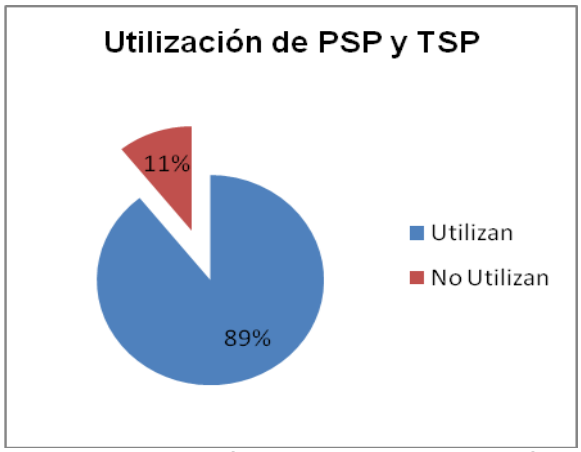


Figura 4.7 Utilización de PSP y TSP (Después)

Por otra parte también se produjo un aumento significativo del 27% en cuanto al conocimiento y dominio de las bases tecnológicas establecidas por el proyecto.

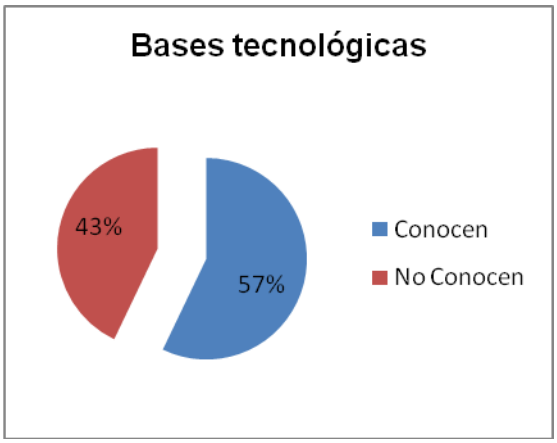


Figura 4.8 Conocimiento de herramientas (Antes)

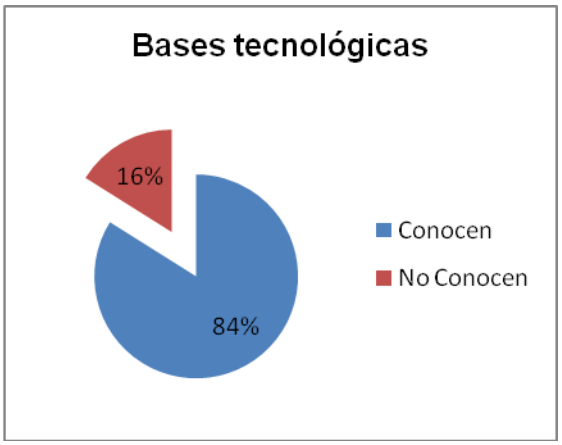


Figura 4.9 Conocimiento de herramientas (Después)

La reutilización de componentes fue otro de los aspectos que se mejoró considerablemente, con un aumento del 53%. Este incremento se debe principalmente a la creación de un repositorio de componentes, el cual permite una adecuada organización y accesibilidad de los mismos. El repositorio brinda la posibilidad de que para la construcción de cada portal se pueda llevar a cabo un proceso de cualificación de componentes basado en los requisitos del portal en cuestión; lo cual reduce en gran medida el tiempo necesario para la construcción del mismo.

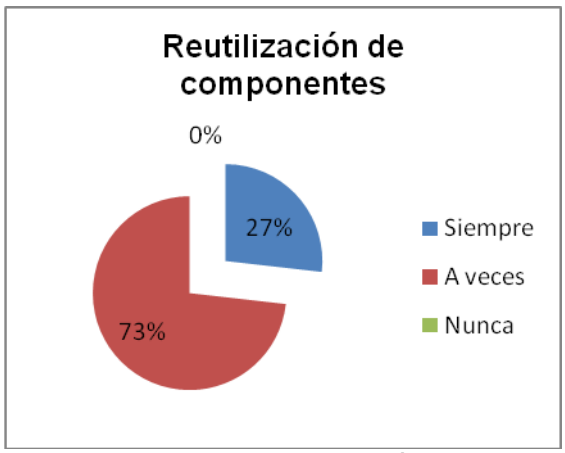


Figura 4.10 Reutilización de componentes (Antes)

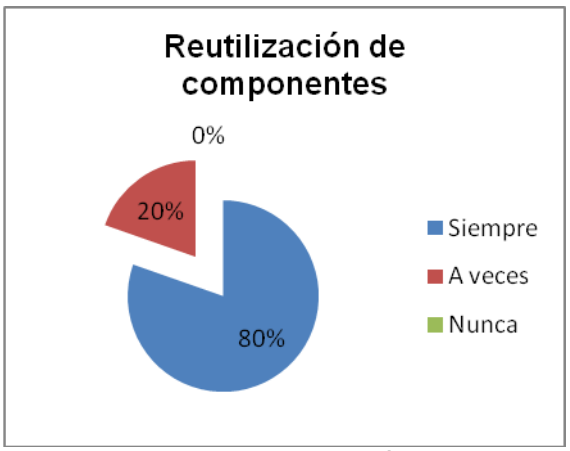


Figura 4.11 Reutilización de componentes (Después)

Conclusiones

- ✓ De los modelos de producción de software analizados durante la investigación, es el de Factoría de Software, el más indicado para ser implantado en la fábrica de portales de la Facultad #10 de la UCI y dentro de los distintos modelos de factoría existentes, el modelo Aplicando Inteligencia es el que agrupa las mejores condiciones.
- ✓ La adopción del enfoque de factoría de software proveyó a la fábrica de portales de una estructura que mejoró la organización, evaluación y control de todos los proyectos.
- ✓ A medida que la factoría gane en madurez va a permitir elevar el grado de especialización de los trabajadores de la misma.
- ✓ El proceso de desarrollo propuesto permitió puntualizar el flujo de trabajo, los roles y las responsabilidades de cada rol, reflejando un considerable aligeramiento y una excelente modelación del producto en cuestión. Además de ganar en organización y estandarización del trabajo.
- ✓ La implantación del modelo Aplicando Inteligencia en la fábrica de portales Web de la Facultad #10, mejora la eficiencia en la producción de portales, ya que siguiendo dicho modelo se logra desarrollar con un menor consumo de tiempo y recursos, disminuyendo así el costo de producción. Por tanto el objetivo general se cumple y se valida la hipótesis, demostrando que se implantó un modelo de producción que responde a las necesidades y características de la actividad productiva en la fábrica de portales de la Facultad #10 de la UCI.

Recomendaciones

Una vez alcanzado un mayor grado de madurez en la factoría, dividir el área de producción en dos sub áreas: la de producción de software y la de componentes.

Referencias bibliográficas

AROUNA WOUKEU, L. C., GARY WILLS, WENDY HALL. *Rethinking Web Design Models: Requirements for Addressing the Content.* University of Southampton. 2007

BASILI, V. R. C., G.; CANTONE. A Reference Archiecture for the Component Factory. ACM Transaction on Software Engineering and Methodology. 1992, vol. 1, nº 1, p. 53-80.

BRAMBILLA, C. T. M. *Semantic personalization of web portal contents*. En *International World Wide Web Conference*. 2007.

CAPRIO, G. *The Software Factory: Refactoring an industry.* 2005,

Calderón, M. (2007). Propuesta de Procedimiento para la Captura de Requisitos en los Proyectos de Portales de la UCI. Ciudad de la Habana, Universidad de las Ciencias Informáticas**:** 53-70.

ESCALONA, M. J. T., J. MEJIAS, M. GUTIERREZ, J.J. VILLADIEGO, D. *The treatment of navigation in web engineering.* 2007,

FACTORY, V. S. *Vector SF afianza su fuerte ritmo de crecimiento y cierra 2007 con un aumento de la facturación del 36%.* 2008,

FERNÁNDEZ, A. A. T., DESCARTES DE SOUZA. *Fábrica de Software:Implementacão e Gestão de Operacões.* 2004,

G. KOUTSOUKOS, L. A., J.L. FIADEIRO, J. GOUVEIA. *Architectural concerns and use of a model-driven development framework.* 2002,

GARCÍA, A. Factoría de software de la empresa Softel. Comunicación personal del Lic. Abdel García, jefe de la factoría de la empresa Softel. La Habana [Consultado: enero de 2010].

GARY WILLS, A. W., LESLIE CARR, WENDY HALL. *The Need for Deeper Design in a Semantic Web*
2009,

GESTIÓN, G. C. D. S. I. Y. Congreso Internacional de Factorías de Software. En *Madrid. 2008*.

GIACHETTI G, M. B., PASTOR O. *Perfiles UML y Desarrollo Dirigido por Modelos: Desafíos y Soluciones para Utilizar UML como Lenguaje de Modelado Específico de Dominio.* 2008,

Greenfield, J. (2004). Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, Microsoft Corporation.

HALBLEIB, H. *Software design using CRC cards.* 1999,

HUMPHREY, W. S. *TSP Principles.* 2003,

INDRA. *Convertir INDRA en una empresa global en 2010.* 2008,

INSTITUTE, S. E. *Compilation Data for Projects Using TSP and PSP.* 2007,

JACOBSON, I. G., MARTIN AND JONSSON, PATRIK. *Software reuse: Arquitecture, process and organization for business success.* 1997,

LEÓN, R. A. H. *Curso básico de gestión de proyectos.* 2005.

---. *El Paradigma Cuantitativo de la Investigación Científica.* Editado por: Eduniv. Universidad de las Ciencias Informáticas, 2002,

LI, C. L., H.; LI, M. *A Software Factory Model Based on ISO 9000 e CMM for Chinese Small Organization*. En *Second Asia-Pacific Conference on Quality Software (APAQS'01)*. Hong Kong. 2001.

MARANTE, Y. R. M. *Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour.* 2005.

NORA KOCH , A. K., ROLF HENNICKER *UML-based Web Engineering Approach.* 2006,

O.M.F. DE TROYER, C. J. L. *WSDM: a user centered design method for Web sites.* 2005,

ORIGIN, A. *Newsletter Noticias.* 2008,

PIRES, J. P. A. A. A. R. D. A. M. V. S. A. L. F. *On the Notion of Abstract Platform in MDA Development*. En *In Eighth IEEE International Conference on Enterprise Distributed Object Computing*. California, USA. September.2004.

PRESSMAN, R. S. *Ingenieríal del Software. Un enfoque práctico.* 2005,

ROCKWELL, R. G., M. H. *The Eureka Software Factory CoRe: A Conceptual Reference Model for Software Factories. Software Engineering Environments Conference.* 2003,

SMS, G. *Grupo SMS patrocinador del Congreso Internacional de Factorías de Software.* 2008,

STEPHEN J. MELLOR, A. N. C., TAKAO FUTAGAMI. *Model-Driven Development.* IEEE Software. 2003

TEAM, C. P. *CMMI for Development, Version 1.2.* 2006

TECNOLOGÍA, M. Y. *Métodos y Tecnología reúne por primera vez en Madrid a los expertos mundiales del lenguaje TTCN-3.* 2008,

Trujillo, Y. (2007). Modelo de factoría de software aplicando inteligencia Universidad de las Ciencias Informáticas.

Tolba, A. M. R. *Integrating V-Model into the Web development process.* 2009,

VALENCIA, R. *SOFTTEC amplia su presencia.* 2007,

Bibliografía

Anaheim (2003) Generative Techniques in the context of MDA.

Arouna Woukeu, L. C., Gary Wills, Wendy Hall (2007). Rethinking Web Design Models: Requirements for Addressing the Content. A. Intelligence, Multimedia Group. Department of Electronics and Computer Science, University of Southampton.

BASILI, V. R. C., G.; CANTONE (1992). "A Reference Archiecture for the Component Factory. ACM Transaction on Software Engineering and Methodology." **1**(1): 53-80.

Brambilla, C. T. M. (2007). Semantic personalization of web portal contents. International World Wide Web Conference.

Calderón, M. (2007). Propuesta de Procedimiento para la Captura de Requisitos en los Proyectos de Portales de la UCI. Ciudad de la Habana, Universidad de las Ciencias Informáticas**:** 53-70.

Caprio, G. (2005) The Software Factory: Refactoring an industry.

CORITEL (2008) CORITEL líder de sistemas y tecnología.

De Troyer, O. (1998). WSDM: A User Centered Design Meted for Web Sites. 7th World Wide Web Conference. Brisbane, Australia.

Diana Marcela Sánchez, J. M. C., Esperanza Marcos (2005) Ontologías y MDA: una revisión de la literatura.

Eisenecker, K. C. a. U. W. (2000) Generative Programming - Methods, Tools, and Applications.

Escalona, M. J. T., J. Mejias, M. Gutierrez, J.J. Villadiego, D (2007) The treatment of navigation in web engineering.

F. García, J. B., M. Laguna y J. Marqués (2002). Líneas de Productos, Componentes Frameworks y Mecanos, Universidad de Salamanca.

FACTORY, V. S. (2008) Vector SF afianza su fuerte ritmo de crecimiento y cierra 2007 con un aumento de la facturación del 36%.

FERNÁNDEZ, A. A. T., DESCARTES DE SOUZA (2004) Fábrica de Software:Implementacão e Gestão de Operacões.

G. Koutsoukos, L. A., J.L. Fiadeiro, J. Gouveia (2002) Architectural concerns and use of a model-driven development framework. ATX Software S.A

GARCÍA, A. Factoría de software de la empresa Softel. Comunicación personal del Lic. Abdel García, jefe de la factoría de la empresa Softel. La Habana [Consultado: enero de 2010].

Gary Wills, A. W., Leslie Carr, Wendy Hall (2009) The Need for Deeper Design in a Semantic Web

Gestión, G. C. d. S. I. y. (2008). Congreso Internacional de Factorías de Software, Madrid.

Giachetti G, M. B., Pastor O (2008) Perfiles UML y Desarrollo Dirigido por Modelos: Desafíos y Soluciones para Utilizar UML como Lenguaje de Modelado Específico de Dominio.

Greenfield, J. (2004). Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, Microsoft Corporation.

Group, O. M. (2003) MDA Guide v1.0.1.

Group, O. M. (2003) Model Driven Architecture Guide.

Halbleib, H. (1999) Software design using CRC cards.

Hennicker, R., Koch, N. (2000). A UML-based Methodology for Hypermedia Design. Lecture Notes in Computer Science. Proc. UML'2000. York, England.

Humphrey, W. S. (2003) TSP Principles.

Ibar Jacobson, G. B., James Rumbaugh (1999). El Proceso Unificado de Desarrollo de Software, Rational Rose Corporation.

INDRA (2008) Convertir INDRA en una empresa global en 2010.

INSA (2008) Inauguración del CENIT de Aldeatejada.

Institute, S. E. (2007) Compilation Data for Projects Using TSP and PSP.

J. Fabri, A. P., L. Begosso, A. L´Erário, F. Fujii y M. de Paula (2004). Techniques for the Development of a Software Factory: Case CEPEIN-FEMA. 17th International Conference Software and Systems Engineering and their Applications. ICSSEA. Paris.

J. GREENFIELD, K. S., S. COOK, Y S. KENT (1997). Software Factories. Assembling Applications with Patterns, Models Frameworks and Tools, Wiley.

JACOBSON, I. G., MARTIN AND JONSSON, PATRIK (1997) Software reuse:

Arquitecture, process and organization for business success.

Juan Bernardo Quintero, R. A. (2007). MDA y el papel de los modelos en el proceso de desarrollo de software. Escuela de Ingeniería de Antioquia**:** 131-146.

León, R. A. H. (2002). El Paradigma Cuantitativo de la Investigación Científica. EDUNIV, Universidad de las Ciencias Informáticas.

León, R. A. H. (2005). Curso básico de gestión de proyectos.

LI, C. L., H.; LI, M (2001). A Software Factory Model Based on ISO 9000 e CMM for Chinese Small Organization. Second Asia-Pacific Conference on Quality Software (APAQS'01). Hong Kong.

M.J. Escalona, M. M., J.Torres, A. Reina (2007). Desarrollo de la navegación en entornos web. Talleres de las Jornadas de Ingeniería del Software y Bases de Datos (TJISBD). Zaragoza. Spain. 1.

MARANTE, Y. R. M. *Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour.* 2005.

Milano, P. d. (2001) WebML User Guide versión 3.0.

N. Lim, S. J., y F. Pavri (1999). Diffusing Software-based Technologies with a Software Factory Approach for Software Development. A Theoretical Framework. Enterprise Architectures for Virtual Organisations, Keuruu, Finland.

Nora Koch , A. K., Rolf Hennicker (2006) UML-based Web Engineering Approach.

O.M.F. De Troyer, C. J. L. (2005) *WSDM: a user centered design method for Web sites*.

Olsina, L. (1998). Building a Web-based information system applying the hypermedia flexible process modeling strategy. 1st International workshop on Hypermedia Development, Hypertext.

ORIGIN, A. (2008) Newsletter Noticias.

PATTERNS, P. C. Y. L. N. S. P. L. P. A. *Software Product Lines. Practices and Patterns.* Addison Wesley, 2001, vol. Segunda Edición, 29-31 p.

Pires, J. P. A. A. a. R. D. a. M. v. S. a. L. F. (2004). On the Notion of Abstract Platform in MDA Development. In Eighth IEEE International Conference on Enterprise Distributed Object Computing. California, USA.

Pressman, R. S. (2005). Ingenierírial del Software. Un enfoque práctico.

R. Hennicker, N. K. (Junio 2001). A UML-based Methology for hypermedia Design. First international workshop on Web-Oriented Software Technology, Valencia.

Robertson, J. (2003) *Looking towards the future of CM*

ROCKWELL, R. G., M. H (2003) The Eureka Software Factory CoRe: A Conceptual Reference Model for Software Factories. Software Engineering Environments Conference.

S.Staab , J. A., S.Decker, M.Erdmann, A.Hotho, A.Maedche, H.-P.Schnurr, R.Studer, Y.Sure (2002) Semantic Community Web Portals.

Santana, W. (2010) Estrategia de implantación para una factoría de software.

Schmidt, D. C. (2006) Model Driven Engineering.

Schwabe, D. y. R., G (1999). The Object-Oriented Hypermedia Design Model (OOHDM)

SEI, S. E. I. (2007). Compilation Data for Projects Using TSP and PSP.

SMS, G. (2008) Grupo SMS patrocinador del Congreso Internacional de Factorías de Software.

Sommerville, I. (2002) Ingeniería de Software.

STEPHEN J. MELLOR, A. N. C., TAKAO FUTAGAMI (2003). Model-Driven Development, IEEE Software.

Team, C. P. (2006). CMMI for Development, Version 1.2.

tecnología, M. y. (2008) Métodos y Tecnología reúne por primera vez en Madrid a los expertos mundiales del lenguaje TTCN-3.

Trujillo, Y. (2007). Modelo de factoría de software aplicando inteligencia Universidad de las Ciencias Informáticas.

Tolba, A. M. R. *Integrating V-Model into the Web development process.* 2009,

Valencia, R. (2007) SOFTTEC amplia su presencia.

Voigt, B. J. J. (2004) Dynamic System Development Method.

Anexos

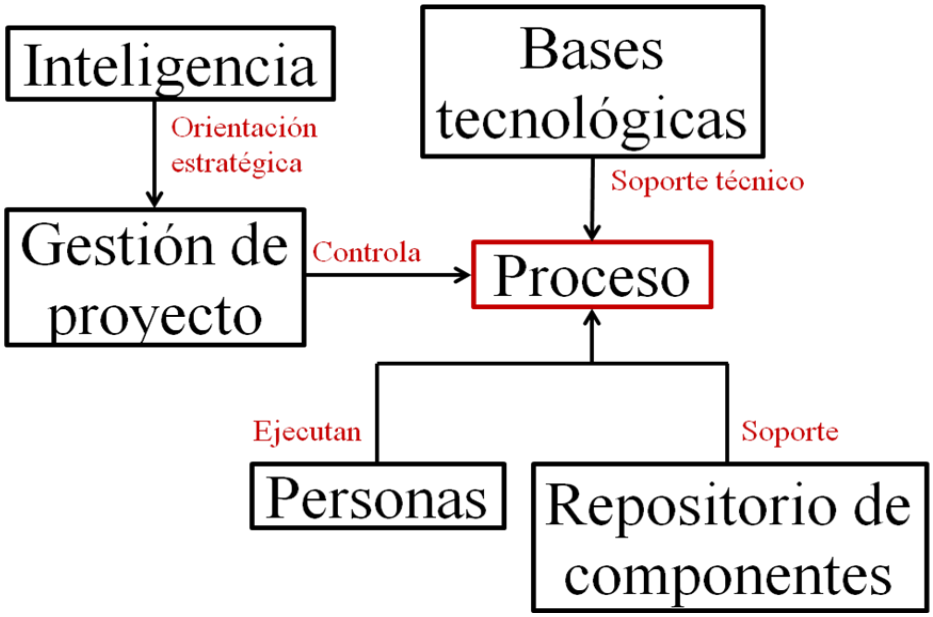
Anexo #1. Épocas fundamentales en la evolución de la estrategia en las factorías de software

Época	Estrategia
1970-1980	. Diseño de estructuras que soporten el proceso de desarrollo de software. . Construcción de un Software Work Bench (SWB) para las actividades del proceso de desarrollo y establecer una organización que controle y monitorice el proceso. . Además se destacan otras disciplinas de gestión de proyectos y calidad, metodologías, programas de formación, etc.
1990-1999	Fábricas basadas en: . Entornos de Desarrollo Integrados, el contexto lo constituyen grandes empresas europeas. . La experiencia, como el SEL (Software Engineering Laboratory) de la NASA, el SEC (Software Experience Center) de DaimlerChrysler o el EPIK (Engineering Process Improvement and Knowledge Sharing) de ICL. . La madurez de procesos, el contexto de esta aproximación lo constituye el modelo CMM. . La reutilización, basadas en familias de soluciones relacionadas. . Otras técnicas de gestión de la calidad, como TQM (Gestión de Calidad Total), generadores de código y herramientas CASE.
2000 - actualidad	Desarrollo basado en componentes, el desarrollo dirigido por modelos y las líneas de producto software. En línea con las recientes propuestas sobre el modelo de fábrica de software para organizaciones chinas, que consideran: Fábrica Software = (Especificaciones de Gestión, Líneas de producto) x (Procesos, Personas, Técnicas)

Anexo #2. Modelos de producción de software

Modelo	Características	Ventajas	Desventajas
Modelos tradicio-nales	-Se basan en lenguajes de programación y la generación de documentos. -Los modelos se utilizan principalmente como representaciones directas de las entidades del software en desarrollo (clases y tipos de datos de algún lenguaje de programación concreto, referencias entre clases, etc.).		-Sensibles a los cambios en el ambiente de desarrollo y en el dominio de aplicación. -Dependientes de la infraestructura tecnológica.
MDA	-Utiliza modelos como el recurso principal en el proceso de desarrollo, generando directamente los sistemas de software a partir de modelos de dicho sistema.	-Define la tecnología que recomienda utilizar para aplicar su enfoque. -Cuenta con lenguajes ampliamente conocidos que, en gran parte, no necesitan de descripción y disponen de abundante documentación. -Cuenta con varias herramientas disponibles (OptimalJ, Arcstyler, etc.)	-No define técnicas a utilizar, ni fases del proceso, ni ningún tipo de guía metodológica. Sólo proporciona la infraestructura tecnológica y conceptual con la que construir los modelos.
Factoría de software	-Se centran en el desarrollo de sistemas similares promoviendo la reutilización de arquitecturas, componentes software y conocimiento.	-Proporcionan una completa guía metodológica a los desarrolladores. Se definen de forma más clara y precisa los pasos que deben llevarse a cabo para construir un método que siga este paradigma. -La cantidad de información para la construcción de métodos que proporciona es mayor que la proporcionada por MDA. -Integra muchas áreas de la Ingeniería del Software que ya han sido investigadas y puestas en práctica (líneas de productos, patrones de diseño, construcción de frameworks, etc.).	-No propone ninguna pauta sobre cómo se deberían implementar los sistemas. -Las herramientas específicas del dominio, que propone Microsoft, todavía se encuentran en fase de desarrollo, por lo que no son muy robustas y no pueden aplicarse en entornos de producción industrial.

Anexo #3. Modelo de Factoría de Software Aplicando Inteligencia



Anexo #4. Diseño de la entrevista #1

- ✓ ¿Considera que en la fábrica de portales de la Facultad #10 hay un modelo de producción de software?
- ✓ ¿Considera que se necesita uno?
- ✓ ¿Cuáles elementos cree vitales en ese modelo?
- ✓ ¿Considera que la producción de software debe responder a características como?:
    - Proceso definido y estandarizado para el desarrollo de software basado en una metodología y con el uso de los principios de la industrialización.
    - Control y almacenamiento en bibliotecas de componentes de software (documentos, código, métodos, etc.).
    - Producción de software fuertemente basada en métodos y técnicas estandarizadas.
    - Estimación de costos y tiempo basados en el conocimiento real de la capacidad productiva, mediante métodos de obtención basados en datos históricos.
    - Producción a gran escala con productos de diferentes magnitudes
- ✓ ¿Sabe que el enfoque de factoría de software responde a esas características?

- ✓ ¿Qué cree si se propone un modelo de factoría de software que además de responder a esas características permita?:

    - Uso de la inteligencia para la orientación estratégica a corto, mediano y largo plazo.
    - Gestión de proyecto, de la calidad y de los recursos.
    - Definición del mapa de proceso y estructura organizacional basado en una metodología y en estándares.
    - La producción basada en componentes donde exista un área de producción de software y otra de componentes.
    - El uso de estándares como CMMI, PSP y TSP, ISO.
    - La definición de reglas que permitan la coordinación de cada una de las personas que intervienen en el proceso y el ensamblaje de cada uno de los componentes.
    - La clasificación de las factorías según el alcance.

Anexo #5. Diseño de la encuesta #1

Cuestionario

Con este cuestionario pretendemos identificar potencialidades y deficiencias en el proceso productivo de la fábrica de portales Web de la Facultad #10. Le pedimos sinceridad a la hora de responder las preguntas, le aseguramos confidencialidad y anonimato a su respuesta.

1. ¿Conoce en qué consiste su proyecto y qué objetivos tiene?

   Sí ___ No ___

En caso afirmativo explique en qué consiste su proyecto y que objetivos persigue el mismo.
_____
_____
_____

2. ¿Qué rol desempeña en el proyecto? _____

a. ¿Conoces las funciones de tu rol?

   Sí ___ No ___

En caso afirmativo responda: ¿Cuáles son?
_____
_____
_____

3. ¿Realiza usted una planificación de las tareas de acuerdo al cronograma del proyecto?

Siempre __ A veces __ Nunca__

En caso de que realice alguna planificación responda: ¿La planificación la realiza individual o en colectivo?

Individual __ Colectivo __

a. ¿Aplican técnicas como PSP y TSP?
Sí ___ No ___

En caso afirmativo responda: ¿Cuál utilizan?

TSP___ PSP___ Ambas___ Otras _____

4. Teniendo en cuenta que las cuestiones técnicas importantes para el desarrollo del sistema y las tecnologías sobre las que el sistema va a ser implantado. Responda las siguientes preguntas:

a. ¿Se definen las bases tecnológicas?
Sí_____ No_____

En caso afirmativo responda:

Lenguaje de Programación	
Entorno Integrados de Desarrollo (IDE)	
Sistema Gestor de Bases de Datos	
Sistema Operativo	
Sistema de Control de Versiones	
Sistema de Gestión Documental	
Herramienta de Modelado	
Herramienta de Gestión de Proyecto	
Estándares de Información	
Sistema para Seguimiento de Errores	
Framework o Componentes	

b. ¿Se definen los estándares a utilizar en el proyecto?
Sí____ No____
c. ¿Se reutilizan componentes para la continuidad del trabajo en otras etapas del proyecto o en proyectos futuros?
Siempre____ Nunca____ A veces____

d. ¿Los componentes realizados se almacenan en un repositorio?
Sí____ No____

5. ¿Utilizan alguna metodología para modelar el sistema?

   Sí____ No____

En caso afirmativo responda las siguientes preguntas:

a. ¿Cuál utilizan? _____
b. ¿Es representativa toda la documentación que genera esta metodología?

   Sí____ No____

En caso que sea negativo explique su respuesta

_____
_____
_____
_____

Anexo #6. Guía para informar el peso de los criterios

Modelo No. 1
Guía para informar el peso de los criterios.
Fecha de recepción…__/__/__…….
Fecha de entrega....__/__/__.....

Nombre y Apellidos del evaluador……..

Le otorgará un peso a cada criterio de acuerdo a su opinión y el peso total de cada grupo debe sumar:

- ✓ Grupo No.1…… 20
- ✓ Grupo No.2…… 25
- ✓ Grupo no.3……. 20
- ✓ Grupo No.4……..35

Para que el peso total asignado sea 100.

Grupo No. 1: Criterios de mérito científico

- ✓ Calidad de la investigación.
  Peso.....

- ✓ Novedad científica.
  Peso.....

- ✓ Valor científico de la propuesta.
  Peso.....

- ✓ Aporte científico.
  Peso.....

Grupo No. 2: Criterios implantación

- ✓ Satisfacción de las necesidades de la producción.
  Peso.....
- ✓ Garantía de principios básicos de la Ingeniería de Software.
  Peso.....
- ✓ Uso de estándares de calidad.
  Peso.....
- ✓ Necesidad del empleo del modelo.
  Peso.....

Grupo No.3: Criterios de generalización

- ✓ Atractividad para su uso.
  Peso.....
- ✓ Ventajas del producto, proceso o servicios que se desarrollan.
  Peso.....
- ✓ Adaptabilidad a diferentes entornos de producción de software.
  Peso.....

Grupo No.4: Criterios de impacto

- ✓ Estandarización en la producción.
  Peso.....
- ✓ Reutilización de componentes.
  Peso.....
- ✓ Organización del proceso de producción.
  Peso.....
- ✓ Ventajas competitivas.
  Peso.....
- ✓ Posibilidades de aplicación.
  Peso.....

Anexo #7. Guía para la evaluación

Modelo No. 2
Guía para la evaluación.
Fecha de recepción…__/__/__……
Fecha de entrega....__/__/__.....

Nombre y Apellidos del evaluador.....…

Criterios de medida que se evalúan en una escala de 1 - 5

Grupo No. 1: Criterios de mérito científico

- ✓ Calidad de la investigación.
  Evaluación.....
- ✓ Novedad científica.
  Evaluación.....
- ✓ Valor científico de la propuesta.
  Evaluación.....
- ✓ Aporte científico.
  Evaluación.....

Grupo No. 2: Criterios implantación

- ✓ Satisfacción de las necesidades de la producción.
  Evaluación.....
- ✓ Garantía de principios básicos de la Ingeniería de Software.
  Evaluación.....
- ✓ Uso de estándares de calidad.
  Evaluación.....
- ✓ Necesidad del empleo del modelo.
  Evaluación.....

Grupo No.3: Criterios de generalización

- ✓ Atractividad para su uso.
  Evaluación.....
- ✓ Ventajas del producto, proceso o servicios que se desarrollan.
  Evaluación.....
- ✓ Adaptabilidad a diferentes entornos de producción de software.
  Evaluación.....

Grupo No.4: Criterios de impacto

- ✓ Estandarización en la producción.
  Evaluación.....
- ✓ Reutilización de componentes.
  Evaluación.....
- ✓ Organización del proceso de producción.

Evaluación.....

- ✓ Ventajas competitivas.
  Evaluación.....
- ✓ Posibilidades de aplicación.
  Evaluación.....

Categoría final del proyecto

___ Excelente: Alta novedad científica, con aplicabilidad y resultados relevantes.

___ Bueno: Novedad científica, resultados destacados.

___ Aceptable: Suficientemente bueno con reservas.

___ Cuestionable: No tiene relevancia científica y los resultados son malos.

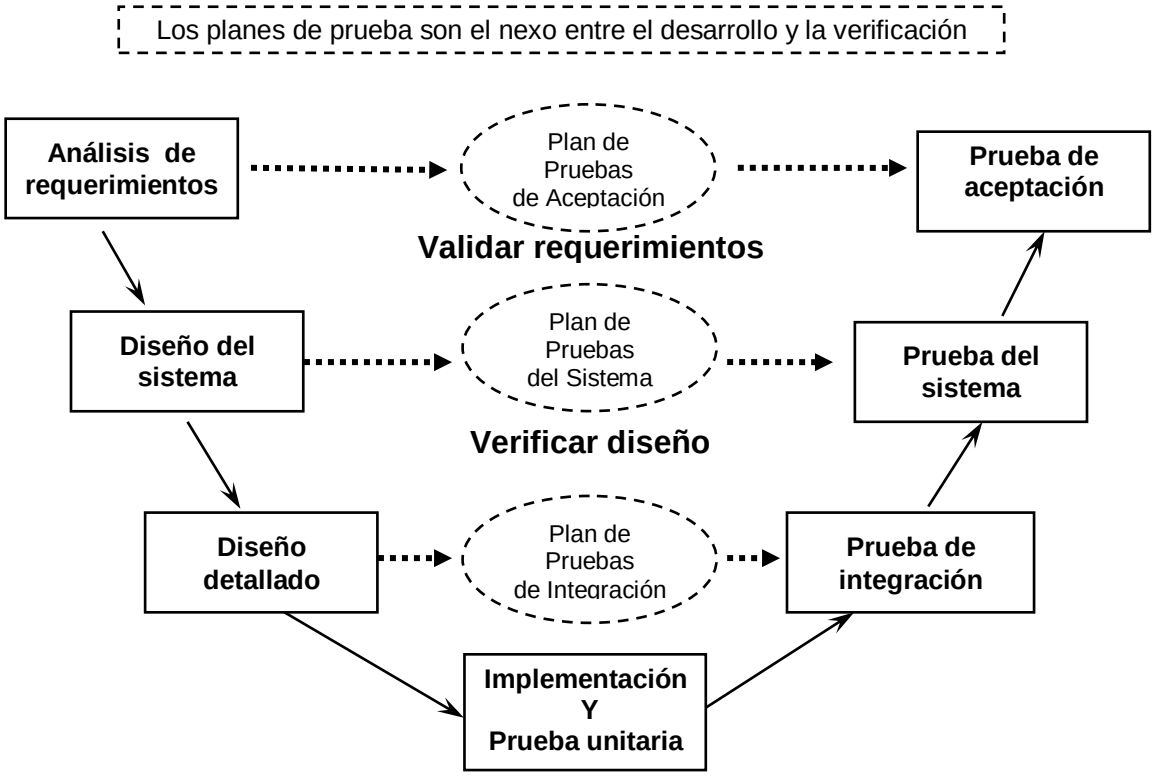
___ Malo: No aplicable.

Valoración final.

Sugerencias del evaluador para mejorar la calidad del proyecto.

Elementos críticos que deben mejorarse.

Anexo #8. Modelo V



Anexo #9. Tabla Informe del Levantamiento de Información para la Arquitectura de Información.

1. Introducción

1.1 Propósito

1.2 Alcance

1.3 Definiciones, Acrónimos y Abreviaturas

1.4 Referencias

[Lista de documentos a los que se hace referencia]

Código	Título
[1]	Documento 1

2. Definición de los objetivos

[Los objetivos del producto serán especificados en forma de texto con viñetas, se recomienda la utilización de verbos que indiquen las acciones a realizar]

3. Definición de la audiencia

3.1 Clasificación de la audiencia

[Se determina cuál es el público al que estará orientado el producto organizado por categorías. Deberán tenerse en cuenta características como capacidad física, capacidad técnica, conocimientos, necesidades de información y ubicación geográfica]

3.2 Necesidades de la audiencia

[Especifica qué información relevante se le puede proporcionar a través del producto a cada grupo de usuarios]

3.3 Expectativas de la audiencia

[En este punto se explica qué espera el usuario del producto aún cuando no lo tenga definido concretamente como una necesidad de información, de aquí pudieran salir posibles temas para banners, secciones, servicios]

4. Definición de los contenidos y servicios

[Se definen todos los contenidos y servicios que el producto va a tener de acuerdo a sus objetivos y necesidades de la audiencia, dejándolo establecido en el siguiente inventario de contenidos]

4.1 Inventario de contenidos

Categoría	Nombre	Formato	Actualización	Disponibilidad	Responsable
Fuentes					
Servicios					
Sistemas					

5.Anexos

Anexo #10. Arquitectura de información

1. Introducción

1.1 Alcance

1.2 Definiciones, Acrónimos y Abreviaturas

1.3 Referencias

2. Esbozo de la estructura o taxonomía

[Representación simple de la estructura del portal en cuanto a etiquetas y jerarquía de los contenidos

Inicio

1.1 Quiénes somos

1.1.1 Nuestra historia

1.1.2.Nuestro Personal

1.1.2.1 Biografías del personal

1.2 Qué hacemos

1.2.1 Productos

1.2.2 Servicios

2.1 Descripción de los elementos de la arquitectura

[Descripción textual de cada uno de los elementos de la estructura, características, comportamiento. Etc]

3. Definición de la estructura

3.1 Mapa de navegación

[Representación en forma de árbol de secciones, niveles y contenidos relacionados]

3.1.1 Elementos del sistema de navegación

[Listado de los elementos del Sistema de Navegación]

3.2 Diseño de la estructura de las pantallas tipo

[Representación lineal de cada uno de los elementos que componen las pantallas tipo, con el objetivo de verificar la ubicación de cada uno de ellos.]

3.2.1 Descripción de los elementos que componen las pantallas

[A cada elemento se le asigna un número en la pantalla y se describe la funcionalidad de ese grupo de contenido, en los casos donde se realicen transacciones se debe incluir un diagrama de flujo sencillo que ejemplifique las posibles interacciones y sus resultados con las pantallas correspondientes]

Anexo #11. Plan de gestión de requisitos

1. Introducción

1.1 Propósito

1.2 Alcance

1.3 Definiciones, Acrónimos y Abreviaturas

1.4 Referencias

[Lista de documentos a los que se hace referencia]

Código	Título
[1]	Documento 1

2. Gestión de Requisitos

2.1 Organización, responsabilidades e interfaces

[Describe quien es responsable del desarrollo de las actividades descritas en el flujo de trabajo de requisito]

2.2 Herramientas, ambientes e infraestructura

[Describe las herramientas de software que serán utilizadas para la gestión de los requisitos. Describe las herramientas y procedimientos que usados para el control de versiones de los Requisitos generados durante todo el ciclo de vida]

3. Programa de Gestión de Requisitos

3.1 Identificación de requisitos

[Describe como los elementos serán nombrados, marcados y numerados. Puede utilizarse la siguiente tabla]

Artefacto (Tipo de documento)	Elemento de traceabilidad	Descripción
Solicitud de los involucrados (STR)	Solicitud de los involucrados (STRQ)	Principales solicitudes de los involucrados

Glosario (GLS)	Términos (TERM)	Términos necesarios para comprender el modelo
Visión (VIS)	Necesidades de los involucrados(NEC)	Principales necesidades de los involucrados y usuarios
Visión (VIS)	Características Básicas (CAB)	Condiciones o capacidades de esta libración del sistema
Modelo de Casos de Uso	Caso de Uso (CU)	Caso de Uso para esta versión del documento
Especificaciones Adicionales (EA)	Requisitos Adicionales (RA)	Requerimientos no funcionales que no fueron capturados en el modelo de CU

3.2 Seguimiento

3.2.1 Criterio de < elemento de seguimiento>

[Para cada elemento se definen un grupo de reglas o guías para aplicar la traceabilidad. Por ejemplo: Toda característica básica aprobad se le asocial uno o más casos de uso o uno o más requisitos complementarios.]

3.3 Atributos

3.3.1 Atributo para < elemento de seguimiento >

[Para cada elemento se identifican la lista de los atributos que serán utilizados para caracterizarlo. Algunos de los atributos pueden ser los siguientes:]

Estado

[Conjunto de posibles estado de un elemento.]

Propuesto	[Se usa para describir características que están en discusión pero que aun no han sido revisadas y aceptadas]
Aprobado	[Característica que ya fue aprobada para su implementación]
Rechazada	[Característica rechazada]
Incorporada	[Característica incorporada en un momento específico del proyecto]

Beneficio

[Que beneficio reporta al negocio]

Crítico	[Es una característica esencial para el negocio]
Importante	[Es una característica importante que no incluirla pudiera afectar la satisfacción de

	los clientes y usuarios]
Útil	[No es una característica que afecte en la satisfacción del cliente]

Esfuerzo
[Los requisitos son diferentes en cuanto al tiempo de desarrollo y otros recursos.]

Bajo	[< 1 día]
Medio	[1 a 20 días]
Alto	[>20 días]

Estabilidad
[% aceptable de cambios que pudiera tener el requisito.]

Bajo	[< 10%]
Medio	[10-50%s]
Alto	[>80 %]

4. Entregables del proyecto
[Una lista tabular de los artefactos que serán creados durante el proyecto, incluso las fechas de entrega designadas.]

5. Gestión de cambios a los requisitos
[Hacer referencia al documento de los procesos de Gestión de Configuración establecidos en la organización]

Anexo #12. Especificación de requisitos de software

1. Introducción
- **1.1 Propósito**
- **1.2 Alcance**
- **1.3 Definiciones, Acrónimos y Abreviaturas**
- **1.4 Referencias**

[Lista de documentos a los que se hace referencia]

Código	Título
[1]	Documento 1

2. Funcionalidad

[Esta sección describe los requisitos funcionales del sistema expresados en lenguaje natural. Típicamente se organiza por características pero también son apropiados métodos alternativos de organización como por ejemplo por usuario o subsistema.]

2.1 <Requisito de dato 1..n>

[Descripción del requisito]

2.2 <Requisito de interfaz 1..n>

[Descripción del requisito]

2.3 <Requisito de navegacional 1..n>

[Descripción del requisito]

2.4 <Requisito de personalización 1..n>

[Descripción del requisito]

2.5 <Requisito funcional 1..n>

[Descripción del requisito]

3. Usabilidad

[Esta sección incluye todos lo requisitos que afectan la usabilidad. Ejemplos:

- *Especificar el tiempo de entrenamiento requerido para que usuarios normales y avanzados sean productivos operando el sistema.*
- *Especificar requisitos acordes con estándares de usabilidad establecidos]*

3.1 < Requisito de Usabilidad 1..n>

[Descripción del requisito]

4. Fiabilidad

[En esta sección se especifican los requisitos relacionados con la Fiabilidad. Ejemplos:

- *Disponibilidad – especificar porciento de tiempo disponible (xx.xx%), horas de uso, acceso para mantenimiento, modo de funcionamiento degradado etc.*
- Tiempo medio entre fallos – usualmente se especifica en horas pero puede también especificarse en términos de días, meses o años.
- Tiempo medio de reparación – Cuanto tiempo está permitido que el sistema quede fuera de operación luego de haber fallado?
- Exactitud – especificar la precisión y exactitud requerida en las salidas del sistema.
- *Máximo de errores – usualmente es expresado en términos de errores/MLC (miles de líneas de código) o errores/puntos de función.*
- *Errores – categorizar los errores en términos de menores, significativos y críticos: los requisitos deben definir que se entiende por error crítico (ej. Pérdida total de los datos o inhabilitadas para el uso ciertas partes del funcionamiento del sistema).]*

4.1 < Requisito de Fiabilidad 1..n>

[Descripción del requisito]

5. Eficiencia

[Deben perfilarse en esta sección las características de la eficiencia del sistema. Incluir los tiempos de respuesta específicos. Donde sea aplicable, hacer referencia a los Casos de Uso por el nombre.

- *Tiempo de respuesta por transacción (promedio, máximo).*

- *Rendimiento (ej. transacciones por segundo, cantidad de datos que pueden ser transferidos en un segundo).*
- *Capacidad (ej. número de clientes o transacciones que el sistema puede alojar).*
- *Modos de degradación (cual es el modo de operación aceptable cuando el sistema de alguna forma ha sido degradado).*
- *Utilización de recursos (memoria, disco, comunicaciones, etc.]*

5.1 < Requisito de Eficiencia 1..n>

[Descripción del requisito]

6. Soporte

[Esta sección indica cualquier requisito que refuerce el soporte o mantenimiento del sistema a construir, incluyendo normas de codificación, convenciones para nombrado, bibliotecas de clase, el acceso y utilidades de mantenimiento.]

6.1 < Requisito de Soporte 1..n>

[Descripción del requisito]

7. Restricciones de diseño

[Esta sección debe indicar cualquier restricción de diseño en el sistema a construir. Las restricciones representan decisiones de diseño que se han tomado y a las cuales es necesario adherirse. (Ej. lenguajes de programación, requisitos de proceso de software, el uso prescrito de herramientas de desarrollo, restricciones de arquitectura y diseño, componentes comprados, las bibliotecas de la clase, etc.]

7.1 < Requisito de Soporte 1..n>

[Descripción del requisito]

8. Requisitos para la documentación de usuarios en línea y ayuda del sistema.

[Describe los requisitos para la documentación de usuarios en línea, la ayuda del sistema, ayuda relacionada con avisos, etc.].

9. Componentes Comprados

[Esta sección describe cualquier componente comprado y a ser usado en el sistema, cualquier licencia aplicable o restricciones del uso, y cualquier compatibilidad/interoperabilidad asociada o estándares de interfaz.]

10. Interfaz

[Esta sección define las interfaces que deben soportadas por la aplicación. Debe contener la especificidad adecuada, protocolos, puertos y direcciones lógicas, etc., para que el software pueda desarrollarse y verificarse contra los requisitos de la interfaz.]

10.1 Interfaces de usuario

[Describe las interfaces de usuario que deben ser implementadas por el software.]

10.2 Interfaces Hardware

[Esta sección define cualquier interfaz del hardware que será soportada por el software, incluyendo la estructura lógica, direcciones físicas, el comportamiento esperado, etc.]

10.3 Interfaces Software

[Esta sección describe las interfaces del software a otros componentes del sistema del software. Éstos pueden ser componentes comprados, componentes reutilizados de otra aplicación o componentes que se desarrollan para subsistemas fuera del alcance de este documento, pero con esta aplicación debe actuar recíprocamente.]

10.4 Interfaces de Comunicación

[Describe cualquier interfaz de comunicaciones a otros sistemas o dispositivos como las redes de área locales, los dispositivos remotos, etc.]

11. Requisitos de Licencia

[Define cualquier requisito de licencia o restricción de uso que serán seguidos por el software.]

12. Requisitos Legales, de Derecho de Autor y otros.

[Esta sección describe cualquier denegación legal necesaria, garantías, notificaciones de derecho de autor, patentes, marca comercial o complacencia con logotipo para el software.]

13. Estándares Aplicables

[Esta sección describe por referencia cualquier norma o estándar aplicable y las secciones específicas que aplicadas al sistema. Por ejemplo, podría incluir estándares legales, de calidad, regulatorios, normas de la industria para la usabilidad, interoperabilidad, internacionalización, integración con el sistema operativo, etc.]

Anexo #13. Modelo de sistema

1. Introducción

1.1 Propósito

1.2 Alcance

1.3 Definiciones, Acrónimos y Abreviaturas

1.4 Referencias

[Lista de documentos a los que se hace referencia]

Código	Título
[1]	Documento 1

2. Actores del Sistema

[Si los requisitos se describen con la técnica de Casos de Uso únicamente, entonces se utilizan los epígrafes 2, 3 y 4 sin variaciones. En los proyectos donde se utilicen otras técnicas de descripción se incorporarán los epígrafes en dependencia de las necesidades específicas]

[Se especifican todos los actores del negocio y se le asocia una descripción simple de cada uno de ellos]

Actor	Descripción

3. Diagrama de Casos de Uso del Sistema

[Figura que ilustre del modelo de casos de uso del sistema]

4. Especificación de los Casos de Uso

4.1 <Primer Caso de Uso del Sistema>

4.1.1 Descripción de Casos de Uso

[Se describe en la tabla los detalles del caso de uso en fusión de acción del actor y respuesta del sistema.
Si se decide tener un documento independiente para la definición de cada Caso de Uso, en esta sección se haría referencia a ese documento]

Caso de Uso:	
Actores:	
Resumen:	
Precondiciones:	
Referencias	
Prioridad	
Flujo Normal de Eventos	
Sección “”	
Acción del Actor	Respuesta del Sistema
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
Prototipo de Interfaz	
Poscondiciones	

Anexo #14. Arquitectura de Software

1. Introducción

1.1 Propósito
1.2 Alcance
1.3 Definiciones, Acrónimos y Abreviaturas
1.4 Referencias

[Lista de documentos a los que se hace referencia]

Código	Título
[1]	Documento 1

2. Ambiente de desarrollo

[En esta sección se definen el conjunto de herramientas y plataformas tecnologías (especificando en cada caso su versión) y así como su integración. En este sentido se han realizado las siguientes clasificaciones que permitan completar correctamente esta sección.]

2.1 Herramientas Horizontales

2.1.1 Sistema operativo

2.1.2 Seguridad (Antivirus, Niveles de Acceso a código fuente, documentación)

2.1.3 Gestión de recursos (Levantamiento tecnológico, asignación de recursos)

2.1.4 Salvas automáticas.

2.1.5 Control de versiones

2.1.6 Gestión de proyecto

2.1.7 Gestión documental

2.1.8 Seguimiento de errores

2.2 Herramientas verticales

2.2.1 Herramientas de modelado

2.2.2 Compilación (Compilador, Maquina Virtual, Interprete)

2.2.3 Prueba (Pruebas Unitarias)

2.2.4 Refactorización y "formateadores" de Código. (Basado en un estándar de codificación)

2.2.5 Sistemas automatizados de compilación y prueba.

2.2.6 Entornos de desarrollo integrados (IDE)

2.2.7 Lenguajes de programación

2.2.8 Framework o Componentes

2.2.9 Servidor de Aplicaciones.

2.2.10 Sistema Gestor de Bases de Datos. (Servidor, Cliente, o Modelo propio de persistencia)

3. Representación Arquitectónica

[Incluir en esta sección una descripción de que es la arquitectura de software para el sistema, y como esta es representada. La descripción de la representación aborda sobre el cómo la arquitectura y el diseño son representados, las convenciones de modelado y los artefactos usados para presentar la información, patrones arquitectónicos, tendencia, clasificación.]

4. Objetivos y Restricciones Arquitectónicas

[Incluir en esta sección una descripción de los requerimientos y objetivos del software que tienen un impacto significativo en la arquitectura.

• *Interoperabilidad con sistemas legados*

• *Uso de estándares de datos*

• *Política de manejo de datos (cifrado)*

• *Distribución geográfica o física de la entidad a automatizar*

• *Otras restricciones que puedan atentar con el cumplimiento de los objetivos.]*

5. Tamaño y Rendimiento

[Incluir en esta sección una descripción las características de dimensiones importantes del software que afectan la arquitectura, así como las restricciones de rendimiento para la puesta a punto.

• *Prever el crecimiento de los datos.*

• *Tiempo de respuesta.*

• *Niveles concurrencia]*

6. Vista de casos de uso

[Incluir en esta sección el modelo de casos de uso y de forma opcional el modelo de casos de uso del negocio]

6.1 Casos de uso arquitectónicamente significantes

[Incluir en esta sección el diagrama de casos de uso arquitectónicamente significantes. Pueden ser añadidas notas al diagrama para explicar porque un caso de uso en específico es arquitectónicamente significante]

6.2 Nombre de caso de uso: Breve descripción.

[Realización del caso de uso: Documentación de la realización del caso de uso, Diagrama de interacción.]

7. Estilos arquitectónicos.

7.1 Patrones arquitectónicos

[Incluir en esta sección los patrones arquitectónicos que se utilizan en la vista lógica, en la misma estarán expresadas las partes correspondientes de cada patrón. Se puede dar una visión general si se quiere hacer el próximo punto menos extenso.]

8. Vista Lógica

[Incluir la descripción de las clases más importantes, su organización en paquetes y subsistemas, y la organización de estos subsistemas en capas. La descripción se realiza a través de diagramas de clases para ilustrar la relación entre las clases arquitectónicamente significantes, subsistemas, paquetes y capas.]

8.1 Elementos del modelo arquitectónicamente significantes

[Incluir un diagrama con los elementos del modelo de diseño que son arquitectónicamente significantes

8.2 Visión general de la arquitectura – Alineamiento de paquetes, subsistemas y capaz

[Incluir en esta sección un diagrama que ilustre la organización del modelo de diseño en capas lógicas e incluir la descripción de cada una de ellas. Enumeración de las tecnologías a aplicar por cada una de las capas y subsistemas.]

9. Vista de procesos

[Incluir en esta sección la descripción de las tareas (procesos, hilos, tareas programadas, eventos y notificaciones) involucrados en la ejecución del sistema. Además de incluir la ubicación de las clases y objetos necesarios para estas tareas de ser necesario.]

9.1 Diagrama de vista de procesos que muestra la composición de los procesos e hilos, y la distribución de clase en estos procesos e hilos.

9.1.1 Nombre del proceso: *[Incluir una descripción sobre el role del proceso en el sistema.]*

9.1.2 Nombre del hilo: *[Este hilo pertenece al proceso 7.1 debe incluirse una descripción del la tarea a realizar por el mismo.]*

10. Vista de despliegue

[Incluir en esta sección la descripción de los nodos físicos para la mayoría de las configuraciones tanto para usuarios finales como para desarrolladores y probadores. Además ubicar las tareas (de la vista de procesos) y las capas lógicas en los nodos físicos. Esta descripción se realiza a través del diagrama de despliegue seguido por la distribución de los procesos y las capas lógicas en cada uno de los procesadores.]

10.1 Diagrama de despliegue.

10.1.1 Nombre de dispositivo: *[Descripción de la capacidad que el dispositivo provee al sistema.]*

10.1.2 Nombre del procesador: *[Descripción de la funcionalidad y capacidad del nodo.]*

10.2 Descripción de elementos e interfaces de comunicación

10.2.1 <<Nombre tipo de conexión>>: Características físicas de la conexión

11. Vista de Implementación

[Incluir en esta sección la colección de componentes y subsistemas de implementación mediante el modelo de implementación (diagrama de componentes). Esta vista define además, ejecutables, bibliotecas, ficheros, subsistemas, dependencias entre ellos.]

12. Vista de Datos

[Incluir en esta sección los elementos persistentes arquitectónicamente significativos en el modelo de datos. Una vista general de la tecnología usada para lograr la persistencia de las entidades del sistema además de la descripción del modelo de datos en términos de tablas, vistas, triggers y procedimientos almacenados.]

13. Calidad

[En esta sección, listar las dimensiones claves de calidad del sistema que forman la arquitectura. La información presentada puede incluir:

- *Requerimientos de rendimiento operacional, tales como tiempo medio entre fallas.*
- *Objetivos de calidad, tales como "apagados no programados".*
- *Objetivos de extensibilidad, tales como "el software puede ser actualizado mientras el sistema está corriendo".*
- *Objetivos de portabilidad, tales como plataformas de hardware, sistemas operativos, idiomas.*

Para cada dimensión, abordar como la arquitectura soporta el requerimiento. Puede organizar esta sección por las diferentes vistas (lógica, implementación,...) o por calidad. Cuando una característica particular es importante para el sistema, por ejemplo seguridad, privacidad, el papel que juega la arquitectura en ese requisito debe ser cuidadosamente delineado en esta sección.]

Anexo #15. Plan de prueba

1. Introducción

1.1 Propósito

1.2 Alcance

1.3 Definiciones, Acrónimos y Abreviaturas

1.4 Referencias

[Lista de documentos a los que se hace referencia]

Código	Título
[1]	Documento 1

2. Roles y responsabilidades
[Descripción del equipo de probadores, por quienes está compuesto, responsabilidad de cada miembro.]

Rol	Cantidad	Responsabilidad
Nombre del rol	Cantidad	Lista de responsabilidades dentro del proceso de revisión

3. Escenario de pruebas.
[Descripción del escenario en el que se ejecutarán las pruebas]

3.1 Despliegue del sistema
[Modelo de despliegue del sistema]
3.2 Recursos del sistema

3.2.1 Servidores

Recurso	TIpo
[Servidor de datos]	*2 procesadores Intel Xeon, 1 GB RAM*

3.2.2 PC Clientes

PC Clientes	
Cantidad	
Descripción	
Software base	
Servicios	

4. Requerimientos a probar
[Listado de requerimientos a probar. Puede hacerse referencia al documento de requerimientos]

5. Estrategia de pruebas de aceptación
[Se describe el flujo de trabajo que se utilizará para la ejecución de las pruebas]

6. Evaluación de las pruebas
[Se describen los criterios de evaluación para las pruebas, como clasificación de las No Conformidades, Pedidos de cambios y listas de chequeo]

7. Cronograma
[Cronograma de ejecución de las pruebas]

No.	Tarea	Fecha	Responsable	Participantes	Observaciones
	Nombre de la tarea				

8. Anexos

Anexo #16. Diseño de casos de prueba

1. Descripción general

[Descripción general del caso de uso]

2. Condiciones de ejecución

[Precondiciones del caso de uso]

3. Secciones

[Para cada sección los escenarios van a ser flujo básico + flujos alternos]

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
[Nombre de la sección 1]	*EC 1.1: Nombre del escenario.*	*[Descripción de la funcionalidad.]*	*[Pasos a desarrollar para probar la funcionalidad que se indicó.]*
	EC 1.n: Nombre del escenario.	*[Descripción de la funcionalidad.]*	*[Pasos a desarrollar para probar la funcionalidad que se indicó.]*
[Nombre de la sección m]	*EC m.1: Nombre del escenario.*	*[Descripción de la funcionalidad.]*	*[Pasos a desarrollar para probar la funcionalidad que se indicó.]*
	EC m.n: Nombre del escenario.	*[Descripción de la funcionalidad.]*	*[Pasos a desarrollar para probar la funcionalidad que se indicó.]*

3.1 SC <Sección a revisar>

ID del escenario	Escenario	Variable 1	Variable 2	Variable n	Respuesta del sistema	Resultado de la prueba
[EC 1]	*[Nombre del escenario]*	*V*	*V*	*V*	*[Se escribe el resultado que se espera al realizar la prueba.]*	*[Se escribe el resultado que se obtiene al realizar la prueba.]*
		V	*V*	*V*		
[EC 2]	*[Nombre del escenario]*	*NA*	*NA*	*NA*	*[El sistema emite un mensaje para que llene los campos obligatorios.]*	*[Se escribe el resultado que se obtiene al realizar la prueba.]*

[Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

4. Registro de defectos y dificultades detectadas

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Resp. equipo de desarrollo
[Nombre]	*[1]*	*[Descripción de la*	*[Descripción del aspecto*	*[Etapa de detección del error]*	*[X]*	*[X]*	*[X]*	*[Se coloca el estado de la NC y*	*[Esta columna se comienza a*

		NC]	*correspondiente]*					*la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.]* *RA: Resuelta* *PD:Pendiente* *NP:No Procede*	*llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]*

5. Anexos

5.1 Anexo 1

Anexo #17. Registro de pruebas unitarias y de integración

1. Introducción

1.1 Propósito

1.2 Alcance

1.3 Definiciones, Acrónimos y Abreviaturas

1.4 Referencias

2. Registro de Prueba Unitaria o Integración.

2.1. <Nombre de Prueba 1…n>

[La estructura del nombre puede ser tan flexible como intuitivo y unívoco sea el nombre (Módulo_EspacioNombre_Clase_Método)]

Prueba de: *[Unidad o Integración]*		
Nombre Prueba:		
Estado: *[Estado actual de la prueba. Ejemplo: Satisfactoria (Cuando arrojó los resultados esperados), Fallida(No arrojó los resultados esperados), Pendiente (Cuando se pospone su ejecución), Cancelada (Cuando se decide no realizarse)]*	**Tipo**: *[El tipo de prueba. Ejemplo: Cargar , Regresión]*	**Última ejecución**: DD/MM/AA *[Fecha en que la prueba se ejecutó por última vez]*

Ejecutado por: *[Nombre de la persona que ejecutó la prueba]*	**Verificado por**: *[Nombre de la persona que verificó la ejecución de la prueba]*
Descripción *[Descripción de Aspectos Generales de la prueba unitaria o de integración ya sean sus características particulares, incidencias en el momento de su desarrollo y otros aspectos relevantes.]*	
Entrada: *[Dato de entrada]*	
Criterio de aceptación: *[Aceptación o condiciones de éxito de las pruebas]*	
Resultado: *[Resultados de la última prueba]*	

Anexo #18. Documento de No conformidades

1. Descripción General

[Descripción de Aspectos Generales a tener en cuenta a la hora de realizar el diseño de las pruebas, incidencias en el momento de su desarrollo y otros aspectos relevantes.]

2. Elementos probados

[Descripción general o lista de los Elementos Probados, y otros aspectos importantes a tener en cuenta a la hora analizar las No Conformidades Detectadas.]

3. Elementos no probados y causas

[Descripción de Aspectos Generales a tener en cuenta a la hora de realizar el diseño de las pruebas, incidencias en el momento de su desarrollo y otros aspectos relevantes.]

4. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	*< 1>*	*<Descripción de la No Conformidad>*	*<Descripción del Aspecto correspondiente>*	*<Etapa de detección del error>*	*S: Significativa* *NS: No Significativa* *R: Recomendación*	*[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.]* *RA: Resuelta* *PD:Pendiente* *NP:No Procede*	*[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]*

[La NC puede tener solo una de las tres clasificaciones: Significativa, No Significativa o Recomendación]

5. Anexos

5.1 Anexo <1>

Anexo#19. Tabla de valores del peso relativo de cada criterio

G	C/E	E1	E2	E3	E4	E5	E6	E7	∑E	Ep
	C1								0	0
	C2								0	0
	C3								0	0
20	C4								0	0
	C5								0	0
	C6								0	0
	C7								0	0
25	C8								0	0
	C9								0	0
	C10								0	0
20	C11								0	0
	C12								0	0
	C13								0	0
	C14								0	0
	C15								0	0
35	C16								0	0
T		0	0	0	0	0	0	0	0	0

Anexo #20. Tabla para el cálculo de la concordancia

G	C/E	E1	E2	E3	E4	E5	E6	E7	∑E	Ep	ΔC	ΔC^2
	C1								0	0	0	0
	C2								0	0	0	0
	C3								0	0	0	0
20	C4								0	0	0	0
	C5								0	0	0	0
	C6								0	0	0	0
	C7								0	0	0	0
25	C8								0	0	0	0
	C9								0	0	0	0
	C10								0	0	0	0
20	C11								0	0	0	0
	C12								0	0	0	0
	C13								0	0	0	0
	C14								0	0	0	0
	C15								0	0	0	0
35	C16								0	0	0	0
T		0	0	0	0	0	0	0	0	0	0	0

Anexo #21. Tabla de calificación de cada criterio

Calificación	Calificación (c)					P	P x C
	1	2	3	4	5		
C1							
C2							
C3							
C4							
C5							
C6							
C7							
C8							
C9							
C10							
C11							
C12							
C13							
C14							
C15							
C16							
TOTAL							

Glosario de términos

Actividad: conjunto de operaciones o tareas propias de una persona o entidad que permite que el trabajo a realizar sea descrito y entendido de manera precisa por aquellos que tienen que ejecutarlo.

Activos del proceso: entiéndase como activos del proceso modelos, patrones, algoritmos utilizados como artefactos en el proceso. Los activos del proceso también pueden ser denominados como componentes de infraestructura, componentes de valor en el proceso.

Arquitectura de software: la arquitectura de software representa la estructura o las estructuras del sistema, que consta de componentes de software, las propiedades visibles externamente y las relaciones entre ellas.

Artefacto: un artefacto es una pieza de información que es producida, modificada o usada por el proceso, define un área de responsabilidad para un rol y está sujeta a control de versiones. Un artefacto puede ser un modelo, un elemento de modelo o un documento.

Componente: un componente es un parte encapsulada de un sistema; idealmente, una parte sustituible, casi independiente y segura de un sistema no trivial que desempeñe una función clara en el contexto de una arquitectura bien definida.

Cultura corporativa: no es otra cosa que el conjunto de valores, actitudes y creencias que comparten los miembros de la organización como consecuencia de las prácticas, procedimientos, estructuras y sistemas establecidos desde hace tiempo.

Equipo de desarrollo: es un grupo de trabajo constituido por una serie de profesores, investigadores, colaboradores y alumnos unidos en la ilusión de acometer un determinado proyecto o avanzar en el conocimiento y en la investigación teórica y aplicada.

Estándar: lo que es establecido por la autoridad, la costumbre o el consentimiento general. En este sentido se utiliza como sinónimo de norma.

***Framework*:** es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede

incluir soporte de programas, librerías y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Herramientas: utensilios o provisiones necesarias para poder emprender un proyecto de software. Soportan los procesos de desarrollo de software modernos.

Hipermedia: se refiere a la existencia de más de un medio (texto, imagen sonido, video, etc.).

Ingeniería de Software: se puede definir como el tratamiento sistemático de todas las fases del ciclo de vida del software.

Mecanismo: manera de producirse una actividad, una función o un fenómeno.

Modelo: un modelo es una representación abstracta de alguna cosa que puede o no existir en la realidad; es una simplificación que muestra ciertos aspectos de lo que se desea modelar, escondiendo aquellos elementos que no son de interés a los que usaran el modelo.

Módulo: los módulos son componentes enchufables que extienden la funcionalidad del núcleo de Drupal.

***Outsourcing*:** sistema utilizado por empresas grandes para rentar los servicios de compañías chicas y efectuar proyectos pequeños en lugar de ellas.

Portal Web: no todos los sitios Web pueden ser considerados como un portal, a pesar de que pueda tratarse de un sitio robusto y con información relevante. Un portal Web funciona como una plataforma de despegue para la navegación de las personas que se conectan al World Wide Web. Se utilizan para localizar la información y otros sitios de interés, la idea consiste en agrupar en el portal toda una serie de información y servicios de forma tal que el usuario obtenga todo lo que busca sin tener que salir del portal.

Proceso: es un conjunto de actividades y resultados asociados que producen un resultado.

Proceso de desarrollo de software: es la definición del conjunto completo de actividades necesarias para transformar los requisitos de un usuario en un producto. Un proceso es una plantilla para crear proyectos.

Producto: conjunto de artefactos que se crean durante la vida del proyecto, como los modelos, código fuente, ejecutables y documentación.

Proyecto: combinación de recursos humanos y no humanos reunidos en una organización temporal para conseguir un propósito, tiene un punto de de comienzo definido y con objetivos definidos mediante los que se identifican.

Proyecto de Software: el elemento organizativo a través del cual se gestiona el desarrollo de software. El resultado de un proyecto es una versión de un producto.

Rol: un rol define un conjunto de habilidades, competencias y responsabilidades que están relacionadas.

Sistema de gestión de contenido (*Content Management System*, en inglés, abreviado CMS): puede considerarse como un software que facilita la creación, mantenimiento, publicación y presentación de contenidos principalmente en páginas Web. En esencia consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. Permite manejar de manera independiente el contenido y el diseño del sitio; además de admitir la fácil y controlada publicación en el sitio a varios editores.