CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN

TESIS

IMPLEMENTACIÓN Y VALIDACIÓN DE UNA E-GUÍA DE LA METODOLOGÍA SoSE-LR

PRESENTA

L.I. Mauricio Silva Camacho

PARA OBTENER EL GRADO DE MAESTRÍA EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES

DIRECTOR DE TESIS

Dr. José Manuel Mora Tavarez

SINODALES

Dr. Héctor Alejandro Duran Limón

Dra. Laura Cecilia Rodríguez Martínez

Aguascalientes, Ags. 5 de Junio de 2012

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

Autorizaciones

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

MAURICIO SILVA CAMACHO
ALUMNO DE LA MAESTRÍA EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES
P R E S E N T E

Estimado Sr. Silva:

Por medio de este conducto me permito comunicarle a usted, que el trabajo tesis o caso práctico titulado "**Implementación y validación de un E-Guía de la metodología SoSE-LR**", está autorizado y será bajo la dirección del Dr. José Manuel Mora Tavares; y como revisores el Dr. Héctor Alejandro Durán Limón y la Dra. Laura Cecilia Rodríguez Martínez, para que pueda obtener así el grado de Maestría en Informática y Tecnologías Computacionales.

Sin otro particular me permito saludarle (a) muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., 1 de junio del 2012
"SE LUMEN PROFERRE"
LA DECANO

MTRA. MARTHA CRISTINA GONZÁLEZ DÍAZ

UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
CENTRO DE CIENCIAS BÁSICAS

VoBo M en C JORGE EDUARDO MACÍAS LUEVANO
SECRETARIO TÉCNICO MAESTRÍA EN INFORMÁTICA
Y TECNOLOGÍAS COMPUTACIONALES

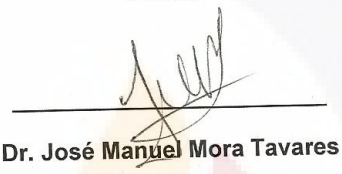
c.c.p. Dr. Alejandro Padilla Díaz.- Secretario de Investigación y Posgrado.
c. c .p. M en C Jorge Eduardo Macías Luevano.- Secretario Técnico de la Maestría en Informática y Tecnologías Computacionales.
c. c. p. Dr. José Manuel Mora Tavares
c. c. p. Dr. Héctor Alejandro Durán Limón
c. c. p. Dra. Laura Cecilia Rodríguez Martínez
c. c. p. Archivo.

Por este conducto autorizamos al tesista:

Mauricio Silva Camacho

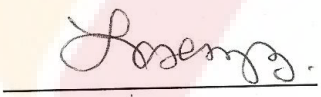
La impresión de su documento final de tesis ya que cumple con los requisitos de contenido y forma exigidos en la Universidad Autónoma de Aguascalientes.

Asesor

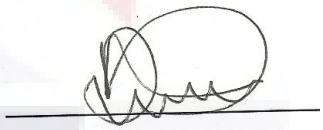


Dr. José Manuel Mora Tavares

Sinodales



Dra. Laura Cecilia Rodríguez Martínez



Dr. Héctor Alejandro Durán Limón

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Agradecimientos

Agradecimientos

El trabajo desarrollado ha sido posible gracias a la colaboración inestimable de muchas personas, por ello deseo expresar mi más sincero agradecimiento comenzando por los profesores de la Maestría, con un enorme respeto por compartir conmigo sus conocimientos y por haberme guiado en esta etapa de mi formación académica, a todos ellos por su comprensión, paciencia y dedicación.

Quiero agradecer muy especialmente al Dr. José Manuel Mora Tavarez, Director de Tesis, por las largas jornadas de asesorías y consultas, por sus consejos y orientaciones para el desarrollo de la tesis.

A la Dra. Laura Cecilia Rodríguez Martínez y al Dr. Héctor Alejandro Duran Limón, por sus aportaciones y comentarios, muchas gracias.

A mis compañeros y profesores, quiénes compartimos esta hermosa experiencia, al Consejo Estatal de Ciencia y Tecnología del Estado de Aguascalientes (CONCyTEA).

Y un agradecimiento general a todas aquellas personas que de una u otra manera ayudaron para la realización de esta Tesis.

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Dedicatorias

Dedicatoria

A Dios.

Por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor.

A mis padres

Porque siempre existieron palabras de apoyo cuando más lo necesité, por comprenderme y alentarme a no claudicar. Porque con su amor, confianza, ejemplo y enseñanza, fortalecieron mi vida. Esta meta lograda también es de ustedes.

Con cariño y admiración.

A mis familiares.

A mis hermanos, por darme ánimo en los momentos difíciles para seguir adelante, por creer en mí y estar conmigo, a mis cuñadas, sobrina y sobrinos con todo mi afecto, que, de forma incondicional me acompañaron en esta aventura de conocimiento.

A nuestra casa de estudios por haberme dado la oportunidad de ser parte de una generación de triunfadores y gente productiva para el país y cumplir este gran sueño.

A todas y todos quienes de una u otra forma han colocado un granito de arena para el logro de este Trabajo de Grado, mil palabras no bastarían para agradecerles su apoyo, su comprensión y sus consejos en los momentos difíciles.

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Índice General

Índice

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Índice de Tablas

Índice de tablas

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Índice de Ilustraciones

Índice de ilustraciones

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Resumen

Resumen

La comunidad de Ingeniería del Software en los últimos años ha expresado especial interés en la mejora de procesos de software con el fin de aumentar la calidad y productividad del mismo. El aseguramiento de calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto satisface los requisitos dados de calidad, así como la reducción de costos y el cumplimiento a la funcionabilidad independientemente de la metodología a utilizar.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, dará un resultado no satisfactorio. Sin embargo, muchas veces no se toma en cuenta el utilizar una metodología y las herramientas adecuadas.

SoSE-LR es un proceso de desarrollo de software bajo el enfoque de servicios. SoSE-LR puede ser considerado mínimamente suficiente pero completo (e.g. balanceado en agilidad y rigor), lo que significa que sólo el contenido fundamental está incluido.

Se pretende implementar un artefacto (EPG de SoSE-LR) que permita representar la experiencia y el conocimiento de los ingenieros de software expertos sobre cómo hacer un producto software usando dicho proceso. Evaluar empíricamente la utilidad, facilidad de uso, compatibilidad e intención de uso de la EPG generada con un grupo piloto de desarrolladores de software.

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Abstract

Abstract

The Software Engineering Community in recent years has expressed a particular interest in software process improvement to increase quality and productivity of the same.

The software quality assurance is a set of actions planned and systematic necessary to provide confidence that the product given the requirements in quality, and reducing compliance costs and the functionality regardless of the methodology used.

All software development is risky and difficult to control, but unless there is a methodology involved, it'll be give an unsatisfactory result. However, some times didn't use a methodology and tools in the right way.

SOSE-LR is a software development process focused on services. SOSE-LR can be considered minimally sufficient but complete (eg balanced in agility and rigor), which means that the only fundamental content is included.

The objective is to implement an artefact (EPG SOSE-LR) which allows to represent the experience and knowledge of expert software engineers on how to make a software product using this process. Empirically evaluate the usefulness, usability, compatibility and the use intention of the EPG generated with a pilot group of software developers.

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Introducción

1 Formulación del problema

1.1 Antecedentes

La comunidad de Ingeniería del Software en los últimos años ha expresado especial interés en la mejora de procesos de software con el fin de aumentar la calidad y productividad del mismo (Scott, Carvalho, & Jeffery, 2002).

El aseguramiento de calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto satisface los requisitos dados de calidad, así como la reducción de costos y el cumplimiento a la funcionabilidad independientemente de la metodología a utilizar.

Una metodología de desarrollo de sistemas se refiere a la estructura que se utiliza para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información. Una gran variedad de estos marcos se han desarrollado en los últimos años, cada uno con sus propias fortalezas y debilidades reconocidas. Una metodología de desarrollo del sistema no es necesariamente adecuado para su uso por todos los proyectos. Cada una de las metodologías disponibles se adapta mejor a tipos específicos de proyectos, en base a varios técnicos, el proyecto de la organización, y las consideraciones del equipo (CMS, 2008).

La mejora de procesos de desarrollo de software requiere de un gran esfuerzo humano, incurren en largos periodos de ejecución y por consiguiente muy costosos. Estos proyectos son críticos para la organización, puesto que implican un cambio de su proceso de producción, con el fin de lograr una mayor productividad y calidad de los productos que elaboran.

De los modelos de calidad existentes como puede ser CMMI (Capability Maturity Model Integrated) (Armas, Chamorro, Montes, & Gutierrez, 2007). Estos modelos de calidad necesitan el cumplimiento de una serie de requisitos para obtener un determinado nivel de calidad (madurez en CMMI). Uno de estos requisitos es el establecimiento de un estándar para el desarrollo de software de la organización (Organizacional Set of Standard Software Process). Este elemento es de especial importancia ya que muchos de los proyectos de desarrollo de software que se realizan no acaban en costo y tiempo estimado (Petter & Vaishnavi, 2008).

En estos proyectos hay actividades claves: la definición del proceso de software, la mejora de su calidad y llevar a cabo su ejecución (Oktaba & Ibarguengoitia, 1998).

Actualmente, la definición de un proceso carece de una formalización que proporcione mecanismos para su caracterización, búsqueda, facilidad de uso, captura de información de experiencia por las personas que han usado el proceso en proyectos anteriores, medidas de uso, etc.

Por lo tanto la definición de procesos es difícil y costosa puesto que cada vez que se aborda la definición de un nuevo proceso se parte de cero. Es necesario siempre tener expertos en el área del proceso y expertos de gestión de procesos para abordar esta actividad. Por consiguiente, las personas implicadas en el proceso deben realizar sus actividades de manera diferente a cómo lo estaban realizando hasta el momento.

Este trabajo aporta a la optimización de los procesos de producción de software mediante la automatización de las metodologías de desarrollo mediante la gestión de proyectos de software la cual nos ayudara a planificar, obtener, agilizar y organizar talentos y administrar recursos, con el fin de terminar todo el trabajo requerido para desarrollar un proyecto y cumplir con el alcance dentro de límites de tiempo y costo definidos lo cual requiere liderar los talentos, evaluar y regular continuamente las acciones necesarias y suficientes. Para esto se trabajó sobre la hipótesis de que el proceso de desarrollo de software es un tipo proceso de negocio particular, y los procesos de negocio pueden ser automatizados en todo o en parte a través de una herramienta, el objetivo es soportar un proceso de desarrollo de software a través de una Guía Electrónica de Procesos (EPG por sus siglas en Inglés) la cual es una herramienta que proporciona orientación del proceso y el apoyo a la gestión del conocimiento en el proceso de desarrollo de software (Kurniawati, Kitchenham, & Jeffery, 2004).

A través de estudios, se demostró que la herramienta es realmente útil y eficaz en el logro de Mejora de Procesos de Software (SPI) con éxito en una organización pequeña de software con sede en Sydney, Australia (Kurniawati et al., 2004).

La EPG a desarrollar contendrá el proceso de desarrollo de software SoSE-LR (referencia de Tesis Doctoral de Dra. Rodríguez). SoSE-LR es un proceso recientemente reportado, orientado al desarrollo de software con un enfoque de servicios.

1.2 Descripción del Problema

El desarrollo de software es un proceso tecnológico de alta complejidad que consume tiempo, requiere mucho esfuerzo humano y demanda costos, generalmente elevados. Un porcentaje muy alto de proyectos de software fracasan debido, entre otros factores, a una gestión deficiente del proyecto. El éxito de un proyecto de software se mide en función de tres variables fundamentales: costo, tiempo y calidad. Un proyecto exitoso es aquel que se entrega a tiempo, bajo el presupuesto asignado y con la calidad especificada. Para manejar estas tres variables, los ingenieros de software emplean modelos, procesos y técnicas gerenciales. (CMMI Product Team, 2006)

La documentación la cual va a reflejar el proyecto ha de estar bajo ciertos estándares y deberá cumplir con normas de acuerdo a la naturaleza y dimensiones del proyecto

Es por esto que se debe de utilizar un método o proceso para guiar las actividades que se tienen que realizar y poder dar solución a los problemas manteniendo preferentemente solo el contenido fundamental y necesario que debe ser incluido.

En particular, el problema direccionado por esta tesis es la falta de disponibilidad de EPGs de diversos procesos de software y de hecho la falta de uso de las pocas EPGs disponibles (RUP (IBM, 2008a), OpenUP (Balduino, 2007), Upedu (Jaakkola & Thalheim, 2005)).

Existen varios problemas generados por no utilizar una EPG (que reporta un modelo de procesos) entre las cuales nos encontramos con:

- Gran cantidad de documentos de texto y gráficos difíciles de actualizar.
- Inconsistencias no validadas entre las entradas y salidas de los procesos.
- Despliegue de los procesos difícil de realizar.
- Documentos monolíticos difíciles de leer.
- Cualquier persona puede modificar (no seguir) los procesos de la organización

Mientras que en otros países se ha reportado su uso (EUA, Alemania y Australia principalmente), en nuestro país tales artefactos son poco conocidos. En esta Tesis, se cree que el uso de tales EPGs podrá fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances.

1.3 Justificación

Cuando se va a iniciar un desarrollo de software debemos preguntarnos si debemos tener un plan en que apoyarnos y como desarrollarlo.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, dará un resultado no satisfactorio. Sin embargo, muchas veces no se toma en cuenta el utilizar una metodología y las herramientas adecuadas.

En algunas ocasiones se realiza el diseño de manera rígida, con los requerimientos que el cliente solicitó, de tal manera que cuando el cliente solicita un cambio en las etapas finales, se hace muy difícil realizarlo, pues altera muchas cosas que no se habían previsto y es uno de los factores que ocasiona un atraso en el proyecto. Obviamente para evitar estos incidentes se debe llegar a un acuerdo formal con el cliente al inicio del proyecto, de tal manera que cada cambio o modificación no perjudique al desarrollo del mismo.

Los proyectos en problemas son los que se salen del presupuesto, tienen importantes retrasos, o simplemente no cumplen con las expectativas del cliente.

Debido al gran esfuerzo de mejora de procesos de software se propone que se desarrollen y utilicen EPGs basado en web con enlaces a plantillas electrónicas para mejorar la documentación del proyecto. El uso de esta herramienta da lugar a la consideración de gestión del conocimiento en general. Debido a la relación proceso de software con el proceso de gestión orientado al conocimiento apoyando la creación del aprendizaje de las organizaciones. Este enfoque proporciona los conocimientos pertinentes, relacionados con las tareas y como una tarea está siendo ejecutada. El enfoque centrado en el proceso sigue la filosofía de que la gente no simplemente necesita más información, sino que la información debe ser pertinente a las tareas que se llevando a cabo en determinado momento (Scott et al., 2002).

¿Qué es un EPG y cómo nos ayudara en la gestión de procesos? La Guía Electrónica de Procesos (EPG) es una herramienta que proporciona una orientación del proceso y apoya a la gestión del conocimiento en un proceso de desarrollo de software. Está diseñada para ayudar a los profesionales en creación de procesos en definir los procesos de software. (Kurniawati et al., 2004)

En este documento se hablara sobre una Guía Electrónica de Procesos EPG, la cual puede ser construida con diversos tipos de herramientas (software general de HTML o software específico). En esta Tesis se considera que como primer propuesta, la utilización de una herramienta generadora de documentos HTML aportará los elementos necesarios para una EPG básica. En caso de ampliar esta

Tesis, se sugiere utilizar herramientas avanzadas (como EPF Eclipse Process Framework (Haumer, 2007)).

En resumen, una EPG ayuda a agilizar la creación de proyectos y como ayuda al ser utilizada en el uso de mejores prácticas, cómo aprovechar las propias lecciones aprendidas, proporcionar un cierto nivel de coherencia y un lenguaje común en toda la organización al desarrollar, atender las necesidades específicas de los distintos proyectos. Esta automatización parcial del proceso de software SoSE-LR debe ayudar a realizar constantemente ajustes a los mismos, para minimizar los errores, los cuellos de botella y el incumplimiento de metas.

SoSE-LR es un proceso de desarrollo de software bajo el enfoque de servicios. SoSE-LR puede ser considerado mínimamente suficiente pero completo (e.g. balanceado en agilidad y rigor), lo que significa que sólo el contenido fundamental está incluido. Por lo tanto, no proporciona orientación sobre muchos temas que los proyectos pueden tratar, tales como el tamaño de gran equipo, el cumplimiento, las situaciones contractuales, seguridad o aplicaciones de misión crítica, de orientación tecnológica específica, etc. SoSE-LR es completo en el sentido que se puede manifestar como un proceso para la construcción del sistema.

1.4 Objetivo

El propósito de esta tesina es implementar un EPG del proceso de desarrollo de software SoSE-LR que ayude a mejorar su proceso de consulta y aprendizaje.

Objetivo 1: Implementar un artefacto (EPG de SoSE-LR) que permita representar la experiencia y el conocimiento de los ingenieros de software expertos sobre cómo hacer un producto software usando dicho proceso.

Objetivo 2: Evaluar conceptualmente la calidad del diseño de la EPG por un panel de expertos en Ingeniería de Software.

Objetivo 3: Evaluar empíricamente la utilidad, facilidad de uso, compatibilidad e intención de uso de la EPG generada con un grupo piloto de desarrolladores de software.

Pregunta de Investigación 1: ¿Es posible implementar un artefacto (EPG de SoSE-LR) que permita representar la experiencia y el conocimiento de los ingenieros de software expertos sobre cómo hacer un producto software usando dicho proceso?

- Proposición de Investigación Alternativa.1: Si es posible implementar un artefacto (EPG de SoSE-LR) que permita representar la experiencia y el conocimiento de los ingenieros de software expertos sobre cómo hacer un producto software usando dicho proceso.
- Proposición de Investigación Nula.1: No es posible implementar un artefacto (EPG de SoSE-LR) que permita representar la experiencia y el conocimiento de los ingenieros de software expertos sobre cómo hacer un producto software usando dicho proceso.

Pregunta de Investigación 2: ¿Cuál es la calidad del diseño de la EPG generada por un panel de expertos en Ingeniería de Software?

- Proposición de Investigación Alternativa.2: la calidad del diseño de la EPG generada será mayor o igual a 3.0 (usando una escala de Likert de 1 a 5) por un panel de expertos en Ingeniería de Software.

- Proposición de Investigación Nula.2: la calidad del diseño de la EPG generada no será mayor o igual a 3.0 (usando una escala de Likert de 1 a 5) por un panel de expertos en Ingeniería de Software.

Pregunta de Investigación 3: ¿Cuáles son los valores percibidos de utilidad, facilidad de uso, compatibilidad e intención de uso de la EPG generada por un grupo piloto de desarrolladores de software?

- Proposición de Investigación Alternativa.3: los valores percibidos de utilidad, facilidad de uso, compatibilidad e intención de uso de la EPG generada por un grupo piloto de desarrolladores de software, será mayor o igual a 3.0 (usando una escala de Likert de 1 a 5) por un panel de expertos en Ingeniería de Software.
- Proposición de Investigación Nula.3: los valores percibidos de utilidad, facilidad de uso, compatibilidad e intención de uso de la EPG generada por un grupo piloto de desarrolladores de software, será mayor o igual a 3.0 (usando una escala de Likert de 1 a 5) por un panel de expertos en Ingeniería de Software.

Para poder cumplir con los objetivos antes mencionados debemos de utilizar un framework el cual es un esqueleto sobre el cual varios objetos son integrados para una solución dada. Cabe mencionar que este no es consciente de todos los requerimientos sin importar las aplicaciones que con las cuales se desarrollara. A esto le sumamos la capacidad de extenderse sin prejuicios para diversificar la expresión del programa mismo. En términos generales, es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de software, un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Provee una estructura y una metodología de trabajo.

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Marco Teórico

2 Marco Teórico

2.1 Especificación de proceso en Ingeniería de Software

Un proceso del software es un conjunto de actividades que conducen a la creación de un producto de software. Estas actividades pueden consistir en el desarrollo de software desde cero en un lenguaje de programación estándar como Java o C. sin embargo, cada vez que se desarrolla nuevo software ampliando y modificando sistemas existentes y configurando e integrando software comercial o componentes del sistema (Sommerville, 2005).

Aunque existen muchos procesos diferentes de software, algunas actividades fundamentales son comunes para todos ellos:

- Especificación del software. Se debe definir la funcionabilidad del software y las restricciones en su operación.
- Diseño e implementación del software. Se debe producir software que cumpla con su especificación
- Validación del software. Se debe validar el software para asegurar que hace lo que el cliente desea
- Evolución del software. El software debe evolucionar para cubrir las necesidades cambiantes del cliente (Sommerville, 2005)

Como vimos anteriormente el proceso de software es una composición de fases, actividades, artefactos y recursos (incluyendo los humanos). Las fases, son mejor conocidas como fases del ciclo de vida del software.

Cada fase contiene varias actividades que se llevan a cabo con el propósito de generar un importante producto (artefacto), como por ejemplo, el documento de especificación de requerimientos o el documento de diseño. Ejemplos de fases son: análisis, diseño, código, instalación, etc. (Oktaba & Ibarguengoitia, 1998).

2.2 Modelos de proceso de Software

Cada modelo de proceso representa un proceso desde una perspectiva en particular y así proporciona solo información parcial sobre ese proceso, a continuación se describen algunos de los modelos:(Sommerville, 2005)

Modelo en cascada. Considerada de las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa fases separadas del proceso tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas, etc.(Sommerville, 2005)

Desarrollo evolutivo. Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones abstractas. Este se refina basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades.(Sommerville, 2005)

Ingeniería de software basada en componentes. Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema más que en desarrollarlos desde cero.(Sommerville, 2005)

Incremental. En un proceso de desarrollo incremental, los clientes identifican a grandes rasgos, los servicios que proporcionara el sistema. Identifican que servicios son más importantes y cuales menos. Entonces se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema.(Sommerville, 2005)

Desarrollo en espiral. Cada ciclo en la espiral representa una fase del proceso del software, cada ciclo de la espiral se divide en cuatro sectores.(Sommerville, 2005)

Como vemos existen varias y muy completas metodologías que se acoplan a las necesidades de cada empresa, cabe mencionar que también existe: Rapid Application Development (RAD), Programación orientada a objetos (OOP), Virtual Finite State machine (VFSM), Dynamic Systems Development Method, Scrum (desarrollo), Rational Unified Process (RUP), Extreme Programming(XP), Enterprise Unified Process (EUP), Constructionist Design Methodology (CDM), Agile Unified Process (AUP) entre otras.

2.3 Guías electrónicas de proceso en (EPG) en Ingeniería de Software

En el momento de definir la metodología a usar en un proyecto específico los ingenieros de proceso y los líderes de proyecto invierten un gran esfuerzo, para decidir que ciclo de vida van a aplicar. En muchos casos es un proceso manual y depende de la experiencia y habilidad de las personas. Como alternativa han surgido un conjunto de herramientas que facilitan el mantenimiento, actualización, difusión de las metodologías y las buenas prácticas de la empresa.(Moe & Dybå, 2004)

Una guía electrónica de procesos debe de contener la información que describa y detalle los siguientes elementos básicos de los procesos:(Balduino, 2007)

- Actividades
- Artefactos
- Roles
- Agentes
- Herramientas

La interface de EPG básicamente es un sitio Web, al que se puede acceder a través de la intranet de la organización, en él se publican las buenas prácticas de la compañía, los procesos e información relacionada con las metodologías de desarrollo. Específicamente se publica información que es de mucha utilidad para los equipos de proyecto como por ejemplo: definiciones de fases, actividades, hitos, diagramas de flujo, políticas, procedimientos, estándares, listas de verificación, y plantillas para los documentos de entrega.

EPG más que una herramienta es un marco conceptual que define las características que debería tener una guía electrónica de procesos, que puede implementarse por las organizaciones de acuerdo con su entorno tecnológico. En este sentido los requisitos y principios básicos que una EPG debería satisfacer según son:(Kurniawati & Jeffery, 2005)

- Proporcionar una descripción de los procesos, que contenga: actividades, artefactos, roles, agentes, recursos y las principales relaciones entre ellos, preferiblemente que permita una descripción gráfica.
- Proveer de una interface efectiva al usuario
- Que use enlaces de manera efectiva para permitir la navegación a través de la guía
- Permitir el acceso fácil y rápido a la información solicitada por el usuario
- Usar una notación conocida que facilite la orientación y la búsqueda a través de la guía
- No “inundar” al usuario con mucha información
- Que se pueda implementar por medio de tecnologías compatibles con aplicaciones Web y el entorno de la organización.

Tanto en el ámbito académico como industrial se han desarrollado herramientas que plasman estos principios. A continuación describimos algunas de las más importantes, y que conforman el punto de partida para la integración de herramientas que soportan un proceso de desarrollo.
Planificación y control de procesos. La planificación necesita como entrada, los elementos de un proyecto (fases, roles, tareas, actividades, productos, etc). De esta forma las EPGs, y su producto, se integran con las herramientas de planificación para gestionar el ciclo de vida de cada proyecto.

Repositorios y gestión. La integración de un entorno de desarrollo debe facilitar la gestión de todos los activos de una empresa, entre ellos los elementos, las actividades y los resultados de un proyecto de desarrollo de software.

2.4 Estructura genérica

El esquema conceptual que aquí se presenta (Kellner, Becker-Kornstaedt, Riddle, Tomal, & Verlage, 1998) ofrece una única e integrada vista de los elementos e interrelaciones anteriormente citados.

Este esquema conceptual es suficientemente general como para ser aplicado en un amplio rango de métodos de representación de procesos y se construye sobre el marco de trabajo conceptual.

Los elementos básicos de este esquema y sus interconexiones se agrupan en las siguientes clases:

Productos de Trabajo (Artifacts): Descripciones de productos creados o modificados durante el proceso, tanto en su resultado intermedio o final como en su estado temporal interno.

Actividades (Activities): Descripciones de “cómo se hacen las cosas”. Actividades y Artefactos se asocian a través de las relaciones “produce” y “usa”.

Agentes (Agents): Descripciones de entidades que pueden llevar a cabo actividades. Las descripciones son en términos de técnicas, costes y disponibilidad. Un agente puede ser un individuo o un grupo.

Roles (Roles): Descripción de un conjunto de obligaciones y permisos relacionados con las actividades. Agentes y roles están relacionados por la relación “asume”. Roles y actividades están asociados por la relación “implicado en”.

Recursos (Resources): Descripciones de programas informáticos u otras ayudas que pueden ser utilizadas para dar soporte o automatizar el desarrollo de una actividad.

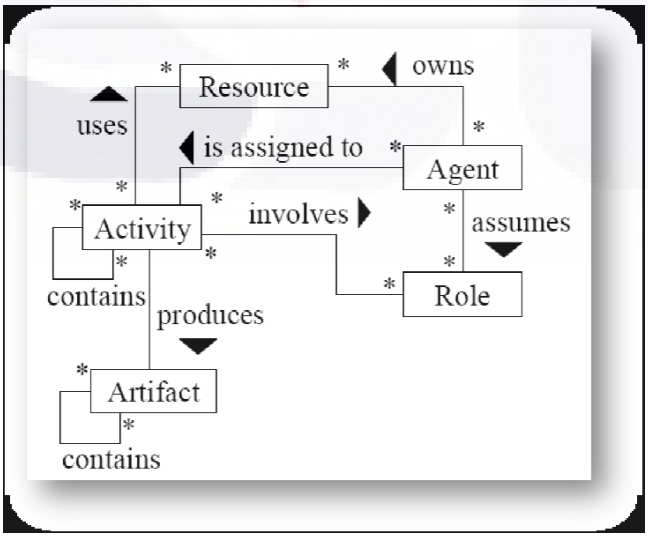


Ilustración 1 Interrelaciones entre clases

2.5 Herramientas para construir EPG

2.5.1 Editores de HTML

HTML, siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. Un editor de páginas web es una aplicación diseñada con el fin de facilitar la creación de documentos HTML o XHTML. Su complejidad puede variar desde la de un simple editor de texto plano, entornos WYSIWYG, hasta editores WYSIWYM.

Amaya

Amaya es una herramienta combinada del W3C compuesta por un navegador web y una herramienta de autor. Cualquier página web que se abra puede ser editada inmediatamente. Se pueden ver y generar páginas HTML y XHTML con hojas de estilo CSS, expresiones MathML y dibujos SVG. Una gran característica consiste en que puede ver los enlaces que se crean con el editor.(Quint & Vatton, 2005)

Renderiza imágenes, por ejemplo en PNG y un subconjunto del formato de Gráficos Vectoriales Escalables (SVG), como figuras básicas, texto, imágenes y foreignObject (el último es útil para incluir fragmentos HTML o expresiones MathML en los dibujos). Los gráficos están escritos en XML y pueden ser mezclados libremente con HTML y MathML.

Kompozer

Kompozer es un proyecto derivado de otros editores WYSIWYG creados en inicio por la fundación Mozilla, como Composer, que actualmente se ha dejado de desarrollar por Mozilla, aunque se encuentra algo similar dentro de la suite de programas Sea Monkey. A partir del momento que se abandonó Composer de Mozilla, el desarrollo de este editor se continuó en un proyecto llamado Nvu, que también se dejó de lado por la empresa que había tomado el testigo. Nvu ya mejoró bastante el Composer de Mozilla, agregando unas cuantas cosas, pero no era muy estable y tenía diversos errores que se están terminando de solucionar en Kompozer.

Lo bueno es que Kompozer ha heredado la tecnología de Mozilla para crear un editor web que funciona con el mismo motor de Firefox.

Editplus

Está especializado en HTML, tiene un asistente para tablas, una lista enorme de Activex que pueden usarse, y en general varios botones y teclas abreviadas para introducir etiquetas y otras cosas, una paleta que saca el color hexadecimal, control + b inserta las etiquetas y al principio del texto seleccionado, etc.

2.5.2 Editores avanzados para Web (Dreamweaver, Flash, etc).

Adobe Dreamweaver

Adobe Dreamweaver es una aplicación en forma de estudio (basada en la forma de estudio de Adobe Flash) que está destinada a la construcción y edición de sitios y aplicaciones Web basados en estándares. Creado inicialmente por Macromedia (actualmente producido por Adobe Systems) es uno de los programas de este tipo más utilizado en el sector del diseño y la programación web, por sus funcionalidades, su integración con otras herramientas como Adobe Flash y, recientemente, por su soporte de los estándares del World Wide Web Consortium. Su principal competidor es Microsoft Expression Web y tiene soporte tanto para edición de imágenes como para animación a través de su integración con otras. Hasta la versión MX, fue duramente criticado por su escaso soporte de los estándares de la web, ya que el código que generaba era con frecuencia sólo válido para Internet Explorer, y no validaba como HTML estándar. Esto se ha ido corrigiendo en las versiones recientes.

La gran ventaja de este editor sobre otros es su gran poder de ampliación y personalización del mismo, puesto que en este programa, sus rutinas (como la de insertar un hipervinculo, una imagen o añadir un comportamiento) están hechas en Javascript-C, lo que le ofrece una gran flexibilidad en estas materias. Esto hace que los archivos del programa no sean instrucciones de C++ sino, rutinas de Javascript que hace que sea un programa muy fluido, que todo ello hace, que programadores y editores web hagan extensiones para su programa y lo ponga a su gusto.

Las versiones originales de la aplicación se utilizaban como simples editores WYSIWYG. Sin embargo, versiones más recientes soportan otras tecnologías web como CSS, JavaScript y algunos frameworks del lado servidor.

Dreamweaver ha tenido un gran éxito desde finales de los años 1990 y actualmente mantiene el 90% del mercado de editores HTML. Esta aplicación está disponible tanto para la platafoma MAC como para Windows, aunque también se puede ejecutar en plataformas basadas en UNIX utilizando programas que implementan las API's de Windows, tipo Wine.

Como editor WYSIWYG que es, Dreamweaver permite ocultar el código HTML de cara al usuario, haciendo posible que alguien no entendido pueda crear páginas y sitios web fácilmente sin necesidad de escribir código.

Algunos desarrolladores web criticaban esta propuesta ya que crean páginas HTML más largas de lo que solían ser al incluir mucho código inútil, lo cual va en detrimento de la ejecución de las páginas en el navegador web. Esto puede ser especialmente cierto ya que la aplicación facilita en exceso el diseño de las páginas mediante tablas. Además, algunos desarrolladores web han criticado Dreamweaver

en el pasado porque creaba código que no cumplía con los estándares del consorcio Web (W3C).

Expression Web

Está enfocada al cumplimiento de estándares, ya que permite validar nuestro contenido contra el estándar que el usuario desee seguir o navegadores en los que se planea montar la aplicación Web. Además, Expression Web es compatible con hojas de estilo CSS y con Microsoft Visual Studio. Asimismo, incluye la capacidad de procesar archivos XML mediante JavaScript.

Desde la versión 2 del programa, se permite la integración con lenguajes de servidor como ASP.NET o PHP sin necesitar de instalar un servidor. También permite la interacción con Adobe Photoshop para generar imágenes.

	Editor de texto plano	Hojas de estilo	Despliegue de contenidos	Soporta RUP
HTML	SI			
Amaya	SI	SI	SI	SI
Kompozer	SI	SI		si
Editplus	SI	SI		
DreamWeaver	SI	SI		

Tabla 1 Editores HTML

2.5.3 Lenguajes de Programación (Java).

Java

La primera versión de Java empezó en 1991 y fue escrita en 18 meses en Sun Microsystems. De hecho, en ese momento, ni siquiera se llamó Java; se llamó Oak y se utilizó en Sun para uso interno. La idea original para Oak era crear un lenguaje orientado a objetos independiente de la plataforma. Por entonces, muchos programadores se limitaban a la programación del IBM PC, pero el entorno corporativo podía incluir toda clase de plataformas de programación, desde el PC hasta los grandes sistemas. Lo que había detrás de Oak era crear algo que se pudiera usar en todos los ordenadores (y ahora que Java se ha hecho popular gracias a la red Internet, cada vez más corporaciones están adoptándolo para uso interno en lugar de C++, precisamente por esa razón). El lanzamiento original de Oak no fue especialmente fascinante; Sun quería crear un lenguaje que se pudiera usar en electrónica. Oak pasó a llamarse Java en 1995, cuando se lanzó para el uso público y supuso un éxito casi inmediato. En ese momento, Java había adoptado un modelo que lo hizo perfecto para la red Internet, el modelo bytecode.(Holzner, 2000)

2.6 Editores especiales EPF

2.6.1 Lisa

Es una herramienta propuesta dentro del marco del proyecto ZEN. El objetivo de LiSA es almacenar el conocimiento de los procesos de desarrollo de la organización (conocimiento metodológico), por medio de patrones de proceso. Permite almacenar información sobre: roles, actividades, patrones de procesos, artefactos.(Gnatz, Marschall, Popp, Rausch, & Schwerin, 2009)

Eclipse Process Framework Composer (EPF)

Proporciona a los profesionales del desarrollo de software una base de conocimiento del capital intelectual, que facilita la revisión, la gestión y el despliegue de contenidos como: definiciones de métodos, artículos, guías de proceso, plantillas, buenas prácticas, normas, procedimientos internos, material para entrenamiento y otros aspectos acerca de cómo desarrollar software. Proporcionar a los ingenieros y a los gerentes de proyecto la capacidad de seleccionar, adaptar y ensamblar de manera ágil, los procesos para un proyecto de desarrollo concreto.(Haumer, 2007)

2.6.2 IBM Rational Method Composer (RMC)

Es una herramienta comercial de IBM Rational, que incluye los módulos donados al proyecto EPF, junto con la integración a la herramienta Rational, las características de migración y la versión completa de RUP.

RMC, incluye plug-ings que se instalan sobre RUP y que ofrecen una variedad de métodos y prácticas para una gran variedad de dominios de tecnología de la información. Soporta la metodología RUP, para diferentes dominios.(IBM, 2008)

	Almacena Conocimiento	Fácil revisión	Despliegue de contenidos	Soporta RUP	Modelado UML	Acceso Libre
LiSA	SI					NO
Eclipse EPF	SI	SI	SI	SI	SI	SI
IBM RMC				SI		NO

Tabla 2 Herramientas crear EPG

2.6.3 Editores Especiales (EPF).

Eclipse Process Framework

Herramienta de desarrollo de procesos open source. Con ella se pueden crear procesos consistentes visualizados a partir de un sitio, con navegación ágil para los usuarios de los procesos.

Eclipse es un entorno libre y de código abierto de desarrollo integrado (IDE), originalmente iniciado por IBM en 2001.Tres años más tarde, la fundación Eclipse fue creado como una organización independiente sin fines de lucro, para actuar como administrador de la comunidad Eclipse. Hoy en día, la comunidad de código abierto Eclipse está hecho de personas y organizaciones de una sección transversal de toda la industria del software. La plataforma Eclipse es utilizado por miles de empresas de TI el desarrollo de software. Este éxito se debe principalmente al diseño de la plataforma Eclipse, que ofrece un API documentado y el medio ambiente facilitando el desarrollo y la integración de plug-ins. En la actualidad, cientos de plug-ins (prueba de la unidad, por ejemplo, el modelado UML, y el control de versiones) se puede utilizar con Eclipse.

El proceso de Eclipse Compositor Framework (EPF Composer) comenzó en febrero de 2006 y publicó su primera versión 1.0 en septiembre de 2006. EPF Composer es una plataforma de herramientas de gestión de procesos y el marco conceptual para su creación, adaptación e implementación de procesos de desarrollo de software. EPF Composer ha sido desarrollado para abordar los problemas relacionados con el proceso que enfrentan los gerentes de proyecto o programa y por los ingenieros de proceso que se encarga de definir y mantener los procesos de desarrollo. Dos grandes problemas son abordados por la herramienta. En primer lugar, facilita la comprensión de los métodos de desarrollo de software, procesos y conceptos por los equipos de proyecto. En segundo lugar, explica que el equipo de desarrollo la manera de aplicar una metodología determinada mediante la descripción de la secuencia de pasos, explicando las técnicas y herramientas a utilizar para realizar una tarea determinada(Kurniawati et al., 2004)

2.6.4 NetBeans

NetBeans es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto con una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

2.6.5 Ventajas y desventajas de Eclipse y NetBeans

Eclipse

Fue desarrollado originalmente por IBM para la programacion en JAVA. Ees gratuito, tiene integración con CVS (Concurrent Versioning System) ade mas funciona con (PHP, C/C++, Python, RUby, C, C++, Flex, Corba, etc), pero para ello se necesitan adicionales.

Muestra los errores de PHP incluyendo un tooltip mostrando cual es el error, se puede cerrar bloques de código como funciones, clases o incluso comentarios, auto completa tags, atributos html, incluso auto completa comentarios de PHPDoc, tiene multiples proyectos a la vez

Uso de SWT en vez de Swing mejora el rendimiento debido al uso de widgets nativos.

Eclipse compila en tiempo real, es decir, no espera a que tenga el código listo para compilarlo tras la pulsación de sus botones, sino que, conforme se va escribiendo el código, va avisando de los errores que va cometiendo.

Contra

Para aplicaciones web en Java hay que instalar unos plugins funcione bien, mientras unos pueden ver ventajas en el uso de Eclipse al usar plugins, unos lo pueden ver como desventaja, ya que Netbenas trae todo en un mismo paquete, pero al mismo tiempo Netbeans genera mucho código innecesario y por la misma razón consume muchos recursos.

NetBeans

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans. Es gratuito y se puede programar webs en Java sin instalarle ningún plugin ya que posee un tomcat de serie, El depurado de las aplicaciones es más sencillo.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo.

Contras: Consume más recursos que Eclipse.

	Netbeans	Eclipse
Gratuito	SI	SI
Funciona con otros lenguajes	NO	SI
Auto compilado	NO	SI
SWT	NO	SI
SWING	SI	NO

Tabla 3 NetBeans VS. Eclipse

Podemos mencionar que ambos son buenos la utilización de uno u otro es más una cuestión de comodidad, pero debido a las necesidades del estudio el que se acopla más a las necesidades del mismo el IDE optimo es eclipse.

Como hemos visto la herramienta que nos puede ayudar por la versatilidad y manejo de opciones a utilizar ser Eclipse Framework ya que nos permite mantener una base de metodologías y a partir de ellas componer un ciclo de proyecto

2.7 Estudio de Casos Similares

2.7.1 Caso No. 1

Un proceso para proyectos pequeños y ágiles

Los proyectos pequeños tienen necesidades diferentes procesos de los proyectos más grandes. Pequeños equipos de proyecto en general quiere bajos costos, para que puedan centrarse en la entrega del producto. Sin embargo, existen buenas prácticas que beneficien a los pequeños proyectos, y les ayudará a ser más eficaces.

¿Qué es el Proceso Unificado Básico?

Basic Unified Process (BUP) es una versión simplificada de IBM Rational Unified Process (RUP) optimizado para pequeños proyectos. Los proyectos pequeños constituyen equipos de 3 a 6 personas y la participación de 3 a 6 meses de esfuerzo de desarrollo. BUP conserva las características esenciales del RUP, que incluye el desarrollo iterativo, casos de uso y escenarios que impulsan el desarrollo, gestión de riesgos, y el enfoque centrado en la arquitectura. El resultado es un proceso mucho más simple que sigue siendo fiel a los principios de RUP.

¿Cómo funciona el proceso unificado?

BUP se organiza en dos dimensiones diferentes (pero correlacionadas): método y el proceso. La dimensión de método es el método que los elementos (es decir, las funciones, tareas, los artefactos, y orientación) se definen, independientemente de cómo se aplican en un ciclo de vida del proyecto. La dimensión del proceso es donde los elementos de método se aplican en un sentido de comportamiento, donde desde el mismo conjunto de elementos de método, muchos ciclos de vida diferentes pueden ser creados para diferentes tipos de proyectos.

El método BUP método se centra en las siguientes disciplinas: requisitos, arquitectura, desarrollo, pruebas, gestión de proyectos y gestión del cambio, lo que permite la selección de contenidos que sólo se desean a la hora de crear una configuración para su publicación.

Roles

Las habilidades esenciales que necesitan los equipos pequeños están representados por los roles de BUP.

Analista - responsable de reunir los requisitos y la documentación cuando sea necesario.

Arquitecto - responsable de la arquitectura de software.

Desarrollador - crear una solución (o parte de ella), haciendo el diseño, implementación, pruebas unitarias y de integración de los componentes.

Tester - responsable de probar el sistema desde una perspectiva más grande que el desarrollador hace, por lo que el sistema funciona tal como se define y es aceptado por el cliente.

Gerente de Proyecto - planifica y gestiona el proyecto, coordina las interacciones con las partes interesadas, y mantiene el equipo del proyecto enfocado.

Tareas

Algunas tareas RUP se transformaron en pasos y se incluyó dentro de otras las principales tareas que realiza el mismo papel y en el mismo punto de tiempo. Un proyecto puede decidir si ese paso se lleva a cabo o no, dependiendo de lo que se necesita.

Artefactos

Existe una reducción del número de artefactos (en comparación con RUP) En general, estas directrices recomiendan la captura de la información en un artefacto existente, hoja de cálculo, base de datos, tabla, e-mail, etc.

Un ejemplo de bloque de construcción básico de BUP es *desarrollar la solución,* como se muestra a continuación:

Desarrollar la solución (por requerimiento) (en contexto)	Actividad
Diseño de la solución	
Implementar pruebas de desarrollo	
Implementar la solución	**>** Tareas
Ejecutar pruebas de desarrollo	
Integrar y crear Construir	

Tabla 4

Tabla 4. Bloque clásico BUP

BUP apoya la agilidad en el sentido de que tiene un pequeño número de funciones, tareas y artefactos informales. Es compatible con la flexibilidad, ya que su organización en paquetes y la separación entre el método y el proceso permite la selección de elementos de medición deseado (sólo un subconjunto, si se quiere) para crear el proceso que tenga sentido para la realización del proyecto.

2.7.2 Caso No. 2

La implementación de un proceso basado en el modelo de desarrollo de software en un entorno de apoyo: comparación de un componente de código abierto con una herramienta propia.

En este trabajo se examina el uso de TikiWiki como una herramienta de gestión de contenido de código abierto en el contexto de su uso como parte de un modelo basado en el proceso descriptivo entorno de desarrollo de software. Este entorno está formado por un modelo de proceso basado en una guia de procesos electrónicos (EPG) vinculado a un servidor de repositorio de experiencia y una base de datos para la gestión de tareas y el tiempo de grabación. Se lleva a cabo en un medio de organización de empresas de desarrollo de software que se centra en las aplicaciones web, los grandes proyectos de mark-up y las tecnologías web.

El proceso de guía electrónica / Experiencia Repository (EPG / ER) es una herramienta basada en Web que proporciona una guía de procesos y apoyo a la gestión del conocimiento para el proceso de desarrollo de software.

Cada página de la EPG / ER consiste en dos marcos. La visión esquemática de la izquierda muestra el diagrama UML del proceso y admite la navegación y la hiper-navegación del proceso. La sección de descripción en la parte derecha muestra la descripción de la entidad que se muestra actualmente y los atributos de texto como los criterios de entrada y salida, responsabilidades, etc Para navegar por el modelo de proceso, el usuario simplemente selecciona la actividad que él o ella está interesado para aprender.

TikiWiki normalmente se ejecuta en un servidor UNIX, pero la versión que se ejecuta en una plataforma Windows fue utilizado para esta investigación. Después de arreglar el problema de inicio de sesión, se hicieron varios intentos para generar una imagen similar a la presentada por la implementación PageSeeder. Una serie de problemas se encontraron en estos intentos.

A pesar de tener la mayor ventaja de ser de código abierto (y por lo tanto libre), se decidió que el uso de TikiWiki en esta etapa no era una opción viable.

Ser una aplicación de código abierto relativamente nuevo, un TikiWiki se encontró que era propenso a errores. Y a pesar de que ofrece la alternativa más barata, a diferencia de PageSeeder, carece de un grupo de usuarios de soporte dedicado. A TikiWiki también permite que todos los derechos de edición a cualquier usuario por lo tanto, la presentación de una alta posibilidad de corrupción repositorio. Además, la cantidad de esfuerzo para personalizar un TikiWiki en el diseño de EPG / ER parece ser innecesario, dado que las funciones de la herramienta actual perfectamente con la herramienta PageSeeder actuar de sus funciones de gestión del conocimiento. Debido a estas razones, se decidió que el uso de la herramienta

de TikiWiki para manejar las funciones de gestión del conocimiento de una herramienta de EPG / ER no es actualmente una opción viable.

2.7.3 Caso No. 3

Proceso de centrado en la experiencia de repositorio para una pequeña organización de software

El repositorio de experiencia está diseñado para ayudar con tareas generales de desarrollo de software, teniendo en cuenta los presupuestos más pequeños, los recursos y los plazos típicos de las organizaciones más pequeñas. En el dominio de software de La creciente importancia de la gestión del conocimiento se refleja en el establecimiento de enfoques tales como la creacion de experiencia y la inclusión de la gestión del conocimiento como un factor clave en los enfoques de muchos programas de mejora de procesos.

Los enfoques desarrollados para las grandes organizaciones son por lo general no simpatizan con los limitados recursos, incluyendo tiempo, dinero y personal, disponibles en las organizaciones pequeñas.

El depósito de la experiencia (ER) se implementa como una simple extensión de la Guía de Procesos Electrónica (EPG) que ya está en uso en la empresa. La EPG es una presentación basada en web. La EPG se genera directamente a partir de modelos UML y las actividades de los procesos. Los modelos de definir las actividades, objetos, flujos de control y los flujos de objeto del proceso. El generador de EPG toma los modelos UML, reformatea ellos y proporciona una navegación y enlaces para ayudar en la ejecución del proceso.

La propia ER está estructurado en cuatro categorías: listas de verificación, ejemplos, plantillas y experiencias genéricas.

Para implementar la herramienta de EPG / ER se amplió la EPG original para incluir las páginas de enlace rápido. Las páginas fueron luego sembradas estratégicamente para proporcionar posibles vínculos con las entradas de ER.

Para hacer el sistema utilizable desde el principio y ver los beneficios rápidamente, se obtuvieron a dos expertos. Un experto para la "puesta en marcha" y el otro para todas las etapas del proceso.

Después de la sesión de entrenamiento de la EPG / ER se puso a disposición para su uso general. Los usuarios se les animó a navegar por la EPG / ER y entrar y recuperar experiencias como lo exige más que a través de un proceso de gestión del conocimiento formal. Este documento informa sobre el uso desde el primer mes de instalación de la herramienta para dar algunas ideas de su éxito o fracaso. Los datos provienen de un modelo de aceptación de la tecnología (TAM) basada en una encuesta llevada a cabo después de la formación, EPG / ER en los registros del servidor, los registros de PageSeeder, el registro de tarea EPG / ER y encuestas a los usuarios.

Pregunta	Datos	Medida
¿Se utiliza la EPG / ER?	EPG los registros del servidor	Página accedidas a través del tiempo
	Tarea registros	Cantidad de tiempo dedicado al uso de EPG / ER
	PageSeeder Registros	Número de nuevas entradas PageSeeder
	Encuesta de usuarios	Informes de la utilización
¿La EPG / ER bien aceptada?	TAM encuesta	Facilidad percibida y utilidad percibida de uso previo
¿Cómo es la EPG / ER utiliza?	EPG los registros del servidor	Páginas más visitadas
	Encuesta de usuarios	Informó características más utilizadas
	PageSeeder Registros	Tipo de nuevas entradas PageSeeder
¿Cuál es la EPG / ER utiliza?	Encuesta de usuarios	El uso informado
¿Cuáles son los efectos del uso de la EPG / ER /	Encuesta de usuarios	Beneficios reportados / perjuicios
¿Qué mejoras podrían hacer la EPG / ER?	Encuesta de usuarios	Mejora sugeridas

Tabla 5 Preguntas sobre EPG

Una encuesta de usuarios se llevó a cabo al final de cuatro semanas. La encuesta contenía preguntas abiertas acerca de las opiniones de uso y los usuarios de las herramientas, como son las ventajas o inconvenientes de la utilización de la herramienta y cómo la herramienta se puede mejorar. Las primeras respuestas fueron muy positivas. De los doce posibles usuarios nueve reportaron el uso de la herramienta. Si cinco usuarios han encontrado la herramienta beneficiosa para su trabajo, mientras que cuatro usuarios reservaron su juicio. Las características más utilizadas fueron los procesos de desarrollo en cuanto a la adición, la navegación o la descarga de plantillas y ejemplos. Dos usuarios citó el tiempo necesario para agregar las experiencias y para acceder y utilizar las experiencias como menores gastos en detrimento de utilizar la EPG / ER. La mejora más solicitada fue una facilidad de búsqueda.

Los resultados de uso y aceptación de la EPG / ER después del primer mes de uso general fueron muy alentadores. La compañía está utilizando la herramienta y los usuarios parecen estar encontrando lo verdaderamente útil. La tasa de participación (9 / 12 usuarios) también es buena. Los datos revelan algunas ideas interesantes sobre la aplicación y uso del repositorio de experiencia en la empresa.

Las respuestas a las preguntas de la encuesta abierta revelan algunas ideas interesantes sobre la adopción y el éxito de la EPG / ER. Por ejemplo, el usuario que informó que no podía encontrar lo que quería en el repositorio de experiencia, pero, cuando se encontró en otro lugar, lo puso en el repositorio de experiencia "para el futuro", sugiere que la empresa ya tiene algunas características importantes de una organización de aprendizaje.

Los datos sugieren que un depósito simple y de bajo costo de la experiencia puede ser útil en una organización pequeña y pueden obtener beneficios en poco tiempo.

Se va a seguir evaluando el uso de la EPG / ER en los próximos seis meses. Un tiempo de evaluación ya se revelan las tendencias en el uso (aumentos o disminuciones), así como proporcionar más datos sobre los aspectos más útiles de la experiencia de depósito y en los beneficios o perjuicios.

2.7.4 Caso No. 4

Introducción a OpenUP (Proceso Unificado Abierto)

Los diferentes proyectos tienen diferentes necesidades de proceso. Los factores típicos dictan la necesidad de un proceso más formal o ágil, como el tamaño y la ubicación del equipo, la complejidad de la arquitectura, la novedad de la tecnología, de conformidad con las normas, entre otros. Sin embargo, existen buenas prácticas de desarrollo de software que beneficia a cualquier equipo de proyecto a ayudarles a ser más eficaces. Este documento presenta los elementos fundamentales de OpenUP - un proceso ágil y unificado que contiene el conjunto mínimo de prácticas que ayudan a que los equipos san más eficaces en el desarrollo de software. OpenUP abraza una filosofía pragmática y ágil que se centra en la naturaleza colaborativa de desarrollo de software.

¿Qué es OpenUP?

OpenUP es un proceso de desarrollo de software mínimamente suficiente - lo que significa que sólo el contenido fundamental se incluye. Por lo tanto, no proporciona orientación sobre muchos temas que los proyectos pueden tratar, tales como tamaño de los equipos grandes, el cumplimiento, las situaciones contractuales, la seguridad o aplicaciones de misión crítica, la tecnología de orientación específica, etc. Sin embargo, OpenUP es muy completo en el sentido de que se manifiesta como un proceso para construir un sistema. Para atender las necesidades que no están cubiertos en su contenido, OpenUP es extensible para ser utilizado como base sobre lo que se debe de tener en el proceso y se puede agregar o adaptar, según sea necesario.

OpenUP se basa en los casos de uso y escenarios, la gestión del riesgo, y un enfoque centrado en la arquitectura para impulsar el desarrollo.

OpenUP principios

OpenUP es impulsada por los cuatro principios fundamentales que se citan a continuación.

-Colaborar para alinear los intereses y la comprensión social.

-Equilibrar las prioridades que compiten para maximizar el valor para los accionistas.

-Se centran en la arquitectura de principios para minimizar los riesgos y organizar el desarrollo.

-Evolucionar para obtener continuamente retroalimentación y mejorar.

¿Cómo se organiza OpenUP?

OpenUP se organiza en dos dimensiones diferentes, correlacionadas: El contenido de método y contenido del proceso. El contenido de método es el método que los elementos (es decir, las funciones, tareas, los artefactos, y orientación) se definen, independientemente de la forma en que se utilizan en un ciclo de vida del proyecto. El contenido del proceso es donde los elementos de método se aplica en un sentido temporal. Muchos ciclos de vida diferentes para diferentes tipos de proyectos pueden ser creados desde el mismo conjunto de elementos de método

Áreas de Contenido

A nivel personal, los miembros del equipo en un proyecto de OpenUP contribuyen con su trabajo en micro-incrementos, que normalmente representan el resultado de unas pocas horas hasta unos pocos días de trabajo. La aplicación se desarrolla un micro-incremento en el tiempo y el progreso es visto con eficacia todos los días. Los miembros del equipo comparten abiertamente sus progresos diarios en micro-incrementos, lo que aumenta la visibilidad de trabajo, la confianza y el trabajo en equipo.

El proyecto se divide en iteraciones: planeacion, el tiempo de los intervalos se mide en semanas. OpenUP ayuda al equipo adecuado a centrar sus esfuerzos a través del ciclo de vida de la iteración, con el fin de ofrecer un valor incremental a los interesados de una manera predecible.

Las estructuras OpenUP del ciclo de vida del proyecto estan dentro de cuatro fases: Concepción, Elaboración, Construcción y Transición.

Roles

Las habilidades esenciales que necesitan los equipos pequeños están representados por los roles de OpenUP:

-Las partes interesadas.- representado los grupos de interés cuyas necesidades deben ser satisfechas por el proyecto.

-Analista.- Representa las preocupaciones del cliente y del usuario final mediante la recopilación de aportaciones de los interesados para entender el problema a resolver y por la captura y fijación de prioridades para los requisitos.

-El arquitecto es responsable de diseñar la arquitectura de software, que incluye la toma de decisiones técnicas clave que limitan el diseño y la implementación del proyecto.

-Desarrollador.- Es responsable de desarrollar una parte del sistema, incluyendo el diseño que se ajuste a la arquitectura, y luego la ejecución, la unidad de pruebas-, e integrar los componentes que forman parte de la solución.

-Tester.- Es responsable de las actividades centrales del esfuerzo de la prueba, tales como identificar, definir, implantar y llevar a cabo las pruebas necesarias, así como el registro de los resultados de las pruebas y análisis de los resultados.

-Gerente de Proyecto.- Es el de la planificación del proyecto, en colaboración con los actores y el equipo, las interacciones en coordinación con las partes interesadas, y mantiene el equipo del proyecto se centraron en el cumplimiento de los objetivos del proyecto.

-Cualquier Rol.- Es representado por cualquiera del equipo que pueda realizar tareas generales.

Disciplinas

El contenido del método OpenUP se centra en las siguientes disciplinas: requisitos, arquitectura, desarrollo, pruebas, gestión de proyectos y administración de configuración y cambio.

Otras disciplinas y áreas de interés se han omitido, como el modelado de negocio, medio ambiente, gestión de requisitos y la configuración avanzada de administración de configuración de herramientas. Estas preocupaciones son consideradas innecesarias para un proyecto pequeño o son manejados por otras áreas de la organización, fuera del equipo del proyecto.

Tareas

Una tarea es una unidad de trabajo de un rol que se le puede pedir para llevar a cabo. En OpenUP, hay 18 tareas que realizan los roles, ya sea como actores principales (el responsable de la ejecución de la tarea) y los intérpretes adicionales (de apoyo y suministro de información utilizados en la ejecución de la tarea). La naturaleza colaborativa de OpenUP se manifiesta por tener los actores principales de trabajo con un conjunto de otras personas al realizar una tarea.

Artefactos

Un artefacto es algo que es producido, modificado, o utilizado por una tarea. Los roles son responsables de la creación y actualización de los artefactos. Los artefactos están sujetas al control de versiones en todo el ciclo de vida del proyecto.

Son 17 artefactos los que son considerados escenciales en proyectos OpenUP los cuales deben ser utilizados para la captura de información de productos y relacionados con el proyecto. No hay obligación capturar información en artefactos formales. La información puede ser capturada de manera informal en la pizarra (por ejemplo, para el diseño y la arquitectura), notas de reuniones (por ejemplo, para la evaluación de estado), plantillas, etc.

Proceso

El método de contenido reutilizable se crea por separado de su aplicación en los procesos. El método proporciona paso a paso las explicaciones, describiendo cómo los objetivos específicos de desarrollo se consiguen independientemente de la ubicación de los elementos dentro de un método de ciclo de vida del desarrollo.

Método elementos se organizan en partes reutilizables de los patrones de la capacidad del proceso llamado, proporcionando un enfoque de desarrollo coherente a las necesidades del proyecto común. Estos patrones se hacen de la organización de tareas (a partir del contenido del método) en las actividades, agrupándolas en una secuencia que tiene sentido para el área específica donde se aplica ese modelo.

Los patrones pueden ser pequeños y enfocados en áreas específicas como la gestión de la iteración, la iniciación del proyecto, definición de la arquitectura y así sucesivamente. Estos se consideran los bloques básicos de construcción para crear patrones más grandes o los procesos de entrega.

OpenUP principios	**Declaración Manifiesto Ágil**
Colaborar para alinear los intereses y la comprensión de compartir	Individuos e interacciones sobre procesos y herramientas
Equilibrar las prioridades que compiten para maximizar el valor para los accionistas	Colaboración con el cliente sobre negociación de contratos
Se centran en la arquitectura de principios para minimizar los riesgos y organizar el desarrollo	De trabajo de software sobre la documentación completa
Evolucionar para obtener continuamente retroalimentación y mejorar	Responder al cambio sobre seguir un plan de

Tabla 6 Mapeo OpenUP y Manifiesto Ágil

Esta actividad provee una forma para llevar a cabo la planeación basado en objetivos y la ejecución de los trabajos. El trabajo es asumido por los desarrolladores, y el progreso del trabajo se hace un seguimiento sobre la base de los logros alcanzados mediante el diseño, el desarrollador a prueba, y el código fuente integrada.

El elemento de trabajo puede ser un caso de uso, una situación, un requerimiento de soporte o una solicitud de cambio. Un contexto se puede especificar cuando un elemento de trabajo se asigna a desarrollar, así como especificar en términos generales un elemento de trabajo se va a desarrollar en ese incremento. Desarrollo puede centrarse, por ejemplo, en una capa (por ejemplo, la interfaz de usuario, la lógica de negocio o acceder a bases de datos), o en un componente. Si el contexto se especifica o no, la responsabilidad de los desarrolladores es crear un diseño e implementación para que los elementos de trabajo y para escribir y ejecutar pruebas de desarrolladores en contra de la aplicación para asegurarse de que la ejecución de las obras tal como fue diseñado, tanto como una unidad y se integran en el código base

Como se mencionó antes, los bloques básicos de construcción se utilizan para crear modelos más grandes, por ejemplo, los patrones que se pueden utilizar como plantillas para las iteraciones - los patrones que contienen todas las actividades necesarias para una iteración particular dentro de una fase del proyecto.

OpenUP aplica las fases del proceso unificado: Concepción, Elaboración, Construcción y Transición. En conjunto, estos bloques de construcción básicos también se utilizan para hacer frente a los objetivos de cada fase.

OpenUP tiene un proceso de entrega para el desarrollo iterativo a lo largo de cuatro fases. Los patrones de plantilla de la iteración se aplican tantas veces como sea necesario, dependiendo del número de iteraciones que el equipo decide ejecutar en cada fase. Proyectos con necesidades diferentes pueden crear instancias de los patrones de repetición de plantilla diferente.

Un proceso de base

El proceso OpenUP, aunque completa en su cobertura del objetivo y el contexto, también sirve como un proceso base sobre el cual puede ser el contenido de proceso adicional integrado. Plug-ins pueden extender OpenUP para incluir la orientación a gran escala de técnicas. Proveedores de herramientas pueden crear sus propios plug-ins.

Además de esta comunidad de plug-ins que se pueden mezclar y combinar para construir un proceso a medida y posiblemente más importante para el objetivo de tener un proceso que cumpla con las necesidades específicas de un proyecto en particular, el proceso de la herramienta Eclipse Framework puede ser utilizado para crear contenidos. Plantillas específicas de la organización se pueden integrar en el contenido del proceso y los nuevos elementos de proceso de listas de control y orientación a las funciones de nuevo, las tareas y los productos del trabajo se pueden introducir.

OpenUP es un proceso mínimo, completo y extensible. Promueve técnicas ágiles y principios, mientras que tiene un ciclo de vida estructurado demostrado que destaca la entrega continua de la calidad del software que es valioso para los interesados.

2.7.5 Caso No. 5

Visión: Proceso Unificado para la Educación

UPEDU es un proceso de ingeniería de software. Proporciona un enfoque disciplinado para la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es garantizar la producción de software de alta calidad que satisfaga las necesidades de sus usuarios finales dentro de un horario predecible y presupuesto.

UPEDU tiene dos dimensiones: el eje horizontal representa el tiempo y muestra los aspectos del ciclo de vida del proceso de medida que se desarrolla el eje vertical representa los flujos de trabajo de procesos básicos, que las actividades de grupo, lógicamente por la naturaleza.

La primera dimensión representa el aspecto dinámico del proceso como entre en vigor, y se expresa en términos de fases, iteraciones e hitos.

La segunda dimensión representa el aspecto estático del proceso: la forma en que se describe en términos de los componentes del proceso, las actividades, flujos de trabajo, artefactos y roles.

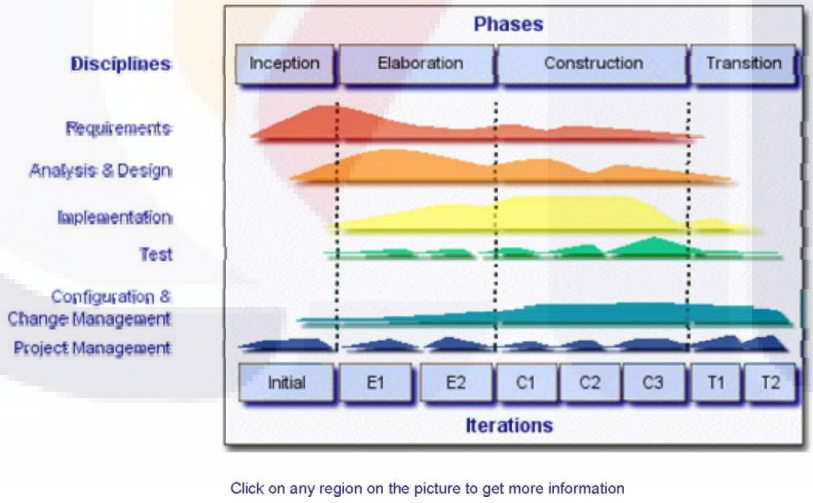


Ilustración 2 Ciclo de vida y de las Fases

Las fases y los hitos de un proyecto

Desde una perspectiva de gestión, el ciclo de vida del software del proceso unificado para la Educación (UPEDU) se divide en cuatro fases secuenciales, cada una es al final importante, cada fase es esencialmente un espacio de tiempo entre dos hitos importantes. En cada fase final de una evaluación se realiza para determinar si los objetivos de la etapa se han cumplido. Una evaluación satisfactoria permite que el proyecto pase a la siguiente fase.

Todas las fases no son idénticas en términos de lo previsto y el esfuerzo. Aunque esto varía considerablemente en función del proyecto, un ciclo típico de desarrollo inicial de un proyecto de tamaño mediano debe anticipar la siguiente distribución entre el esfuerzo y el horario:

	Comienzo	Elaboración	Construcción	Transición
Esfuerzo	~5 %	20 %	65 %	10%
Horario	10 %	30 %	50 %	10%

Tabla 7 Tiempos estimados

Para un ciclo de evolución, las fases de creación y elaboración serían considerablemente menores. Herramientas que puede automatizar una parte de los esfuerzos de construcción pueden mitigar esta situación, haciendo la fase de construcción mucho más pequeña que el inicio y las fases de elaboración en conjunto.

Una sola pasada a través de las cuatro fases es un ciclo de desarrollo, cada uno pasan a través de las cuatro fases produce la creación de software. A menos que el producto "termine", que se convertirá en su próxima generación mediante la repetición de la misma secuencia de fases creación, elaboración, construcción y transición, pero esta vez con un énfasis diferente en las distintas fases. Estos ciclos posteriores se denominan ciclos de evolución. A medida que el producto pasa por varios ciclos, las nuevas generaciones se producen.

Los ciclos de la evolución pueden ser provocados por las mejoras sugeridas por el usuario, los cambios en el contexto de usuario, los cambios en la tecnología subyacente, la reacción a la competencia, y así sucesivamente. Los ciclos de evolución suelen tener lapsos mucho más cortos a las fases de elaboración, desde la definición del producto básico y la arquitectura están determinadas por los ciclos de desarrollo anterior.

El UPEDU ofrece conceptos clave básicos claves del proceso son: repeticiones, las fases, los riesgos, pruebas de rendimiento, y así sucesivamente, se introducen en secciones separadas del proceso. El UPEDU proporciona directrices con información práctica sobre las técnicas para ayudarle a realizar ciertas tareas durante todo el proceso y el uso de artefactos específicos.

Introducción

En el UPEDU, muchos artefactos se derivan de un estudio de caso del proyecto y se utilizan como ejemplos o plantillas. Estos ejemplos se pueden utilizar como punto de partida sobre el uso de las plantillas de los artefactos y darle una mejor comprensión de los propios artefactos. En las secciones siguientes se explica brevemente qué es un caso de estudio y cómo se puede visualizar estos ejemplos en el UPEDU.

2.7.6 Caso de Estudio TMT (herramienta de monitoreo en tiempo)

Breve reseña de la TMT

El sistema de software diseñado es una herramienta de seguimiento de tiempo, lo que se conoce como "TMT".

El Time Monitoring Tool (TMT) permite a los desarrolladores que trabajan dentro de un proceso de desarrollo de software definido, registrar el tiempo dedicado a las diferentes actividades de desarrollo de software, como el diseño, codificación, pruebas, o la depuración. El TMT también permite a un administrador obtener análisis e informes sobre la base de los datos en el sistema.

El TMT se puede utilizar en cualquier aplicación de desarrollo de software para registrar los recursos destinados a las actividades de desarrollo de software. Los desarrolladores utilizan TMT para registrar las actividades que se llevan a cabo. La ordenación del uso de los datos registrados a partir de TMT para validar su planificación, presupuestos y horarios. El administrador de procesos de software utiliza los datos de TMT para comprender mejor los procesos de software de diversas prácticas prescritas y en la guía se refiere a la mejora de procesos software.

El equipo del caso de estudio

Los ejemplos de casos de estudio se han creado durante un curso llamado Estudio de Ingeniería de Software (INF 4315) dado en la École Polytechnique de Montreal a los estudiantes de cuarto año durante el invierno de 2001.

El proyecto debía llevarse a cabo dentro de un semestre de 13 semanas por un equipo de cinco estudiantes. A cada alumno se espera que pasen 135 horas en este proyecto que suma hasta 675 horas-persona. Los estudiantes ya ha completado un primer curso de ingeniería de software y un curso avanzado sobre la ingeniería de software o en proceso de ingeniería de software. El conocimiento de JAVA se recomienda para este proyecto.

Uno de los proyectos presentados fue seleccionado como el caso de estudio que se presentará en el UPEDU. Entonces, todos los artefactos se han revisado, actualizado y controlado por el equipo de UPEDU pero la información directa se mantiene intacta.

¿Cómo puedo acceder a los ejemplos de estudios de caso?

En primer lugar, todos los ejemplos de estudios de caso-se encuentran en el Formato de Documento Portátil (.PDF). La mayoría de los artefactos presentados en el UPEDU se utilizaron en el proyecto TMT. Por lo tanto, con cada página Descripción artefacto en cuestión, hay una sección titulada Plantillas, caso de estudio, informes.

Allí, se encuentran las plantillas relacionadas (en formato Word), ejemplos de casos de estudio y la plantilla del informe con el artefacto. El icono rojo se utiliza para todos los casos de estudio. Al hacer clic en ese enlace se abrirá una ventana nueva del navegador que contiene el ejemplo elegido. Todas las páginas de descripción de los artefactos se encuentran en los conjuntos de artefactos.

Esta página está disponible directamente en el marco UPEDU navegación. O puede utilizar el siguiente enlace: Plantillas de desarrollo de software.

La página ofrece una visión general de todas las plantillas de los artefactos disponibles, ejemplos de estudios de caso y los informes ordenados por disciplinas. El mismo patrón es utilizado

2.8 Lecciones aprendidas

Cómo hemos visto en los capítulos anteriores, podemos establecer por tanto una serie funcionalidades básicas que oferta una EPG:

Entender y desarrollar tanto el proceso como las actividades y tareas que lo componen, de forma que la gestión de los elementos de trabajo se convierta en una tarea sencilla.

Generar la documentación del proyecto sin que esto suponga un problema para el equipo de trabajo.

Conocer en todo momento cada una de las responsabilidades de cada miembro del equipo de trabajo.

Se considera necesaria la formalización del proceso de generación de una EPG, debido a que ayuda a gestionar la calidad del proceso de generación y del producto final desarrollado.

La formalización del proceso de generación de una EPG además permite la automatización de las actividades involucradas en el proceso de generación al tener una definición formal de cada actividad.

Finalmente, cabe destacar el hecho de que las Guías Electrónicas de Procesos, pueden presentarse hoy como una alternativa real, eficaz y eficiente dentro del marco definido por la Mejora del Proceso Software (SPI, en inglés).

En base a lo anterior, los principales beneficios del uso de una guía de procesos es que nos facilitan la comunicación en las cuales los participantes del proceso pueden cumplir pasos definidos en donde están relacionados con alguna parte del proceso.

El avance de la tecnología permite que los procesos se desarrollen y aumenten a la par según las necesidades de la empresa que emplea una guía

En cuanto a los stakeholders involucrados pueden estar al pendiente de los pasos que se tienen que cumplir o como van a ir en comparación a proyectos anteriores, creando con esto una mayor eficiencia en el desarrollo eligiendo mejores alternativas para realizar el trabajo de manera ágil y eficaz.

Ayudar a los participantes a informarse sobre el proceso a realizar (por ejemplo, la consulta de la información de estado podría ayudar a los participantes a comprender lo que necesitan hacer para volver a empezar a trabajar después de una interrupción).

También cabe recalcar que el mal diseño o el mal uso de una guía pueden provocar que la información clave no se encuentre reflejada o que aquellos qué interactúan con la guía no puedan navegar de manera eficaz.

Se debe de cuidar que la información no se encuentre mezclada y sea exclusiva del proyecto en desarrollo, por lo cual un mal control de versiones puede provocar muchos atrasos.

Sin embargo, no todo son ventajas puesto que también existen algunos inconvenientes. Entre ellos la resistencia al cambio y la dificultad de codificación cuando los procesos son largos y complejos. Existen organizaciones donde hay equipos de trabajo que están acostumbrados a hacer uso de las guías de procesos tradicionales, que se encuentran familiarizados con este tipo de guías y que no quieren cambiar. Esta resistencia al cambio puede suponer un problema grave si no se afronta de forma adecuada. Además, en algunas ocasiones las organizaciones tienen definidos procesos largos y complejos, cuya codificación en una EPG puede llegar a resultar una tarea difícil, y su mantenimiento puede complicarse en gran medida si los procesos cambian frecuentemente. A pesar de las desventajas, las EPG pueden llegar a ser una forma eficaz de transferir el conocimiento de los procesos, siempre y cuando las EPG se desarrollen teniendo en cuenta un conjunto de objetivos y requisitos básicos.

En los siguientes capítulos se deberá analizar el desarrollo de una EPG, desde su diseño, arquitectura, elementos de trabajo, procesos, actividades y roles

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Metodología

3. Metodología

Tras el análisis exhaustivo efectuado sobre las Guías de Proceso, y centrándonos ya únicamente en las Guías Electrónicas de Procesos que son las que realmente nos interesan, se intentará formalizar el proceso de desarrollo de una EPG para establecer un método que sea útil posteriormente que ayude a mejorar el proceso de consulta y aprendizaje.

3.1 Diseño de la EPG

Cada uno de los Procesos se compone de un conjunto de Actividades que definen cómo debemos hacer las cosas, e involucra un conjunto de Roles. A su vez, las actividades se componen de un conjunto de Tareas que definen cómo realizar las actividades. En su realización participan unos determinados roles y, como resultado, se obtienen un conjunto de Productos de Trabajo. Además, cada actividad gestiona uno o varios Elementos de Trabajo a través de la realización de las tareas que la componen.

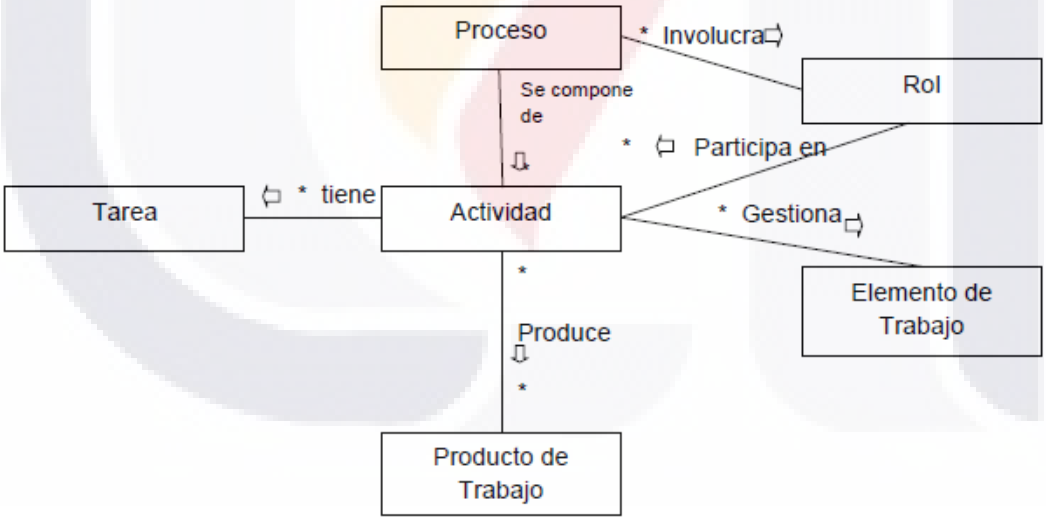


Ilustración 3 Estructura de EPG

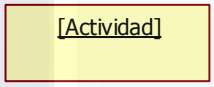
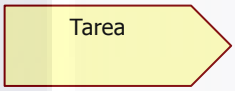
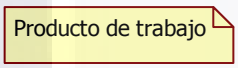
Imagen	Definición
	Un proceso representa una relación entre las instancias de actividades y el uso de los roles en las instancias.
 **Fase**	La fase representa un periodo significante en un proyecto
	La actividad es una unidad general de trabajo asignable a ejecutores específicos representados por el uso de un rol.
	Una tarea es un elemento que contiene el método y la definición de trabajo que indica como el trabajo es ejecutado por los roles.
	Un producto de trabajo es un elemento que es usado, modificado, y producido por tareas.
	Los roles son usados por las tareas para definir quién las ejecuta, así como también para definir un conjunto de productos de trabajo de los cuales esta encargado.
	Una herramienta de definición es un método especial de elementos de contenido que puede ser usado para especificar la participación de una herramienta en una tarea Definición.

Tabla 8 Elementos de un EPG

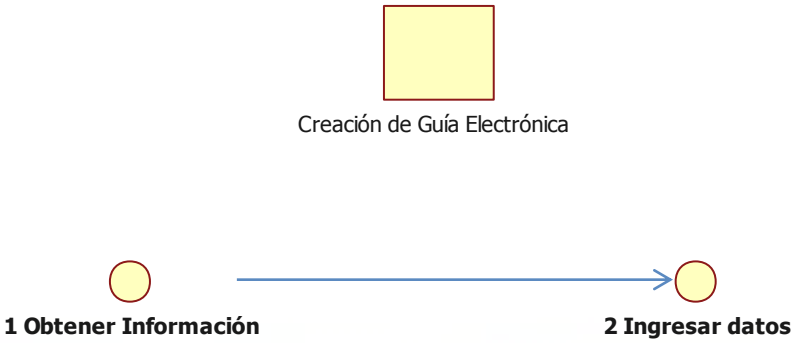


Ilustración 4 Fase general de proceso

La obtención de información no es relevancia para el diseño de la creación de la guía y por tanto no se entrará en más detalles relacionados con la recopilación de datos. Su función principal es recopilar todos los datos necesarios para la creación de una guía electrónica de procesos.

La segunda fase se divide en las siguientes actividades, cada una de ellas dividida en tareas, las cuales serán descritas a continuación.

- Insertar datos iniciales en la Guía de Procesos
- Creación de Elementos de Trabajo
- Creación de Proceso
- Creación de Actividad
- Creación de Rol
- Creación de Producto de Trabajo
- Creación de nueva versión de la EPG

3.2 Ingresar datos a la guía electrónica de procesos.

En la primera actividad se introducirán una descripción inicial sobre la Guía de procesos. En esta primera actividad se intentará introducir los datos básicos que el usuario debe conocer a la hora de realizar una Guía Electrónica de Procesos.

- Conceptos
- Gobierno de la Organización Software
- Ciclos e iteraciones
- Modelo de Equipo
- Principios
- Cultura Organizativa

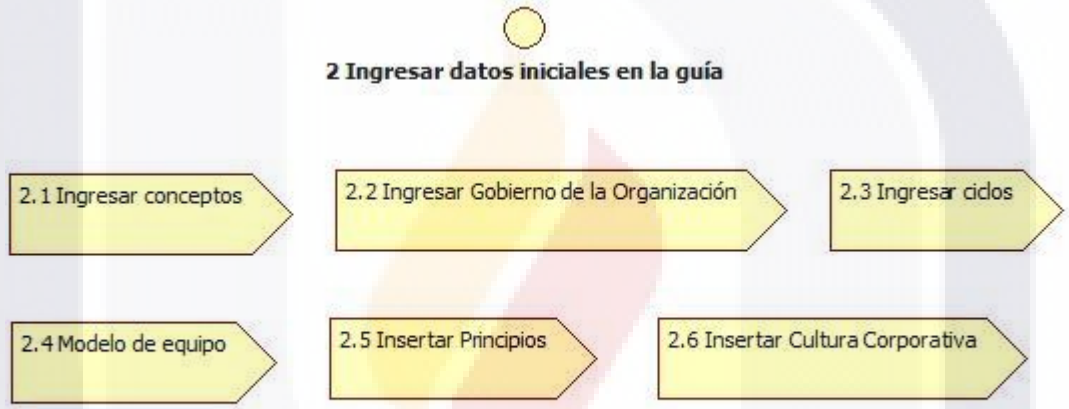


Ilustración 5 Ingresar datos iniciales

Desglosando las actividades antes mencionadas quedaría de la siguiente manera:

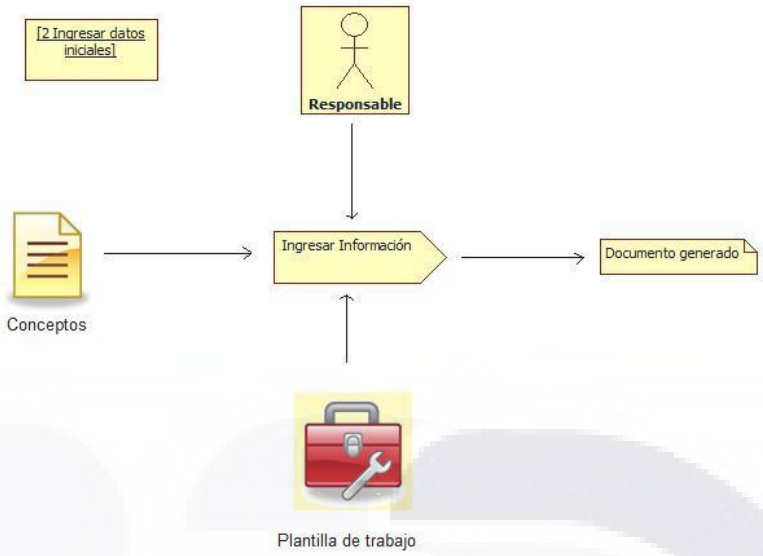


Ilustración 6 Introducción de conceptos

En 2.1 Ingresar datos iniciales, se introducirán los conceptos que más se van a usar a lo largo del desarrollo de la EPG.

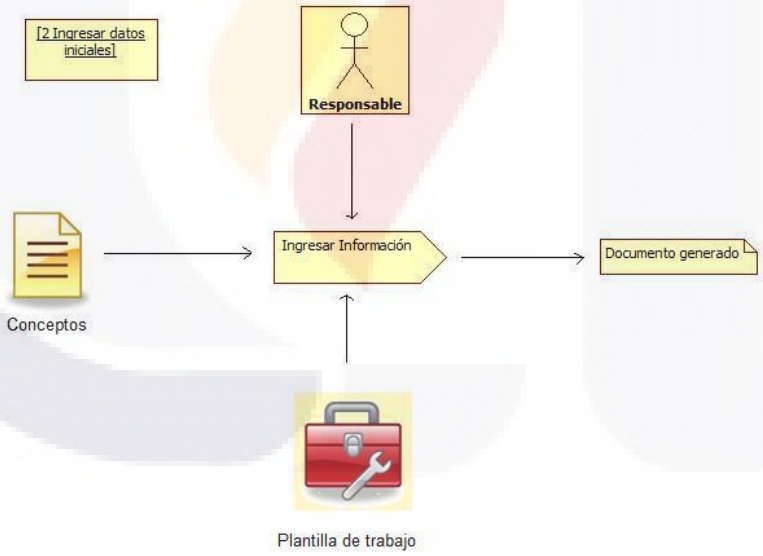


Ilustración 7 Introducción de Gobierno

En 2.2 Ingresar Gobierno de la Organización hace referencia al control de tiempo y dinero en relación al flujo de valor.

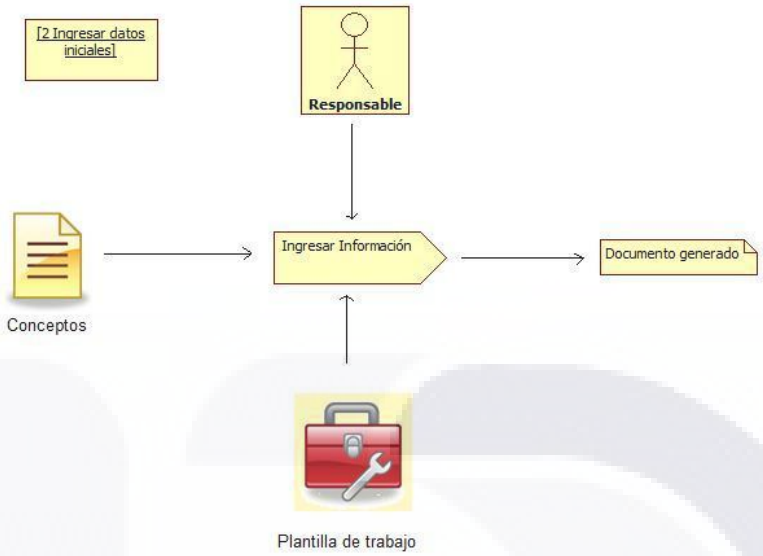


Ilustración 8 Introducción de ciclos

En 2.3 Ciclos se definen los ciclos que se llevan a cabo en el proyecto.

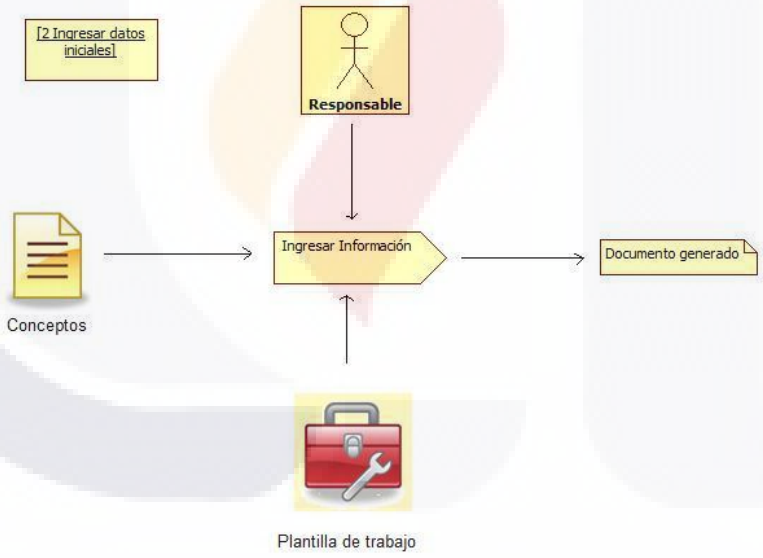


Ilustración 9 Introducción de modelo

En 2.4 En Modelo de Equipo a seguir, se describe un breve enfoque para estructurar personas y actividades para realizar con éxito proyectos de desarrollo de software.

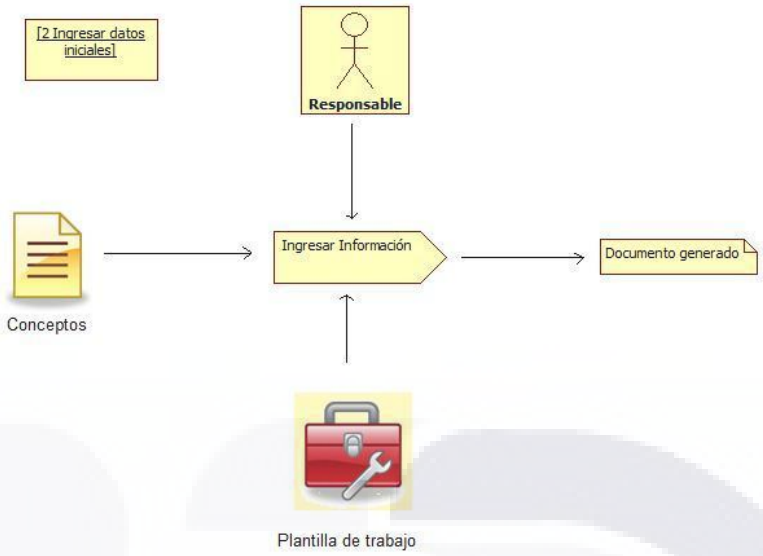


Ilustración 10 Introducción de principios

En 2.5 Principios que se han de seguir para desarrollar una EPG

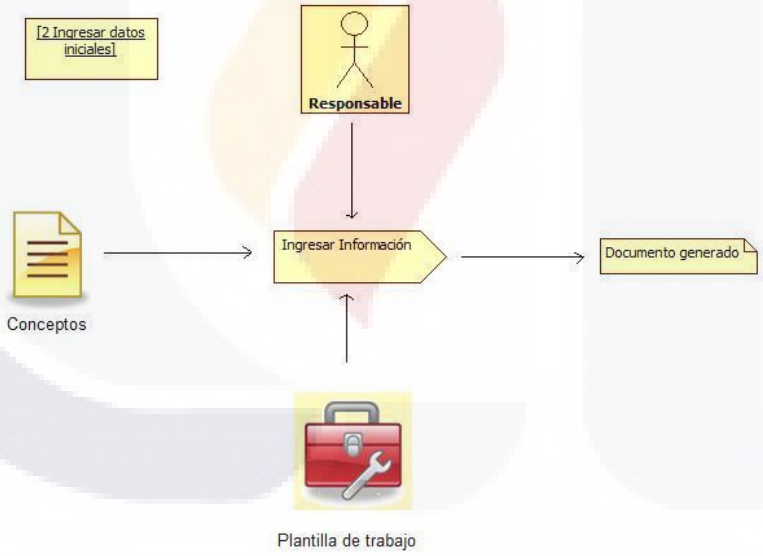


Ilustración 11 Introducción de cultura

En 2.6 En Cultura Corporativa se definen algunos conceptos abstractos que se utilizarán para guiar y restringir actividades concretas.

3.3 Creación de Elemento de Trabajo

En la segunda actividad que crearán de los Elementos de Trabajo. En ella se tienen en cuenta tanto la inserción de los Elementos de Trabajo en la Lista, como la definición detallada de cada uno de ellos.

Ilustración 12 Creación elementos de trabajo

Desglosando las actividades antes mencionadas quedaría de la siguiente manera:

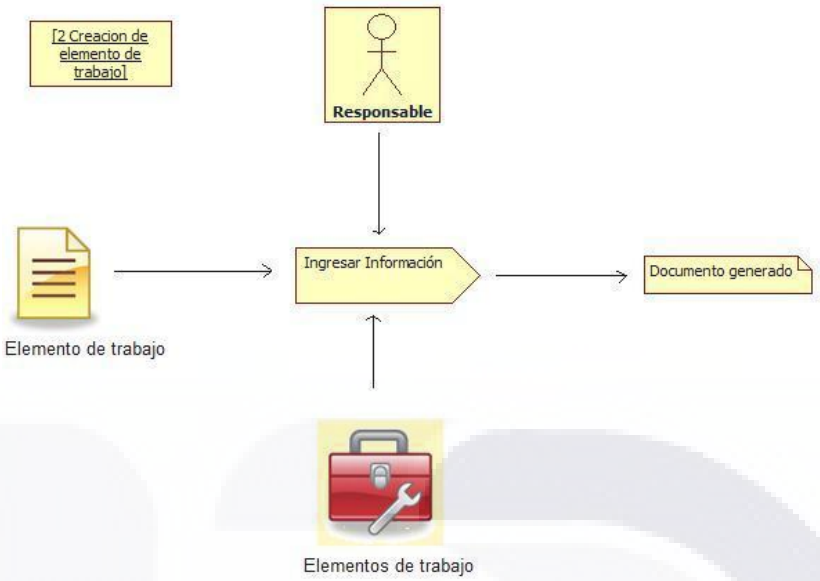


Ilustración 13 Insertar elemento de trabajo

En la primera tarea de esta actividad se inserta el Elemento de trabajo en la lista, para luego poder definirlos.

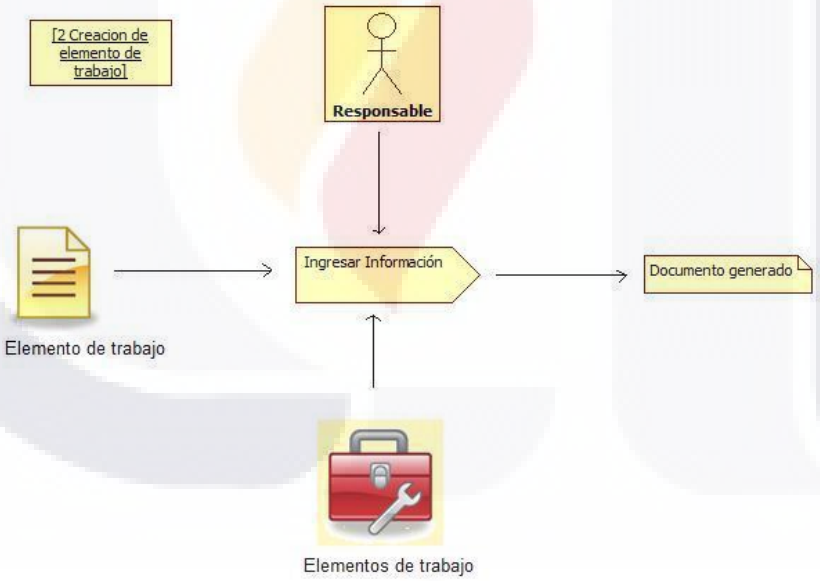


Ilustración 14 Insertar descripción

Una vez que se tiene el elemento insertado en la lista, se abrirá la plantilla de Elemento de Trabajo, seleccionando el nombre del Elemento de Trabajo que se va a definir y una breve descripción del mismo.

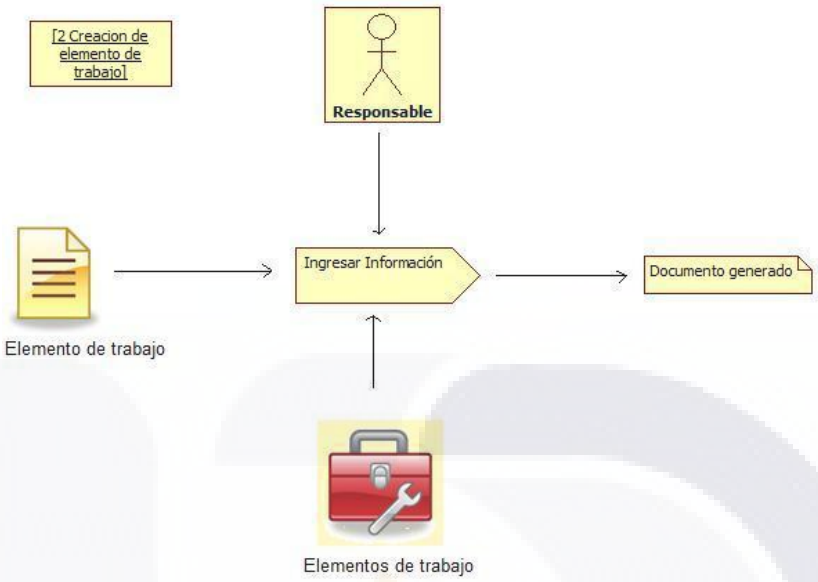


Ilustración 15 Insertar diagrama de estados

En la segunda tarea de esta actividad, se introduce en la plantilla la ruta donde tenemos almacenada la imagen del diagrama de Estados de este Elemento.

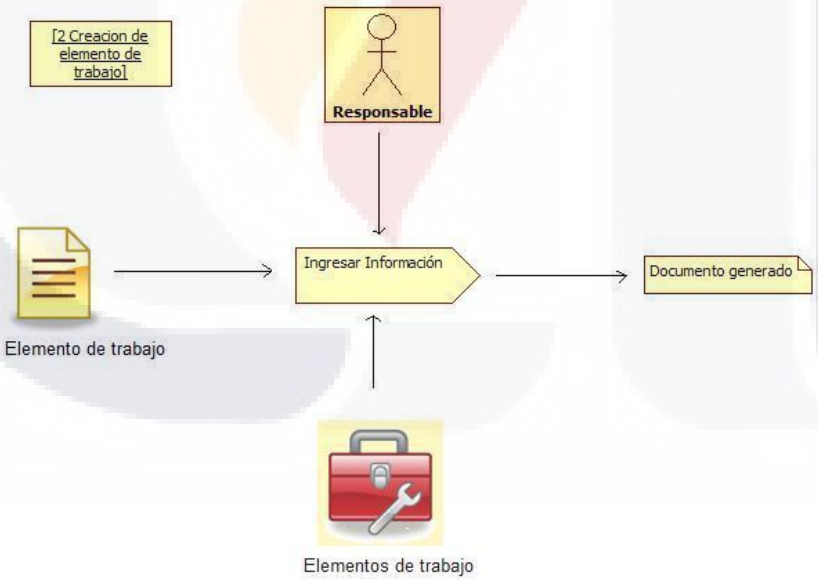


Ilustración 16 Definir estados

Una vez que se definen los Estados, en cada uno de ellos se describirá las transiciones que les afecten.

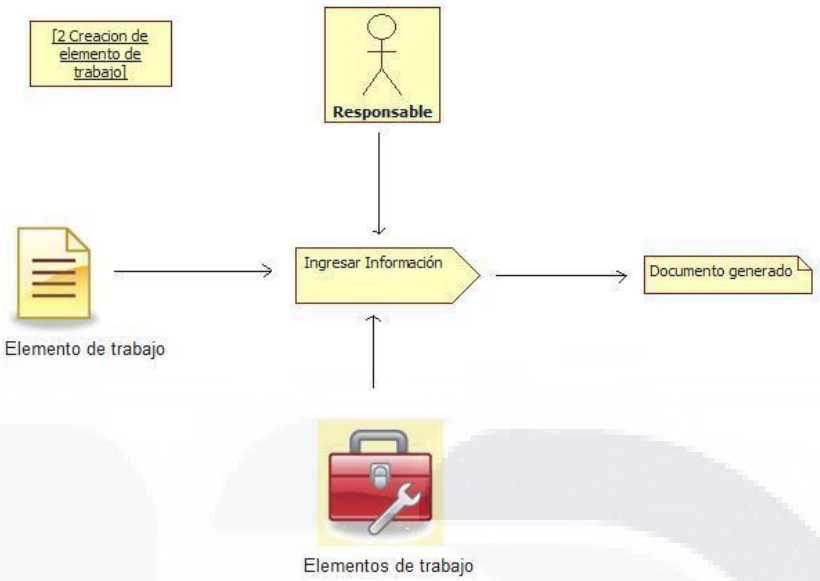


Ilustración 17 Crear transiciones

Una vez que se definen los Estados, en cada uno de ellos se describirá las transiciones que les afecten.

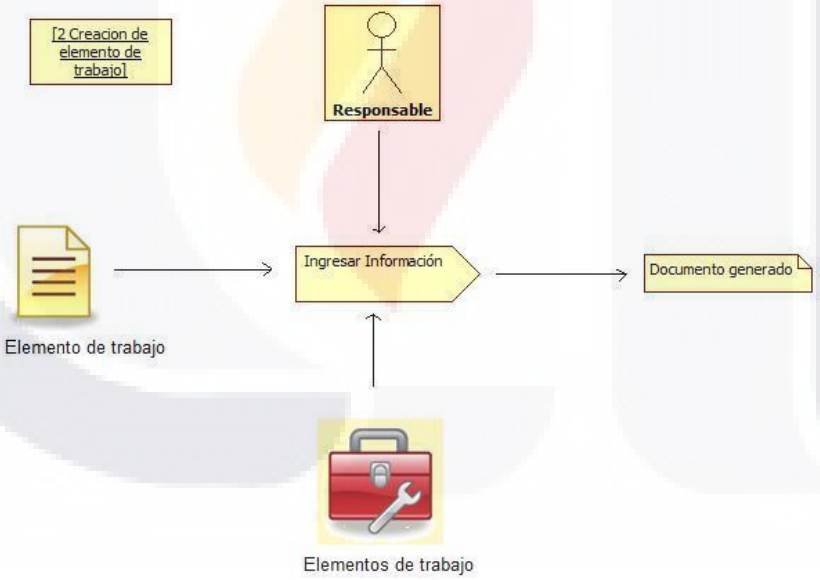


Ilustración 18 Insertar campos

Para poder utilizar un campo en un Elemento de Trabajo, antes es necesario haberlo definido en la lista de Campos

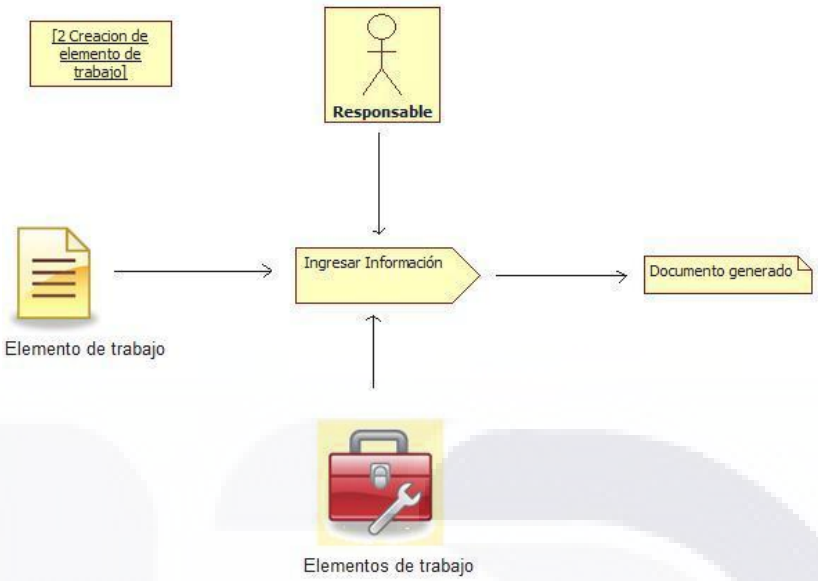


Ilustración 19 Introducir campos

En la séptima tarea se utilizarán los campos definidos para “describir” un Elemento de Trabajo.

3.4 Creación de Proceso

Es muy similar al Elemento de Trabajo. Es imprescindible insertar el elemento en la lista, antes de empezar a rellenar la plantilla con la descripción. La secuencia entre las siguientes tareas es la siguiente:

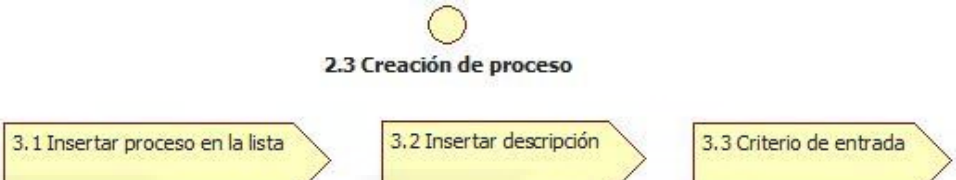


Ilustración 20 Creación de proceso

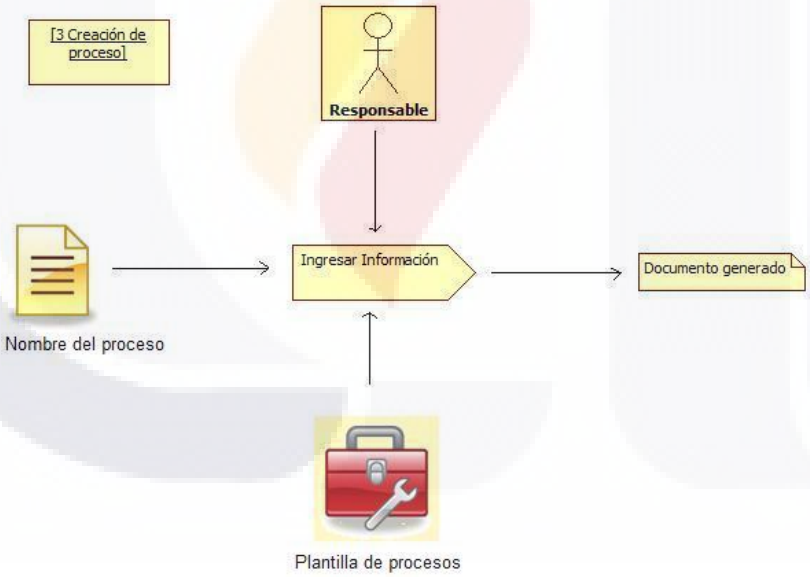


Ilustración 21 Introducción de proceso en la lista de procesos

En la primer actividad e introduce el proceso a la lista de procesos

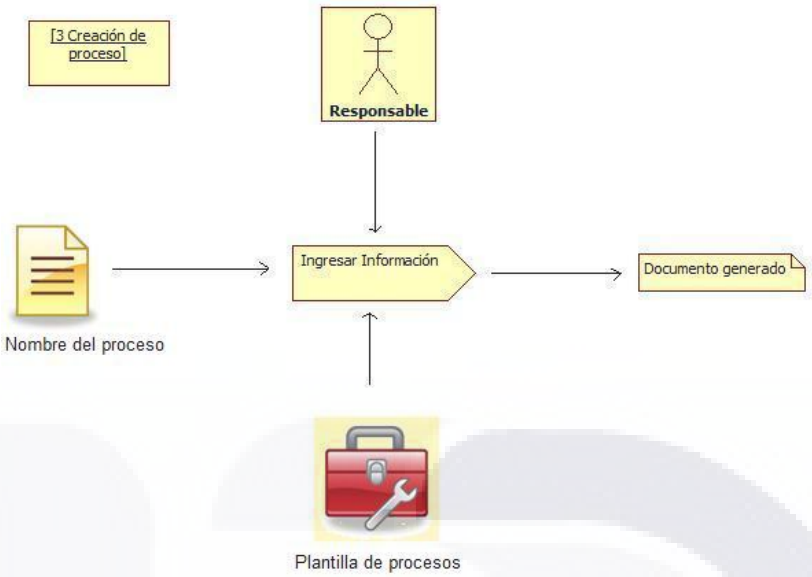


Ilustración 22 Introducción de descripción de procesos

En esta actividad se introduce una breve descripción del mismo.

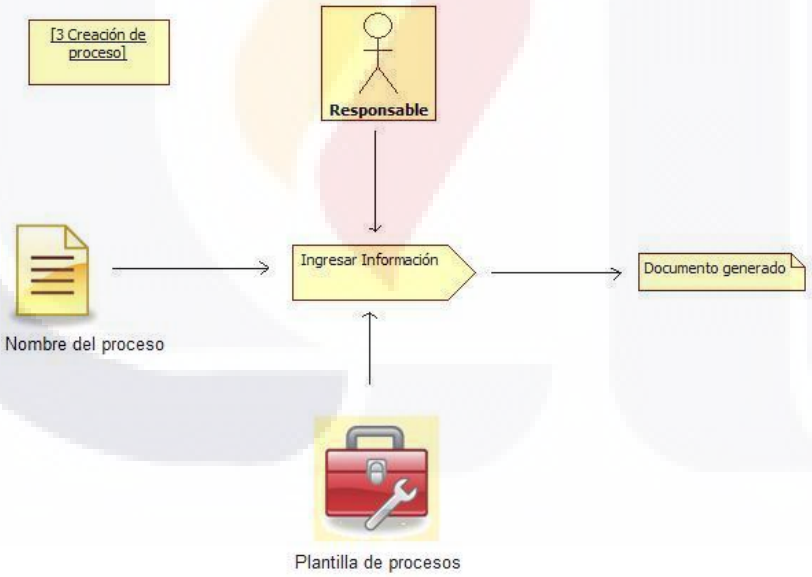


Ilustración 23 Introducción de criterios de entrada

En la tercera tarea de esta actividad, se debe introducir los criterios de entrada que se deben cumplirse antes de ejecutar dicho proceso.

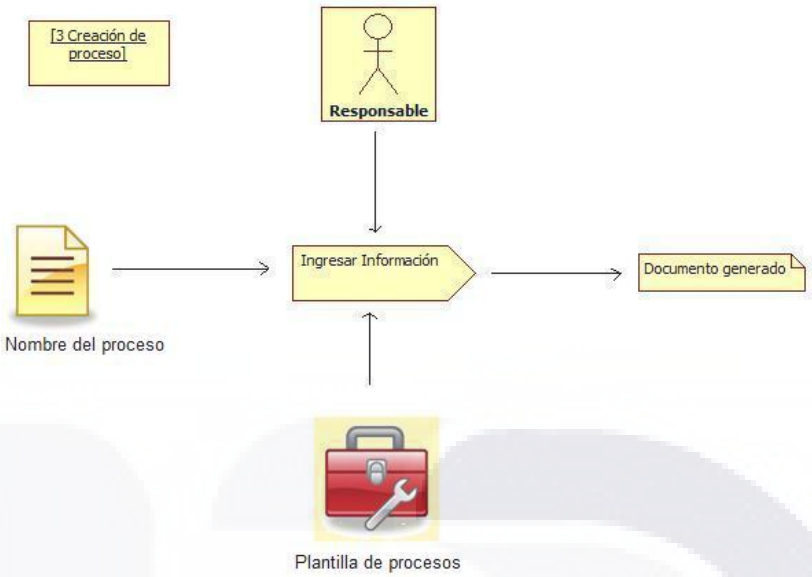


Ilustración 24 Introducción de tiempos de ejecución

Se debe introducir en esta tarea el momento en el que se realizará dicho proceso.

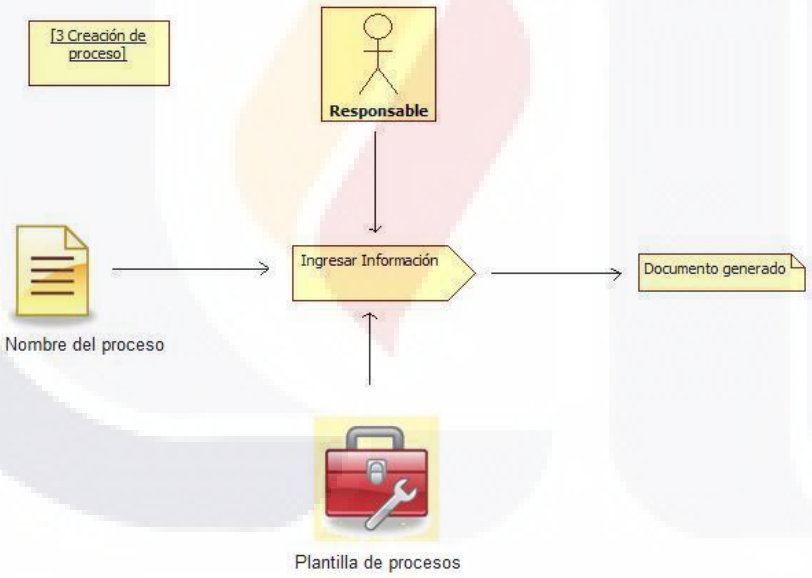


Ilustración 25 Introducción de criterios de salida

Se introducen los criterios de salida que se cumplen después de realizar el proceso.

3.5 Creación de Actividad

En esta fase se crean las actividades y todos los campos que la contienen.

La secuencia entre las siguientes tareas es la siguiente:

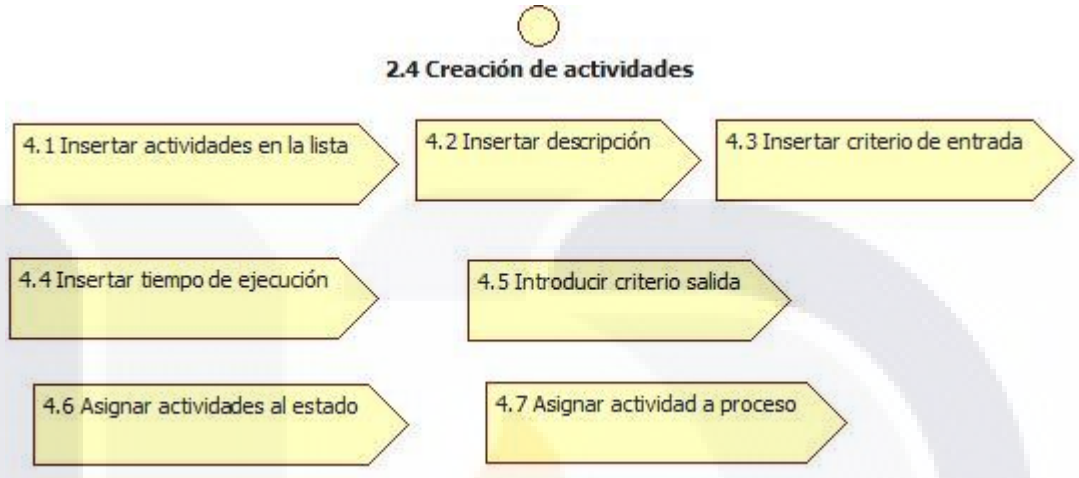


Ilustración 26 Creación de actividades

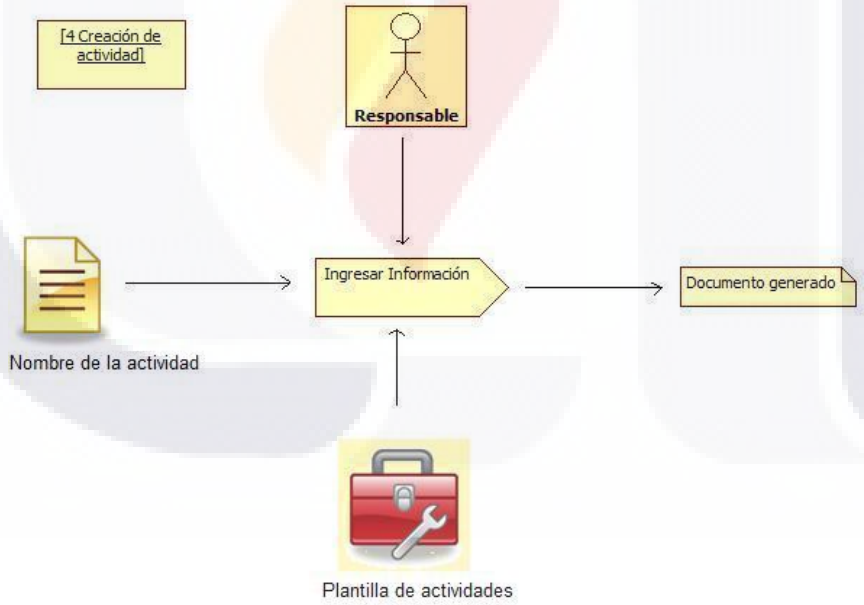


Ilustración 27 Introducción de actividades

Se realiza la introducción de actividades en la lista de actividades

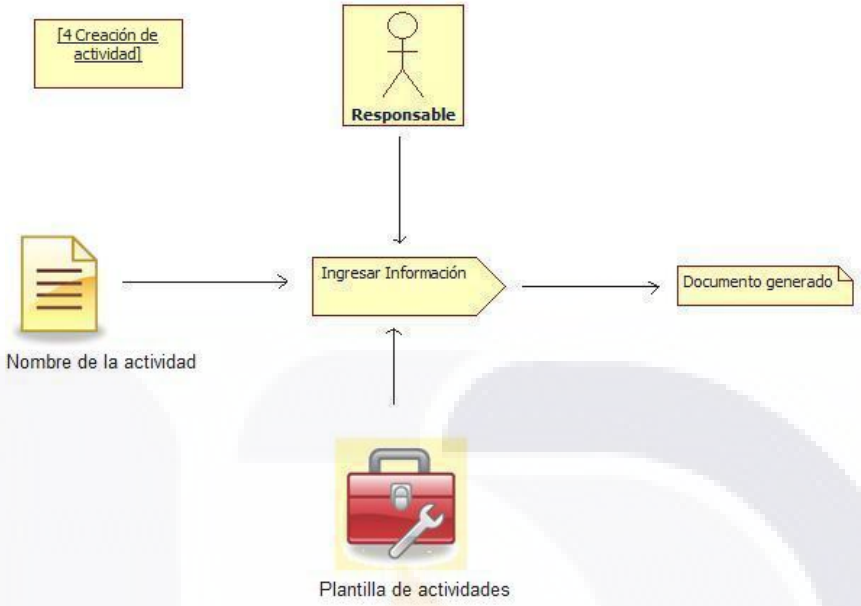


Ilustración 28 Introducción de descripción

Se realiza una breve descripción de las actividades que fueron agregadas

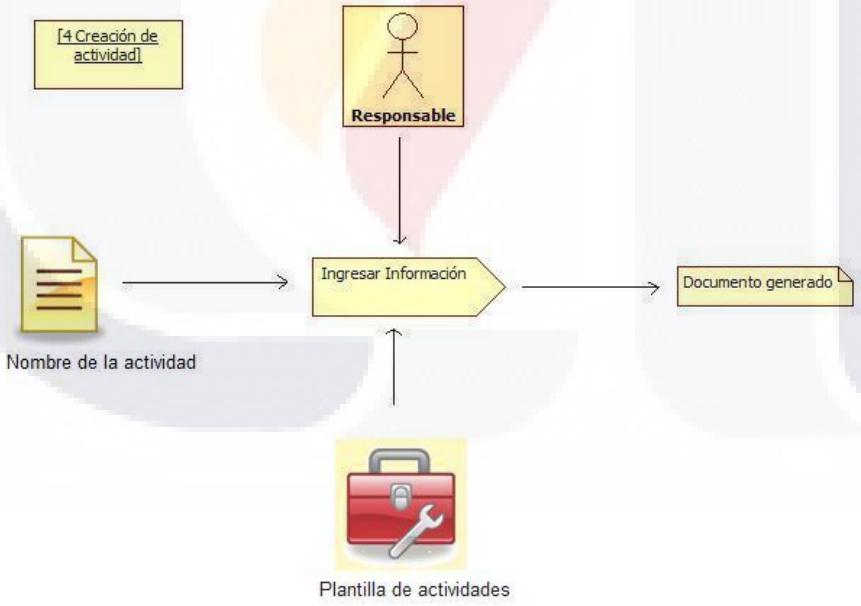


Ilustración 29 Introducción de criterios de entrada

Se Introducen los criterios de entrada para llevar a cabo la actividad

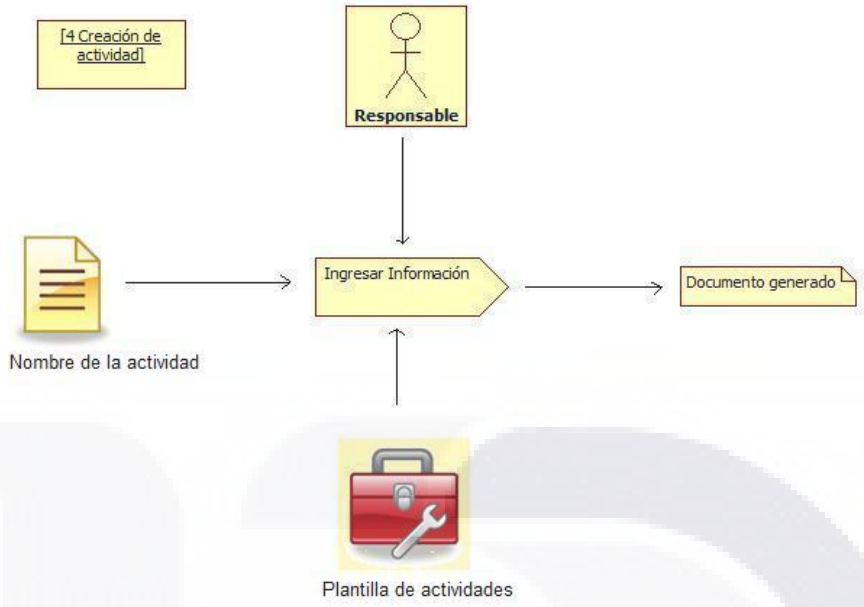


Ilustración 30 Introducción de tiempos de ejecución

Se Introducen los criterios de cuando se llevaran a cabo las actividades.

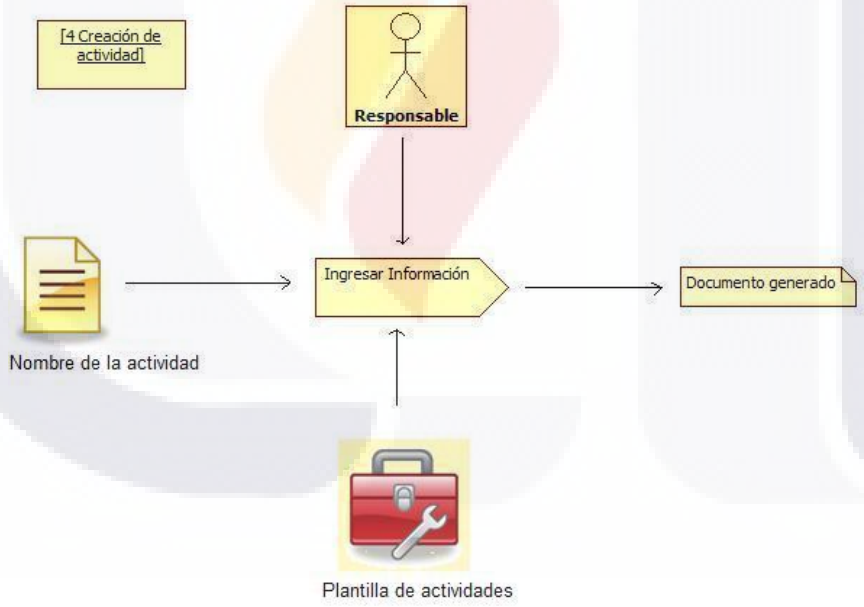


Ilustración 31 Introducción de criterios de salida

Se Introducen los criterios de salida de las actividades

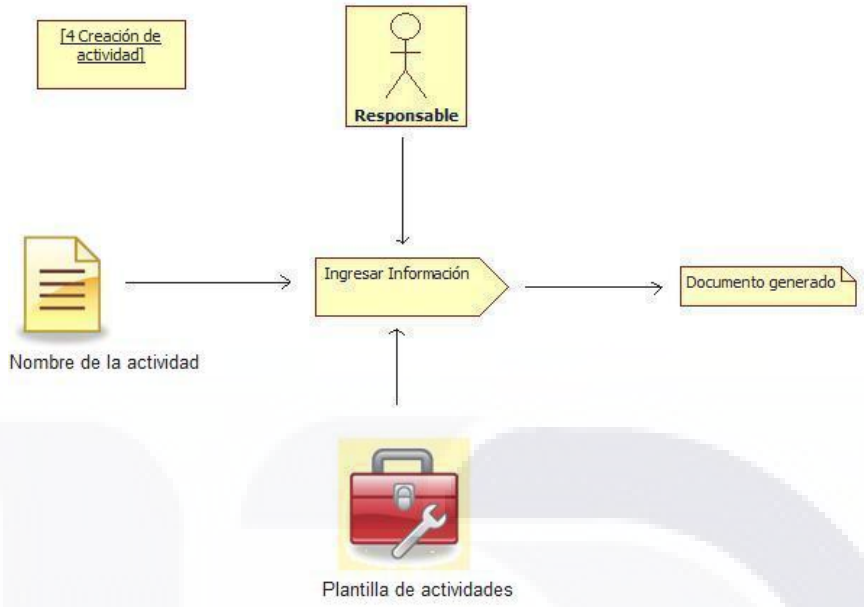


Ilustración 32 Vincular actividad al estado

Una vez que se tienen definidos los Elementos de Trabajo, las Actividades y los procesos, es necesario vincularlos entre ellos con el estado.

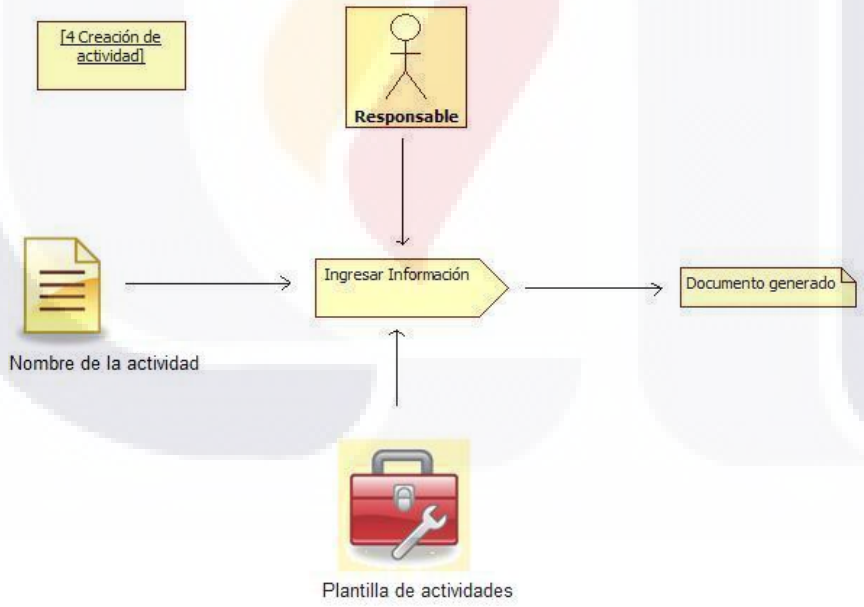


Ilustración 33 Vincular actividad al proceso

Una vez que se tienen definidos los Elementos de Trabajo, las Actividades y los procesos, es necesario vincularlos entre ellos con el estado.

3.6 Creación de roles

Se deben definir también los roles que van a participar en la creación elementos de la Guía de Procesos. Para ello debemos introducir todos los datos necesarios. La secuencia entre las anteriores tareas es la siguiente:

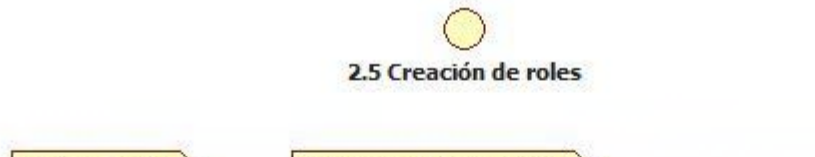


Ilustración 34 Creación de roles

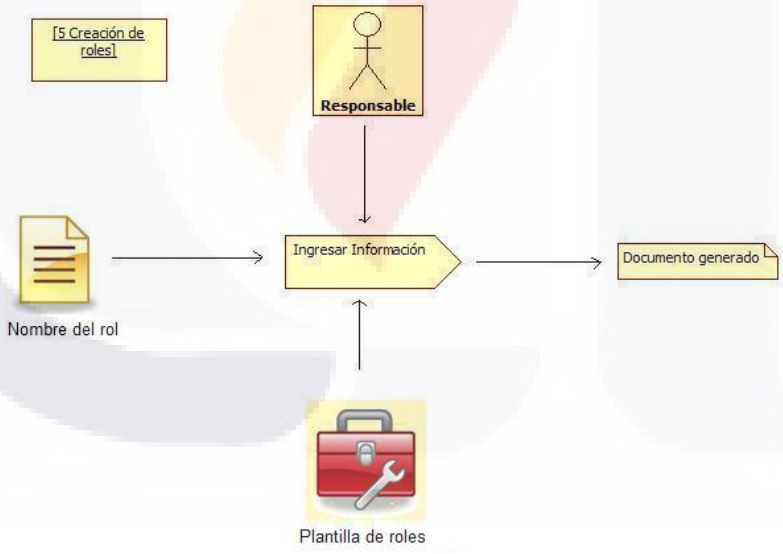


Ilustración 35 Ingresar nuevo rol

En esta actividad se introducirá el nuevo rol

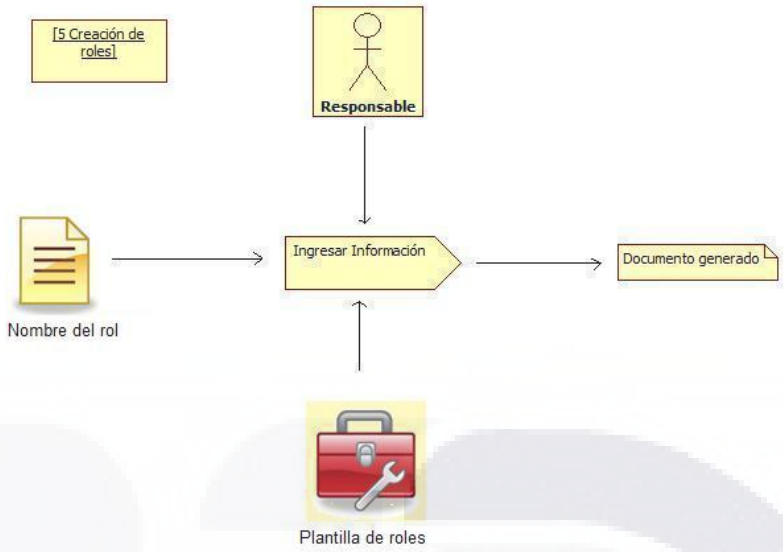


Ilustración 36 Ingresar descripción del rol

En esta actividad se introducirá la descripción del nuevo rol

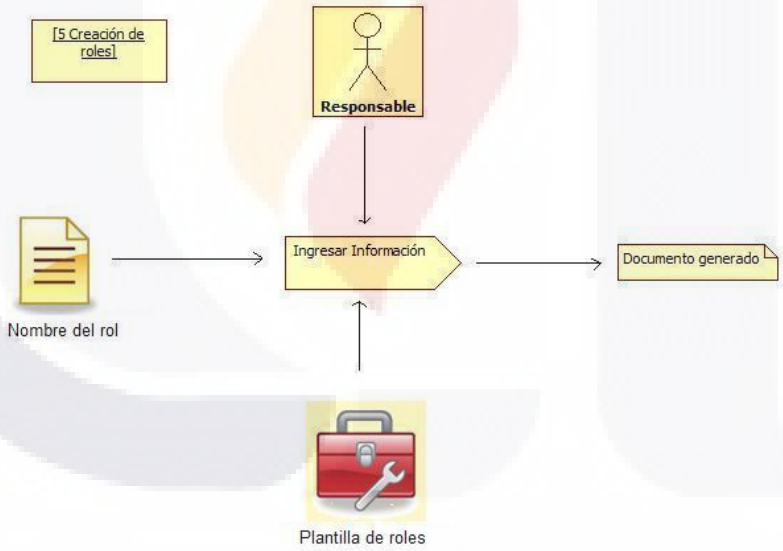


Ilustración 37 Ingresar rol en proceso

Igual que insertamos las actividades en los procesos y en los Elementos de Trabajo, ahora debemos insertar los roles en los procesos y Actividades en los que dicho Rol esté involucrado.

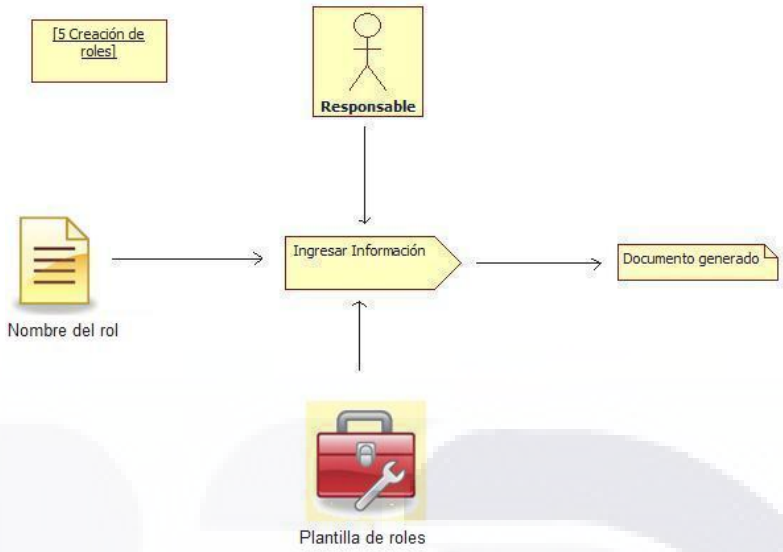


Ilustración 38 Ingresar rol en actividad

Igual que insertamos las actividades en los procesos y en los Elementos de Trabajo, ahora debemos insertar los roles en los procesos y Actividades en los que dicho Rol esté involucrado.

3.6 Creación de productos de trabajo

Un Producto de Trabajo es una abstracción general que representa el resultado de un proceso. Los Productos de Trabajo incluyen: Artefactos, Entregables, Resultados.

Las tareas poseen producto de trabajo de entrada y de salida. Los roles utilizan productos de trabajo para realizar tareas, y generar otros productos de trabajo al realizar estas tareas. Los productos de trabajo son de responsabilidad de un solo rol, de modo que las responsabilidades sean fáciles de identificar y comprender, y para promover la idea de que cada pieza de información del proceso requiere un conjunto de habilidades apropiado. Aunque un rol puede poseer un producto de trabajo, otros roles pueden usar ese producto de trabajo, para actualizarlo o para otros fines

Ilustración 39 Productos de trabajo

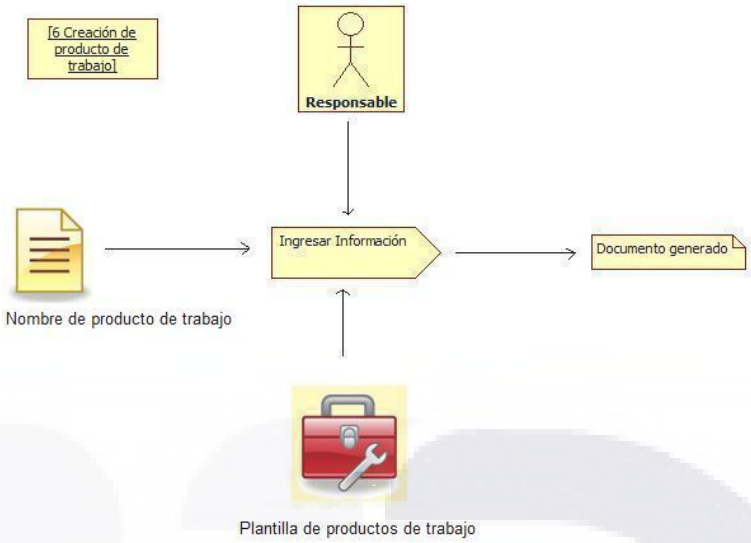


Ilustración 40 Introducir productos de trabajo

En esta actividad se introducen los productos de trabajo

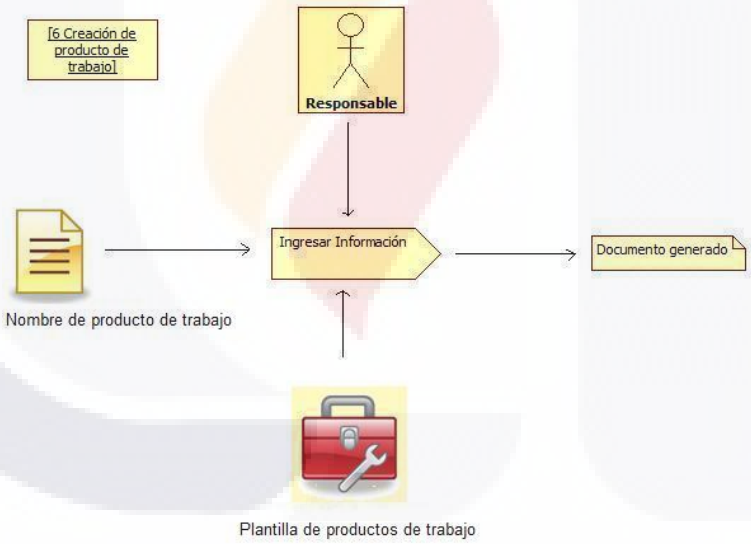


Ilustración 41 Introducir descripción de los productos de trabajo

En esta actividad se introduce la descripción de los productos de trabajo

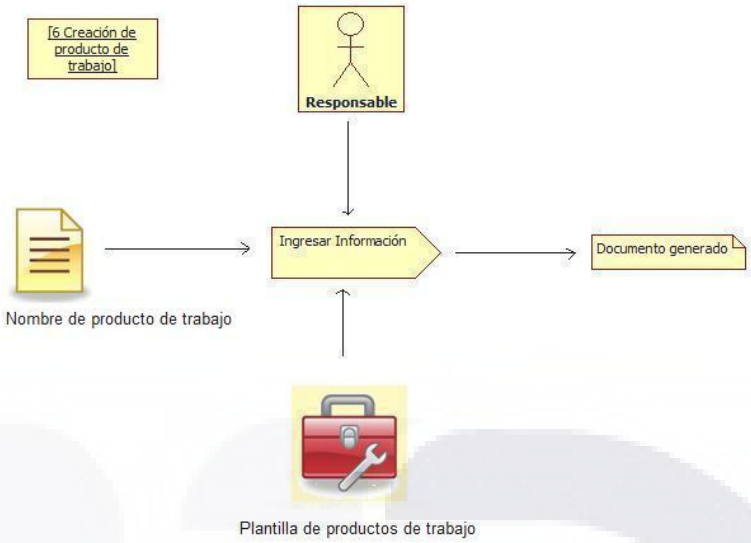


Ilustración 42 Relacionar actividades a procesos

Ahora se deben asociar los Productos de trabajo creados a los procesos y a las actividades.

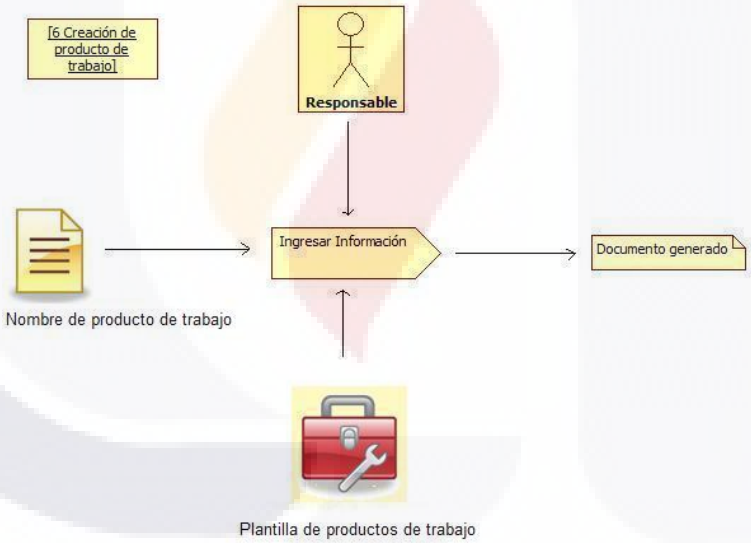


Ilustración 43 Relacionar productos a actividades

Ahora se deben asociar los Productos de trabajo creados a los procesos y a las actividades.

3.7 Identificación de fases SoSELR

Puede observarse que la Fase de diseño implica una parte que es independiente de la plataforma, es decir, que conformará el Modelo Independiente de la Plataforma (PIM – Platform Independent Model); y una parte que es dependiente de la plataforma, es decir que se diseña para una plataforma específica y por lo tanto conforma el Modelo de Plataforma Específica (PSM – Platform Specific Model).

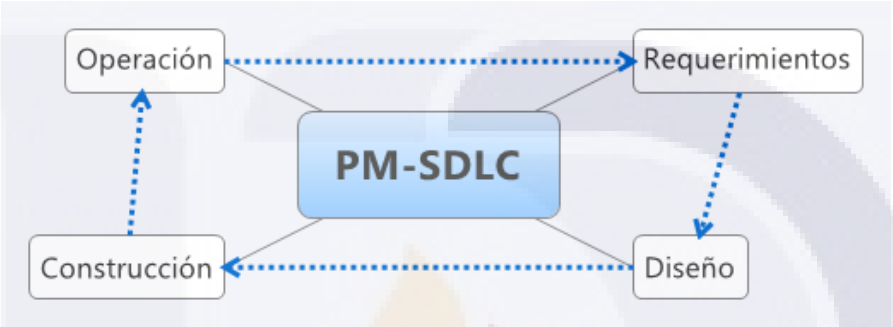


Ilustración 44 Modelo de Proceso de ciclo de software

Por otro lado, el diseño se ubica dentro del desarrollo, ya que al iniciar el diseño, ya quedó atrás la parte independiente de la computación.

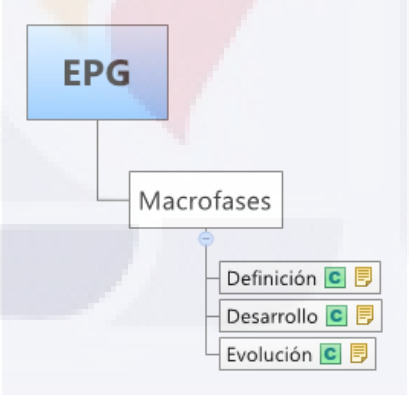


Ilustración 45 Integración de las macrofases

A continuación se define cada componente del nuevo PM-SDLC: Macrofases, Fases-Actividades, Productos-Herramientas, Controles (modo iterativo) y Roles.

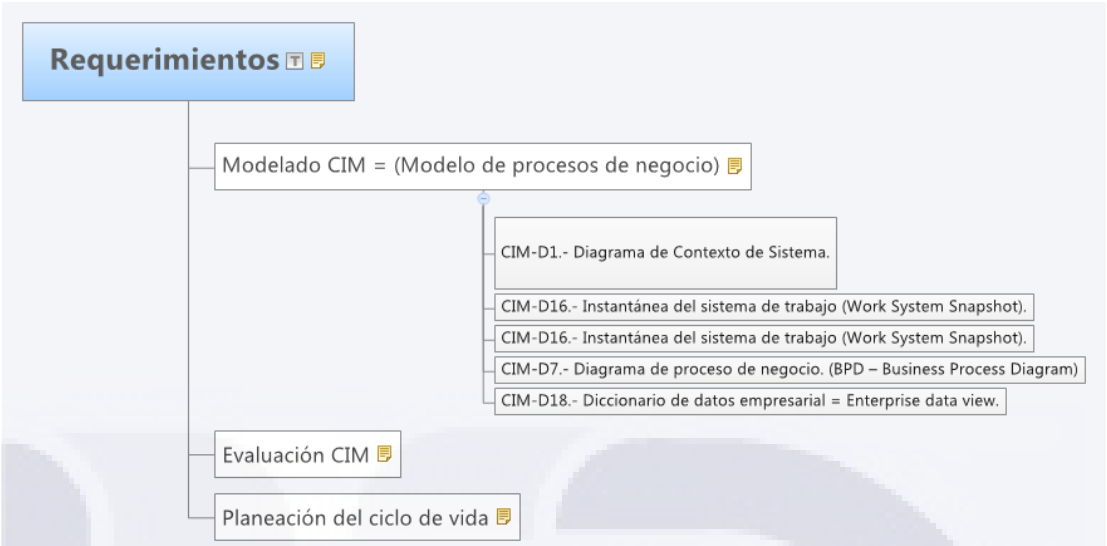


Ilustración 46 Fase de diseño

Se refiere a la macro fase en la que se planea el proceso completo de desarrollo del sistema. En ella se definen los requerimientos, el alcance y entrega del sistema.

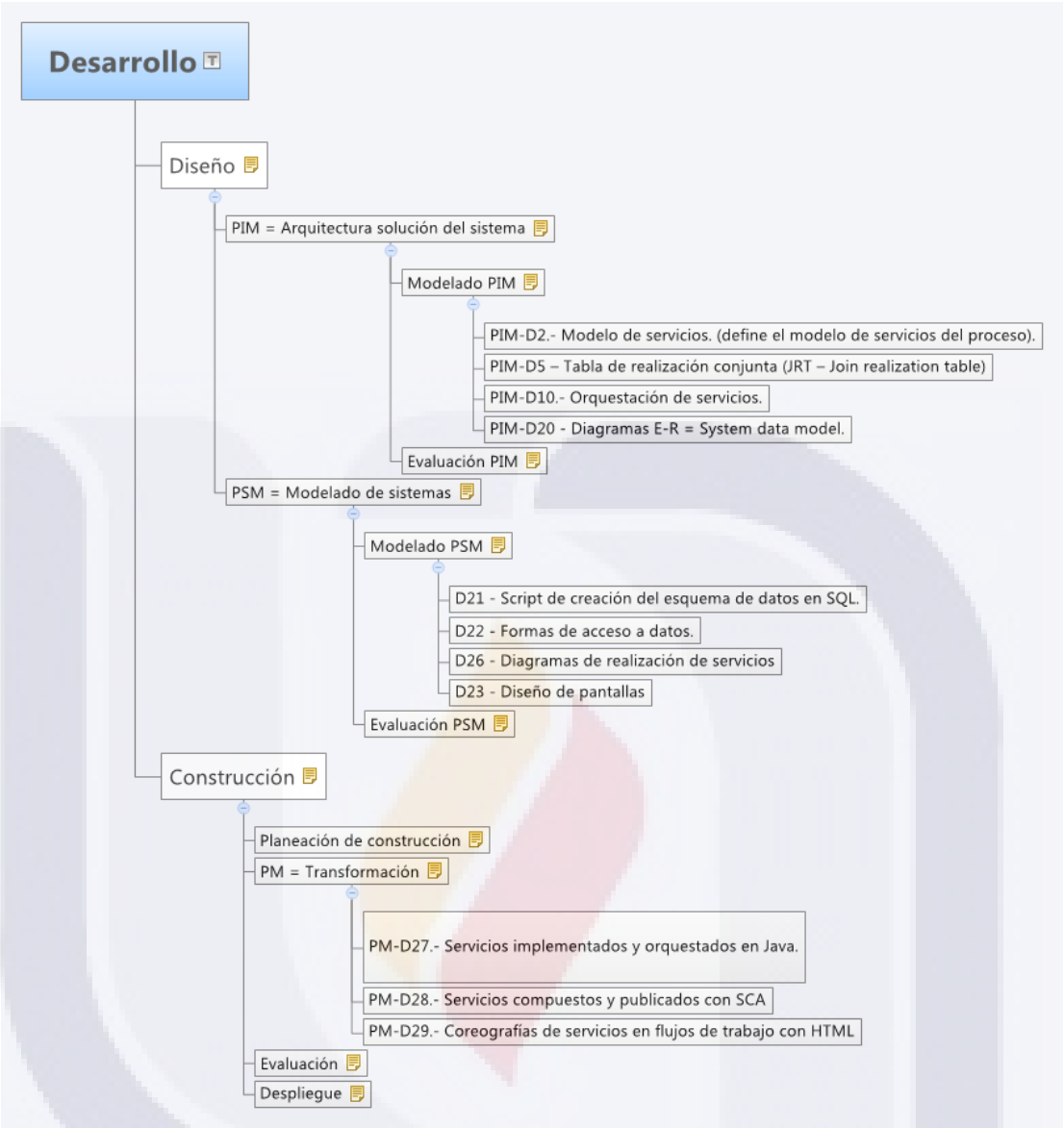


Ilustración 47 Fase de desarrollo

En esta macro fase se construye el sistema de acuerdo a la planeación. En ella se diseña y construye el sistema de software, obteniendo un sistema implementado y en operación.

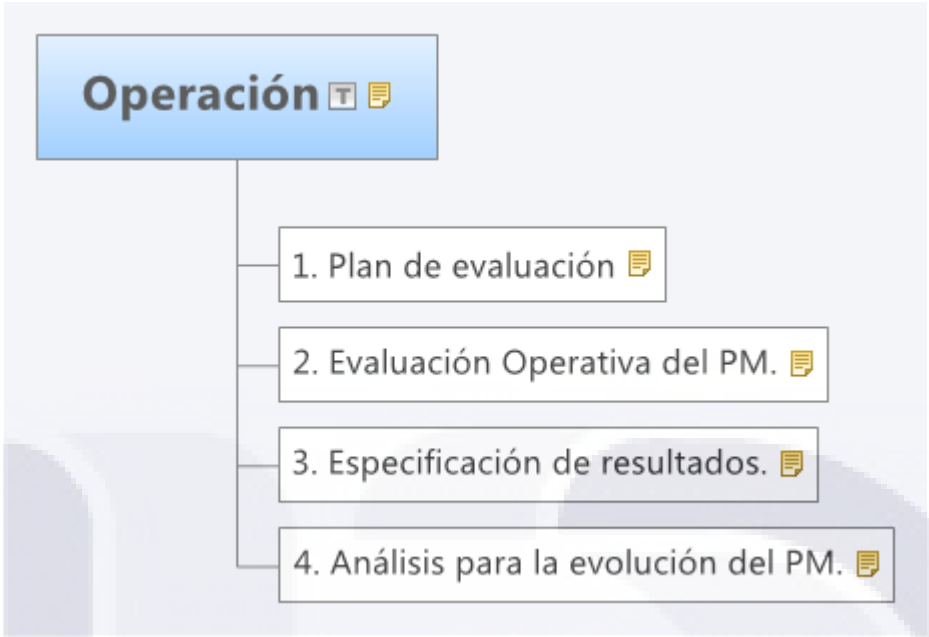


Ilustración 48 Fase de Evolución

En esta macro fase se construye el sistema de acuerdo a la planeación. En ella se diseña y construye el sistema de software, obteniendo un sistema implementado y en operación.

3.7.1 Definición de fases y actividades

Requerimientos.

En esta fase se hace un levantamiento de requerimientos, se especifican los objetivos de acuerdo a esos requerimientos, y se planea el alcance y entrega del sistema de software. Esta fase incluye tres actividades:

1. Modelado CIM.- Levantamiento de requerimientos y especificación de requerimientos, para crear un modelo independiente de la computación:
    a. Se crea un modelo de procesos de negocios que identifica (conoce los proceso de negocio sus flujos de trabajo y los define conceptualmente)
    b. Se especifican los procesos y la comunicación entre ellos, los servicios y la comunicación entre ellos, y los flujos de trabajo
    c. Se define la arquitectura empresarial.
2. Evaluación.- Evaluación de CIM, para detectar y corregir errores en él, tanto en su definición como en su especificación.
3. Planeación del ciclo de vida.
    a. Plan para el alcance y entrega del sistema.
    b. Plan de desarrollo.

Diseño.

En esta fase se diseña el sistema, diseño que implica la construcción de dos modelos, el modelo PIM (que corresponde con la “diseño de la arquitectura solución”=Solution Architecture Design) a partir de CIM, y el modelo PSM (que corresponde con el “modelado del sistema”=System Modeling) a partir de PIM. Esta fase se divide en dos partes que incluyen las siguientes actividades:

Primera parte.

1. 1Modelado PIM.- Transformación del modelo CIM en modelo PIM para definir la arquitectura solución del sistema:
    a. Se define y especifica la coreografía de servicios (se especifica la interface de cada servicio y de los flujos de trabajo, la orquestación y coordinación de servicios, y la semántica de negocios)
    b. Se diseñan las bases de datos.
2. Evaluación.- Evaluación de PIM, para detectar y corregir errores en él, tanto en su definición como en su especificación.

Segunda parte.

1. Modelado PSM.- Transformación del modelo PIM en el modelo PSM que es el modelo de sistemas (System Modeling), que incluye:
    a. Especificación de composición de servicios (se construye un repositorio de meta-modelos)
    b. Semántica de los procedimientos.
2. Evaluación.- Se evalúa el modelo PSM, para detectar y corregir errores, tanto en su definición como en su especificación.

Construcción.

En esta fase se construye el sistema, primero construyendo un modelo para una plataforma específica (PSM), y segundo, construyendo un modelo ejecutable (PM). Esta fase implica cuatro actividades:

1. Planeación de construcción.- Se realiza un plan para la transformación de PSM en PM.
2. Modelado PM.- Se transforma PSM en PM para conseguir un modelo ejecutable del sistema, esta transformación es propiamente la construcción del sistema de software: se implementa y prueba cada servicio y la integración de los mismos. Aquí implementación se refiere a la codificación, tanto de los servicios como de su composición, orquestación y coordinación).
3. Evaluación.- Se evalúa el PM, para detectar errores y corregir.
4. Despliegue.- Se pone en operación el PM.

Operación.

En esta fase se monitorea y evalúa la operación del sistema (PM), y en base a los registros de monitoreo y evaluación se analiza para evolucionar el sistema mejorándolo y agregándole funcionalidad.

1. Plan de evaluación.- Se planea para evaluar la operación del sistema.
2. Evaluación Operativa del PM.- Se monitorea y evalúa la operación del sistema.
3. Especificación de resultados.- Se registran los resultados de la evaluación operativa.
4. Análisis para la evolución.- Se analizan los resultados para decidir acerca de evolucionar el sistema.

3.7.2 Modo iterativo del PM-SDLC

Una de las bases para asegurar el rigor del PM-SDLC que se propone, es el modelo de proceso MBASE. Con esta base se propone una forma espiral de iteración como la de MBASE. Los productos de cada fase del ciclo de vida de desarrollo propuesto son, los modelos CIM, PIM, PSM y PM (solución entregable SOSS) y una evaluación de ese SOSS. Se trata de repetir en cada iteración en forma de espiral, todas las fases del ciclo de vida propuestas en este PM-SDLC, logrando en cada iteración los mismos productos (modelos, solución y evaluación). La diferencia en cada iteración respecto a estos productos, será el grado de completitud y madurez alcanzado (considerando que un modelo “espiral” se refiere a un modelo “iterativo e incremental”).

A continuación se definen los objetos de control del espiral y los criterios de término para cada parte del ciclo de desarrollo.

Objetos de control = lazos de repetición y milestones.

Para poder controlar el ciclo de desarrollo el nuevo PM-SDLC define dos “objetos de control” que además aseguran la iteración en espiral, estos objetos de control son: (1) “lazos de repetición” (Inception, Elaboration, Construction, Transition) (2) los milestones que indican cuando un lazo de repetición ha terminado

Criterios para terminar un lazo de repetición.

Durante el desarrollo del proyecto, en un “lazo de repetición, independientemente de su nombre, se repiten las fases del ciclo de vida de desarrollo (requerimientos, diseño, construcción, evaluación de forma concurrente)

3.7.3 Roles

Los roles definidos como equipos de desarrollo se tienen cada uno un administrador de proyecto y un líder técnico tal como lo indica la siguiente tabla:

Equipo No.	Roles	Funciones
1	Equipo de modelado empresarial.	Realiza las actividades de la fase de Requerimientos. - Modelado CIM.
		Y una planeación para la iteración (plan para el alcance y entrega. Plan de desarrollo).
2	Equipo de arquitectura de sistemas	Realiza las actividades de la fase de Diseño que corresponden con el modelado PIM.
3	Equipo de modelado de sistemas	Realiza las actividades de la fase de Diseño que corresponden con el modelado PSM.
4	Equipo de desarrollo servicios.	Realiza las actividades de construcción que tienen que ver con la construcción de servicios (una parte del modelado PM).
		Plan para la construcción (Administradores de los equipo 4, 5, 6).
5	Equipo de desarrollo de flujos de trabajo e integración del SOSS.	Realiza las actividades de construcción que tienen que ver con la composición de servicios que conforman los flujos de trabajo y el SOSS al final (una parte del modelado PM).
		Plan para la construcción. (Administradores de los equipos 4, 5, 6).
6	Equipo de despliegue	Se encarga de poner en marcha el SOSS: asegura el funcionamiento del SOSS en los sitios de trabajo, entrena a los usuarios Y asegura el buen funcionamiento inicial hasta que los usuarios puedan ser “independientes” en el uso del SOSS.
		Plan para la construcción. (Administradores de los equipos 4, 5, 6).
7	Equipo de operación, evaluación y evolución.	Se encarga de asegurar la operación continua, el monitoreo, la evaluación y evolución del sistema.

Tabla 9 Definición de roles

3.7.4 Herramientas de análisis y diseño

Para la construcción de CIM.

1. Definición del proceso de negocio:
    a. D1 = Diagrama de Contexto de Sistema (System Context Diagram),
    b. D16 = Instantánea del sistema de trabajo (Work System Snapshot),
    c. D17 = Tabla de responsabilidades del sistema (SRT – System Responsability Table)
2. Definición del flujo de trabajo:
    a. D7 = Diagrama de Proceso de Negocio (BPD – Business Process Diagram).
3. Datos:
    a. D18 = Diccionario de datos empresarial = Enterprise data view.

Para la construcción de PIM.

1. Definición de casos de uso:
    a. D2 = Modelo de servicios (Service model diagram).
    b. D5 = Tabla de Realización Conjunta (JRT – Join Realization Table), para el proceso de negocio empresarial de caja blanca (White Box enterprise business process).
2. Definición de la colaboración de servicios:
    a. D10 = Orquestación de servicios (Diagramas de colaboración de servicios en flujos de trabajo.
    b. D9 = Diagramas de interacción de servicios en work flows (no a detalle para todo el sistema)
3. Datos:
    a. D20 = Diagrama E-R = System data model.

Para la construcción de PSM.

1. Datos:
    a. D21 = Script de creación del esquema de datos en SQL.
    b. D22 = Formas de acceso a datos.
2. Procesos de negocio:
    a. D26 = Diagramas de especificación de servicios (interfaces, operaciones = los dibujos de interfaces + implementación en base a componentes o unidades, etc.).
3. Interfaces de usuario:
    a. D23 = Diseño de pantallas

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Resultados

4. Resultados

Como parte de la evolución hacia la digitalización en la sociedad en la que vivimos, los manuales y guías que se utilizan para orientar a sus respectivos lectores está siendo generados y consultados cada vez más en formato electrónico, de ahí el origen del concepto de “Guía Electrónica de Procesos”

Para concluir con la tesis se debe comentar que se han cumplido los objetivos marcados de implementar un EPG del proceso de desarrollo de software SoSE-LR que ayude a mejorar su proceso de consulta y aprendizaje.

Se implemento un artefacto (EPG de SoSE-LR) permitiendo representar la experiencia y el conocimiento de 10 ingenieros de software expertos sobre cómo hacer un producto software usando dicho proceso.

Según los expertos la calidad conceptual de la calidad del diseño de la EPG ha tenido aceptación además de que de acuerdo a su experiencia esto les facilitaría su trabajo, además de ser de fácil utilización en sus empresas.

De los resultados de la Encuesta Demográfica, podemos concluir lo siguiente.

De acuerdo a la pregunta 1, el 45% de la población encuestada, se encuentra en un puesto de trabajo profesional, su nivel de escolaridad es licenciatura, se encuentra entre los 35 y 40 años, el 73% de la población pasa de 5 a 9 horas pendiente del correo electrónico, aunque el 45% se considera un usuario normal en el uso de esta herramienta, en el área funcional en la que se desempeñan se observa muy variada la ocupación, únicamente 27% de la Población ocupa un área funcional administrativa y otro 27% se encuentra en el área de fabricación y diseño.

El 100% de la población encuestada, indica que las operaciones comerciales de su trabajo son a nivel nacional.

En cuanto al instrumento de métricas de aceptación de metodologías se tuvieron las siguientes estadísticas:

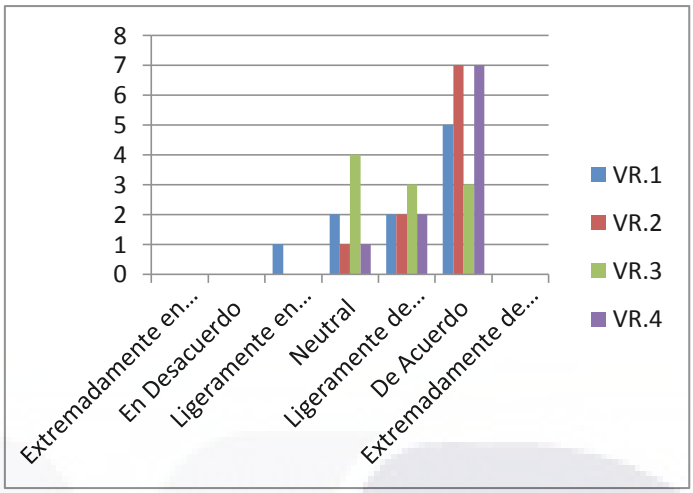


Ilustración 49 Cumplir tareas de desarrollo mas rápidamente

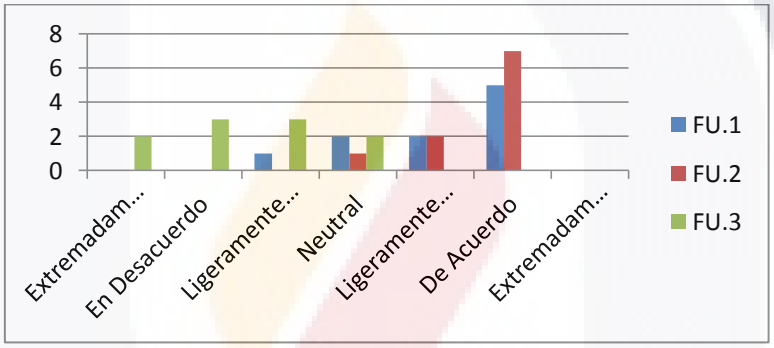


Ilustración 50 Facilidad de Uso

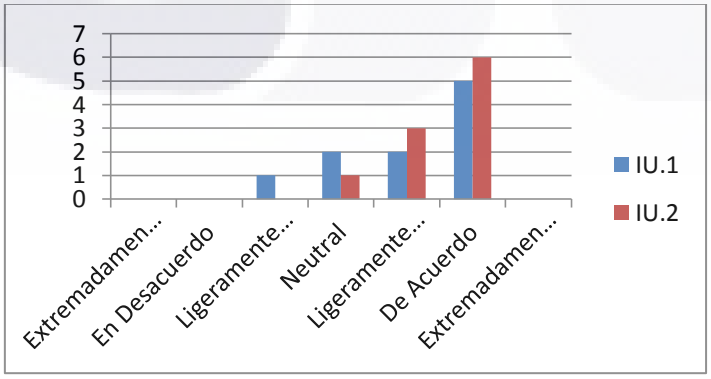


Ilustración 51 intención de Uso

Resumiendo la información resultante mostrada en las graficas anteriores tenemos que:

En el apartado de utilidad.

La población encuestada indica que esta de acuerdo en que el utilizar una metodología SoSE-LR los habilitaría para cumplir con tareas de desarrollo mas rápidamente, con mejor calidad, realizándolo de manera efectiva y teniendo un mayor control sobre el proceso de desarrollo

En el apartado de Facilidad de Uso.

La población encuestada indica que esta de acuerdo en que en caso de tener que aprender a utilizar la metodología será fácil aprenderla, aunque se tendría que tener cierta capacitación para un buen manejo de la misma.

En el apartado de Intensión de Uso

La población encuestada indica que tienen intención de utilizar una metodología en sus área de trabajo, aunque aun no tienen definida cual herramienta, indican que está es una buena opción para ser aplicada en el área.

Como resultante del estudio realizado a través de este documento podemos llegar a contestar las preguntas antes formuladas en los objetivos los cuales concluimos que el implementar una EPG del proceso de desarrollo de software SoSE-LR que ayuda a mejorar su proceso de consulta y aprendizaje, la experiencia de la población encuestada nos ha arrojado resultados satisfactorios para validar que es una metodología viable de fácil uso y aprendizaje.

Aunque a pesar de que algunos tienen años desarrollando sistemas aun hay áreas donde se no utilizan herramientas de modelado de software por lo que la calidad en el diseño les ha parecido buena.

Mostrando con esto que el utilizar esta metodología les permitiría tener una mejor planeación y desarrollo del sistema durante todo el ciclo de vida, se mantendría a todos los integrantes del equipo trabajando bajo un mismo esquema y estandarizando el producto.

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Conclusiones

5. Conclusiones

Del presente documento se desprenden una serie de conclusiones relevantes no tan solo para entender el papel de una Guía Electrónica de Procesos en el Ciclo de Vida de un Sistema sino también como influye de manera económica a la empresa que decide utilizar este tipo de herramientas.

La población del área de sistemas utiliza cada vez mas este tipo de herramientas, con la tendencia a tener sistemas de calidad tanto en el Área Administrativa como en el Área de TI buscando mejoras para integrar información relevante, estandarizar resultados y obtener un mejor producto de manera eficiente y eficaz.

No obstante se admite que el crecimiento de su uso también requiere la capacidad del personal para manejar las herramientas, implicando con esto, cambios de paradigmas al utilizar EPG's

La Metodología SoSe-LR se da en respuesta en mejorar un PM- SDLC especifico mostrando con esto mejoras en el desarrollo de software, de acuerdo a los resultados obtenidos por expertos ingenieros en software indican algunos que la experiencia reflejada bajo este artefacto implica un ahorro en tiempo y dinero para la empresa. La información que anteriormente era utilizada de manera empírica, pero al utilizar una metodología especifica, su utilización y la información generada llega a ser parte del sistema de conocimiento de la misma lo cual influye de manera automática a la estandarización y uso para próximos proyectos, no implicando que la utilización de servicios utilizados para un proyecto se pueda utilizar en otros sin mayor problema.

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Recomendaciones

6. Recomendaciones

Cómo sabemos la formalización y uso de una Guía Electrónica de Procesos esta en auge, y la metodología SOSE-LR es de gran utilidad y de fácil uso para el desarrollo de sistemas, teniendo en cuenta esto y viendo el desarrollo de la misma, se han llegado a las siguientes líneas de ampliación:

Se podrían añadir servicios asociados con la gestión de configuración: ya que las definiciones de procesos tienen a evolucionar con el tiempo, el control de versiones es algo a tener en cuenta, para evitar trabajar con información obsoleta o fuera de lugar.

Los equipos de trabajo que participan en el proceso deberían ser capaces de añadir sus notas, comentarios o marcas sobre cualquier parte de una EPG, que podrían ser integradas y mostradas en una ventana adicional o con otro tipo de letra.

Se podrían añadir cuestionarios para recabar información de los equipos de trabajo que participan en el proceso para tratar de mejorar la guía en base a sus opiniones y recomendaciones

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Glosario

Glosario

CMMI	Capability Maturity Model Integrated
EPF	Eclipse Process Framework
EPG	Es una herramienta que proporciona orientación del proceso y el apoyo a la gestión del conocimiento en el proceso de desarrollo de software
Framework	Es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado
OpenUp	Es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido
OSSP	Organization's set of software process

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Bibliografía

Referencias

Armas, R., Chamorro, A., Montes, M., & Gutierrez, J. A. (2007, enero 1). DESDE ISO 9001 HACIA CMMI, PASOS PARA LA MEJORA DE LOS PROCESOS Y MÉTRICAS. Grupo Gesfor. Recuperado a partir de http://moodle.univo.com.mx/ingenieria/moodledata/temp/backup/1293679217/course_files/semana5/RPM_v4_01_1_.03.pdf

Balduino, R. (2007). Introduction to OpenUP (Open Unified Process).

CMMI Product Team. (2006, agosto). CMMI® for Development Version 1.2. Recuperado a partir de http://repository.cmu.edu/cgi/viewcontent.cgi?article=1378&context=sei

CMS. (2008, marzo 27). SELECTING A DEVELOPMENT APPROACH. Recuperado a partir de http://www.cms.gov/SystemLifecycleFramework/Downloads/SelectingDevelopmentApproach.pdf

Gnatz, M., Marschall, F., Popp, G., Rausch, A., & Schwerin, W. (2009, marzo 31). Towards a Tool Support for a Living Software Development Process.

Haumer, P. (2007, abril). Eclipse Process Framework Composer Part 1: Key Concepts.

Holzner, S. (2000). *Java 2*.

IBM. (2008a). *IBM Rational Unified Process*.

IBM. (2008b). *IBM Rational Method Composer*. Recuperado a partir de http://public.dhe.ibm.com/common/ssi/ecm/en/rad14029usen/RAD14029USEN.PDF

Jaakkola, H., & Thalheim, B. (2005). Software Quality and Life Cycles. Recuperado a partir de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.5848&rep=rep1&type=pdf

Kellner, M. I., Becker-Kornstaedt, U., Riddle, W. E., Tomal, J., & Verlage, M. (1998). Process Guides: Effective Guidance for Process Participants. Presentado en Computer Supported Organizational Work, Held at Lisle, Illinois, USA, 14-17 June 1998. Recuperado a partir de ftp://ftp.sei.cmu.edu/pub/documents/articles/pdf/epg.pdf

Kurniawati, F., & Jeffery, R. (2005, julio 19). The use and effects of an electronic process guide and experience repository: a longitudinal study.

Kurniawati, F., Kitchenham, A., & Jeffery, R. (2004). Implementing a process model-based software development support environment: Comparing an open source component with a proprietary tool. IET Digital Library.

Moe, N. B., & Dybå, T. (2004). The Adoption of an Electronic Process Guide in a Company with Voluntary Use. Recuperado a partir de http://www.springerlink.com/content/nvbx1t96wan40mdr/

Oktaba, H., & Ibarguengoitia, G. (1998). Software Process Modeled with objects: Static view.

Petter, S., & Vaishnavi, V. (2008, abril). Facilitating experience reuse among software project managers, pp. Pages 1783–1802.

Quint, V., & Vatton, I. (2005, diciembre 28). Towards Active Web Clients. Recuperado a partir de http://wam.inrialpes.fr/publications/2005/DocEng05-Quint.html

Scott, L., Carvalho, L., & Jeffery, R. (2002). A Process-Centred Experience Repository for a Small Software Organisation.

Sommerville, I. (2005). *Ingeniería de software* (Septima.). Madrid.

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Anexos

Anexos

ANEXO A

Resumen de la metodología SoSe

Este nuevo PM-SDLC es resultado de una investigación Conceptual y de Diseño, que conjuga varias teorías y criterios de diseño; además tratado de aprovechar la experiencia previa de los modelos de proceso para el desarrollo, tanto de sistemas tradicionales como de Sistemas de Software Orientados a Servicios (SOSS - Service-Oriented Software System).

Los fundamentos teóricos (principalmente para proponer las fases y actividades) considerados son: el estándar MDA [Miller2003], el concepto de orientación a servicios desde la referencia [Zirnrnerman2004], y la experiencia conjunta de los modelos tradicionales tomada de [Rodríguez2006], y la experiencia conjunta de los modelos emergentes en Ingeniería de Software Orientada a Servicios (SOSE - Service-Orientes Software Engineering) [Rodríguez2009]. Como criterios fundamentales de diseño se toma el balance entre Agilidad y Rigor de acuerdo a Boehm (2004) y Rodríguez (2007).

1.1 Definición de Macrofases:

Definición. Se refiere a la macro fase en la que se planea el proceso completo de desarrollo del sistema. En ella se definen los requerimientos, el alcance y entrega del sistema.

Desarrollo. En esta macro fase se construye el sistema de acuerdo a la planeación. En ella se diseña y construye el sistema de software, obteniendo un sistema implementado y en operación.

Evolución. En esta macro fase se monitorea la operación del sistema y se evalúa para una posible evolución del sistema, ya sea para mejorarlo o incrementar su funcionalidad.

1.2 Definición de fases y actividades:

Requerimientos. En esta fase se hace un levantamiento de requerimientos, se especifican los objetivos de acuerdo a esos requerimientos, y se planea el alcance y entrega del sistema de software. Esta fase incluye tres actividades:

1. Modelado CIM.- Levantamiento de requerimientos y especificación de requerimientos, para crear un modelo independiente de la computación: a) Se crea un modelo de procesos de negocios que identifica (conoce los proceso de negocio sus flujos de trabajo y los define conceptualmente), b) se especifican los procesos y la comunicación entre ellos, los servicios y la comunicación entre ellos, y los flujos de trabajo y, c) se define la arquitectura empresarial.
2. Evaluación.- Evaluación de CIM, para detectar y corregir errores en él, tanto en su definición como en su especificación.
3. Planeación del ciclo de vida.- a) Plan para el alcance y entrega del sistema. b) Plan de desarrollo.

Diseño. En esta fase se diseña el sistema, diseño que implica la construcción de dos modelos, el modelo PIM (que corresponde con la "diseño de la arquitectura solución"=Solution Architecture Design) a partir de CIM, y el modelo PSM (que corresponde con el "modelado del sistema"=System Modeling) a partir de PIM. Esta fase se divide en dos partes que incluyen las siguientes actividades: Primera parte.

1. Modelado PIM.- Transformación del modelo CIM en modelo PIM para definir la arquitectura solución del sistema: a) se define y especifica la coreografía de servicios (se especifica la interfase de cada servicio y de los flujos de trabajo, la orquestación y coordinación de servicios, y la semántica de negocios); b) se diseñan las bases de datos.
2. Evaluación.- Evaluación de PIM, para detectar y corregir errores en él, tanto en su definición como en su especificación.

Segunda parte.

1. Modelado PSM.- Transformación del modelo PIM en el modelo PSM que es el modelo de sistemas (System Modeling), que incluye: a) especificación de composición de servicios (se construye un repositorio de meta-modelos), y b) semántica de los procedimientos. 2. Evaluación.- Se evalua el modelo PSM, para detectar y corregir errores, tanto en su definición como en su especificación.

Construcción. En esta fase se construye el sistema, primero construyendo un modelo para una plataforma específica (PSM), y segundo, construyendo un modelo ejecutable (PM). Esta fase implica cuatro actividades:

1. Planeación de construcción.- Se realiza un plan para la transformación de PSM en PM.
2. Modelado PM.- Se transforma PSM en PM para conseguir un modelo ejecutable del sistema, esta transformación es propiamente la construcción del sistema de software: se implementa y prueba cada servicio y la integración de los mismos. Aquí implementación se refiere a la codificación, tanto de los servicios como de su composición, orquestación y coordinación).
3. Evaluación.- Se evalúa el PM, para detectar errores y corregir.
4. Despliegue.- Se pone en operación el PM.

Operación. En esta fase se monitorea y evalúa la operación del sistema (PM), y en base a los registros de monitoreo y evaluación se analiza para evolucionar el sistema mejorándolo y agregándole funcionalidad.

1. Plan de evaluación.- Se planea para evaluar la operación del sistema.
2. Evaluación Operativa del PM.- Se monitorea y evalua la operación del sistema.
3. Especificación de resultados.- Se registran los resultados de la evaluación operativa.
4. Análisis para la evolución.- Se analizan los resultados para decidir acerca de evolucionar el sistema.

2. Modo iterativo del nuevo PM-SDLC.

Una de las bases para asegurar el rigor del PM-SDLC que se propone, es el modelo de proceso MBASE. Con esta base se propone una forma espiral de iteración como la de MBASE (ver Figura A.1). Los productos de cada fase del ciclo de vida de desarrollo propuesto son, los modelos CIM, PIM, PSM y PM (solución entregable SOSS) y una evaluación de ese SOSS (Tabla A.1). Se trata de repetir en cada iteración en forma de espiral, todas las fases del ciclo de vida propuestas en este PM-SDLC, logrando en cada iteración los mismos productos (modelos, solución y evaluación). La diferencia en cada iteración respecto a estos productos, será el grado de completitud y madurez alcanzado (considerando que un modelo "espiral" se refiere a un modelo "iterativo e incremental").

A continuación se definen los objetos de control del espiral y los criterios de término para cada parte del ciclo de desarrollo.

1) Objetos de control = lazos de repetición y milestones.

Para poder controlar el ciclo de desarrollo el nuevo PM-SDLC define dos "objetos de control" que además aseguran la iteración en espiral, estos objetos de control son: (1) "lazos de repetición" (Inception, Elaboration, Construction, Transition) que se muestran en la Tabla A.2 y que son los que propone MBASE como fases; y (2) los milestones14 que indican cuando un lazo de repetición ha terminado (la Tabla A.2 indica los milestones para cada lazo de repetición).

2) Criterios para terminar un lazo de repetición.

Durante el desarrollo del proyecto, en un "lazo de repetición, independientemente de su nombre, se repiten las fases del ciclo de vida de desarrollo (requerimientos, diseño, construcción, evaluación de forma concurrente), propuestas por el nuevo PM-SDLC en la Tabla A.1. Dentro de cada lazo de repetición se pueden ejecutar una o varias iteraciones en espiral. Cuando se cumpla con el milestone correspondiente a cada lazo de repetición, el lazo en turno terminará.

Cada lazo de repetición tiene un nombre I. Incepción, E. Elaboración, C. Construcción, T. Transición y S. Soporte. En cada lazo de repetición hay uno o dos productos que son clave, de modo que comúnmente el avance y madurez de dichos productos en el lazo respectivo definen el término del lazo. Para el lazo de Incepción es el modelo CIM al 40% y la planeación, para el lazo Elaboración es el modelo PIM al 70%, para el lazo Construcción es el modelo ejecutable PSM y PM al 100%, para el lazo Transición es el Despliegue al 100%, y, para el lazo Soporte la evaluación al 100%. Como se muestra en la Tabla A.3

3. Definición de Roles:

Cada rol se encarga también de las actividades de administración implicadas de acuerdo a su responsabilidad (por ejemplo el plan de iteración, el plan para la evolución y el plan de construcción), en las actividades de cada fase, y para todas estas actividades de administración hay un equipo más, que es el "equipo de administración", que coordina estas actividades y se asegura de su cumplimiento. Los equipos 4, 5 y 6 pueden llamarse de manera genérica "equipo de desarrollo".

Es importante notar que para desarrollos mayores, posiblemente será necesario tener todos los equipos cubriendo cada rol, mientras que a medida que disminuye el tamaño del desarrollo, se requerirá en determinado momento una sola persona por rol, o incluso una misma persona puede llevar a cabo varios roles a la vez; y llegando al punto en el que par desarrollos muy pequeños, será necesaria una, dos o tres personas que se cubran en conjunto todos los roles.

4. Explicación de las herramientas de análisis y diseño propuestas.

La mayoría de las explicaciones tiene un ejemplo, las que no lo tienen indican que se trata de alguna herramienta ya muy conocida. Algunas explicaciones son algo extensas por ejemplo el diagrama BPD, por la novedad del tema, y por la importancia que la parte de negocios tiene cuando se traba de orientación a servicios.

Framework

El framework de sistema de trabajo es la base del "work system snapshot", el cual sumariza el sistema de trabajo sobre una sola página (single page) identificando sus clientes, productos y servicios, procesos y actividades, participantes, información y tecnología. Al inicio de un análisis, crear y discutir un "work system snapshot" de ejemplo para una aplicación de préstamos (loan application) (Work system snapshot for a loan application and underwriting system for loans to new clients) de acuerdo acerca del alcance y propósito del sistema de trabajo que se analiza. El ambiente, infraestructura, y estrategia no están incluidos en la imagen instantánea del sistema para ajustar a una página. Estos tópicos se consideran como un análisis más amplio. Este "instantánea del sistema de trabajo" y esta "aproximación a sistema de trabajo" es útil para resumir sistemas dentro de organizaciones y para ayudar a pensamientos individuales no técnicos acerca de situaciones en términos de sistemas.

Las SRTs pueden también ser usadas para sumarizar las recomendaciones acerca de ejecutar pasos específicos más exitosamente o acerca de agregar o eliminar pasos. Así mismo, versiones extendidas de los SRTs pueden resumir la extensión para la cual cambios recomendados que probablemente resolverían problemas relacionados con responsabilidades específicas y la extensión a la que podrían causar problemas.

Un BPD es el diagrama especificado con BPMN. Un BPD usa los elementos gráficos y la semántica que soporta esos elementos, definidos en la especificación BPMN.

BPMN es un soporte solo para los conceptos de modelado aplicables a procesos de negocio. Esto significa que otros tipos de modelado hecho por organizaciones para propósitos de negocios quedan fuera del alcance de BPMN.

El modelado de procesos de negocio se usa para comunicar una amplia variedad de información para una amplia variedad de audiencias. BPMN está diseñado para cubrir muchos tipos de modelado y permite la creación de procesos de negocio de fin a fin (end-to-end business processes).

Hay tres tipos básicos de sub modelos con dentro de un modelo BPMN de fin a fin (end-to-end BPMN model):

- Procesos de negocio privados (Private or internal business processes)
- Procesos abstractos (Abstract or public processes)
- Procesos colaborativos (Collaboration or global Processes)

Algunos términos de especificación de BPMN con respecto al uso (regarding) de "Carriles de Alberca" (Swimlanes) (es decir, Albercas y Carriles (Pools and Lanes)) son usadas en las descripciones de abajo (below).

Procesos de negocio privados (internos) (Private (Internal) Business Processes).

Private business processes son aquellos que procesos de negocio internos para una organización específica y son el tipo de procesos que han sido generalmente llamados flujos de trabajo (workflow) o procesos BPM.

Si se usan Swimlanes entonces un proceso de negocio privado puede ser contenido dentro de una sola alberca (Pool). El flujo de secuencia (Sequence Flow) del proceso es de cualquier manera contenido dentro de la alberca (Pool) y no pueden cruzar los límites de la alberca. El flujo de mensajes (Message Flow) puede cruzar el límite de la alberca para mostrar las interacciones que existen entre los procesos de negocio privados. De modo, que un solo (single) Diagrama de Procesos de Negocio (BPD - Business Process Diagram) puede mostrar múltiples procesos de negocio privados, cada uno con mapeos separados.

Un proceso abstracto representa las interacciones entre un proceso de negocio privado y otros procesos o participantes. Solo muestra todas las actividades que son usadas para comunicar el lado de afuera (outside) del proceso de negocio privado, más los mecanismos de control de flujo apropiados, que son incluidos en el proceso abstracto. Todas las demás actividades "internas" del proceso de negocio privado no se muestran en el proceso abstracto. De modo que el proceso abstracto muestra la secuencia de mensajes hacia afuera (to the outside world) que son requeridos para interactuar con estos procesos de negocio.

Un proceso abstracto es contenido dentro de una alberca (Pool) y puede modeladarse separadamente o dentro de un largo Diagrama BPMN para mostrar el Flujo de Mensajes entre las actividades del proceso abstracto y otras entidades. Si

el proceso abstracto está en el mismo Diagrama con sus correspondientes procesos de negocio, entonces las actividades que son comunes a ambos procesos pueden asociarse.

Procesos de colaboración (globales). (Collaboration (Global) Processes).

Un proceso de colaboración describe (depicts) la interacción entre dos o más entidades de negocio. Estas interacciones se definen como una secuencia de actividades que representan los patrones de intercambio de mensajes (message exchange patterns) entre las entidades involucradas.

Los procesos de colaboración pueden mostrarse como dos o más procesos abstractos comunicándose unos con los otros (communicating with each other). Con un proceso abstracto, las actividades para los participantes de la colaboración pueden considerarse los "puntos de contacto" ("touch-points") entre los participantes.

Carriles de nado. (Swimlanes = Pools and Lanes)

Aquí se trata de la definición de procesos de negocio dentro de un ambiente colaborativo B2B (collaborative B2B environment). BPMN usa el concepto conocido como "swimlanes" para ayudar a particionar y organizar actividades. BPEL4WS se enfoca en un proceso privado específico que es interno para un Participante dado (por ejemplo una organización o compañía). BPEL4WS también pueden definir un proceso abstracto, pero desde el punto de vista de un solo participante. Es posible que un Diagrama BPMN puede describir más que un proceso privado, así como los procesos que muestran la colaboración entre procesos privados o entre participantes. De modo que cada proceso de negocio privado puede ser considerado para iniciar su ejecución por diferentes participantes. Gráficamente, cada participante será particionado; esto es, será contenido dentro de una caja rectangular llamada "Alberca" ("Pool"). Las Albercas pueden tener sub-Swimlanes llamadas simplemente "Carriles" ("Lanes"). Albarcas y Carriles están diseñados para soportar los usos de BPMN para definir diagramas de procesos de negocio (BPD - Business Process Diagram).

Una "Swimlane" tiene:

- "Nombre" como atributo, que es el texto que describe a la Swinlane.
- Albercas (Pools)
- Carriles (Lanes)

Alberca (Pool).

Una alberca (Pool) representa un participante dentro de un proceso. Un participante puede ser una entidad de negocio específico (por ejemplo una compañía) o puede ser un rol de negocio más general (por ejemplo un comprador, un vendedor o un fabricante). Gráficamente, una alberca es un contenedor para particionar un Proceso a partir de otras albercas, cuando se modelan situaciones B2B, aunque una alberca no necesita tener detalles internos (es decir puede ser una "caja negra ("black box")). Una alberca (Pool) es un rectángulo que debe dibujarse con una línea simple sólida y negra.

Para una Alberca de "Caja blanca", las actividades dentro de ella se organizan mediante un flujo de secuencia (Sequence Flow). El flujo de mensajes (Message Flow) puede cruzar el límite de la Alberca (Pool boundary) para adjuntarle (o colgarle) las actividades apropiadas.

Todos los BPDs contienen al menos una Alberca. En la mayoría de los casos, un BPD que consiste de una sola Alberca desplegará solo las actividades del proceso y no los límites de la Alberca. Aún más, muchos BPDs pueden mostrar la Alberca "principal" sin límites. Esto es, las actividades que representan el trabajo ejecutado desde el punto de vista del modelador o de la organización del modelador son consideradas actividades "internas" y pueden no ser rodeadas por los límites de una Alberca, mientras las otras Albercas del Diagrama tendrán sus límites.

Atributos de una Alberca:

- Proceso (0-1): Proceso.- El atributo proceso define el proceso que contiene la Alberca. Cada Alberca debe tener un proceso.
- Participante: Participante.- El modelador debe definir el participante para la Alberca. El participante puede ser un Rol o una Entidad. Este define el rol que una entidad o rol particular que la Alberca jugará en un Diagrama que incluye colaboración.
- Carriles (1-n): Carril.- Debe haber una o más carriles dentro de la alberca. Si hay un solo carril, entonces este carril comparte el nombre de la alberca y solo se despliega el nombre de la alberca. Si hay más de un carril, entonces cada carril tendrá su propio nombre y todos los nombres serán desplegados.
- Límite Visible Verdadero: Boolean.- Los atributos definen si el límite rectangular de la alberca es visible. Solo una alberca en el Diagrama puede poner el valor de este atributo en Falso.

Carril (Lane).

Un carril es una subpartición dentro de una Alberca, que se extiende a lo largo de la Alberca, en cualquier dirección, ya sea vertical u horizontal . El texto asociado con la línea (por ejemplo su nombre y/o atributo) puede ubicarse dentro del cuadro, en alguna dirección o localización, dependiendo de la preferencia del modelador o vendedor de la herramienta de modelado. Nuestros ejemplos ponen los nombres como un "banner" en el lado izquierdo (para Albercas horizontales) o en el tope (para Albercas verticales) sobre el otro lado de la línea que separa el nombre de la Alberca, sin embargo, esto no es un requerimiento. Los Carriles se usan para organizar y categorizar las actividades dentro de una Alberca.

Atributos de los Carriles.

- Alberca Padre (ParentPool): Pool.- La alberca padre debe ser especificado. Puede haber un solo padre.
- Carril Padre (ParentLane) (0-1): Carril.- El carril padre es un atributo opcional que se usa si el carril está anidado dentro de otro carril. El anidamiento puede ser multi-nivel, pero solo el padre inmediato se especifica.

Para el caso de este PM-SDLC se hará una tabla PIM-D5 (Tabla de realización conjunta = caso de uso) para cada actividad dentro de la definición de cada diagrama BPD de proceso interno.

CIM-D18.- Diccionario de datos empresarial = Enterprise data view.

Utilizar cualquier forma de diccionario de datos existente en la literatura de Análisis y Diseño de Sistemas de Información.

Para la construcción de PIM.

- Definición de casos de uso:

D2 = Modelo de servicios (Service model diagram).

D5 = Tabla de Realización Conjunta (JRT - Join Realization Table), para el proceso de negocio empresarial de caja blanca (White Box enterprise business process).

- Definición de la colaboración de servicios:

D10 = Orquestación de servicios (Diagramas de colaboración de servicios en flujos de trabajo.

D9 = Diagramas de interacción de servicios en work flows (no a detalle para todo el sistema)

- Datos:

D20 = Diagrama E-R = System data model.

Este diagrama representa el punto de vista lógico de procesos (para RUP-SE se trata de sistemas y subsistemas, representados por paquetes [Rational2002]). Este diagrama usa clasificadores de "procesos" («servicio»), para lograr paquetes estereotipados como «servicio».

La Figura A.19 representa el modelo de procesos de un subsistema o sistema, este utiliza estos paquetes estereotipados de proceso:

Cada servicio de este proceso puede a su vez ser una composición de servicios, o una simple lista de operaciones. Si es una composición quizá deba crearse un submodelo de servicios que defina los servicios de la composición en cuestión; si no es una composición es suficiente con listar los servicios.

(El flujo de caja negra que RUP llamó business worker survey no se pone porque corresponde con los diagramas BPD = CIM-D7 Diagramas de Proceso de Negocio).

Esta tabla define como los procesos colaboran y los eventos de caja blanca que se presentan dentro del proceso de negocio empresarial, con enfoque de sistemas (Caso de Uso para RUP). Esto es, esta tabla define el flujo del caso de uso de un proceso de negocio detallado (subsistemas a detalle en RUP) del sistema. Un ejemplo es la Figura A.17 en donde se define el flujo de trabajo de las actividades previamente definidas en CIM-D7 para procesos internos [Rational2003]:

Cada caso de uso utiliza las operaciones de los servicios definidos en PIM-D2.

PIM-10.- Orquestación de servicios.

Diagramas de composición de servicios similar a los que usa SCA (esto va en el proveedor de servicios).

Se trata de diagramas entidad relación existentes en la literatura de bases de datos relacionales.

Para la construcción de PSM.

- Datos:

D21 = Script de creación del esquema de datos en SQL. D22 = Formas de acceso a datos.

- Procesos de negocio:

D26 = Diagramas de especificación de servicios (interfaces, operaciones = los dibujos de interfaces + implementación en base a componentes o unidades o etc.).

- Interfaces de usuario:

D23 = Diseño de pantallas,

PSM-D26.- Diagramas de realización de servicios (interfaces, operaciones = los dibujos de interfaces + implementación en base a componentes).

Estos diagramas muestran la descripción (especificación de servicios en términos de sus interfaces, operaciones), representados como componentes. Se adapta un estereotipo con una notación tomada directamente de UML en combinación con la notación de estereotipo de servicios definido en RUP-SE (2003):

Servicio = interfaz + implementación

Estereotipo de servicio.- Proporciona un conjunto de operaciones de servicio, se trata de un estereotipo tal como lo maneja UML, de la siguiente manera:

Interfaz o Interface (de un servicio) = es una clase estereotipada que se usa para mostrar las operaciones y otras propiedades de un servicio. Define el conjunto de operaciones de un servicio

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Anexo B

ANEXO B

Pantallas de la EPG

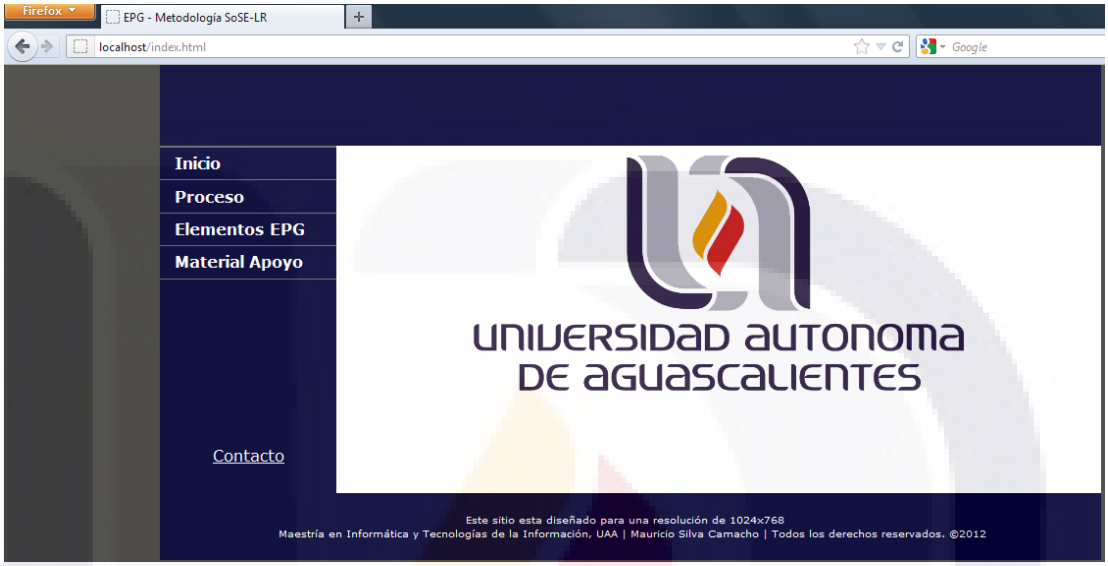


Ilustración 52 Pantalla de inicio al sitio

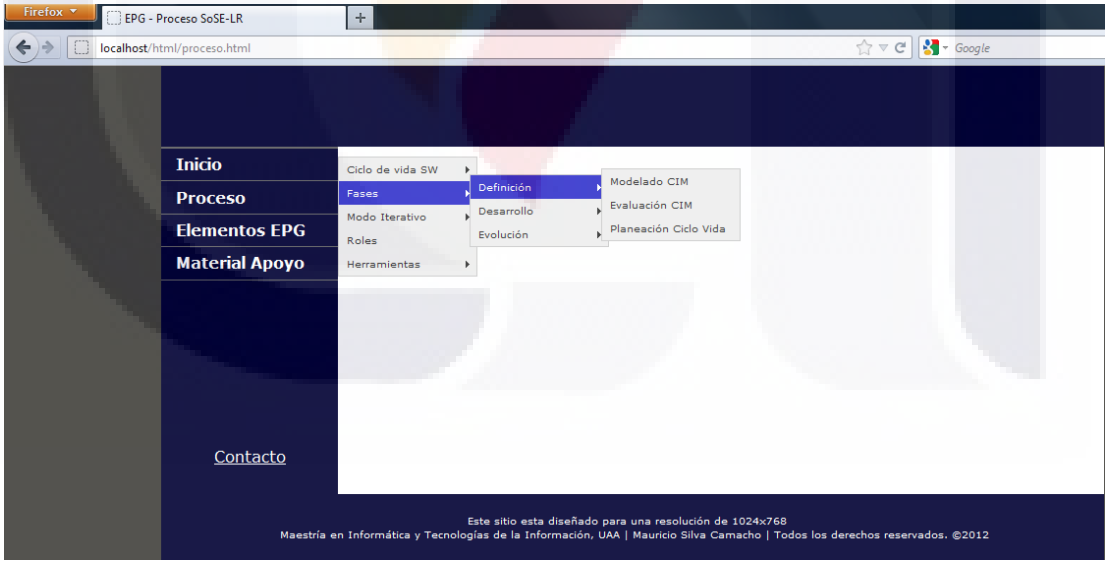


Ilustración 53 Elementos del Proceso SoSE-LR

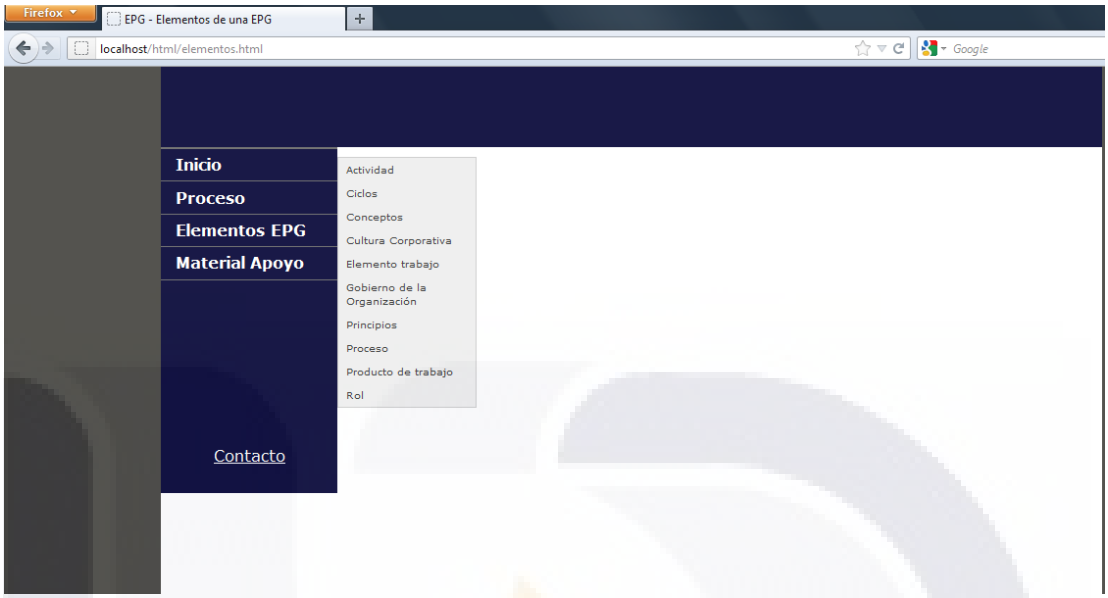


Ilustración 54 Elementos de la Guía Electrónica de Procesos

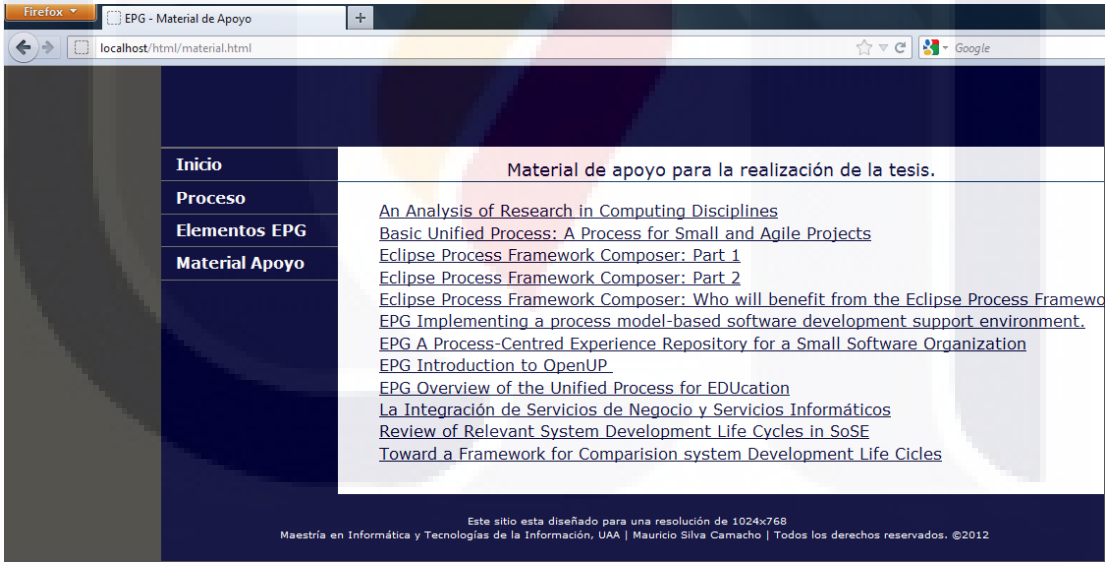


Ilustración 55 Material de Apoyo

Ilustración 56 Ciclo de Vida de Desarrollo de Software

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Anexo C

ANEXO C

Encuesta Demográfica

Encuesta Demográfica

INSTRUCCIONES. Por favor, antes de llenar los siguientes cuestionarios, responda a las siguientes preguntas con fines demográficos:

1.- Marque **solo una** Respuesta que describa mejor su actividad laboral en su organización:
( ) Trabajo de oficina
( ) Trabajo Profesional
( ) Jefe de departamento
( ) Subdirector de área
( ) Gerente

2.- Marque **solo una** Respuesta que describa mejor su nivel escolar:

( ) Preparatoria
( ) Carrera Técnica
( ) Licenciatura
( ) Maestría (MSc)
( ) Doctorado (PhD)

3.- Marque **solo una** Respuesta que describa mejor su rango de edad:

( ) 18-24
( ) 25-34
( ) 35-44
( ) 45-54
( ) 55- o más

4.- Marque **solo una** Respuesta que describa mejor su uso de servicio de correo electrónico:

( ) 0-4
( ) 5-9
( ) 10-14
( ) 15-19
( ) 20- o más (e.g. via BITNET, 1989)

5.- Marque **solo una** Respuesta que describa mejor su descripción de la área dentro de la organización en la que labora:

( ) Dirección General
( ) Área de Administración
( ) Área Financiera
( ) Área de Ingeniería / Diseño
( ) Área de Fabricación / Producción
( ) Área de Recursos Humanos

( ) Marketing y área de ventas
( ) Área de TI

6.- Marque **solo una** Respuesta que describa mejor el alcance de las operaciones comerciales de su organización de trabajo:

( ) Regional. ( ) Nacional. ( ) Mundial.

7.- Marque **solo una** Respuesta que describa mejor su nivel de cursos cortos tomados dentro de la organización relacionados con herramientas de productividad de TI :

( ) 0 cursos
( ) 1 – 2 cursos
( ) 3 o mas cursos

8.- Marque **solo una** Respuesta que describa mejor su nivel de experiencia en el uso de herramientas de productividad para desarrollo de software

( ) Principiante (Un dominio del 20% de uso de herramientas de productividad para desarrollo de software)
( ) Novato (Un dominio del 40% de uso de herramientas de productividad para desarrollo de software))
( ) Usuario Normal (Un dominio del 60% de uso de herramientas de productividad para desarrollo de software))
( ) Usuario avanzado (Un dominio del 80% de uso de herramientas de productividad para desarrollo de software))
( ) Usuario Experto (Un dominio del 100% de uso de herramientas de productividad para desarrollo de software))

Muchas gracias por su valiosa participación

Cuestionario Instrumento Conceptual de Métricas de Aceptación de Metodologías 2012

INSTRUMENTO CONCEPTUAL DE METRICAS DE ACEPTACIÓN DE METODOLOGIAS 2012

(Basado en Moore & Benbasat, 1991; Karahanna et. al. 1999).
Elaboración: L.I. Mauricio Silva Camacho.
Sistemas de Información / Ciencias Básicas
Universidad Autónoma de Aguascalientes

Favor de asignar de manera personal a cada estatuto el grado de acuerdo o desacuerdo que perciba, en base a la experiencia adquirida por utilizar la metodología <SoSE-LR> para desarrollar el caso asignado. Gracias por su colaboración en esta investigación.

CONSTRUCTO	Extremadamente en Desacuerdo	En Desacuerdo	Ligeramente en Desacuerdo	Neutral	Ligeramente de Acuerdo	De Acuerdo	Extremadamente de Acuerdo
<utilidad>							
VR.1 Utilizar una metodología (SOSE-LR) me habilita a cumplir mis tareas de desarrollo de SOSYSTEMS más rápidamente.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
VR.2 Utilizar una metodología (SOSE-LR) mejora la calidad de mi desarrollo de SOSYSTEMS.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
VR.3 Usar una metodología (SOSE-LR) realza la efectividad de mi proceso de desarrollo de SOSYSTEMS.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
VR.4 Usar una metodología (SOSE-LR) me da mayor control sobre el proceso de desarrollo de SOSYSTEMS.	**1**	**2**	**3**	**4**	**5**	**6**	**7**

CONSTRUCTO	Extremadamente en Desacuerdo	En Desacuerdo	Ligeramente en Desacuerdo	Neutral	Ligeramente de Acuerdo	De Acuerdo	Extremadamente de Acuerdo
<facilidad de uso>							
FU.1 Aprender a utilizar/operar SoSE-LR, sería fácil para mi	**1**	**2**	**3**	**4**	**5**	**6**	**7**
FU.2 En caso de obligación de usar SoSE-LR, sería fácil para mi.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
FU.3 En caso de obligación de usar SoSE-LR, sería difícil para mi.	**1**	**2**	**3**	**4**	**5**	**6**	**7**

CONSTRUCTO	Extremadamente en Desacuerdo	En Desacuerdo	Ligeramente en Desacuerdo	Neutral	Ligeramente de Acuerdo	De Acuerdo	Extremadamente de Acuerdo
<intención de uso>							
IU.1 Planeo y tengo la intención de usar la metodología (SOSE-LR) para desarrollar SOSYSTEMS, en mi próximo proyecto profesional en mi organización de trabajo.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
IU.1 Dentro de los próximos 6 meses, planeo y tengo la intención de experimentar parcialmente el uso de la metodología (SOSE-LR) para desarrollar SOSYSTEMS, en mi organización de trabajo.	**1**	**2**	**3**	**4**	**5**	**6**	**7**