UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 9

PRINCIPIOS ESTRATÉGICOS PARA LA GUÍA DEL PROCESO DE DESARROLLO DE SOFTWARE EDUCATIVO EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORES: Yordanys Piñeiro Gómez

Jacqueline Gallardo Collazo

TUTOR: Ing. Febe Ángel Ciudad Ricardo

CO-TUTOR: Ing. Alleyne Antonio Formoso Mieres

ASESORA: Lic. Elianis Cepero Fadraga

Ciudad de la Habana, Julio de 2007
“Año 49 de la Revolución"

Todos los seres estamos en el mundo para algo. Nuestra existencia tiene un sentido. Cada uno tenemos una misión que cumplir. Un camino que seguir. Un sueño que conquistar y que vivir. Un tesoro para buscarlo y encontrarlo. Una Leyenda Personal. Una vocación.

Pero muchas veces los hombres no vivimos nuestra Leyenda Personal, sino que nos instalamos en la seguridad de lo que ya tenemos y abandonamos la búsqueda de nuestro tesoro y no vivimos nuestro sueño.

Una búsqueda comienza siempre con la “Suerte del Principiante”. Y termina siempre con la “Prueba del Conquistador”.

Paulo Coelho

DEDICATORIA

A mis padres por ser mi principal inspiración de querer llegar a convertirme en una persona preparada, digna y trabajadora.

Jacqueline Gallardo Collazo

A mis padres y hermana, por todo el esfuerzo, entrega, amor y apoyo incondicional. A ustedes especialmente va dedicado todo mi esfuerzo.

A toda mi familia, por el apoyo brindado y la confianza depositada en mí.

A mamá (Modesta), por cuidarme y quererme tanto de niña y mantener ese amor hasta ahora.

A Fidel y a la Revolución, por hacer realidad sueños como este.

A mi tutor Febe, quien ha puesto todo su empeño en lograr el éxito del trabajo de diploma realizado.

A mi novio Alejandro, por su amor, comprensión y ayuda en todo este tiempo. Te quiero mucho.

A todos los amigos que de una forma u otra me han brindado su cariño, su ayuda y que han deseado que salga adelante en todos los aspectos de la vida.

A todas aquellas personas que han logrado ser una parte especial de mi vida, a las que alguna vez se lo he dicho y posiblemente no han dado tanto crédito.

A todos aquellos que forman parte de este logro.

Yordanys Piñeiro Gómez

AGRADECIMIENTOS

A Fidel y a la Revolución por darnos la oportunidad de hacer realidad nuestros sueños y permitirnos ser profesionales sencillos.

A mi madre por darme su cariño y ayudarme siempre a salir airosa de los momentos difíciles que se imponen en la vida.

A mi padre por darme su amor, dedicación y sentirse siempre tan orgulloso de las cosas buenas que hago.

A mis tres hermanos queridos por ser parte importante de mi vida personal.

A Febe Ángel Ciudad Ricardo por ser un tutor excepcional, que me brindó sus conocimientos para lograr superar una de las pruebas decisivas que te pone la vida.

A todas mis compañeras que aún están conmigo y a las que no están que me dieron su amistad y apoyo a lo largo de la vida universitaria que estamos por culminar.

A Yordanys Piñeiro Gómez que fue además de mi compañera de tesis, mi amiga que me dio su apoyo indispensable.

A Roberto Millet Luaces por el ser el profe que más influyó en mi desarrollo profesional durante estos años y por enseñarme a confiar en mí misma.

Y a todas aquellas personas que de una forma u otra, directa o indirectamente me dieron su aporte imprescindible en mi paso por la universidad y a las que hicieron posible la conformación de este Trabajo de Diploma.

Muchas gracias a todos.

Jacqueline Gallardo Collazo

Este agradecimiento no solo va dirigido a las personas que han hecho posible el desarrollo de este trabajo de diploma, sino además, a todos aquellos que me han apoyado a lo largo de mi vida, y que han estado junto a mí en los buenos y malos momentos. A todos ustedes: Muchas gracias, no saben cuanto ha significado que hayan estado ahí para mí. Especialmente quería agradecer:

A mi mamita linda por todo el amor, comprensión, dedicación, ayuda y entrega depositada en todos los años de mi vida. Te quiero mucho, gracias por estar siempre a mi lado, por confiar siempre en mi buen juicio para hacer las cosas, por darme un buen ejemplo como madre, por ser mi amiga y mi guía, por sostenerme cuando algo puede ir mal. Gracias por esto y por mucho más.

A mi hermanita, que la quiero con todo mi corazón, por cuidarme y estar siempre tan pendiente de mí.

A mi papito lindo, que aunque por desgracias de la vida no puede estar hoy conmigo, me dijo que pasara lo que pasara yo debía seguir adelante, y que confiaba en que todo iba a salir muy bien. A él le agradezco todos los años de dedicación, de ayuda profesional, de amor. Gracias por haber estado siempre a mi lado, por indicarme cuál debía ser el camino a seguir. Siempre te llevaré en mi corazón.

A nuestro tutor, Ing. Febe Ángel Ciudad Ricardo, por todo el apoyo, ideas, esfuerzos, dedicación y ayuda incondicional para el desarrollo de este trabajo de diploma.

A mi familia, y a todas aquellas personas que han contribuido a lo largo de los años en mi formación profesional y a ser cada día mejor persona.

A Fidel y a la Revolución por haber sido creadores y mantenedores de este gran proyecto. Por la confianza depositada en la juventud que se forma como profesional.

A Alejandro por quererme, cuidarme, apoyarme y estar junto a mí en los momentos difíciles. Te quiero.

A Jacqueline Gallardo, mi compañera de tesis, por su ayuda y comprensión. Por apoyarme en los buenos y malos momentos.

A Zoraida, Alcides, Isabel, Yaneisis, Alleyne y Osmel quería agradecerles especialmente, porque más que profesores han sido excelentes amigos. Gracias por haberme cuidado, por preocuparse y estar junto a mí cuando más los he necesitado. No se imaginan cuanto los quiero.

A los amigos que me han acompañado en el transcurso de estos 5 años y que han contribuido en gran medida a que la estancia sea más placentera.

Los quiero a todos.

Yordanys Piñeiro Gómez

DECLARACIÓN DE AUTORÍA

Por este medio declaramos que somos las únicas autoras de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que hagan el uso que estimen pertinente del mismo.

Para que así conste firmamos la presente a los __ días del mes de junio del 2007.

_____	_____
Yordanys Piñeiro Gómez	Jacqueline Gallardo Collazo
Autor	Autor

_____

Ing. Febe Ángel Ciudad Ricardo

Tutor

DATOS DE CONTACTOS:

TUTOR: Ing. Febe Ángel Ciudad Ricardo
Profesor Instructor de Ingeniería, Gestión de Software, Metodología de la Investigación Científica y Seminario de Tesis – UCI
Jefe del Departamento de la Especialidad – Facultad 9
Dirección: Universidad de las Ciencias Informáticas (UCI), Edificio: 40, Apto: 40104
Teléfono Oficina: +53 – 7 – 8372557 Teléfono Apto: +53 – 7 – 8358825 E-mail: fciudad@uci.cu

CO-TUTOR: Ing. Alleyne Antonio Formoso Mieres
Ingeniero Industrial
Graduado en la CUJAE en Julio de 2004
Profesor de la UCI desde Septiembre de 2004
Ha impartido clases de Contabilidad y Finanzas, Probabilidades y Estadística, Investigación de Operaciones
Dirección: Universidad de las Ciencias Informáticas (UCI), Edificio: 43, Apto: 43203
Teléfono Oficina: +53 – 7 – 8372559 Teléfono Apto: +53 – 7 – 8358848
E-mail: alleyne@uci.cu

ASESORA: MSc. Elianis Cepero Fadraga

MSc. Elianis Cepero Fadraga
Profesora Asistente. Licenciada en Educación Especialidad Idioma Inglés 1996. Instituto Superior Pedagógico “José Martí”, Camagüey, Cuba.
Master en Ciencias de la Educación Superior 2003 Universidad “Camilo Cienfuegos”, Matanzas, Cuba.
Miembro del "Grupo de Especialistas en Lengua Inglesa” de la Asociación de Lingüistas de Cuba.
Imparte clases de Idioma Extranjero I, II, III y IV en la Universidad de las Ciencias Informáticas desde el año 2002.
Profesora de la Disciplina Idiomas Extranjeros, Facultad 9.
Dirección: Universidad de las Ciencias Informáticas (UCI), Edificio: 21, Apto: 21103
Teléfono Oficina: +53 – 7 – 837 2560 Teléfono Apto: +53 – 7 – 835 8800 E-mail: ecepero@uci.cu

OPINIONES Y AVALES

El Trabajo de Diploma, titulado *Principios Estratégicos para la guía del proceso de desarrollo de software educativo en la UCI*, fue realizado en la Universidad de las Ciencias Informáticas. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

□ Totalmente

□ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

_____

_____

_____

_____

_____

_____

_____

Como resultado de la implantación de este trabajo se reportará un efecto económico que asciende a <valor en MN o USD del efecto económico>

Y para que así conste, se firma la presente a los ____ días del mes de Junio del año 2007.

_____ _____

Representante de la entidad Cargo

_____ _____

Firma Cuño

OPINIÓN DEL TUTOR

Título: Principios Estratégicos para la guía del proceso de desarrollo de software educativo en la UCI.

Autores: Yordanys Piñeiro Gómez

Jacqueline Gallardo Collazo

_____ _____/___/_____

Firma Fecha

Ing. Febe Ángel Ciudad Ricardo

RESUMEN

El avance vertiginoso de la Ciencia y la Tecnología en la actualidad, ha traído consigo numerosos cambios en disímiles sectores y esferas de la sociedad a través de la introducción de las Tecnologías de la Información y las Comunicaciones (TIC). Su impacto ha propiciado que se hayan realizado numerosas modificaciones en la concepción del proceso de enseñanza-aprendizaje, el cual está siendo apoyado por la gran gama de productos de software educativo que han sido elaborados a nivel internacional.

La Universidad de las Ciencias Informáticas (UCI) se ha fusionado hace algunos años a este amplio y competitivo mercado que tiene como fuerte línea de producción el software educativo, como vía de integración de estos procesos de enseñanza-aprendizaje con las tecnologías de la información y las comunicaciones a las cuales se hizo referencia anteriormente. Los estudios que han sido realizados, sobre el proceso que se lleva a cabo para la elaboración de estos productos, han arrojado un significativo número de problemas que atentan contra el correcto desarrollo de los proyectos de producción de software educativo.

Teniendo en cuenta dichas dificultades, se decidió que a partir de un riguroso estudio de los modelos de desarrollo de software que se conocen y utilizan en la actualidad, de las numerosas propuestas metodológicas, que van desde las más tradicionales hasta las que hoy en día conocemos como ágiles, de las actividades y tareas que forman parte de la fases genéricas de la ingeniería del software planteadas por Pressman, de los estándares internacionales relacionados con el desarrollo de software, así como, del flujo productivo que se ha establecido en la Universidad para guiar el proceso de desarrollo de software educativo, plantear un conjunto de principios que sirvan de guía para el desarrollo de software educativo en la UCI, y que a su vez, contribuyan con su aplicación a minimizar los tiempos de desarrollo, reducir los costos del proyecto y garantizar la calidad del producto software.

Este trabajo está estructurado en dos capítulos. El Capítulo 1, contempla la fundamentación teórica de esta investigación, en la cual son expuestos los principales conceptos y argumentos que esclarecen el objeto de estudio. Por su parte, en el Capítulo 2 se realiza un acercamiento al flujo productivo de la UCI y se describe un nuevo enfoque para la producción de software educativo en la misma.

Palabras Claves

Software educativo, Proceso, Modelos, Metodologías, Principios, Guía.

ÍNDICE DE TABLAS Y FIGURAS

ÍNDICE

INTRODUCCIÓN

“Las organizaciones que desarrollan o mantienen software pueden optar por trabajar a la buena de Dios, o por seguir una metodología. Hacerlo a la buena de Dios no es tan raro. Es la primera forma que se empleó para desarrollar programas: Aquí tenemos unos ordenadores, aquí unos señores a los que les encanta leer los manuales de programación, y se trata sencillamente de conseguir que estas máquinas terminen imprimiendo facturas (o haciendo lo que sea)”. (PALACIO 2005)

Trabajar a la “buena de Dios” ha sido y sigue siendo en la actualidad la alternativa con la cual algunas empresas o determinados proyectos han trabajado y continúan trabajando, no constituyendo ésta, la vía más adecuada para el desarrollo de software. El éxito de aquellos equipos que logran producir eficientemente, en el tiempo planificado y con los costos establecidos bajo el enfoque anteriormente mencionado, depende íntegramente del conocimiento tácito del personal que está involucrado en el mismo, adquirido por sus propios esfuerzos, que tienen un gran talento y que son capaces de construir software con alta calidad, cuyo trabajo dependerá de cuán capaces sean para lograr los resultados que se necesitan y que el cumplimiento de tiempos de desarrollo y de costos del proyecto dependerá de la forma en que sean capaces de organizarse, de prever los riesgos que puedan presentarse y tener la solución más adecuada a la mano.

Al modo de producción referenciado anteriormente, se le suele llamar “programación heroica” y representa al conjunto de personas que indudablemente saben hacer software, no a las empresas que saben hacer software; de esta forma se está trabajando sin seguir método alguno, lo cual responde a lo que se conoce en la actualidad como producción artesanal, caracterizada a lo largo de la historia por no tener definido un conjunto de tareas por las cuales todos los involucrados puedan guiarse a fin de obtener resultados similares. Dicha producción no contempla un patrón con el cual se pueda realizar una planificación organizada, y por tanto predecir cuál será el resultado a obtener, el tiempo a emplear en la construcción de determinado producto y por inferencia establecer la posibilidad de terminación de una cantidad de productos en un tiempo establecido. Por estas razones es que se hace necesario que haya una evolución de esta concepción artesanal a una producción industrial, donde se garantice la erradicación de las deficiencias que caracterizan al método anteriormente descrito, donde se establezca

un proceso estándar que garantice de antemano qué es lo que se va a hacer, cómo se va a hacer, qué herramientas se emplearán para obtener el producto resultante, qué tiempo será necesario emplear para la concepción de un producto, cuáles son los costos que reportará la construcción del mismo. Bajo este enfoque se debe garantizar que dichas empresas sigan una metodología para el desarrollo del software, pues su estabilidad y nivel no pueden depender de un conjunto de personas que decidan emprender un proyecto determinado contando con un personal que tenga talento solamente, pues solo tendrá buenos resultados si sus creadores son capaces de cumplir con una serie de requisitos que son imprescindibles para lograr el éxito y que muy pocas personas se identifican con ellos.

Además, ¿qué pasaría cuando se quiera lograr un mayor crecimiento en la organización y se necesite trabajar en un proyecto que se caracterice por tener gran envergadura? ¿Acaso ese conjunto finito de personas será capaz de resolver de una forma adecuada el groso del problema que será propuesto? Esto nos conduce a arribar a la conclusión, de que indudablemente, necesitamos de empresas que sepan hacer software, y que produzcan de forma eficiente en todos los proyectos que le sean asignados, obteniendo resultados de alta calidad.

“Las empresas que desarrollan software no pueden ignorar que su negocio es un negocio de software, y que el modelo que cada una adopte para las actividades de desarrollo y mantenimiento tiene implicaciones relevantes en la eficiencia general del negocio. El problema que pueden encontrar quienes deciden implantar métodos más eficientes, es caer en la desorientación ante el abanico de modelos de calidad, de **procesos** y de técnicas de trabajo desplegado en la última década; o abrazar al primero que se presenta en la puerta de la organización como “solución” de eficiencia y calidad”. (PALACIO 2005).

De aquí se deriva otro problema fundamental: se decide trabajar con una metodología para guiar el proceso de desarrollo del software, pero ¿cuál de ellas se debería escoger? Esta es una decisión que demanda tiempo de análisis, pues el éxito en la construcción de un producto depende en gran medida del modelo que se decida escoger, un modelo que se adapte a la naturaleza y características particulares del proyecto, de la aplicación, controles y entregas a realizar, y requiere que para seleccionarlo, se realice un estudio detallado y profundo de todas las alternativas por las cuales se pueda llegar a la solución apropiada. Además, para escoger el modelo de proceso adecuado es necesario tener en cuenta las características del producto, su tamaño o complejidad, el entorno del proyecto en el que trabaja el equipo de software y otras características.

Conforme avanzan los años, se han ido desarrollando y perfeccionando los modelos, procesos y metodologías que son utilizados para el desarrollo del software; las modificaciones que han sufrido cada uno de ellos sientan las bases en experiencias anteriores o simplemente en nuevas necesidades que se van concibiendo de acuerdo a las características propias y externas con las que se identifique el software, tratando siempre de limar las deficiencias que hacen que dicha selección no sea la más óptima.

En la actualidad se han podido identificar una gran gama de modelos y metodologías, que se han ido desplegando a nivel internacional y adaptando a las condiciones de la empresa donde se esté desarrollando y a las características del producto en particular. Por la importancia que se le adhiere a los métodos relacionados con el proceso de desarrollo de software, muchos autores han dedicado un preciado tiempo a la organización, explicación y mejora de los mismos. Además existe un significativo número de clasificaciones para los modelos de desarrollo de software y las metodologías que se conocen.

Actualmente la Ciencia y la Tecnología están ocupando un lugar cimero a escala internacional. Su desarrollo acelerado en los últimos tiempos ha provocado numerosos cambios en disímiles sectores y esferas de la sociedad a través de la introducción de las Tecnologías de la Información y las Comunicaciones (TIC). Dichas tecnologías han ido evolucionando de forma vertiginosa debido a la propia necesidad de intercambio de información, comunicación y socialización que existe en el mundo actual. El insaciable interés por conocer qué recursos, material o innovación tecnológica existe en cualquier lugar, al que por diversas razones no se tenga acceso, ha permitido una consecuente investigación que ha provocado la integración de varios medios de información. Con ellos se ha logrado un avance tan vertiginoso, que se ha llegado a afirmar por diversos autores, que la sociedad ha entrado en un nuevo milenio, inmerso en lo que podríamos llamar “sociedad o era de la información, las comunicaciones y el conocimiento”. Se podría agregar a esta frase una parte que dijera “…y de las transformaciones”, porque no solo los términos información, comunicación y conocimiento recogen los numerosos cambios que trae consigo la introducción y aplicación de las TIC, sino que a todo esto viene unido un conjunto de transformaciones políticas, económicas, sociales y laborales que repercuten directamente en la sociedad de forma involuntaria, pues forman parte de los resultados propios de los avances tecnológicos.

El impacto que han tenido las TIC ha propiciado numerosas modificaciones en la concepción del modelo de enseñanza, que ha pasado de un modelo de enseñanza tradicional, donde el profesor tiene el papel protagónico, y de su conocimiento depende el éxito o fracaso del estudiante que está educando, las habilidades que va adquiriendo, los valores que va forjando, el conocimiento que va acumulando y desarrollando; a un modelo que se adapte a los cambios que se están produciendo en la sociedad, que incluye las nuevas concepciones pedagógicas adaptadas a los medios que servirán de apoyo al desarrollo de la clase y la puesta en marcha de un estudio individual con un espacio diferente al habitual.

Para lograr el éxito y avance en el sistema de conocimientos y habilidades que debe tener cada individuo de la sociedad actual, ha sido trascendental la inserción del sistema de enseñanza en un espacio donde la tecnología juega un papel primordial pues las propias necesidades y exigencias predominantes abogan por un cambio radical que permita ver más allá de lo que está reflejado en un libro de texto, en las palabras de un profesor o simplemente en lo que ves o escuchas a tu alrededor.

El deseo de lograr un mecanismo que se convirtiera en rector de la asimilación tecnológica, del conocimiento desde el punto de vista educacional e instructivo que se necesita, lleva a la concepción y creación dentro del gran campo que abarca el software, al **software educativo** en particular, el cual, además de tener las características comunes del software en general como programa computacional; estructural y funcionalmente deben servir de apoyo, de guía, de facilitador del proceso de enseñanza-aprendizaje en todas sus modalidades. La concepción del mismo ha ganado gran aceptación por parte de la sociedad en toda su dimensión, creciendo constantemente el índice de demandas de variadas tipologías de software educativo de calidad en gran parte de las instituciones a nivel internacional, fundamentalmente en aquellas que pertenecen al sector educacional.

Es importante señalar que la creciente demanda que ha tenido el software educativo en los últimos años, ha provocado que se hayan descuidado un poco algunos aspectos que son elementales para garantizar la calidad de los productos resultantes. Muchos de ellos se han desarrollado de forma desorganizada, poco documentada, sin una planificación real y en muchos casos sin seguir una metodología de software determinada.

Nuestro país, se encuentra inmerso en un proceso de transformaciones dentro del cual está priorizado el sector educacional. Se han creado muchas instituciones en el país que contribuyen al crecimiento informático, y como parte de los aportes que cada día sustentan ese desarrollo se encuentra la

elaboración de diversos software educativos. La Universidad de las Ciencias Informáticas (UCI) se ha unido desde hace algunos años a los programas conque cuenta nuestra nación para alcanzar un logro inmediato y superior al existente en el sector informático. Se ha dedicado tiempo, costo y esfuerzo para la creación de diversos productos que vayan ayudando a lograr los objetivos anteriormente mencionados, pero el propio hecho de ser una institución que actualmente se encuentra en formación, donde no existe mucha experiencia por parte del personal y donde se han tenido que realizar diversas modificaciones a partir de los resultados obtenidos según experiencias anteriores, ha provocado la aparición de diversas dificultades en el desarrollo de los productos de software educativo, las cuales vienen dadas por numeroso factores que atentan contra su desarrollo exitoso. Todo esto, unido a la falta de procedimientos para guiar al personal que se encuentra involucrado en el desarrollo de los mismos, es lo que hace que se publique en el presente año un Sistema Metodológico para el desarrollo de software educativo en la UCI, para el cual se realizó un estudio basado en los resultados de proyectos anteriores. Este sistema metodológico se realiza como solución inmediata y versión inicial de lo que sería una guía que contribuyera a reducir los problemas que se han ido presentando en los proyectos que se identifican con el área educativa. Dicho Sistema Metodológico no contempla un conjunto de características que son necesarias para la adecuada ejecución de los proyectos, de ahí que aún no cumpla con todas las expectativas que se requieren para elaborar un software de alta calidad, en el tiempo planificado y con los costos establecidos.

Por todo lo anteriormente expuesto se puede plantear que: “La cada vez más creciente necesidad de integración entre los procesos de enseñanza-aprendizaje y las tecnologías de la Información y las Comunicaciones ha generado una alta demanda de productos de software educativo a nivel mundial. Ante este fenómeno un amplio y competitivo mercado de desarrollo de software educativo incrementa la presión sobre los proyectos de esta rama en materia de eficiencia y calidad de las soluciones en todos los aspectos del desarrollo: tecnológicos, financieros, contractuales, etc.”(MARTÍNEZ *et al.* 2007) Para su creación no se ha podido contar con un conjunto amplio de metodologías que permitan la guía de estos procesos productivos pues no existe un estándar internacional que regule la gestión de los mismos. La UCI constituye hoy en día un eslabón esencial en la producción de software educativo y es por ello que necesita la documentación necesaria para guiar dichos procesos, la cual es muy escasa y la existente no resuelve en sí las particularidades de cada uno. Esto conlleva a que en muchos casos los

tiempos de desarrollo que se emplean sean mayores a lo planificado, lo que trae consigo un aumento en los costos, y podría repercutir en gran medida en la calidad del producto resultante.

Precisamente, retomando algo que se decía anteriormente, sí existen un significativo número de modelos, metodologías y procesos para guiar el desarrollo de software, pero la inserción del software educativo a esta gran industria ha traído consigo, el replanteamiento de nuevos métodos que estén dirigidos básicamente a la solución de determinadas características que hacen que se tengan que tomar en cuenta nuevas alternativas que se adapten a un software de nuevo tipo. Su reciente aparición en el mercado del software trae consigo la no existencia de un conjunto vasto de metodologías que se dediquen especialmente a tratar esta nueva concepción, son muy pocos los autores que han escrito al respecto, y los que lo han hecho básicamente han realizado un conjunto de adaptaciones a los modelos o procesos existentes, agregándoles nuevas actividades que se corresponden con dichas particularidades. Con respecto a esto Cataldi afirmó:

“Uno de los problemas más importantes con los que se enfrentan los ingenieros en software y los programadores en el momento de desarrollar un software de aplicación, es la falta de marcos teóricos comunes que puedan ser usados por todas las personas que participan en el desarrollo del proyecto informático.

El problema se agrava cuando el desarrollo corresponde al ámbito educativo debido a la inexistencia de marcos teóricos interdisciplinarios entre las áreas de trabajo”. (CATALDI 2000)

Como se hace alusión en la cita anterior, no existe un estándar que pueda ser usado para el desarrollo de software por todas las personas que se encuentren vinculadas al desarrollo de proyectos informáticos, mucho menos, cuando se trata de productos correspondientes al plano educativo, en el cual se hace necesario tener en cuenta los aspectos pedagógicos-didácticos y de comunicación con el usuario para cada caso en particular, de ahí que se necesite investigar y arrojar conclusiones que definan cuál es el modelo o proceso más idóneo que se deba utilizar y qué adaptaciones son necesarias que se hagan en el mismo de acuerdo a las teorías de aprendizaje, para que se pueda lograr un producto óptimo.

Las condiciones en las cuales se desenvuelve la producción de software educativo a nivel internacional y de forma particular en la universidad, conlleva a que se haya identificado el problema científico:

Escasez de un conjunto de principios que regulen el proceso productivo de software educativo en la Universidad de las Ciencias Informáticas (UCI).

Para resolver el problema referenciado anteriormente se hizo necesario que el objetivo general de esta investigación esté dirigido a plantear un conjunto de principios que permitan la guía de los procesos productivos de software educativo en la UCI.

Para darle cumplimiento a dicho objetivo fue necesario centrar la investigación en el **proceso de desarrollo de software educativo**, lo cual constituye el objeto de estudio.

El conjunto de principios que serán planteados como solución de esta investigación tendrán su incidencia final en la **gestión de los proyectos productivos de software educativo en la Universidad de las Ciencias Informáticas**, lo cual determina el campo de acción de la misma.

Para darle cumplimiento al objetivo trazado se determinó que las tareas a realizar en esta investigación estarían dirigidas a:

1. Estudiar el arte del tema en Cuba y el mundo.
2. Estudiar las condiciones actuales en la cuales se desarrolla el software educativo en la UCI.
3. Estudiar los aspectos contenidos en los estándares internacionales y nacionales para el desarrollo de software según las normas de la IEEE y la ISO.
4. Recopilar información sobre el funcionamiento de los proyectos productivos de software educativo en la UCI.
5. Describir un conjunto de principios que sirvan de guía para el desarrollo de software educativo en la UCI.

El desarrollo exitoso de las tareas expuestas anteriormente contribuirá al cumplimiento de la **idea a defender** de esta investigación: Si se realiza un estudio riguroso de los modelos, metodologías, procesos y estándares internacionales para el desarrollo de software, así como del contexto de producción del software educativo en la UCI, se podrán definir un conjunto de principios que sirvan de guía para el desarrollo de software educativo en la UCI.

Para el desarrollo de la investigación se utilizaron métodos teóricos y empíricos. Dentro de los métodos teóricos serán utilizados los de análisis y síntesis, el histórico-lógico y el causal y dentro de los métodos empíricos serán utilizadas la entrevista y la encuesta. A continuación se especifica el por qué de la selección de los mismos.

1. Entrevistas y Encuestas a líderes de proyecto de software educativo en la UCI, así como a directivos de producción educativa en la Universidad, para recopilar toda la información que estas nos puedan suministrar, para conocer cómo se desarrolla el proceso de desarrollo de software educativo de forma general en la universidad.
2. Se aplicará el "Método Histórico" para investigar si existe un proceso planteado en la universidad que guíe en la actualidad el proceso de desarrollo de software educativo, y de ser así se indagará en función de conocer a ciencia cierta cuáles son los resultados de la aplicación de dicho proceso en la elaboración de un producto determinado.
3. El método "Causal" para estudiar los factores que provocan la necesidad de un proceso de este tipo en la institución.
4. El método de "Análisis" para comprender y enunciar los principios para la guía del proceso de desarrollo de software educativo y el de "Síntesis" para plantear, describir y resumir el proceso que se desarrollará a partir de la investigación en la UCI.

Con el objetivo de lograr resultados que le dieran credibilidad a esta investigación fue necesario seleccionar una población, de la cual se extrajo una muestra que fue analizada. De los resultados obtenidos a partir de este análisis se pudo inferir como se iba a comportar dicha población. A continuación se presentan:

Población: Conjunto de líderes de proyecto de software educativo de la Universidad de las Ciencias Informáticas.

Muestra: Siete líderes de proyecto de software educativo

Unidad de estudio: Un líder de proyecto de software educativo.

Para la selección de la muestra fue utilizada la técnica de muestreo aleatorio simple, la cual establece que teniendo en cuenta la población escogida, deben ser enumerados cada uno de los proyectos que

forman parte de dicha población y a partir de este resultado se debe realizar un sorteo, eligiendo casos particulares hasta obtener el tamaño de la muestra necesaria para la investigación.

El trabajo que se presenta a continuación está estructurado en dos capítulos. El Capítulo 1 contempla la fundamentación teórica de esta investigación, en la cual son expuestos los principales conceptos que contribuyen al mejor entendimiento del problema en cuestión, se especifican detalladamente todos los argumentos que esclarecen el objeto de estudio y se explican sintetizadamente otras soluciones existentes.

Por su parte, en el Capítulo 2 se realiza un acercamiento al flujo productivo de la Universidad de las Ciencias Informáticas y a partir del análisis de un conjunto de factores que intervienen indistintamente en el desarrollo de software se hace necesario realizar un conjunto de observaciones que sirven como eslabón fundamental para el planteamiento de un conjunto de principios estratégicos para la guía del proceso de desarrollo de software educativo en la UCI.

Para finalizar se presentan las conclusiones, las recomendaciones, las referencias bibliográficas, la bibliografía, un glosario de términos y el conjunto de anexos para un mejor entendimiento de lo expuesto a lo largo del trabajo.

CAPÍTULO I

Fundamentación Teórica

1.1 Introducción

"El problema del software se reduce a la dificultad que afrontan los desarrolladores para coordinar las múltiples cadenas de trabajo de un gran proyecto de software. La comunidad de desarrolladores necesita una forma coordinada de trabajar. Necesita un **proceso** que integre las múltiples facetas del desarrollo. La presencia de un proceso bien definido y bien gestionado es una diferencia esencial entre proyectos hiperproductivos y otros que fracasan". (JACOBSON *et al.* 2000)

"Un **proceso** define *quién* está haciendo *qué*, *cuándo* y *cómo* alcanzar un determinado objetivo. En la ingeniería del software el objetivo es construir un producto software o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad. Captura y presenta las mejores prácticas que el estado actual de la tecnología permite. En consecuencia, reduce el riesgo y hace el proyecto más predecible. El efecto global es el fomento de una visión y una cultura enormes". (JACOBSON *et al.* 2000)

Es precisamente el proceso de desarrollo de software el objetivo esencial sobre el cual gira la investigación realizada para el desarrollo del Capítulo 1, pues constituye la base para el análisis y entendimiento del objeto de estudio que rige la misma, el cual se centra en el desarrollo del software educativo en la Universidad de las Ciencias Informáticas. De ahí que se haya hecho necesario plantear los fundamentos teóricos que contribuyen a esclarecer el origen y desarrollo del mismo, hasta llegar al planteamiento y estudio de un conjunto de soluciones que son utilizadas en la actualidad para la producción de software de forma general, así como, propuestas para el software educativo en particular.

1.2 Conceptos asociados al dominio del problema

Con el objetivo de proporcionarle al lector, la vía más adecuada para una mayor comprensión de los temas que serán abordados en este capítulo, directamente relacionados con el objeto de estudio de la investigación, se describen detalladamente a continuación un grupo de conceptos asociados al dominio del problema, entre los que se encuentran: el software educativo, el proceso de enseñanza – aprendizaje y las Tecnologías de la Información y las Comunicaciones (TIC).

1.2.1 Software educativo

Sánchez J. (1999), en su libro "Construyendo y Aprendiendo con el Computador", define al software educativo como un programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar. Un concepto más restringido de software educativo lo define como aquel material de aprendizaje especialmente diseñado para ser utilizado con un computador en los procesos de enseñar y aprender.(ÁLVAREZ and RODRÍGUEZ 2006)

La definición anterior es reflejada en al artículo: “Software Educativo. Su influencia en la escuela cubana”, cuyas autoras expresan posteriormente:

“Finalmente, los Software Educativos se pueden considerar como el conjunto de recursos informáticos diseñados con la intención de ser utilizados en el contexto del proceso de enseñanza – aprendizaje. Se caracterizan por ser altamente interactivos, a partir del empleo de recursos multimedia, como videos, sonidos, fotografías, diccionarios especializados, explicaciones de experimentados profesores, ejercicios y juegos instructivos que apoyan las funciones de evaluación y diagnóstico.” (ÁLVAREZ and RODRÍGUEZ 2006)

El software educativo está formado por los programas educativos y didácticos creados con la finalidad específica de ser utilizados para facilitar los procesos de enseñanza – aprendizaje, como se había reflejado en las definiciones anteriores, es decir, tiene la finalidad de lograr que el estudiante adquiera conocimientos, habilidades, procedimientos, y esto es posible gracias a la interactividad que surge entre el estudiante y el computador, a través del cual se puede individualizar el trabajo de ese estudiante,

pues este tipo de software permite adaptar el ritmo de trabajo a cada uno, así como las actividades que tiene implícitas según la actuación de los mismos.

Es importante destacar que existen programas que son utilizados en los centros educativos con funciones didácticas o instrumentales, que aunque pueden desarrollar dichas funciones, no han sido elaborados específicamente con esa finalidad, por tanto se considera que está excluido de la categoría de software educativo. Ejemplo de ello tenemos a los procesadores de textos, gestores de bases de datos, hojas de cálculo, editores gráficos, entre otros.

1.2.2 Proceso de enseñanza – aprendizaje

"El **proceso de enseñanza** – **aprendizaje** constituye la vía mediatizadora esencial para la apropiación de conocimientos, habilidades, hábitos, normas de relación, de comportamiento y valores, legados por la humanidad, que se expresan en el contenido de enseñanza, en estrecho vínculo con el resto de las actividades docentes y extradocentes que realizan los estudiantes". (J. ZILBERSTEIN 1999)

"Enseñanza y aprendizaje forman parte de un único proceso que tiene como fin la formación del estudiante". (MANCEBO 1999)

"El proceso de enseñanza – aprendizaje es un proceso esencialmente interactivo y comunicativo, de intercambio de información, compartiendo experiencias, conocimientos y vivencias, que logran una influencia mutua en las relaciones interpersonales". (FERNÁNDEZ 2000)

El objetivo final de este proceso es la formación del estudiante y esto se logrará a través de como el profesor muestra a sus alumnos los contenidos, hábitos y habilidades, utilizando medios y con la intención de lograr objetivos fijados desde un inicio en un determinado contexto.

1.2.3 Tecnologías de la Información y las Comunicaciones (TIC)

"Las TIC brindan condiciones óptimas para transformar una enseñanza tradicional, pasiva, fundamentalmente centrada en la transmisión del contenido, el profesor y la clase, en otro tipo de educación más personalizada, participativa, centrada en alcanzar aprendizajes diversos y que posea una real significación para cada estudiante, y que esté dirigida a lograr una dimensión profundamente

humana y capaz de desarrollar la personalidad de todos los participantes conjuntamente con una determinada transmisión de contenidos y actualización cultural. Pero ellas por si solas no garantizan el éxito". (HEVIA ?)

"Las *NTIC*[1], se convierten en una indispensable herramienta para acelerar los procesos de enseñanza-aprendizaje, elevar la calidad de los mismos, convertirlo en un proceso permanente de la sociedad y no solo durante la etapa de estudios académicos. Las NTIC deben contribuir a fomentar los procesos de investigación e innovación en los ámbitos curricular, metodológico, tecnológico y organizativo del *proceso enseñanza – aprendizaje."* (LAMAS 2000)

" Las *nuevas tecnologías no sólo mejoran el entorno de la enseñanza y el aprendizaje, sino que lo están cambiando"* (BATES 2001). La inserción en la educación de estas tecnologías ha tenido un gran impacto y aunque aparentemente les proporciona facilidad a los profesores son muy factibles para lograr el entendimiento del contenido impartido al estudiante.

1.3 Objeto de Estudio

1.3.1 Descripción General

La introducción de esta investigación refleja que el objeto de estudio sobre el cual versa este trabajo de diploma es el proceso de desarrollo de software educativo, enmarcado específicamente en la Universidad de las Ciencias Informáticas.

Para un mejor entendimiento del mismo se hace necesario que se comiencen planteando un conjunto de conceptos que definen al proceso de forma general y al proceso de desarrollo de software de forma particular, pues estos constituyen la base de los posteriores avances que se han ido realizando en función de la organización y desarrollo de los mismos.

[1] NTIC se refiere a las Nuevas Tecnologías de la Información y las Comunicaciones, lo que hoy en día se conoce como TIC (Tecnologías de la Información y las Comunicaciones)

Se parte de que un proceso es “un conjunto de actividades o acciones interrelacionadas entre sí que, a partir de una o varias entradas de información, materiales o de salidas de otros procesos, dan lugar a una o varias salidas también de materiales (productos) o información con un valor añadido”.

De igual forma Blaya plantea que un proceso es “un conjunto de actuaciones, decisiones, actividades y tareas que se encadenan de forma secuencial y ordenada para conseguir un resultado que satisfaga plenamente los requerimientos del cliente al que va dirigido”.(BLAYA 2006)

En el libro “El Proceso Unificado de Desarrollo de Software”, se expresa que el proceso de desarrollo de software en particular, lo podríamos definir como “el conjunto completo de actividades necesarias para convertir los requisitos de usuario en un conjunto consistente de artefactos que conforman un producto software, y para convertir los cambios sobre esos requisitos en un nuevo conjunto consistente de artefactos”. (JACOBSON *et al.* 2000)

Por su parte Roger S. Pressman en “Ingeniería del Software. Un Enfoque Práctico” caracteriza el proceso de desarrollo de software como un marco de trabajo de las tareas que se requieren para construir software de alta calidad. Dicho proceso se muestra en la Figura 1.1 Los elementos que se involucran se describen a continuación:

- **Un marco común del proceso**, definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos de software, con independencia del tamaño o complejidad.
- **Un conjunto de tareas**, cada uno es una colección de tareas de ingeniería del software, hitos de proyectos, entregas y productos de trabajo del software, y puntos de garantía de calidad, que permiten que las actividades del marco de trabajo se adapten a las características del proyecto de software y los requisitos del equipo del proyecto.
- **Las actividades de protección**, tales como garantía de calidad del software, gestión de configuración del software y medición, abarcan el modelo del proceso. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso.”(PRESSMAN 2002)

Figura 1 El proceso del software

"Ningún proceso de desarrollo de software es de aplicabilidad universal. Los procesos varían porque tienen un lugar en diferentes contextos, desarrollan diferentes tipos de sistemas, y se ajustan a diferentes tipos de restricciones del negocio (plazos, costos, calidad y fiabilidad). Por consiguiente, un proceso de desarrollo de software del mundo real debe ser adaptable y configurable para cumplir con las necesidades reales de un proyecto y/u organización concreta".(JACOBSON *et al.* 2000)

Además se debe reflejar que para desarrollar un software es necesario utilizar un proceso soportado por herramientas, pues estas son esenciales en el mismo.

"El proceso se ve influido fuertemente por las herramientas. Las herramientas son buenas para automatizar procesos repetitivos, mantener las cosas estructuradas, gestionar grandes cantidades de información y para guiarnos a lo largo de un camino de desarrollo concreto".(JACOBSON *et al.* 2000)

"Un proceso de software define el enfoque que se toma cuando el software es tratado por la ingeniería. Pero la ingeniería del software también comprende las tecnologías que tiene el proceso – métodos técnicos y herramientas automatizadas –". (JACOBSON *et al.* 2000)

1.3.1.1 Modelos de desarrollo de software

Para que este conjunto de actividades que definen al proceso no se lleven a cabo de forma caótica se ha hecho necesario incorporar una estrategia cuyo objetivo fundamental fuera ordenar dichas actividades en el desarrollo del software. Esta estrategia es lo que se suele llamar Modelo de proceso del software. Dichos modelos han ido evolucionando de forma significativa con el paso de los años y se han ido adaptando a las necesidades y características propias de cada proyecto. De ahí que en la actualidad cada ingeniero seleccione un modelo de proceso para la ingeniería del software según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse, y los controles y entregas que se requieren.

Estos modelos han sido organizados de manera heterogénea por diferentes autores, por lo que se ha decidido tomar como referencia los modelos planteados por Pressman en su libro “Ingeniería del Software. Un Enfoque Práctico”, agrupados de la siguiente forma:

Modelo Lineal Secuencial, Modelo de Construcción de Prototipos, Modelo DRA y Modelos Evolutivos, en el se incluye el Modelo Incremental, el Modelo Espiral, el Modelo Espiral WinWin y el Modelo de Desarrollo Concurrente, entre otros modelos que hace alusión.

Teniendo en cuenta la importancia de estos modelos para el desarrollo de software se decidió que se debía realizar una breve explicación de cuatro modelos que han tenido un gran impacto en el desarrollo de productos software.

1.3.1.1.1 Modelo en Cascada

El Modelo Lineal Secuencial o Modelo en Cascada es el paradigma más antiguo y más extensamente utilizado por la ingeniería del software. Su versión original fue presentada por Royce en 1970, aunque los refinamientos realizados por Boehm (1981), Sommerville (1985) y Sigwart (1990) son mucho más conocidos. Dicho modelo sugiere un enfoque sistemático, secuencial, para el desarrollo del software, es decir, el producto evoluciona a través de una secuencia de fases ordenadas en forma lineal, permitiendo iteraciones al estado anterior. Aunque las etapas o actividades que comprende este modelo suelen variar podríamos resumirlas en: Ingeniería y modelado de Sistemas/Información, Análisis de los requisitos del software, Diseño, Generación de Código, Pruebas y Mantenimiento. (Ver Anexo 1)

En la práctica se ha logrado demostrar que este modelo carece de eficacia, sus problemas se centran en que en los proyectos reales, raras veces se sigue la linealidad que propone el mismo, además de que se ha demostrado que casi siempre hay iteraciones que siguen más allá de la etapa anterior. Conjuntamente, este modelo requiere que el cliente exponga explícitamente todos los requisitos, cosa que es muy difícil para este, lo que trae consigo numerosas dificultades a la hora de acomodar la incertidumbre natural al comienzo de muchos proyectos. También este modelo se caracteriza por la no disponibilidad de una versión de trabajo del programa hasta que el proyecto esté muy avanzado lo que provoca un efecto desastroso del mismo ante la aparición de un grave error que no haya sido detectado hasta revisar el programa.

Aunque este paradigma tenga estas debilidades es significativamente mejor que un enfoque hecho al azar para el desarrollo del software.

1.3.1.1.2 Modelo de Construcción de Prototipos

El Modelo de Construcción de Prototipos (Ver Anexo 2) contribuye a eliminar las debilidades que fueron expuestas en el Modelo en Cascada pues ofrece un mejor enfoque ante situaciones que se puedan presentar tanto por el cliente como por el desarrollador de software. El uso de prototipos se centra fundamentalmente en la idea de comprender los requisitos de entrada, proceso o salida que no han sido identificados por el cliente, el cual, como se había dicho con anterioridad solo es capaz de definir un conjunto de objetivos generales para el software. Simultáneamente, ayuda al desarrollador de software a comprender mejor lo que se necesita hacer y puede utilizarse cuando este tiene dudas acerca de la viabilidad de la solución pensada. Es importante destacar que una vez se tenga el prototipo construido es utilizado para refinar los requisitos del software a desarrollar, el cual se va incrementando gradualmente y va evolucionando hasta llegar a lo que se suele llamar “primer sistema”. En este modelo las etapas del ciclo de vida clásico quedan modificadas de la siguiente forma: Análisis de requisitos del sistema, Análisis de requisitos del software, Diseño, desarrollo e implementación del prototipo, Prueba del prototipo, Refinamiento iterativo del prototipo, Refinamiento de las especificaciones del prototipo, Diseño e implementación del sistema final, Explotación (u operación) y mantenimiento.

Aunque este paradigma le guste a los clientes y a los desarrolladores, puede tornarse problemática, pues cuando se le comunica al cliente sobre la necesidad de reconstruir el producto, que tuvo la oportunidad de ver como una versión del software, pero que en realidad trae asociado un sinnúmero de dificultades que atentan contra la calidad y facilidad de mantenimiento, este no entiende la magnitud del problema, y piensa que para lograr un producto final solo se necesita realizar unos pequeños ajustes, cuando en verdad esta gestión de desarrollo frecuentemente se hace muy lenta. Además, se puede presentar el problema de que lo que en un inicio se utilizó por el desarrollador de software como una elección viable y rápida para la construcción del prototipo y que pudo haber sido la selección menos adecuada ahora es una parte integral del sistema.

1.3.1.1.3 Modelo Incremental

El Modelo Incremental (Ver Anexo 3) combina elementos (aplicados repetidamente) del modelo lineal secuencial (aunque corrige la problemática de la linealidad del modelo en cascada) con la filosofía interactiva de construcción de prototipos. Cada secuencia lineal produce un incremento del software. El modelo de proceso incremental, como la construcción de prototipos y otros enfoques evolutivos, es iterativo por naturaleza. Con cada incremento se va logrando una versión del producto y aunque en los primeros incrementos estas versiones del producto final sean incompletas, a diferencia del modelo de construcción de prototipos, proveen al usuario la funcionalidad que precisa, pues se centra en la entrega de un producto operacional con cada incremento. Para lograr esto en cada paso sucesivo agrega al sistema nuevas funcionalidades o requisitos que permiten el refinado a partir de una versión previa. El modelo es útil cuando la definición de los requisitos es ambigua y poco precisa, porque permite el refinamiento, o sea, se pueden ampliar los requisitos y las especificaciones derivadas de la etapa anterior.

Uno de los problemas que puede presentar es detección de requisitos tardíamente, siendo su corrección tan costosa como en el caso de la cascada.

1.3.1.1.4 Modelo Espiral

"El modelo en espiral propuesto originalmente por Boehm, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. En el modelo espiral, el software se desarrolla en una serie de versiones incrementales".(PRESSMAN 2002)

Se divide en regiones de tareas de las cuales existen generalmente entre tres y seis (Ver Anexo 4). Estas se definen de la siguiente forma:

- **Comunicación con el cliente:** las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
- **Planificación:** las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.
- **Análisis de riesgos:** las tareas requeridas para construir una o más representaciones de la aplicación.
- **Construcción y acción:** las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario (por ejemplo: documentación y práctica)
- **Evaluación del cliente:** las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

Cada una de estas regiones está compuesta por un conjunto de tareas que se adaptan a las características del proyecto que se emprende. En todos los casos se aplican actividades de protección como gestión de configuración de software y garantía de calidad de software.

"El primer circuito de la espiral puede producir el desarrollo de una especificación de productos; los pasos siguientes en la espiral se podrían utilizar para desarrollar un prototipo y progresivamente versiones más sofisticadas del software. Cada paso por la región de planificación produce ajustes en el plan del proyecto. El costo y la planificación se ajustan con la realimentación ante la evaluación del cliente".(PRESSMAN 2002)

El modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software, a diferencia del modelo de proceso clásico que termina cuando se entrega el software.

Durante cada ciclo de la espiral, aparece el análisis de riesgos, identificando situaciones que pueden hacer fracasar el proyecto, demorarlo o incrementar su costo. Es por ello que Pressman expresa que este modelo es un enfoque realista del desarrollo de sistemas y de software a gran escala, pues como el software evoluciona, a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante los riesgos en cada uno de los niveles evolutivos. Para la reducción del riesgo aplica el enfoque de la construcción de prototipo y esta construcción se puede llevar a cabo en cualquier etapa del producto.

1.3.1.2 Metodologías de desarrollo de software

"Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc. Habitualmente se utiliza el término "método" para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y/o diseño". (VALENCIA ?)

Piattini define a la metodología de desarrollo de software como "un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software". (PIATTINI 1996).

Si se quiere construir un software de alta calidad, desarrollado en el tiempo planificado, con los costos establecidos y que satisfaga la necesidad de ser elaborado de una forma más acelerada, es necesario enfocarse en trabajar de forma organizada, donde se controle y documente todo lo relacionado con el proyecto en cuestión y puedan eliminarse los riesgos que podrían presentarse durante el desarrollo del mismo, lo cual no podría lograrse sin el empleo de una metodología eficaz, que se adapte a las características propias del software que se esté desarrollando.

Clasificar las metodologías es una tarea bien difícil por la gran cantidad de propuestas y diferencias en el grado de detalle, información disponible y alcance que poseen. A grandes rasgos, considerando su filosofía de desarrollo se pueden agrupar en: metodologías ágiles o ligeras y metodologías pesadas.

1.3.1.2.1 Metodologías ágiles o ligeras

"A principios de la década del '90, surgió un enfoque que fue bastante revolucionario para su momento ya que iba en contra de la creencia de que mediante procesos altamente definidos se iba a lograr obtener software en tiempo, costo y con la requerida calidad. El enfoque fue planteado por primera vez en 1991 y se dio a conocer en la comunidad de ingeniería de software con el mismo nombre que su libro, RAD o Rapid Application Development. RAD consistía en un entorno de desarrollo altamente productivo, en el que participaban grupos pequeños de programadores utilizando herramientas que generaban código en forma automática tomando como entradas sintaxis de alto nivel". (PRESSMAN 2002)

Estas metodologías están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

Entre las metodologías ágiles tenemos a:

- Extreme Programming (XP).
- SCRUM.
- Crystal Clear.
- Feature -Driven Development (FDD).
- Dynamic Systems Development Method (DSDM).
- Adaptive Software Development (ASD).

1.3.1.2.2 Metodologías pesadas

Las metodologías pesadas son aquellas que están guiadas por una fuerte planificación durante todo el proceso de desarrollo, en la cual se establecen estrictamente las actividades involucradas, los roles definidos, los artefactos que se deben producir, las herramientas y notaciones que serán utilizadas así como el modelado y documentación detallada.

"Este esquema tradicional para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso. Sin embargo este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Ante las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir del "buen hacer" de la ingeniería del software, asumiendo el riesgo que ello conlleva." (LETELIER and SÁNCHEZ 2003)

Ejemplo de estas metodologías tenemos a: SW-CMM (Software Capability Maturity Model), RUP (Rational Unified Process) entre otras.

1.3.1.3 Procesos iterativos e incrementales

"El desarrollo de un producto software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas; esto es, deben seleccionarse y ejecutarse de una forma planificada. Es por esto por lo que son miniproyectos". (JACOBSON *et al.* 2000)

El hecho de que las iteraciones sean controladas trae muchos beneficios, pues reduce el costo del riesgo a los valores de un solo incremento, minimiza el riesgo de no sacar al mercado el producto en el calendario previsto, además, acelera el ritmo del esfuerzo de desarrollo en su totalidad y reconoce una realidad que a menudo se ignora: que las necesidades del usuario y sus correspondientes requisitos no pueden definirse completamente al principio, se van refinando en iteraciones sucesivas.

"Cada iteración resultará, al menos, en una nueva construcción, de forma que se añada un incremento al sistema. Sin embargo, se puede crear una secuencia de construcciones en una sola iteración, cada una de ellas añadiendo al sistema un pequeño incremento. Por tanto una iteración añade al sistema un incremento mayor, posiblemente acumulado a lo largo de varias construcciones". (JACOBSON *et al.* 2000).

1.3.1.4 Proceso de desarrollo de software educativo

Para conceptualizar el proceso de desarrollo de software educativo es necesario partir del concepto de proceso de desarrollo de software, del cual existen disímiles definiciones, siendo referenciadas un conjunto de ellas al inicio de la descripción general del objeto de estudio de esta investigación. Una de ellas podría ser la que se encuentra en el libro *El Proceso Unificado de Desarrollo de Software* de Ivar Jacobson, Grady Booch y James Rumbaugh, donde se expresa que el proceso de desarrollo de software en particular lo podríamos definir como el conjunto de actividades necesarias para transformar los requisitos de usuario en un sistema software.

Con el objetivo de conformar la definición que se quiere, a este concepto se le debe agregar que a este conjunto de actividades que comprende una metodología determinada se le deben adicionar un conjunto de actividades pedagógicas, de carácter educativo. Por lo que se puede plantear que un proceso de desarrollo de software educativo no es más que una secuencia de actividades con la introducción de actividades pedagógicas que tiene el objetivo de trasformar los requisitos de los usuarios en un producto de software educativo.

1.3.1.5 Flujos de Trabajo de Software Educativo

La Universidad de las Ciencias Informáticas se caracteriza por tener como una fuerte línea de producción, el desarrollo de productos de software educativo. La alta demanda de productos de este tipo a nivel mundial ha traído consigo que la Universidad adopte una estrategia para garantizar la adecuada ejecución de los proyectos identificados con esta rama pues en estos casos se hace imprescindible una concepción industrial. Ante esta situación se definió un "Sistema Metodológico para la producción de

Software Educativo" que está compuesto por tres áreas de trabajo, de las cuales solo haremos alusión a la relacionada con los Flujos de Trabajo que guiarán el proceso de desarrollo de los mismos.

En el flujo general de trabajo se identificaron 7 procesos (Ver Anexo 12) que son expuestos a continuación:

- **Definición de Proyecto**

  El cliente entrega su solicitud y el guión de contenidos si lo tiene elaborado. En este flujo se traza la estrategia que se seguirá para el desarrollo de proyecto.

- **Gestión de Requisitos y Análisis**

  Se estudia la solicitud del cliente, del guión de contenidos y la documentación entregada.

- **Evaluación Técnica**

  Se realiza un análisis del proyecto para decidir cual será la arquitectura a utilizar.

- **Diseño Gráfico**

  Este flujo de trabajo incluye la concepción y realización del diseño gráfico de acuerdo a lo que se plantea en el guión técnico.

- **Gestión de Medias**

  La gestión de medias encierra la búsqueda o producción de recursos audiovisuales según los requisitos planteados en el guión técnico. Este flujo de trabajo es bien complejo en su realización y tiene varias etapas.

- **Construcción**

  Se realiza la programación para obtener entregables parciales, el cual es sometido a revisiones hasta estar apto para ser entregado al cliente.

- **Gestión de Configuración**

  Se definen las políticas para el desarrollo de los proyecto, además de las herramientas para la gestión de control de versiones, los servidores en los que se encuentran las herramientas y la documentación básica para el uso de estas.

1.3.1.6 Gestión de Procesos

"La Gestión por Procesos es un sistema de trabajo enfocado a perseguir la mejora continua del funcionamiento de las actividades de una organización, mediante: la identificación, la selección, la descripción, la documentación y la mejora de los procesos. Todas las actividades o secuencias de actividades que se desarrollan en el Servicio constituyen un proceso, y como tal hay que gestionarlas". (TOLEDO 2002)

Blaya plantea que "la gestión por procesos determina qué procesos necesitan ser mejorados o rediseñados, establece prioridades y provee de un contexto para iniciar y mantener planes de mejora que permitan alcanzar objetivos establecidos. Hace posible la comprensión del modo en que están configurados los procesos de negocio, de sus fortalezas y debilidades".(BLAYA 2006)

"La gestión por procesos (Business Process Management) es una forma de organización diferente de la clásica organización funcional, y en el que prima la visión del cliente sobre las actividades de la organización. Los procesos así definidos son gestionados de modo estructurado y sobre su mejora se basa la de la propia organización.

La gestión de procesos aporta una visión y unas herramientas con las que se puede mejorar y rediseñar el flujo de trabajo para hacerlo más eficiente y adaptado a las necesidades de los clientes. No hay que olvidar que los procesos lo realizan personas y los productos los reciben personas, y por tanto, hay que tener en cuenta en todo momento las relaciones entre proveedores y clientes". (TOLEDO 2002)

"La gestión por procesos encierra cuatro factores que hacen posible pasar de grupo de emprendedores a empresa:

- **Repetibilidad de resultados**. Al conseguir que la calidad del resultado sea consecuencia del proceso, producir aplicando el mismo proceso garantiza la homogeneidad de los resultados.
- **Escalabilidad**. Es una consecuencia de la repetibilidad. No sólo un equipo consigue resultados homogéneos en todos los proyectos, sino que los obtienen todos los equipos.
- **Mejora continua**. Al aplicar meta-procesos que trabajan sobre los propios procesos de producción, midiendo y analizando los resultados se obtienen los criterios de gestión necesarios

para aplicar medidas que mejoran de forma continua la eficiencia y calidad de los procesos base, y por tanto de los resultados.

- Un **know-how propio**, consiguiendo finalmente una empresa que sabe hacer, porque su modelo de procesos termina conteniendo un activo valioso de la organización: la clave para hacer las cosas bien, con eficiencia y de forma homogénea. (PALACIO 2005)

1.3.1.7 Administración de Proyectos

"*La administración de proyectos* es la forma de planear, organizar, dirigir y controlar una serie de actividades realizadas por un grupo de personas que tienen un objetivo especifico; el cual puede ser (crear, diseñar, elaborar, mejorar, analizar, etc.) un problema o cosa". (VELA 1997)

"Es el proceso de combinar sistemas, técnicas y personas para completar un proyecto dentro de las metas establecidas de tiempo, presupuesto y calidad", es decir, es administrar los procesos dentro de la organización para obtener los resultados deseados. (ESCOBEDO 2007)

A partir de lo planteado en los conceptos anteriores es oportuno agregar que la administración de proyectos es indispensable para el desarrollo correcto de los mismos, lo cual es fundamental para obtener un producto con la calidad que se requiere.

El proceso de la administración de proyectos cuenta con los siguientes elementos: Planeación, Organización, Dirección y Control, con estos podremos diseñarla y aplicarla. Dichas actividades son realizadas para lograr un objetivo a corto plazo.

Teniendo en cuenta recursos como tiempo, material, capital, recursos humanos y tecnología dicha actividad es llevada a cabo por un conjunto de administradores que actúan como agentes unificadores para proyectos particulares.

Importancia de la Administración de Proyectos

Mediante la administración de proyectos se puede medir el grado de calidad con que se realiza un proyecto. Con una inadecuada administración, una empresa no debe contar con buenos sistemas de información basados en computadoras y no debe existir un buen aprovechamiento de los recursos, ya

sean humanos, financieros o tecnológicos. Una buena administración permitirá a la empresa una correcta toma de decisiones apoyada por un sistema de información con calidad.

Esta es una de las mejores herramientas para desarrollar proyectos de cualquier tipo si se aplica de forma estricta su metodología de trabajo, sin importar el tamaño de la empresa, ni su carácter (pública o privada).

El incremento del fluido de operaciones en una organización ha provocado que los métodos de administrativos convencionales no sean los más adecuados, razón por la cual la administración de proyectos adquiere una gran importancia pues ofrece nuevas alternativas de organización, además de ser muy apropiada para aprovechar de mejor manera los recursos críticos cuando están limitados en cantidad y/o tiempo de disponibilidad. También ayuda a realizar acciones concisas y efectivas para obtener el máximo beneficio.

Beneficios de la Administración de Proyectos

La aplicación de la Administración de proyecto reporta grandes beneficios para las empresas que la desarrollan correctamente. Algunos de estos beneficios están dirigidos a la toma de mejores decisiones de inversión, a la minimización de los costos de un proyecto de inversión, a la propagación y transformación del negocio, a la realización de aquellos proyectos cuya inversión sea viable, a la maximización del retorno de la inversión de los accionistas y por último al trabajar en función de lograr eficiencia del tiempo de realización de un proyecto de inversión

1.3.1.8 Gestión Integrada de Proyecto

“He visto docenas de empresas, buenas y malas, y he observado a muchas administradores de proceso de datos, también buenos y malos. Frecuentemente he visto con horror cómo estos administradores luchaban inútilmente contra proyectos de pesadilla, sufriendo por fechas límite imposibles de cumplir, o que entregaban sistemas que decepcionaban a sus usuarios y que devoraban ingentes cantidades de horas de mantenimiento”. (PAGE-JONES 1985)

La frase anterior es expresada por Meiler Page-Jones en el prefacio de su libro sobre gestión de proyectos, en la que reconoce un problema que se presenta constantemente en los proyectos de

software y que trae consecuencias negativas para los mismos: la débil gestión que se realiza cuando se decide construir un producto; y es que el desarrollo de un software de gran magnitud es un proceso complejo, en el cual están involucradas muchas personas, que tienen diferentes responsabilidades dentro del mismo, así que, si se quiere que el proyecto salga a flote, en el tiempo planificado y con los costos establecidos pero que además el producto resultante sea de alta calidad , es necesario que sea organizado de alguna forma, por tanto es imprescindible que sea gestionado, pues "la gestión de proyectos implica la planificación, supervisión y control del personal, del proceso y de los eventos que ocurren mientras evoluciona el software desde la fase preliminar a la implementación operacional".(PRESSMAN 2002) Dichas actividades varían en función de la persona que esté gestionando algo, depende del rol que juegue dentro del equipo de software y las tareas que deban asumir dentro del mismo.

Para que se realice una adecuada gestión de proyecto es fundamental que se involucren lo que se conoce como las cuatro P's: personas, producto, proceso y proyecto.

"El gestor que se olvida de que el trabajo de ingeniería del software es un esfuerzo humano intenso, nunca tendrá éxito en la gestión de proyectos. Un gestor que no fomenta una minuciosa comunicación con el cliente al principio de la evolución del proyecto se arriesga a construir una elegante solución para un problema equivocado. El administrador que presta poca atención al proceso corre el riesgo de arrojar métodos técnicos y herramientas eficaces al vacío. El gestor que emprende un proyecto sin un plan sólido arriesga el éxito del producto". (PRESSMAN 2002)

En la anterior cita de Pressman se puede apreciar cuán importante es que las 4P's, estén incluidas en la gestión de cualquier proyecto software, la relación de dependencia que existe entre ellas, cuánto podría afectar la elaboración de un producto si una de sus partes no se gestiona de forma adecuada. Es por ello que insiste en que el personal involucrado debe estar organizado para que sea efectivo el desarrollo del trabajo del software, que para comprender el alcance del producto y los requisitos es muy necesaria la comunicación con el cliente, pero que para construir el producto que se desea hay que tener bien definido cuál debería ser el proceso a seleccionar, que se ajuste a las características del mismo, y que además sea adecuado para el personal conque se cuenta. Además expresa que dicho proyecto debe planificarse estimando el esfuerzo y el tiempo que se requiere para que las tareas se cumplan de forma eficiente, y que para esto se debe definir los productos del trabajo y se deben establecer puntos de

control de calidad y mecanismos que permitan controlar y supervisar el trabajo concretado en la planificación.

1.3.2 Descripción actual del dominio del problema

La Universidad de las Ciencias Informáticas ha decidido adoptar una estrategia para la producción de los productos de software educativo y multimedia basándose en las experiencias de proyectos anteriores, que se concreta en un "Sistema Metodológico para el desarrollo de Software Educativo". Dicha estrategia está conformada por un conjunto de elementos que son integrados de forma armónica, de los cuales podemos mencionar la estructura organizativa, los procedimientos, la formación y la documentación técnica que es generada.

Este sistema metodológico está compuesto por tres áreas de trabajo:

1. Flujo de producción
2. Trabajo Técnico-Legal
3. Control de Proyectos

Para describir cómo se comporta el flujo productivo se brinda una representación del flujo general de trabajo, compuesto por siete procesos, que han sido referenciados en el epígrafe 1.3.1.5. Este flujo se definió siguiendo los principios de la metodología RUP, por lo que al igual que este ha sido concebido bajo los principios: centrado en la arquitectura, guiado por casos de uso, iterativo e incremental. En el flujo contenido en el Sistema Metodológico se identifican los puntos de interacción entre áreas y se definen los flujos derivados de estas relaciones. Además se definen en cada uno de los procesos los documentos que son generados así como los documentos que han sido definidos y herramientas que deberán ser utilizadas para la ejecución del flujo general.

La Estrategia Integral para el Trabajo Técnico – Legal que fue adoptada comprende acciones para el trabajo en las áreas de Derechos de Autor, Propiedad Industrial, Confidencialidad y seguridad de la información totalmente integrada al flujo productivo general.

Para el Control de Proyectos se decidió encaminar el trabajo en función de dos objetivos específicos:

1. Auditorias

2. Control de avance

1.3.3 Situación problemática

La creación de la Universidad de las Ciencias Informáticas trajo asociado un factor determinante para el desarrollo profesional y crecimiento económico del país: el establecimiento del vínculo entre la docencia y la producción como modelo de formación estudiantil, relación que establecería que los estudiantes y profesores pertenecientes a la misma estarían vinculados desde los primeros años a la producción, bajo el compromiso de dedicar todo su empeño en función de lograr resultados que pusieran en alto el nombre de esta gran universidad-empresa en la industria del software.

Para lograr la relación entre el estudio y el trabajo productivo se decidió que cada facultad se identificara con un perfil para el desarrollo del software y se crearan en las mismas un conjunto de proyectos productivos, a los cuales se irían asociando gradualmente un significativo número de estudiantes hasta lograr un 100% de incorporación. Estos perfiles han ido sufriendo transformaciones con el decursar de los años con respecto a su concepción inicial, su organización y su asignación respectiva a determinada facultad.

Uno de los perfiles que fue definido en aquel entonces, que se mantiene hasta hoy y que además determina una fuerte línea en la producción en la Universidad, es el de Software Educativo, de cuya línea se derivan un variado número de proyectos relacionados con diferentes entidades en el ámbito nacional e internacional.

Este perfil ha sido asignado a tres facultades diferentes en los cinco años que lleva de creada la institución, lo cual no ha permitido que se sienten correctamente las bases para adquirir la experiencia que se necesita para el desarrollo del software educativo, siendo este uno de los campos en los que se incursiona en el mundo en la actualidad por la problemática que conlleva su desarrollo.

La dificultad principal para la elaboración de productos educativos en la universidad, se debe fundamentalmente a la existencia de un reducido número de metodologías que permitan la guía de los procesos productivos relacionados con esta área de desarrollo, pues aunque existe una gran variedad de procesos, técnicas y metodologías para el desarrollo del software de forma general, entre sus

características no se encuentran contemplados los aspectos pedagógico-didácticos, y esta es una necesidad que además de ser definitoria para la construcción de productos educativos hace que todas las etapas que contempla el proceso de desarrollo que sea seleccionado para guiar a sus desarrolladores, tenga significativas transformaciones.

La reducida experiencia del personal y las condiciones cambiantes por las cuales ha atravesado la organización para el desarrollo de los productos de software educativo, ha traído consigo un conjunto de deficiencias que influyen negativamente en la obtención de productos con la calidad requerida, en el tiempo planificado, con los costos acordados y la completa satisfacción del cliente, para el cual trabajamos.

Los problemas anteriormente mencionados se deben en gran medida a la no utilización y seguimiento de metodologías y estándares o normas de calidad establecidos a nivel internacional para el desarrollo de productos software. La inadecuada planificación y estimación de los tiempos de desarrollo y costos de los proyectos, la ausencia de métricas para evaluar la calidad de los productos, la inadecuada gestión de riesgos, son problema potenciales, que al no ser identificados pueden provocar resultados desastrosos en la producción, la deficiente delegación de responsabilidades a los miembros del proyecto, la insuficiente asignación de roles a las personas involucradas en los mismos.

De casi todos los proyectos se podría afirmar que sus miembros aún trabajan bajo la concepción de "programación heroica", bajo un enfoque de producción artesanal, cuando lo que se necesita en este tipo de institución que genera una alta demanda de productos de software educativo, es el trabajo dirigido desde una perspectiva industrial.

Hoy día existe un Sistema Metodológico para guiar el proceso de producción en la universidad, creado sobre las bases de las dificultades que se fueron presentando en los proyectos anteriores, cuyos resultados arrojan que los problemas más significativos que fueron detectados estuvieron inclinados a un deficiente proceso de gestión de requisitos, a dificultades en la integración de las diferentes áreas que participan en la producción, tales como la gestión de configuración y salvas, no existencia de sistemas de chequeo y control a los proyectos por parte de las entidades involucradas, no existencia de mecanismos de control relacionados con la seguridad de la información y de los procesos de protección a la propiedad intelectual, y falta de definición de los momentos de aprobación de componentes y responsables de esta tarea en cada momento del desarrollo del software educativo.

Es necesario reflejar que a partir de entrevistas (Ver Anexo 11) y encuestas (Ver Anexo 10) realizadas a líderes de proyecto se puede afirmar que en la actualidad existe un casi total desconocimiento de esta metodología, infiriendo por tanto que en estos proyectos se continúa trabajando sin aplicar metodología alguna. Además, esta metodología debe estar sujeta a cambios pues la misma no contempla un conjunto de propiedades que son necesarias y definitorias para desarrollar un software.

1.4 Análisis de otras soluciones existentes

1.4.1 Metodologías ágiles o ligeras

Jim Highsmith, inspirado en su creación y en general en las metodologías ágiles decía:

“En ambientes complejos, el seguir un plan al pie de la letra produce el producto que pretendíamos, pero no el producto que necesitamos”. (HERNÁN)

En los epígrafes siguientes se pretende reafirmar lo expresado por el padre de la metodología Adaptive Software Development (ASD). Se darán un grupo importante de elementos que permitirán adentrar al lector en el tema de las metodologías ágiles y aportará conocimientos importantes para un mejor entendimiento de los resultados de esta investigación.

1.4.1.1 Extreme Programming (XP)

“Muchas de las prácticas propuestas contribuyen a maximizar la comunicación entre las personas, permitiendo de esa forma una mayor transferencia de conocimiento entre los desarrolladores y con el cliente, quien también es parte del equipo”. (HERNÁN)

Esta metodología (Ver Anexo 5) fue desarrollada por Kent Beck. Está centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP está guiada por una rápida programación y se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, en la reutilización de código, en la realización de pruebas a los principales procesos con el objetivo de tratar de obtener los

posible errores futuros (conocido como pruebas unitarias), en la simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Impone además un alto nivel de disciplina entre los programadores, lo cual permite mantener un mínimo nivel de documentación que a su vez se traduce en una gran velocidad de desarrollo y propone que el trabajo de los programadores sea en pares, de forma tal que uno realice lo que el otro no hace en ese instante. XP se define como especialmente adecuada para proyectos de corto plazo con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. Está formada por cuatro partes fundamentales que encierran sus características esenciales, las cuales son: historia de usuarios, roles, procesos y prácticas.

1.4.1.2 SCRUM

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, es un término utilizado en el juego de rugby que llevado a la ingeniería significa que el equipo se agrupará para llegar con el apoyo de cada uno de los miembros del proyecto a obtener un producto con calidad que es el objetivo fundamental. “Scrum define un proceso empírico, iterativo e incremental de desarrollo que intenta obtener ventajas respecto a los procesos definidos (cascada, espiral, prototipos, etc.) mediante la aceptación de la naturaleza caótica del desarrollo de software, y la utilización de prácticas tendientes a manejar la impredictibilidad y el riesgo a niveles aceptables”.(HERNÁN ?)

“Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración”. (LETELIER and SÁNCHEZ 2003)

SCRUM (Ver Anexo 6) define tres roles principales, el propietario del producto (Product Owner), el equipo (Team) y el Scrum Master. El propietario del producto tiene entre sus principales responsabilidades la funcionalidad del sistema, el retorno de la inversión del proyecto y el lanzamiento del proyecto, este rol representa a todos los interesados finales del producto. El rol de equipo se

encarga de transformar el Backlog de la iteración en un incremento de la funcionalidad del software. Por su parte el Scrum Master se encarga de todo lo referente al proceso, además de garantizar el cumplimiento de las responsabilidades según los roles.

Esta metodología dentro de los artefactos que propone se destacan dos fundamentales: el Product Backlog y el Sprint Backlog, en el primero quedan listados todos los requerimientos del proyecto y el encargado de su disponibilidad, priorización y contenido es el Product Owner, este artefacto es dinámico por lo que va avanzando conjuntamente con el producto. En el caso del Sprint Backlog es donde se define por el equipo las tareas para convertir el Product Backlog en una fuente de funcionalidades del software. Solo el equipo puede modificar el Sprint Backlog.

"La intención de Scrum es la de maximizar la realimentación sobre el desarrollo pudiendo corregir problemas y mitigar riesgos de forma temprana. Su uso se está extendiendo cada vez más dentro de la comunidad de Metodologías Ágiles, siendo combinado con otras – como XP – para completar sus carencias. Cabe mencionar que Scrum no propone el uso de ninguna práctica de desarrollo en particular; sin embargo, es habitual emplearlo como un framework ágil de administración de proyectos que puede ser combinado con cualquiera de las metodologías mencionadas". (HERNÁN ?)

1.4.1.3 Crystal Clear

Se trata de un conjunto de metodologías para el desarrollo de software, caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn.

"Crystal "Clear" es la encarnación más ágil de la serie y de la que más documentación se dispone. La misma se define con mucho énfasis en la comunicación, y de forma muy liviana en relación a los entregables. Crystal maneja iteraciones cortas con feedback frecuente por parte de los usuarios/clientes, minimizando de esta forma la necesidad de productos intermedios. Otra de las cuestiones planteadas es la necesidad de disponer de un usuario real aunque sea de forma part time para realizar validaciones sobre la Interfase del Usuario y para participar en la definición de los requerimientos funcionales y no funcionales del software". (HERNÁN ?)

“El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros)”. (HERNÁN ?)

1.4.1.4 Dynamic Systems Development Method (DSDM)

Define el marco para desarrollar un proceso de producción de software. Surge en 1994 con el objetivo de crear una metodología que fuera independiente de las herramientas y que pudiera ser utilizado en proyectos de tipo RAD unificada. La estructura del método fue guiada por nueve principios:

1. El involucramiento del usuario es imperativo.
2. Los equipos de DSDM deben tener el poder de tomar decisiones.
3. El foco está puesto en la entrega frecuente de productos.
4. La conformidad con los propósitos del negocio es el criterio esencial para la aceptación de los entregables
5. El desarrollo iterativo e incremental es necesario para converger hacia una correcta solución del negocio.
6. Todos los cambios durante el desarrollo son reversibles.
7. Los requerimientos están especificados a un alto nivel.
8. El testing es integrado a través del ciclo de vida.
9. Un enfoque colaborativo y cooperativo entre todos los interesados es esencial.

Entre las principales características de esta metodología se destacan que: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio de factibilidad, estudio del negocio, modelado funcional, diseño y construcción, implantación. (Ver Anexo 7)

"El estudio de factibilidad es una pequeña fase que propone DSDM para determinar si la metodología se ajusta al proyecto en cuestión. Durante el estudio del negocio se involucra al cliente de forma temprana, para tratar de entender la operatoria que el sistema deberá automatizar. Este estudio sienta las bases para iniciar el desarrollo, definiendo las features de alto nivel que deberá contener el software. Posteriormente, se inician las iteraciones durante las cuales: se bajará a detalle los features identificados anteriormente, se realizará el diseño de los mismos, se construirán los componentes de software, y se implantará el sistema en producción previa aceptación del cliente". (HERNÁN ?)

En cuanto a los roles, DSDM especifica tres fundamentales:

Visionario: Es el encargado de asegurar que se satisfacen las necesidades del negocio.

Usuario embajador: Este brinda el conocimiento del negocio y define los requerimientos del software.

Coordinador técnico: Es la persona encargada de mantener la arquitectura, verificar la existencia de los componentes construidos respecto a esta y al cumplimiento de los estándares técnicos.

Es importante resaltar que esta metodología ha tenido un refinamiento continuo y es una de las más aceptadas en el mercado.

1.4.1.5 Adaptive Software Development (ASD)

Su impulsor es Jim Highsmith. ASD consiste en un cambio de filosofía en las organizaciones pasando de la transición del modelo Comando-Control al modelo Liderazgo-Colaboración. Se caracteriza por ser iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. Su ciclo de vida está compuesto por tres fases importantes: especulación, colaboración y aprendizaje (Ver Anexo 8). En la primera especulación se inicia el proyecto y se planifican las características del software, se prefija el camino por donde va a ser guiado el desarrollo del software basado en la teoría de los Sistemas Adptativos Complejos; por su parte en la fase de colaboración se construye la funcionalidad que se especifica en la fase anteriormente mencionada y finalmente en aprendizaje se revisa su calidad y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo. Para su revisión se apoyan en las siguientes categorías:

- Calidad del resultado desde la visión del cliente

Se analiza una determinada parte de la aplicación y se recogen los cambios del cliente.

- Calidad del resultado desde la perspectiva técnica

Se realizan las revisiones al código y al diseño y de ahí se obtienen los errores no con el objetivo de culpar a alguien sino de corregirlos y de tenerlos en cuenta para aprender a no volver a cometerlos.

- El funcionamiento del equipo de desarrollo y las prácticas que este utiliza.

Se necesita la interacción entre las partes para de esta forma desechar los aspectos negativos del proceso y lograr un desarrollo correcto del mismo.

- El status del proyecto

Son las revisiones que se llevan a cabo para especificar cual es el estado del proyecto respecto a lo planificado.

Una vez terminado un ciclo se le hace entrega al cliente. La revisión es necesaria para la corrección de errores y para evitar cometerlos nuevamente en posteriores ocasiones, después se procese a volver a iniciar el ciclo de desarrollo.

1.4.1.6 Feature -Driven Development (FDD)

Se propuso por Jeff De Luca y Peter Coad. Esta metodología fue desarrollada alrededor del año 1998.

“FDD se estructura alrededor de la definición de features que representan la funcionalidad que debe contener el sistema, y tienen un alcance lo suficientemente corto como para ser implementadas en un par de semanas” (HERNÁN ?). FDD posee también una jerarquía de *features*. Una de las ventajas de centrarse en las *features* del software es el poder formar un vocabulario común que fomente que los desarrolladores tengan un diálogo fluido con los clientes, desarrollando entre ambos un modelo común del negocio.

Su ciclo de vida está compuesto por cinco procesos, dos de los cuales se realizan tantas veces como iteraciones se planifiquen en el desarrollo. Estos procesos son: Desarrollar un Modelo Global, Construir una Lista de Features, Planificar por Feature, Diseñar por Feature y Construir por Feature (Ver Anexo 9).

Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software.

El proceso de Desarrollar un Modelo Global propone que se realice paralelo a la construcción de la arquitectura, en el desarrollo de este modelo hay una fuerte cooperación de programadores y expertos con el fin de lograr un entendimiento del negocio, una visión global de la aplicación a construir, una primera idea de las funcionalidades y las cuestiones no funcionales del software. Para esto se realizaran diagramas de paquetes, un documento semejante al de Visión y otro documento que es donde se recogen las opciones de modelado realizadas durante este proceso.

En el caso del proceso de Construir una Lista de Features se retoma la idea del proceso anterior de las funcionalidades, para partiendo de estas refinar la nuevas surgidas. Se agrupan siguiendo una jerarquía y se especifica las prioridades según las necesidades del cliente y de acuerdo a estas necesidades se decidirá las que serán implementadas.

Por su parte el proceso de Planificar por Feature utiliza las listas priorizadas especificada anteriormente y basándose en ellas se realiza la planificación de tiempos de las iteraciones. En este proceso interviene el líder del proyecto, el de desarrollo y el de los programadores. Además se delimitan los hitos de terminación de la iteración.

Los procesos de Diseñar por Feature y Construir por Feature son muy parecidos, en la parte donde se construye la aplicación. En el primer proceso el programador líder especifica cuáles van a ser las clases a implementar, mediante la utilización de diagrama de secuencias de UML se verifica si el diseño puede ser realizado, para el segundo proceso se procede a implementar las clases especificadas anteriormente, mas adelante se realizan las pruebas y cuando esta actividad haya terminado se hace entrega al responsable de configuración.

“En conclusión, encontramos en FDD un proceso ágil orientado a la funcionalidad del software por sobre las tareas, sobre las cuales da guías mínimas. El proceso sugiere organizar bloques de *features* a ser construidos en forma incremental mediante iteraciones de dos semanas; provee estrategias de planeamiento para el líder de proyecto; fomenta la colaboración mediante la creación de equipos dirigidos por un programador jefe”. (HERNÁN ?)

1.4.2 Metodologías pesadas o no ágiles

1.4.2.1 Proceso Unificado de Desarrollo de Software (RUP en sus siglas en inglés correspondientes a Rational Unified Process)

"El Proceso Unificado está equilibrado por ser el producto final de tres décadas de desarrollo y uso práctico. Su desarrollo como producto sigue un camino desde el *Proceso Objectory* (primera publicación en 1987) pasando por el *Proceso Objectory de Rational* (publicado en 1997) hasta el *Proceso Unificado de Rational* (publicado en 1998). En este camino de desarrollo ha tenido la influencia mayoritaria de dos grandes métodos: el *Método de Ericsson* y el *Método de Rational*. (JACOBSON *et al.* 2000)

"El *Proceso Unificado de Rational* (RUP), es un proceso de ingeniería de software planteado por Kruchten (1996) cuyo objetivo es producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de una planificación y presupuesto establecido. Cubre el ciclo de vida y desarrollo de software". (DÍAZ-ANTÓN *et al.* 2004)

"RUP toma en cuenta las mejores prácticas en el modelo de desarrollo de software en particular las siguientes:

1. Desarrollo de software en forma iterativa.
2. Manejo de requerimientos.
3. Utiliza arquitectura basada en componentes.
4. Modela el software visualmente (modela con UML).
5. Verifica la calidad del software.
6. Controla los cambios. (JACOBSON *et al.* 2000)

El Proceso Unificado es un proceso de desarrollo de software cuyo ciclo de vida se caracteriza por:

- Dirigido por casos de uso
- Centrado en arquitectura
- Iterativo e incremental

El Proceso Unificado de Desarrollo de Software consta de cuatro fases o etapas: (KRUCHTEN 2004)

1. **Comienzo o Inicio:** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

2. **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

3. **Construcción:** Se obtiene un producto listo pata su utilización que está documentado y tiene un manual de usuario. Se obtienen 1 o varios release del producto que han pasado las pruebas. Se ponen estos release a consideración de un subconjunto de usuarios.

4. **Transición:** El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Además de las fases, el ciclo de vida de esta metodología contiene flujos de trabajos, los cuales se dividen en Flujos de trabajo de desarrollo y Flujos de trabajo de soporte. Ver Figura 1.2.

Flujos de trabajo de desarrollo:

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.

Flujos de trabajo de soporte:

- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, entre otros
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

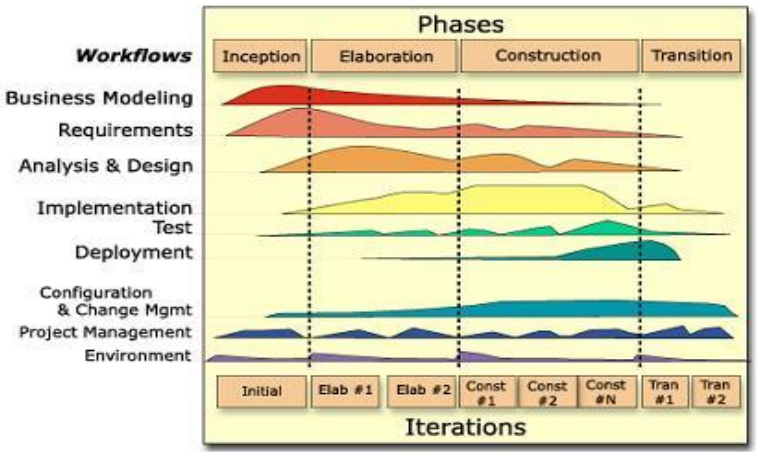


Figura 2 Fases e Iteraciones de la Metodología RUP

En cada una de las iteraciones se obtendrá una versión del producto entregable al cliente.

1.4.2.2 Metodología de desarrollo de software educativo bajo un enfoque de calidad sistémica

Esta propuesta de metodología esta dirigida a la realización de software educativo, es una extensión y adaptación del Proceso Unificado de Desarrollo de Software (RUP), es decir, esta metodología es el producto de la introducción de elementos educativos a RUP y con las mejores prácticas de diseño instruccional, desarrollada bajo el MOdelo Sistémico de CAlidad (MOSCA), modelo propuesto por un colectivo de autores de la Universidad de Bolivia enriquecido por parámetros educativos. Se pretende con ella crear un software educativo con calidad desde su comienzo, que cumpla con características tan necesarias para un material educativo como lo son la usabilidad, funcionalidad y fiabilidad.

A continuación se muestran las actividades que se agregan a RUP, por cada fase.

*"**Fase de Inicio***

MODELO RUP	**ACTIVIDADES AGREGADAS A RUP**
Un documento con la visión del proyecto. Un plan del proyecto que muestre las fases y las iteraciones. Un caso de negocio inicial el cual incluye: contexto del negocio, criterios de éxito y planificación financiera. El modelo de casos de uso con una lista de todos los casos de uso y los actores que puedan ser identificados. Un glosario inicial del proyecto. Un estudio inicial de riesgos.	Un análisis de las necesidades educativas y del entorno educativo. Un estudio sobre las teorías de aprendizaje y diseño instruccional que definen el formato del programa. Una lista de requerimientos pedagógicos relacionados con el contenido y la población estudiantil a la que va dirigida el programa. Revisión de los objetivos y contenidos del material educativo del programa. Establecer los límites de las áreas

Una lista de los requerimientos y restricciones principales del sistema a desarrollar. Estándares para el prototipo inicial. Un mapa de navegación. Una lista inicial de riesgos y su evaluación. Una lista de requisitos funcionales y no funcionales. Un prototipo inicial.	educativas que se van a desarrollar. Establecer un diseño instruccional para el proyecto multimedia, incluyendo los instrumentos de evaluación del usuario sobre lo aprendido. Realizar un estudio sobre las pautas de diseño de interfaz adecuadas a la población estudiantil a la que va dirigida el programa. Establecer los criterios de evaluación del software educativo basados en las características de funcionalidad, usabilidad y fiabilidad (MOSCA).

Tabla 1 Actividades correspondientes a la fase de inicio

Fase de Elaboración

MODELO RUP	**ACTIVIDADES AGREGADAS A RUP**
Actualización del plan de iteración. Generar una lista revisada de riesgos. Realizar la arquitectura del software. Revisar los requerimientos complementarios. Construir un tipo de prototipo de interfaz del usuario. Actualizar el plan de proyecto y	Refinar los modelos instruccionales que se utilizan en el proyecto. Refinar los requerimientos de diseño gráfico y aspectos comunicacionales en base a las pautas pedagógicas establecidas.

elaborar el plan de iteración.	

Tabla 2 Actividades correspondientes a la fase de elaboración

Fase de Construcción

MODELO RUP	**ACTIVIDADES AGREGADAS A RUP**
Actualizar el plan de iteración. Revisar la lista de riesgos. Gerenciar los recursos (herramientas, base de datos). Completar el desarrollo de los componentes (prototipo funcional). Probar los componentes contra los criterios de evaluación definidos. Actualizar el plan de proyecto.	Probar el diseño instruccional, comunicacional y gráfico, contra los criterios de evaluación previamente establecidos.

Tabla 3 Actividades correspondientes a la fase de construcción

Fase de Transición

MODELO RUP	**ACTIVIDADES AGREGADAS A RUP**
Realizar la evaluación del usuario. Realizar los ajustes necesarios. Realizar un ajuste de gastos.	Realizar la evaluación del producto por parte del docente y del estudiante objeto del programa educativo, utilizando MOSCA.

Tabla 4 Actividades correspondientes a la fase de transición" **(DÍAZ-ANTÓN *et al.* 2004)**

1.4.2.3 Metodología Extendida para la creación de Software Educativo desde una visión Integradora

"*Las metodologías propias de la ingeniería del software no cautelan aspectos pedagógicos- didácticos del producto software educativo a desarrollar".* (CATALDI 2000)

La propuesta surge debido a la necesidad de crear una metodología que contemple los aspectos de naturaleza pedagógica que no son tenidos en cuenta en las metodologías convencionales para el desarrollo de software.

Para la solución de la problemática determinaron realizar una extensión de una metodología a partir de una redefinición de la matriz de actividades correspondiente al ciclo de vida del software seleccionado.

Para arribar a una solución realizaron un estudio de algunos de los ciclos de vida más utilizados en el marco de las metodologías de ingeniería del software, entre los cuales estaban el modelo de cascada, el modelo incremental o de refinamientos sucesivos, el prototipado evolutivo, el modelo en Espiral de Boehm (Boehm, 1988) y los modelos orientados al objeto. (PIATTINI 1996).

Después de un análisis exhaustivo de cada uno de ellos determinaron que el ciclo a extender que tenía un mayor acercamiento a una solución que contemplara los aspectos pedagógicos-didácticos era el modelo prototipado evolutivo.

A continuación se describen cada una de las etapas del ciclo de vida elegido que forman parte de la matriz de actividades:

"**Factibilidad (FAC):** En esta etapa se define el producto software y se determina su factibilidad en el ciclo de vida desde la perspectiva de la relación costo –beneficio, como así las ventajas y desventajas respecto de otros productos.

Requisitos del sistema (RES): En esta etapa se deben definir las funcionalidades requeridas para el desarrollo del sistema (o programa), las interfaces y el tipo de diseño.

Especificación de requisitos del prototipo (REP): Consiste en especificar las funciones requeridas, las interfaces y el rendimiento para el prototipo. Aquí se considerarán incrementos en porcentajes de la funcionalidad total del sistema.

Diseño del prototipo (DPR): Es poner en ejecución del plan del prototipo, ya que una vez fijadas las restricciones con el usuario, hay que mostrar el mismo funcionando, aunque sean sólo algunas funcionalidades restringidas. Aquí, hay que hacer un análisis de cómo se va a trabajar, qué módulos se van a hacer, con qué lógica y qué funciones se van a usar.

Diseño detallado del prototipo (DDP): Esta etapa es una especificación verificada de la estructura de control, la estructura de los datos, las relaciones de interfaces, el tamaño, los algoritmos básicos y las suposiciones de cada componente del programa. En esta etapa no sólo se definen y sino que se documentan los algoritmos que llevarán a cabo la función a realizar por cada uno de los módulos.

El diseño de software, es un proceso que se centra en cuatro atributos distintos del programa:

La estructura de datos, la arquitectura del software, el detalle procedimental y la caracterización de la interface. En este proceso deben traducirse los requisitos a una representación del software que pueda ser establecida de forma que se obtenga la calidad requerida antes de que comience la codificación.

Desarrollo del prototipo (codificación) (DEP): Consiste en realizar la codificación o diseño detallado, en forma legible para la máquina".

Implementación y prueba del prototipo (IPP): Consiste en lograr un funcionamiento adecuado del producto software en el sistema informático, funcionando operacionalmente, incluyendo objetivos tales como la conversión del programa y datos (si la hubiere), la instalación y el entrenamiento. La prueba debe asegurar que se han probado toas las sentencias del mismo, y que en las funciones externas se han realizado pruebas que aseguren que la entrada definida produce los resultados que se esperan realmente.

Refinamiento iterativo de las especificaciones del prototipo (RIT): Es un aumento de la funcionalidad del sistema, para luego volver REP a fin de aumentar la funcionalidad del prototipo o continuar, si se logró el objetivo y alcance deseados.

Diseño del sistema final (DSF): Consiste en ajustar las restricciones o condiciones finales e integrar los últimos módulos.

Implementación del sistema final (ISF): Es el sistema de informático funcionando operativamente, incluyendo tales objetivos como conversión del programa y datos, (si la hubiere), la instalación y La capacitación del personal.

Operación y mantenimiento (OPM): Es la puesta en funcionamiento del sistema informático, objetivo que se repite para cada actualización.

Retiro (RET): Es una transición adecuada de las funciones realizadas para el producto y sus sucesores. (CATALDI 2000)

Luego, se definieron las actividades a desarrollar en cada una de las fases anteriormente expuestas y los procesos asociados a cada una de ellas.

Esta metodología hace alusión a que fue necesario la incorporación de nuevos procesos que cautelaran las necesidades pedagógico-didácticas. Estos procesos fueron: proceso de identificación de la necesidad educativa, proceso de análisis de los requisitos educativos, proceso de evaluación de los prototipos de software, proceso de evaluación interna y externa del software, proceso de evaluación contextualizada y proceso de documentación didáctica.

Además de esto hubo que definir nuevas actividades que cautelasen cuestiones pedagógico-didácticas en los procesos ya existentes para el desarrollo del software. Por su envergadura no será posible tratarlas en este documento, por tanto se recomienda el estudio de esta metodología a profundidad a fin de conocer todos los aspectos tratados en la misma.

1.5 Conclusiones parciales

"Para desarrollar un proyecto de software es necesario establecer un enfoque disciplinado y sistemático. Las metodologías de desarrollo influyen directamente en el proceso de construcción y se elaboran a partir del marco definido por uno o más ciclos de vida". (PIATTINI 1996)

La necesidad de modelos, metodologías y procesos que guíen el desarrollo de productos software, ha conducido a la elaboración del capítulo que acaba de ser presentado, como referencia fundamental para el análisis de posibles soluciones que contribuyan a la elaboración de un conjunto de principios para el desarrollo de productos de software educativo en la Universidad de las Ciencias Informáticas.

El estudio del marco conceptual, que rodea al objeto de estudio que rige esta investigación, ha permitido que se tenga un conocimiento general de la situación actual bajo la cual se desarrolla el software educativo en esta institución, la cual ha arrojado que existe una pobre o casi nula aplicación de las

soluciones anteriormente mencionadas en la elaboración de los productos, además de existir un desconocimiento de la estrategia adoptada por la Universidad para guiar el proceso de desarrollo de software educativo.

Aunque las soluciones presentadas no se puedan adecuar íntegramente a las características particulares del tipo de software que se desarrolla en esta institución, la fusión de un conjunto de características de las mismas sí podría formar parte de la propuesta que será presentada en el próximo capítulo.

CAPÍTULO II

Acercamiento al flujo productivo de la UCI

2.1 Introducción

"El requerimiento de experticia en el área por parte de los desarrolladores, y en especial de los líderes de proyecto es el punto clave en la ejecución adecuada de los proyectos de desarrollo de software educativo. Pero como en otras áreas del desarrollo humanos, una solución a este déficit es el uso de guías metodológicas, procedimientos de trabajo, herramientas documentales; en fin, un sistema metodológico que basado en las experiencias de proyectos anteriores permita a los líder y jefes de proyectos tener un apoyo y una herramienta que le ayuda a realizar un mejor y más eficiente proceso de desarrollo". (MARTÍNEZ *et al.* 2007)

La cita anterior es extraída del "Sistema Metodológico para el desarrollo de Software Educativo" que fue elaborado en la Universidad de las Ciencias Informáticas, como respuesta a la necesidad de una guía que contribuya a garantizar la adecuada ejecución de los proyectos que pertenecen a esta línea de producción. Dicha cita ha sido referenciada por tener incluido el mensaje por el cual se debe buscar una alternativa adecuada para el trabajo productivo en la Universidad en todo su contexto: la carencia de un conjunto de elementos que forman parte y definen pautas a la hora de llevar a cabo la realización de un producto software. Por este motivo, se decidió realizar un estudio sobre el funcionamiento de proyectos anteriores que condujeran a la creación de un documento oficial que contribuyera a mejorar el proceso de desarrollo de software.

En este capítulo se pretende realizar un análisis exhaustivo de los procesos que forman parte del flujo general de producción (Ver Anexo 3) que fue definido en este sistema metodológico, a fin de materializar un conjunto de observaciones que servirán de ayuda y complemento para posteriores propuestas que ayudarán a mejorar esta metodología. Para llevar a cabo este análisis se utilizó como referencia las características genéricas del proceso de desarrollo de software que plantea Pressman en Ingeniería del

Software. Un enfoque práctico, las cuales siempre deben estar presentes, independientemente del modelo específico de proceso que se elija cuando se determine construir un software.

También se pretende efectuar un estudio de los proyectos productivos de software educativo que han sido establecidos en la universidad, reflejando de los mismos un conjunto de deficiencias, que han sido obtenidas a partir de entrevistas y encuestas que han sido desarrolladas a los líderes de estos proyectos. Esto queda representado el diagrama de ISHIKAWA (Espina de pescado o diagrama Causa-Efecto) (Ver Anexo 13). Sus resultados servirán de apoyo para el posterior planteamiento de un conjunto de principios estratégicos para la guía de los procesos productivos de software educativo en la UCI. Dichos resultados, unido al análisis del conjunto de modelos, metodologías y procesos definidos con anterioridad, a lo planteado en las normas internacionales sobre el ciclo de vida de los procesos y a las observaciones derivadas de la comparación entre las actividades definidas por Pressman y los procesos especificados en el flujo productivo de la UCI, conducen al surgimiento de un nuevo enfoque para el desarrollo de software educativo.

2.2 ¿Cómo se organiza el desarrollo del software educativo en la UCI?

Desde los inicios de la Universidad de las Ciencias Informáticas se han ido identificando un conjunto de problemas provocados por diversos factores que atentan contra el correcto desarrollo de los proyectos de producción de software educativo. Estas afectaciones han sido provocadas fundamentalmente por la inexperiencia del personal y la falta de procedimientos para guiar a los encargados de dichos proyectos, lo cual se justifica si se tiene en cuenta que estamos en presencia de una nueva institución, en la cual se han ido creando las bases poco a poco para el desarrollo de software, una institución que aún se encuentra en formación y que por tanto existe un desconocimiento en algunas áreas relacionadas con la producción del software. Esta situación empeora cuando se requiere que los productos que sean elaborados correspondan al ámbito educativo, para el cual existen muy pocas metodologías a nivel internacional, y que además no pueden utilizarse de forma directa para la guía de los procesos de desarrollo en la universidad, pues cada proceso de producción se comporta de forma diferente en cada entidad, dependiendo de un conjunto de factores que se asocian al personal involucrado en el proyecto, a la concepción pedagógica, a las características de cada empresa y a su organización.

Todo ello provoca que la universidad defina que “ante esta situación la definición de procedimientos, sistemas de trabajo y herramientas se convierten en la solución para adelante las demandas de producción de software educativo” (MARTÍNEZ *et al.* 2007). De ahí que se decide elaborar un Sistema Metodológico que se adapte a las condiciones de trabajo, tecnología y formas de organización, que se sigue en la universidad, para que sirva de guía a todo el personal que se encuentra vinculado a proyectos de esta índole.

Aunque fue elaborada una propuesta para guiar el proceso de desarrollo de software educativo en la UCI, aún siguen existiendo numerosas dificultades en la gestión de los proyectos que se identifican con los mismos, por lo que se ha decidido detectar, cuáles son los aspectos que no han sido contemplados en el flujo de trabajo que ha sido presentado como solución a la problemática anteriormente mencionada. Para ello se decide realizar un estudio de las actividades definidas para el desarrollo del software a partir de la teoría precisada por Pressman (Ver Anexo 14).

2.3 Fases Genéricas para la ingeniería del software

“El trabajo que se asocia a la ingeniería del software se puede dividir en tres fases genéricas, con independencia del área de aplicación, tamaño o complejidad del proyecto. Cada fase se encuentra con una o varias cuestiones de las destacadas anteriormente”. (PRESSMAN 2002).Dichas fases se encuentran representadas en la **Figura 3.**

Las fases a las cuales hace referencia Pressman en la cita anterior son: fase de definición, fase de desarrollo y fase de mantenimiento. Durante la fase de definición se trata de identificar qué información debe ser procesada, función y rendimiento que se desea, comportamiento del sistema, interfaces que van de ser establecidas, restricciones de diseño existentes y criterios de validación que se necesitan para definir un sistema correcto. Durante la fase de desarrollo se intenta definir cómo han de diseñarse las estructuras de datos, implementarse la función dentro de una arquitectura de software, implementarse los detalles procedimentales, caracterizarse las interfaces, traducirse el diseño en un lenguaje de programación o lenguaje no procedimental y realizarse las pruebas. Durante la fase de mantenimiento es necesario trabajar en función de los cambios asociados a la corrección de errores, a las adaptaciones que se requieren conforme evoluciona el entorno del software y a los cambios que

dependen de las mejoras producidas por los requisitos cambiantes del cliente. Estas fases se complementan con un número de actividades protectoras, entre las cuales se incluyen: seguimiento y control del proyecto de software, revisiones técnicas formales, garantía de calidad del software, gestión de configuración del software, preparación y producción de documentos, gestión de reutilización, mediciones y gestión de riesgos.

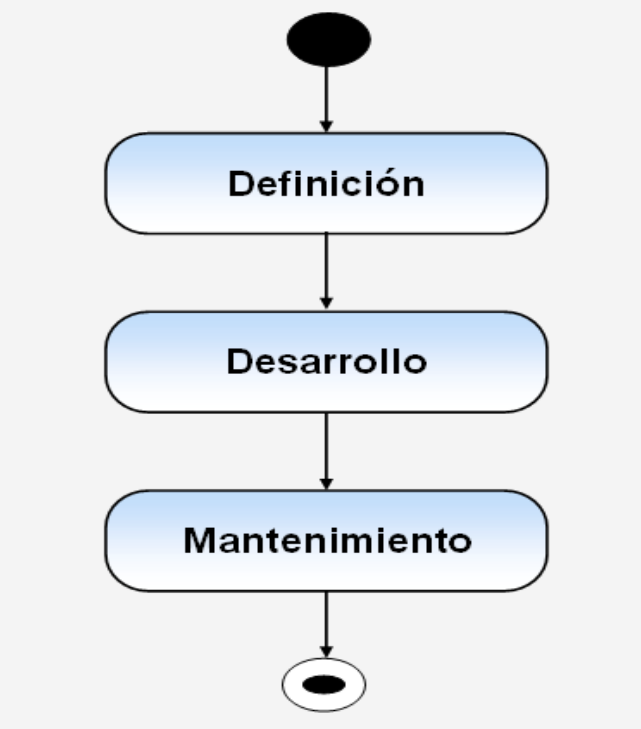


Figura 3 Fases genéricas asociadas a la ingeniería del software

2.3.1 Fase de Definición

Según Pressman la etapa de definición se divide en tres tareas fundamentales, representadas en la **Figura 4**: ingeniería de sistemas o de información, planificación del proyecto del software y análisis de

los requisitos. “La ingeniería de sistemas se centra en diversos elementos, analizando, diseñando y organizando esos elementos en un sistema que pueden ser un producto, un servicio o una tecnología para la transformación de información o control de información”. (PRESSMAN 2002).La ingeniería de sistemas ayuda a traducir las necesidades del cliente en un modelo de sistemas que utiliza uno o varios componentes de hardware, software, personas, bases de datos, documentación y procedimientos. Durante esta etapa se realizan un conjunto de actividades conocidas como ingeniería de requisitos y la misma comprende la identificación, análisis y negociación, especificación, modelización, validación y gestión de requisitos operacionales. Por su parte la planificación del proyecto del software implica la gestión de proyectos, que comprende la planificación, supervisión y control del personal, del proceso y de los eventos que ocurren durante la evolución del software; también involucra las actividades relacionadas con la estimación como el costo, esfuerzo, recursos y tiempo que supone construir un sistema o producto de software; comprende además las acciones para el análisis y la gestión de riesgo, como la identificación, evaluación de la probabilidad de aparición, la estimación del impacto que pueda producir y el plan de contingencia ante la aparición de problemas; y por último abarca todo lo relacionado con la planificación temporal y el seguimiento del proyecto. Durante el análisis de los requisitos se especifican las características operacionales del software (función, datos y rendimiento), se indica la interfaz del software con otros elementos del sistema y se establecen las restricciones que debe cumplir el software. En esta se crean modelos, son particionados los problemas y se desarrollan representaciones para mostrar las particularidades de los requisitos y los detalles de implementación. Además, se crean prototipos que permiten la refinación de los requisitos y se desarrolla la especificación de los requisitos del software. En esta fase se utiliza el modelo de datos y de flujos como base para crear el modelo de análisis, se involucran otros diagramas como el de entidad-relación, el de transición de estados, el modelo de comportamiento y el de contenido de datos con un diccionario de datos.

Figura 4 Tareas principales de la fase de definición

Teniendo en cuenta que la gestión eficaz de un proyecto de software se centra en las cuatro P´s (personal, producto, proceso y proyecto) se hizo necesario que en esta investigación se aludiera a la incidencia de las actividades que forman parte de las tareas correspondientes a la fase de definición definida por Pressman para el desarrollo del software (Ver Anexo 14), sobre los elementos que forman parte de dichas cuatro P´s. Para ello fue necesario agrupar dichas características de acuerdo a su correspondencia con cada uno de estos elementos con el fin de determinar cuáles de ellos toman mayor participación y son imprescindibles para la ejecución de un proyecto determinado. Los resultados son mostrados en la **Figura 5.**

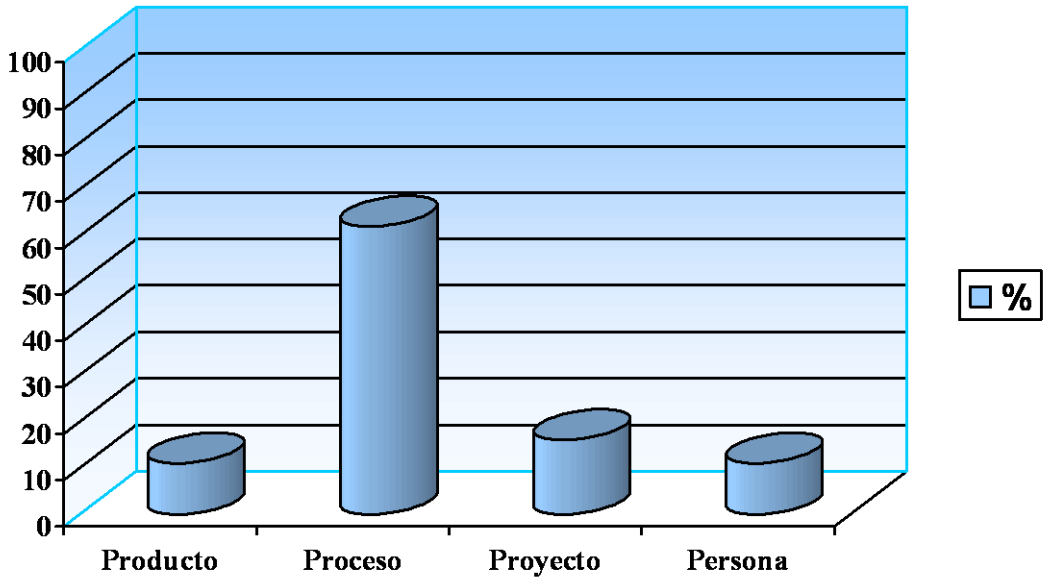


Figura 5 Distribución de las actividades por áreas para la fase de definición

En la figura anterior se puede observar que en la fase de definición, el elemento al cual se le concede mayor importancia es el proceso; y de la gran cantidad de actividades representadas se infiere que para un mejor desarrollo de proyectos software es necesario centrar la atención en las tareas que determinan el desarrollo y organización del mismo, las cuales están dirigidas principalmente, a todo el proceso de selección del modelo adecuado de ingeniería del software, a la organización del personal involucrado en el proyecto, a la estimación y planificación del tiempo y costo que traerá consigo la construcción de un producto, a la identificación, especificación y refinamiento de requisitos del software la selección y diseño de una arquitectura, entre otros aspectos. Este gráfico sugiere además, que independientemente que sean aseguradas todas las actividades que guardan relación con el proceso, no se deben descuidar las que se identifican con el proyecto, mediante el cual se debe garantizar la comunicación con el cliente, el seguimiento y control del proyecto, la definición de un plan de proyecto, entre otras. Asimismo es imprescindible que se garanticen las actividades relacionadas con las personas, pues la adecuada selección de líderes e integrantes de proyectos, constituyen elementos fundamentales para lograr el éxito de un proyecto de software. Además, es necesario determinar desde un inicio el ámbito del

software, se deben descomponer los requisitos del producto en las partes que lo constituyen y se deben crear prototipos del software. Todo ello forma parte de las actividades que debe cumplir el producto que se quiere desarrollar.

2.3.2 Fase de Desarrollo

La etapa de desarrollo se divide en tres tareas fundamentales, representadas en la **Figura 6**: diseño del software, generación de código y prueba del software. El diseño del software comprende el diseño arquitectónico y el de interfaz de usuario. Durante el diseño arquitectónico es necesario el diseño de datos, el cual comprende el modelado de datos, las estructuras de datos, las bases de datos y el almacén de datos; incluye la selección del estilo arquitectónico que se necesita escoger para un sistema determinado, la estructura y las propiedades que ese sistema comprende, así como las interrelaciones que tienen lugar entre todos los componentes arquitectónicos del sistema; el diseño arquitectónico agrupa inicialmente un conjunto de actividades de diseño que conducen a un modelo completo del diseño del software. Por su parte el diseño de interfaz de usuario da especial énfasis a la identificación de los requisitos del usuario, de las tareas y el entorno, para a partir de aquí crear los escenarios de usuario y definir el conjunto de objetos y acciones de la interfaz. “La interfaz de usuario es la ventana del software. En muchos casos, la interfaz modela la percepción que tiene un usuario de la calidad del sistema”. (PRESSMAN 2002). La segunda tarea: generación de código, se centra en el diseño a nivel de componentes, a partir del cual se representan las estructuras de datos, las interfaces y los algoritmos con suficiente detalle como para servir de guía en la generación de códigos fuente de lenguajes de programación. En el caso de las pruebas del software, su objetivo está dirigido a determinar las pruebas que se deben hacer, diseñar la prueba, aplicar las diferentes técnicas de diseño de casos de prueba: prueba de caja blanca y de caja negra y por último la recodificación de errores.

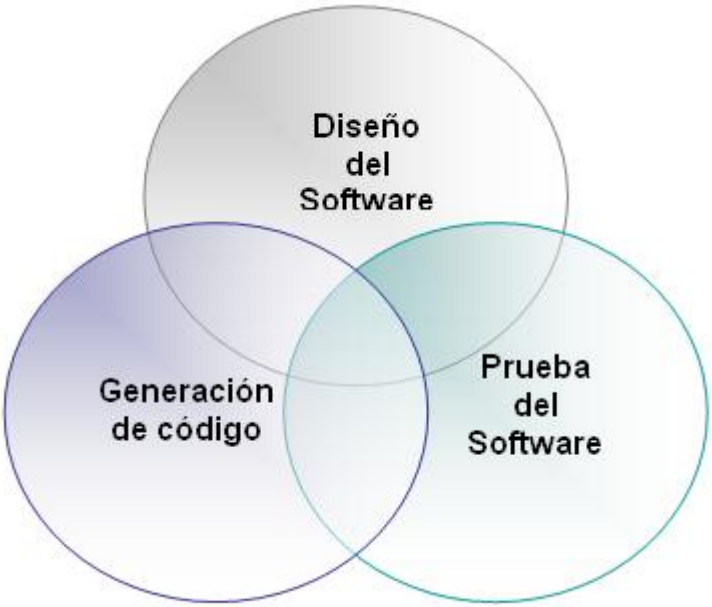


Figura 6 Tareas principales de la fase de desarrollo

En la gráfica de la **Figura 7** queda reflejado que en la fase de desarrollo al igual que en la de definición el mayor énfasis es realizado sobre el proceso, las actividades que guardan relación con el mismo, estarán dirigidas al diseño del software, a la selección y definición de estilos arquitectónicos, a las actividades de diseño de la interfaz de usuario, entre otras. Se puede observar que, seguido del proceso, pasa a tener gran importancia el producto, el cual tiene entre sus actividades esenciales la creación de prototipos a partir de la implementación del modelo de diseño y la realización del diseño a nivel de componentes. Por su parte las actividades correspondientes al proyecto están dirigidas a la aplicación de pruebas (caja blanca y caja negra). Para esta fase las actividades referentes a la personas tienen un peso insignificante.

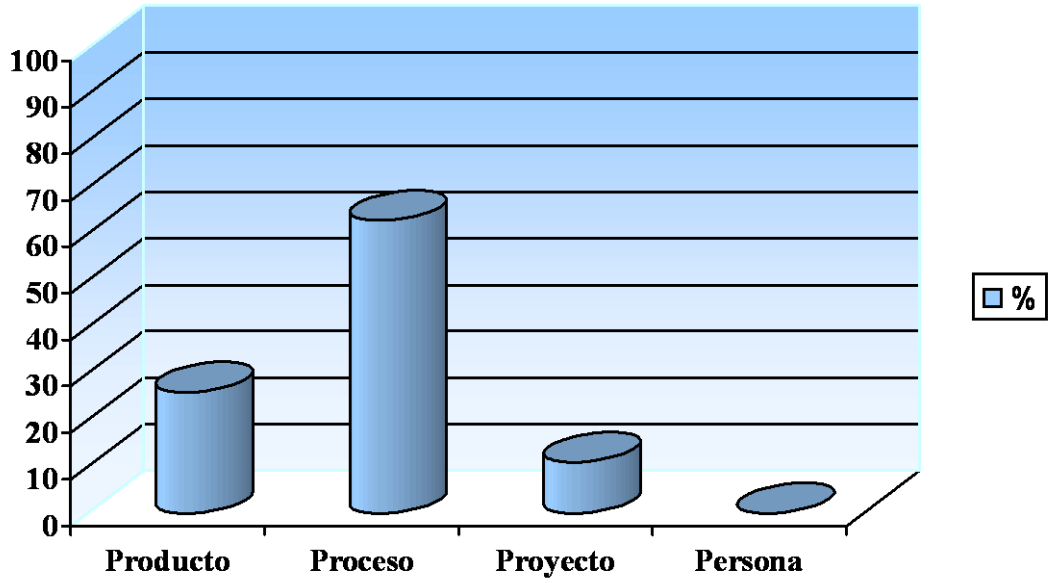


Figura 7 Distribución de las actividades por áreas para la fase de desarrollo

2.3.3 Fase de Mantenimiento

La fase de mantenimiento se centra en el cambio, contemplando 4 tipos de cambios: Corrección, Adaptación, Mejora y Prevención. Ver **Figura 8** "El mantenimiento correctivo cambia el software para corregir los defectos". "El mantenimiento adaptativo produce modificación en el software para acomodarlo a los cambios de su entorno externos". El mantenimiento perfectivo lleva al software más allá de sus requisitos funcionales originales". "El mantenimiento preventivo también llamado reingeniería del software, se debe conducir a permitir que el software sirva para las necesidades de los usuarios finales. En esencia, el mantenimiento preventivo hace cambios en programas de computadora a fin de que se puedan corregir, adaptar y mejorar más fácilmente". (PRESSMAN 2002)

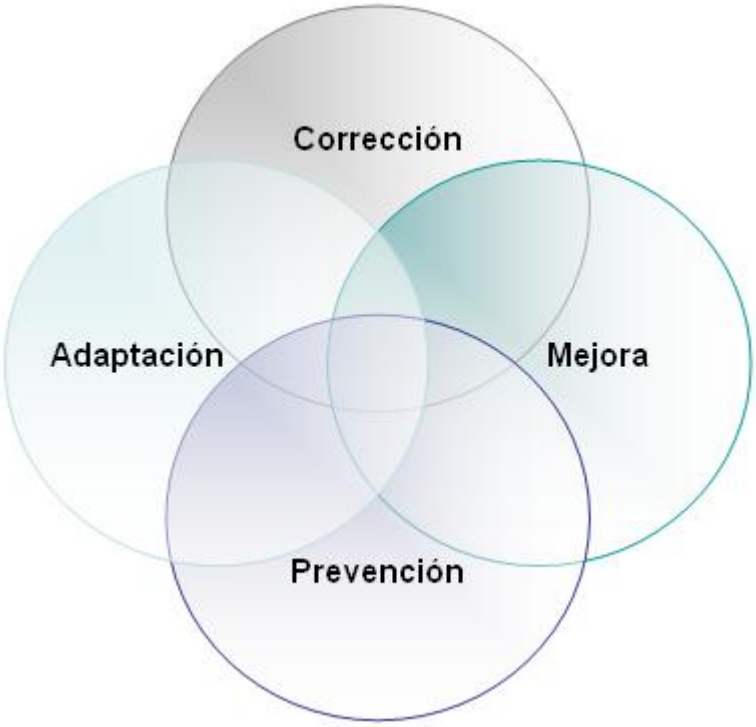


Figura 8 Tareas principales de la fase de mantenimiento

Como se puede observar en la **Figura 9,** las actividades que se realizan en esta fase, mayormente inciden sobre el producto. Están dirigidas fundamentalmente a la corrección de errores. Seguidamente, pueden observarse las actividades que guardan vínculo estrecho con el proceso, dentro de las cuales se puede mencionar la de prevención. Para el proyecto y las personas, no existe un número significativo de actividades definidas en esta fase.

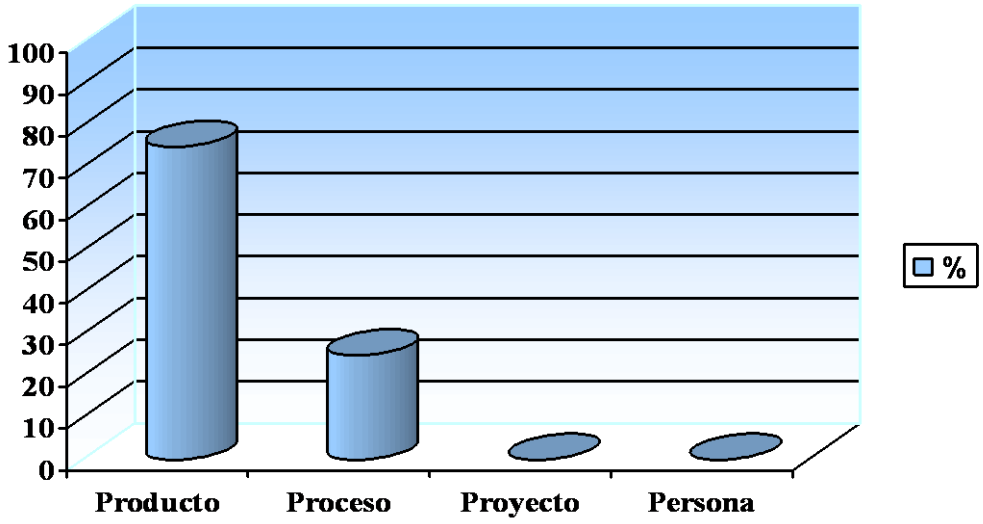


Figura 9 Distribución de las actividades por áreas para la fase de mantenimiento

2.3.4 Conclusiones desde una mirada ingenieril

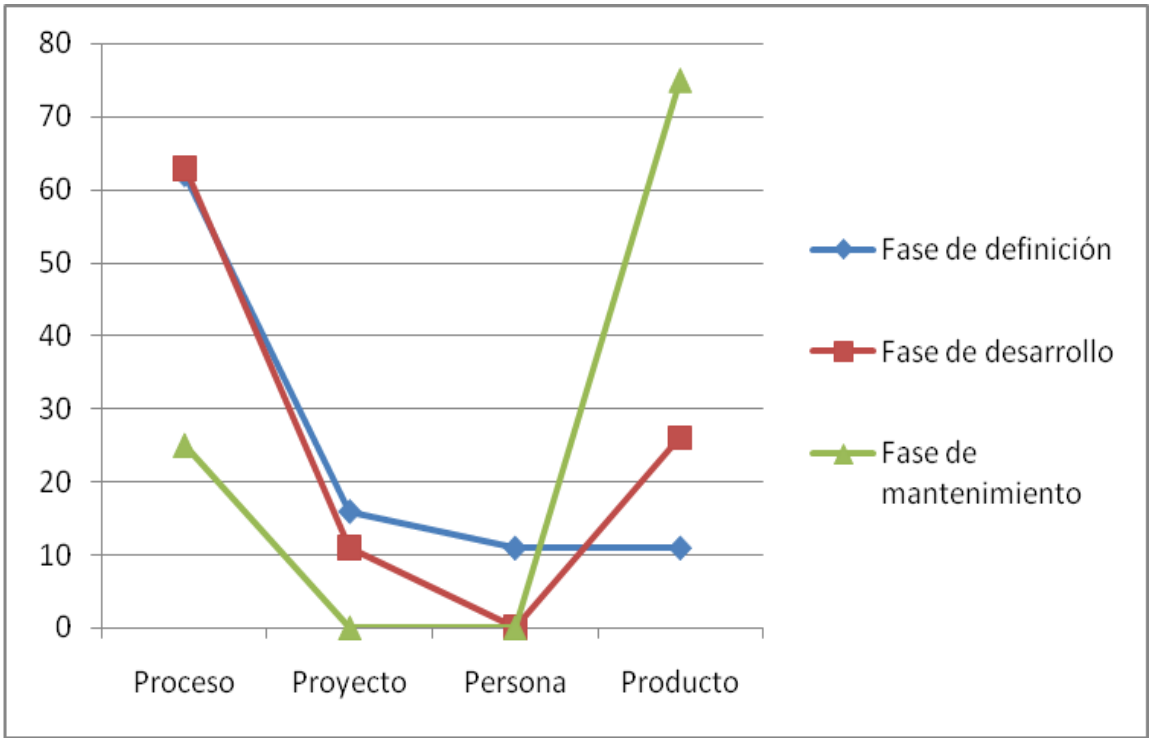
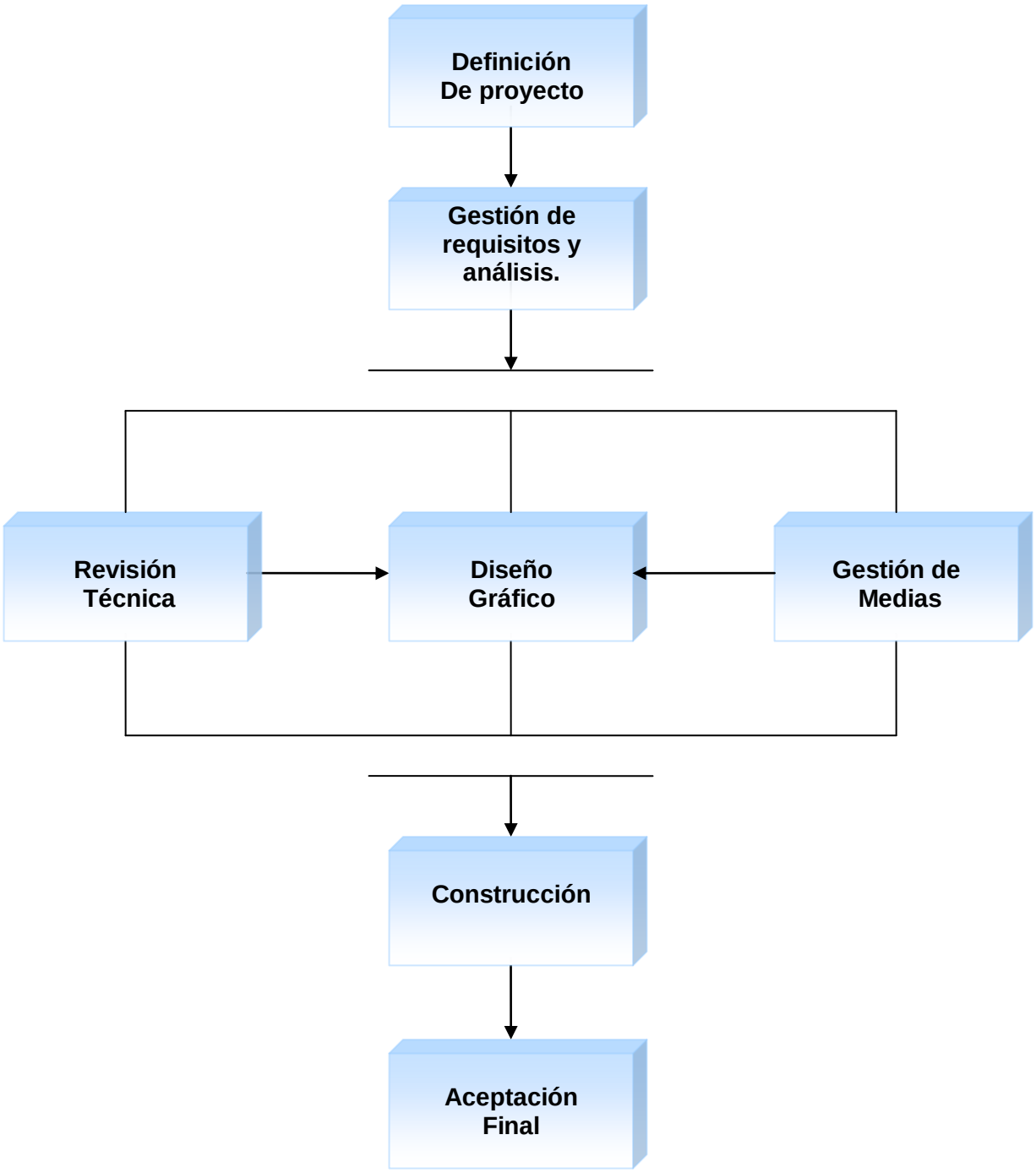


Figura 10 Desglose de actividades por áreas para las fases genéricas

A partir del análisis del conjunto de actividades que forman parte de cada una de las fases definidas por Presuman, se puede llegar a la conclusión, que el trabajo de ingeniería centra especial atención en las fases de definición y desarrollo, constituyendo la primera de ellas la rectora del desarrollo de software, pues de su resultado depende la continuidad y avance de un proyecto de software hacia las restantes fases. El cúmulo de actividades que la caracterizan supera en creces a las definidas en la fase de desarrollo y mantenimiento, significando el 62% del total de actividades que han sido identificadas para el desarrollo de las tres fases anteriormente mencionadas. El 31% de estas actividades son dedicadas a la puesta en práctica de la fase de desarrollo y solo el 7% está destinado a la fase de mantenimiento. Como se puede observar en la **Figura 10,** una gran parte de las actividades que son desarrolladas en las tres fases inciden sobre el proceso, de ahí que se pueda resaltar la importancia y papel fundamental que juega este elemento durante el desarrollo del software. Si se observa este gráfico, teniendo en cuenta los elementos de las 4 P´s se puede determinar que, la atención al proceso se evidencia en las fases de definición y desarrollo. Por su parte, las actividades relacionadas con el producto hacen énfasis en la fase de mantenimiento y en la de desarrollo. Las tareas relacionadas con el proyecto hacen hincapié en las fases de definición y desarrollo y las referidas a las personas en la fase de definición.

2.4 Representación del Flujo de Producción del Sistema Metodológico para el desarrollo de Software Educativo en la UCI



2.4.1 Descripción comparativa entre flujo productivo UCI y las fases definidas por Pressman

Para lograr el cumplimiento del objetivo de esta investigación y después de haber realizado un profundo análisis de las actividades contenidas en las fases genéricas definidas para el desarrollo del software (definición, desarrollo y mantenimiento), planteadas por Pressman, fue necesario realizar una comparación de dichas actividades con las incluidas en los procesos que conforman el flujo productivo del sistema metodológico de la UCI, ya referenciado con anterioridad.

Inicialmente se ubicaron los procesos que forman parte del flujo productivo en las fases anteriormente mencionadas, de acuerdo a la correspondencia existente entre las actividades contenidas en cada uno de ellos. Atendiendo a esto, se determinó que la fase de definición estaría compuesta por los procesos: Definición del proyecto, Gestión de requisitos y análisis y una parte de las actividades que pertenecen a Evaluación técnica; por su parte en la fase de desarrollo están incluidas: las actividades restantes realizadas en el proceso de Evaluación técnica, Diseño gráfico, Gestión de medias y algunas actividades desarrolladas durante el proceso de Construcción. Las demás actividades de este proceso y las incluidas en el proceso de Aceptación del producto pertenecen a la fase de mantenimiento. En esta comparación se realizó mayor énfasis en las diferencias asociadas, con el objetivo de tomarlas como punto de referencia para el posterior planteamiento de otros principios relacionados con actividades que no estén contempladas y que puedan formar parte de la propuesta de solución.

Es importante destacar que aunque no se pueda apreciar de forma directa una correspondencia entre algunas de las actividades pertenecientes a los paradigmas que han sido seleccionados en la comparación, no significa que no sean similares. La diferencia radica en que el hecho de que se esté trabajando con software educativo trae consigo que dichas características sean planteadas siguiendo un enfoque educativo, en los términos que son manejados a nivel internacional para la elaboración de software de este tipo.

A continuación serán mencionadas por cada proceso las actividades definidas por Pressman, que no están incluidas en el flujo productivo. Para la fase de definición, en el proceso de Definición de proyectos, no se refleja que se debe seleccionar las técnicas para la coordinación de proyectos, ni estimar la planificación temporal, el costo inicial, el esfuerzo, los recursos humanos, software reutilizables y herramientas hardware/software; tampoco se plantea que se debe seleccionar un modelo

de proceso, definir un plan de proyecto preliminar, realizar un análisis de riesgo y seleccionar las métricas para el proyecto. Para el proceso de Gestión de requisitos y análisis no se tiene especificado que se deba realizar la estructura del modelo de implementación, la creación del modelo de datos ni del modelo funcional, tampoco el modelado del comportamiento ni la creación del diagrama Entidad-Relación. En el caso de Evaluación técnica se define en el documento Diagnóstico de producción todo lo referente a la arquitectura pero no existe nada reflejado en este proceso con relación a la infraestructura.

Por su parte para la fase de desarrollo, en el proceso de Evaluación técnica no se contempla la realización de un diseño de datos (traduce los objetos de datos definidos en el modelo de análisis en estructuras de datos o arquitectura para base de datos o almacén de datos), la selección de un estilo arquitectónico y de patrones, el análisis de diseños arquitectónicos, el análisis de transformación o de transacción de Diagrama de flujo de datos (DFD) a un estilo arquitectónico, y el refinamiento del diseño arquitectónico. Por su parte las actividades que contempla el proceso de Diseño gráfico tienen correspondencia con lo planteado por Pressman y las actividades del proceso de Gestión de media son más específicas para software educativo. En esta fase también se tiene en cuenta lo referente a las pruebas que se le realizan al producto y en el flujo no está contemplada la determinación de la prueba, el diseño de prueba, la aplicación de pruebas de caja blanca y de caja negra pertenecientes al proceso de Construcción.

Finalmente para la fase de mantenimiento no se tiene en cuenta que debe realizarse la adaptación, la mejora y la prevención, estas actividades corresponden a la Aceptación del producto. Existen también actividades protectoras, estas deben aparecer a todo lo largo del desarrollo del software, de estas actividades no se especifican en el flujo, la garantía de la calidad del software, la preparación y producción de documentos, las mediciones y la gestión de riesgo. Todo lo antes descrito puede contribuir en gran medida al planteamiento de principios estratégicos para la guía del proceso de desarrollo de software educativo que es el objetivo de esta investigación.

2.4.2 Análisis del Diagrama Causa-Efecto

Durante el desarrollo de esta investigación fue necesario obtener los principales problemas que quebrantan el avance sucesivo de los proyectos productivos de software educativo en la UCI, que se

encuentran estrechamente relacionados con el proceso de desarrollo de software. Para revelar estas insuficiencias fue necesaria la realización de entrevistas (Ver Anexo 11) y encuestas (Ver Anexo 10) a líderes de 7 proyectos productivos que pertenecen a las Facultades 8 y 9. Los resultados de las mismas fueron representados en un Diagrama Causa- Efecto (Ishikawa) o Diagrama Espina de Pescado (Ver Anexo 13). Dichos resultados muestran las principales causas que dan origen a la ineficiencia en el proceso de desarrollo de software educativo, lo cual constituye el problema principal determinado durante el estudio realizado. Del mismo se derivan 5 subproblemas fundamentales, cada uno con las causas que los originan, estos subproblemas son: mala asignación de roles, deficiente gestión de personas, inadecuada gestión de recursos materiales (RRMM), ineficiente aplicación de principios y deficiente gestión de proyectos. La propuesta que se obtendrá como resultado de este trabajo de diploma tendrá en cuenta dichos problemas, que pueden ser extensivos al desarrollo de software educativo fuera de la universidad.

2.5 Nuevo enfoque para la producción de software educativo en la UCI

Bajo la premisa de cumplir el objetivo que ha guiado el desarrollo de esta investigación se hace necesario que se realice una referencia sintetizada de las bases sobre las cuales descansa la solución propuesta.

Inicialmente se realizó un estudio riguroso de un conjunto de modelos, metodologías y procesos de desarrollo de software, dentro de los cuales se hizo especial énfasis en los dedicados al desarrollo de software educativo, por sus aspectos novedosos en el ámbito pedagógico-didáctico. Posteriormente, se realizó un estudio de las normas o estándares internacionales para el desarrollo de software, y finalmente, constituyendo el eslabón fundamental para conformar la propuesta de solución se tomó como referencia la comparación realizada entre las actividades contenidas en las fases genéricas de la ingeniería del software, definidas por Pressman y las actividades contempladas en los procesos que forman parte del flujo productivo perteneciente al Sistema Metodológico UCI. Tomando algunos de los principales aspectos de cada uno de los elementos anteriormente mencionados, se conforma un híbrido que ha sido generalizado y contextualizado a la UCI, el cual ha sido denominado “Nuevo enfoque para la producción de software educativo en la UCI”, cuyas siglas lo denotan como NEPSE. Esta propuesta está

integrada por 73 principios estratégicos para la guía del proceso de desarrollo de software educativo en la UCI, basados en actividades de gestión, de corte pedagógico-didácticas y en las actividades ya establecidas en el Sistema Metodológico, muchas de las cuales fueron modificadas, en función de lograr un mejor entendimiento de las mismas. A continuación se realizará una breve descripción de cada uno de los principios que forman parte de esta solución.

Fase de Definición: Definición del proyecto

1. **Solicitud del proyecto**

Establece el proceso de comunicación entre el cliente y representantes de la dirección de producción, así como demás interesados en el desarrollo del proyecto, con el objetivo de realizar la solicitud formal de adquisición de un producto de software educativo. Esta petición queda registrada en una planilla de solicitud de proyecto definida en el "Sistema Metodológico para el desarrollo de software educativo" (SMDSE).

2. **Comunicación preliminar y negociación con el cliente**

Establece que se realice la preparación y negociación de un contrato, en el cual pueden estar involucradas otras partes interesadas en el proyecto. Además se obtienen los requisitos iniciales a partir del guión de contenido entregado por el cliente.

3. **Definición del ámbito del software**

Establece que se debe describir toda la información necesaria para definir el alcance del software (prerrequisito para la estimación), a partir de reuniones o entrevista preliminar entre el cliente y los desarrolladores. Dicha información debe quedar especificada en el documento Visión definido en SMDSE.

4. **Estimación de recursos humanos y materiales**

Establece que una vez definido el ámbito del software se debe determinar la cantidad de personas necesarias para la ejecución del proyecto de acuerdo a las habilidades que se requieren para su desarrollo. Simultáneamente se debe estipular los dispositivos que son necesarios para garantizar el adecuado avance del proyecto.

5. Selección del líder de proyecto

Establece que se debe seleccionar el jefe del equipo (líder), teniendo en cuenta que debe tener habilidades para motivar, organizar y aglutinar personas, además de ser innovador y gran conocedor de su trabajo, con capacidades para resolver cualquier problema que se pueda presentar.

6. Selección de equipo de trabajo

Establece que se debe seleccionar el equipo de software de acuerdo a las necesidades requeridas para el desarrollo del proyecto, así como definir la estructuración del mismo a partir de los diferentes organigramas conocidos.

7. Establecimiento de las políticas de gestión de configuración y salvas

Establece las políticas de configuración por las cuales se regirá el equipo de software y se determina cómo será realizado el proceso para garantizar las salvas, dichas políticas son reflejadas en el documento Políticas de gestión de configuración y salvas planteado en el SMDSE.

8. Definición y asignación de roles

Establece que deben definirse los roles necesarios de acuerdo a las actividades y tareas que fueron definidas en el proyecto, Identificando los roles pedagógicos que deben intervenir para el desarrollo del mismo. Esta asignación debe ser realizada de acuerdo a las capacidades, conocimiento e interés de los integrantes del equipo.

9. Gestionar la documentación del proyecto

Establece que desde un inicio se debe contar con toda la documentación necesaria por la cual se regirá el equipo de software para la elaboración del producto y para la preparación del equipo de software.

10. Identificar la necesidad educativa

Se debe especificar la necesidad del programa educativo y seleccionar la teoría educativa a utilizar para el desarrollo del software de acuerdo a esta necesidad.

11. Definición del modelo de proceso

Establece que se debe seleccionar un modelo de proceso apropiado para la ingeniería del software que aplicará el equipo del proyecto de acuerdo a las características particulares del producto a realizar,

además de ver qué modelo sería el más adecuado para los clientes y los desarrolladores, así como tener en cuenta el entorno del proyecto en el cual se trabaja, la complejidad del producto y el tiempo que haya sido establecido para el desarrollo del mismo de acuerdo a la necesidad.

12. Estimación de la planificación temporal del proyecto, costo y esfuerzo

Establece que desde el inicio del proyecto se deben realizar estimaciones cuantitativas para determinar el tiempo y costo aproximado para el desarrollo del producto así como estimar el esfuerzo (hombres-mes) que se necesita para su ejecución. Esta estimación se realiza a partir del conocimiento del ámbito del software definido con anterioridad, pues no se cuenta aún con el análisis detallado de los requisitos del software.

13. Definición del plan de proyecto preliminar

Establece que una vez seleccionado el modelo de proceso se debe realizar un plan de proyecto donde sean reflejadas las actividades estructurales que se deben llevar a cabo en el transcurso del proyecto. Una vez establecido dicho plan, se debe crear un plan completo reflejando las tareas requeridas a las personas para cubrir las actividades estructurales.

14. Análisis y gestión de riesgo

Establece que desde el inicio del proyecto se realice el análisis y gestión del riesgo, el cual comprende la identificación, estimación, refinamiento y reducción-supervisión-gestión del riesgo, ya que el éxito de un proyecto depende en gran medida de la preparación que se tenga desde el principio sobre las cosas que pueden ir mal en el proyecto, pues su aparición afecta proporcionalmente el plan concebido para el mismo. El resultado de esta actividad debe quedar plasmado en un documento donde figure la lista de los riesgos identificados, además de concebir un plan de contingencia y mitigación por si llegara a ocurrir el problema.

15. Capacitación del equipo de software

Establece que una vez seleccionado el equipo de software se debe garantizar la capacitación de los integrantes del mismo para asegurar el posterior desempeño de cada uno. Esta capacitación puede ser desarrollada a partir de cursos relacionados con los elementos que se definen en el proyecto para el desarrollo del producto.

16. **Realizar un estudio sobre las pautas de diseño de interfaz adecuadas a la población estudiantil a la que va dirigida el programa**

Establece que teniendo en cuenta los usuarios a los cuales va dirigido el software se debe definir cuáles deberían ser los patrones de diseño más adecuados a tener en cuenta para la creación de la interfaz correspondiente al mismo.

Fase de Definición: Gestión de requisitos y análisis

1. **Elaboración del guión técnico**

Establece que una vez estudiada la solicitud del cliente, el guión de contenido, y sean aclaradas las posibles dudas sobre la documentación entregada, se realiza una revisión del guión hasta tener una total comprensión del mismo, cuando se logra esto se elabora el guión técnico.

2. **Identificación y clasificación de requisitos**

Establece que a partir del guión de contenido y las entrevistas con el cliente es posible realizar una identificación de requisitos, los cuales deben quedar registrados en el guión técnico, además de ser clasificados en requisitos funcionales o no funcionales dependiendo del papel que jueguen en el sistema.

3. **Identificación de actores y casos de uso**

Establece que una vez detallados los requisitos en el guión técnico, va a ser posible la identificación de los casos de usos y los actores que intervendrán en el sistema a modelar.

4. **Refinamiento del ámbito del software**

Establece que se deben realizar actualizaciones del ámbito del software, ya que las características del producto que se obtendrá, pueden variar a medida que avance el proyecto. Dichas actualizaciones se deben registrar en el documento Visión especificado en el sistema metodológico.

5. **Elaboración de las planillas de solicitud de medias**

Establece que de acuerdo a los requisitos dados en el guión técnico, se identifican las necesidades de recursos audiovisuales del producto que se esté desarrollando y a partir de esta información se realiza la solicitud a la Dirección de Comunicación Audiovisual, mediante una planilla de solicitud de medias.

6. **Creación de prototipos del software**

Establece que como no es posible especificar completamente un problema en una etapa tan temprana, se deben crear prototipos de software pues esto contribuye a que se pueda producir un modelo ejecutable del software a partir del cual se pueden refinar los requisitos. Esto se evidencia a través de la creación de pantallas que son especificadas en el guión técnico.

7. **Especificación de los requisitos del software**

Establece que una vez identificado los requisitos, se realiza un análisis detallado de los mismos y son especificados en el documento técnico definido en el Sistema Metodológico.

8. **Definir el tipo de programa a utilizar**

Establece que desde el inicio del proyecto se debe definir la tipología o clasificación del software que se desea desarrollar. Ej. Sistemas Tutoriales, Libros Electrónicos, Simuladores, Sistemas Entrenadores, Sistemas Expertos, entre otros.

9. **Definir el tipo de interactividad**

Establece que se debe definir qué mecanismos deben estar presentes en el software que se va a construir para garantizar una comunicación bidireccional, rica y reflexiva.

10. **Priorizar e integrar los requisitos educativos (pedagógicos) con los del software**

Establece que se deben identificar un conjunto de requerimientos pedagógicos de acuerdo a las necesidades educativas especificadas con anterioridad, relacionados con el contenido y la población estudiantil a la que va dirigido el software e integrarlos al conjunto de requerimientos que han sido especificados con anterioridad.

11. Refinar los requerimientos de diseño gráfico y aspectos comunicacionales

Establece que se debe refinar los requisitos que fueron establecidos para el diseño gráfico y aspectos comunicacionales a partir de las pautas pedagógicas establecidas.

12. Refinamiento de requisitos del sistema

Establece que una vez identificados los requisitos del sistema debe realizarse un estudio más detallado de las nuevas funcionalidades o cambios que puedan ser reconocidos a lo largo del proyecto y de acuerdo a esto realizar las actualizaciones y mejoras que sean necesarias.

13. Realización de la modelación del análisis

Establece que se debe realizar la modelación del análisis como punto de partida para un mejor entendimiento del diseño. Los elementos que formen parte de dicha modelación deben quedar registrados en el guión técnico incluido en el Sistema Metodológico.

14. Creación del Diagrama Entidad-Relación

Establece que en caso de ser necesario el empleo de una base de datos en el software que se esté desarrollando se debe realizar inicialmente el diagrama entidad-relación.

Fase de Definición: Evaluación técnica

1. Estudio de los requisitos complementarios tanto de software como educativos

Establece que una vez identificados los requisitos críticos se hace necesario que se realice un estudio exhaustivo de los requisitos de software y educativos que complementan el software a desarrollar.

2. Actualizar el plan de proyecto

Establece que se debe actualizar el plan de proyecto especificado en la definición de proyecto. Dicho plan es consultado repetidamente, actualizando riesgos, estimaciones, planificaciones e información relacionada con el proyecto.

3. **Definición, análisis y diseño de la arquitectura basándose en la teoría educativa elegida**

Establece que inicialmente se debe realizar una recomendación de la arquitectura apropiada a utilizar para el desarrollo del software educativo de acuerdo a la teoría educativa que fue seleccionada con anterioridad así como de la arquitectura organizativa para acometer la producción. De acuerdo a las características particulares de dicho software se decidirá si la propuesta realizada es factible y a partir de ello se realizarán los análisis necesarios para un posterior diseño de la misma. Esta arquitectura quedará especificada en el documento de diagnóstico de producción perteneciente al sistema metodológico.

Fase de Desarrollo: Evaluación técnica

1. **Diseñar la base de datos en caso de existir**

Establece que en caso de que el software a construir requiera de la existencia de una base de datos, esta se diseñará tomando como base el diagrama entidad-relación.

2. **Seleccionar estilo arquitectónico**

Establece que para la construcción de un software se debe seleccionar un patrón arquitectónico, con el cual se obtendrá una visión general de la estructura del software a partir de su arquitectura.

3. **Realización de la modelación del diseño**

Establece que a partir del diseño arquitectónico se agrupan inicialmente las actividades de diseño que conducen a la modelación del diseño del software. Para la realización del modelo del diseño es necesario el guión técnico.

Fase de Desarrollo: Diseño gráfico

1. **Identificación de los requisitos del usuario y las tareas**

Establece que se deben identificar detalladamente las necesidades del usuario final y las tareas de procesamiento necesarias para que el software lleve a cabo la función deseada.

2. **Definir pautas del diseño**

Establece que se debe tener en cuenta la organización de menús, tipos de íconos a usar, efectos a usar, selección de textos, diseño de pantallas y menús, entre otros.

3. **Definir actividades del diseño de la Interfaz**

Establece que una vez finalizado el análisis de las tareas, quedan definidas detalladamente las que requiere el usuario final, lo que sirve como entrada a la actividad del diseño de la interfaz.

4. **Diseñar las interfaces de usuario**

Establece que para diseñar la interfaz de usuario se deben tener en cuenta las características del público al que se dirige el software, las necesidades educativas identificadas y la cumplimentación de las funcionalidades requeridas para la correcta ejecución del software.

5. **Definir criterios de navegación**

Establece que dependiendo de la estructura que se deba seguir para lograr los objetivos pedagógicos que se persiguen en el contenido, se deben determinar los criterios de navegación correspondientes a dicha necesidad.

6. **Definir las actividades y tipos de módulos que se deben utilizar**

Establece que a partir del guión de contenido y del guión técnico se establecerán y organizarán los módulos que conforman el software que se está desarrollando y se definirán las actividades que se realizarán en dichos módulos.

7. **Creación de un prototipo a partir de la implementación del modelo de diseño**

Establece que a partir de la implementación del modelación del diseño definido con anterioridad será creado un prototipo funcional del software que se está desarrollando.

8. **Evaluación del diseño**

Establece que a partir de la creación de un prototipo de interfaz de usuario, se deberá pasar a la evaluación del mismo, con el objetivo de determinar si cumple o no las necesidades del usuario.

9. **Elaborar la documentación correspondiente a este proceso**

Establece que una vez finalizado el diseño gráfico se debe procesar la documentación correspondiente al mismo.

Fase de Desarrollo: Gestión de Medias

1. **Búsqueda o producción de recursos audiovisuales**

Establece que a partir de la solicitud de medias realizada con anterioridad, se debe proceder a la búsqueda o producción de los recursos especificados en dicha solicitud, la cual es posible realizarla de acuerdo a los requisitos especificados en el guión técnico.

Fase de Desarrollo: Construcción

Etapa 1: Implementación

1. **Realizar diseño a nivel de componentes**

Establece que a partir del diseño de datos, arquitectura definida e interfaz se debe realizar el diseño a nivel de componentes, lo cual es evidenciado en el guión técnico.

2. **Revisar y actualizar la lista de riesgos**

Establece que a medida que avanza el proyecto se hace necesario revisar la lista de riesgos y supervisar cuáles de los planteados ha sido mitigado. También se debe tener en cuenta la posibilidad de aparición de riesgos que no fueron previstos. Estos deben ser identificados e incluidos en la lista de riesgos.

3. **Generación de código fuente de lenguaje de programación**

Establece que las representaciones detalladas de las estructuras de datos, interfaces y algoritmos servirán de guía en la generación de códigos fuente de lenguaje de programación.

4. **Concebir la integración de los módulos**

Establece que una vez elaborados los módulos se debe pasar a la integración de los mismos para así lograr el correcto funcionamiento del software.

5. **Probar el diseño instruccional, comunicacional y gráfico**

Establece que debe existir una correspondencia entre el diseño y los criterios de evaluación previamente establecidos.

Etapa 2: Prueba

1. **Determinación de las pruebas que se deben aplicar**

Establece que se deben crear una serie de casos de prueba que intentan "devastar" el software construido. A partir de esto se deben seleccionar las pruebas que sean más adecuadas para comprobar el funcionamiento del software.

2. **Planificar y ejecutar tareas de verificación y validación del software**

Establece que para verificar y validar el software se deben planificar un conjunto de tareas que conduzcan a la evaluación de las funcionalidades que han sido especificadas durante el desarrollo del producto.

3. **Diseño de casos de pruebas**

Establece que se deben diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo posible. Estas pruebas es lo que conocemos como pruebas de caja negra y pruebas de caja blanca.

4. **Aplicación de pruebas de caja blanca**

Establece que se deben realizar pruebas dirigidas al código con el objetivo de encontrar errores procedimentales. Con estas pruebas se comprueban los caminos lógicos del software y se proponen casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Permite que se pueda examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

5. **Aplicación de pruebas de caja negra**

Establece que se deben realizar pruebas sobre la interfaz del software para detectar los errores que puedan existir en la misma. Estos pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce un resultado correcto y que la integridad de la información externa se mantiene. Dichas pruebas se centran en los requisitos funcionales del software.

6. **Registrar los resultados de aplicar las pruebas**

Establece que deben quedar registrados en un documento los resultados de aplicar las pruebas.

Etapa 3: Revisión técnica

1. **Revisión técnica del entregable**

Establece que se debe realizar una revisión de la versión del producto tomando como referencia el guión técnico, los casos de prueba y las pautas técnicas.

2. **Determinación de errores a partir de la revisión técnica**

Establece que la realización de las revisiones técnicas va a dar paso a la identificación de los errores del software.

3. **Realización de un plan de corrección de errores**

Establece que una vez detectado los errores en las revisiones técnicas debe ser realizado inmediatamente un plan para corregir dichos errores.

Fase de Mantenimiento: Construcción

Etapa 1: Corrección de errores

1. **Se corrigen los errores detectados**

Durante la revisión técnica, a partir del listado de errores y del guión técnico son corregidos los defectos que fueron detectados en el software.

Etapa 2: Entrega y revisión del cliente

1. **Entrega del producto al cliente**

Establece que una vez corregidos los errores se realiza una revisión de contenido del entregable por parte del cliente. En caso de que este entregable contenga errores se define un plan de corrección y se regresa a la **Etapa 1** (corrección de errores), de lo contrario se entrega al cliente una carta de aceptación para la entrega parcial.

Fase de Mantenimiento: Aceptación final del producto

1. **Aceptación del producto**

A las entregas realizadas se le hacen revisiones y pruebas de aceptación. Las revisiones y prubas de aceptación deben tener en cuenta los resultados de las revisiones conjuntas, auditorias, pruebas del software, entre otros. Se debe documentar los resultados de las pruebas y revisiones de aceptación. Este proceso culmina con una carta de aceptación del producto emitida por el cliente.

Actividades de Apoyo

1. **Seguimiento y control del proyecto del software**

Establece que a partir de la planificación realizada se debe efectuar un seguimiento del proyecto y un control de todas las tareas y actividades incluidas en dicha planificación. Los pasos que se han establecido para el control del avance del proyecto es descrito en el sistema metodológico.

2. **Revisiones técnicas formales**

Establece que se deben efectuar revisiones técnicas con el fin de evaluar los productos o servicios software bajo consideración.

3. Aseguramiento de la calidad

Establece que se debe asegurar que los productos y procesos del ciclo de vida del proyecto se correspondan con los requerimientos especificados y cumplan con los planes establecidos.

4. Realización de la Gestión de configuración del software

Establece que se deben definir las políticas para el desarrollo de los proyectos, las herramientas para la gestión de control de versiones, las herramientas de trabajo colaborativo Web, los servidores donde se pueden encontrar dichas herramientas., la documentación básica para su uso. Este proceso de gestión de la configuración es ampliado en el Sistema Metodológico.

5. Preparación y producción de documentos

Se debe registrar a lo largo del ciclo de vida del proceso de desarrollo de software la documentación que se irá generando a medida que avanza el proyecto, tanto la interna como la externa. Dicha documentación debe incluir los resultados de las evaluaciones. Se debe elaborar un manual de usuario con información técnica.

6. Preparación y producción de documentación didáctica

Se debe planificar la documentación didáctica y elaborar una guía didáctica, con ejemplos de uso para los futuros usuarios del software. Dicha documentación debe tener la información detallada de las características del programa que haya sido realizado, su forma de uso y posibilidades didácticas. Se debe adjuntar esta información didáctica, los caminos pedagógicos, teorías del aprendizaje y programación didáctica al programa que se ha ido desarrollando.

7. Realizar gestión de reutilización

Establece que se debe definir todo lo referente a la gestión de elementos reutilizables.

8. Auditoria

Establece que se debe chequear el cumplimiento de las políticas y procedimientos establecidos, con el objetivo de determinar si se cumple con los requerimientos, planes y contratos antes especificados. Se realiza a través de listas de chequeo que integran todas las áreas definidas y establecen el nivel de exigencia de la dirección hacia las facultades en la presente etapa.

9. Seguimiento de la gestión de riesgo

Establece que a medida que avanza el proyecto se debe realizar un seguimiento de los riesgos que fueron planteados desde el inicio del mismo para determinar si estos se van haciendo más o menos probables. Se supervisa si se están realizando correctamente los pasos de reducción de riesgo.

10. Aplicación de métricas en el desarrollo del software

Establece que se deben seleccionar y aplicar métricas con el objetivo de evaluar la calidad del software en desarrollo.

2.6 Conclusiones parciales

El desarrollo del capítulo presentado ha permitido que la comparación establecida entre las actividades y tareas contempladas en las fases definidas por Pressman, unido a las actividades especificadas en el flujo productivo perteneciente al Sistema Metodológico UCI, haya contribuido en gran medida al planteamiento de un Nuevo Enfoque para la producción de software educativo en la Universidad de las Ciencias Informáticas, cumpliendo de esta forma el objetivo por el cual se ha trabajado en esta investigación.

Por su parte las encuestas y entrevistas realizadas a líderes de proyectos de software educativo, reflejadas en el diagrama Causa-Efecto (Ver Anexo 13), arrojaron un significativo número de deficiencias que atentan contra el correcto desarrollo de los proyectos productivos, lo cual conduce a la premisa de aplicar los nuevos principios que fueron planteados como solución del trabajo de diploma en dichos proyectos con el objetivo de lograr el exitoso desarrollo de los mismos.

CONCLUSIONES

El desarrollo de la investigación correspondiente al presente trabajo de diploma arrojó que:

- Existe una pobre aplicación de modelos, metodologías o procesos de desarrollo de software en los proyectos productivos de software educativo en la UCI, situación que debe mantenerse si no se crea una cultura de desarrollo industrial, y se le da el máximo de atención a los riesgos potenciales que pueden aparecer ante el afán de trabajar a la "buena de Dios", es decir, sin metodología alguna.
- Aunque la UCI ha adoptado una estrategia que se concreta en un "Sistema metodológico para el desarrollo de software educativo", es importante destacar que esta no contempla un significativo número de actividades que son necesarias para el desarrollo de software educativo, además de resaltar el desconocimiento que sobre la misma se posee por parte de los líderes de los proyectos de software educativo.
- Teniendo en cuenta que no existe una metodología de desarrollo de software universal, se puede recurrir al estudio de algunas de las soluciones existentes para la elaboración de productos software en función de tomar los elementos más significativos de estas y adecuarlos al contexto de la UCI.
- El estudio riguroso de un conjunto de modelos, metodologías y procesos de desarrollo del software permitió el planteamiento de nuevos principios que permiten la guía del proceso productivo de software educativo en la Universidad de las Ciencias Informáticas, lo cual constituye el objetivo fundamental que rigió esta investigación. Los principios que forman la propuesta de solución quedaron registrados bajo las siglas de NEPSE (Nuevo Enfoque para la producción de software educativo).
- La solución propuesta se ajusta a los estándares internacionales para el desarrollo de software.

RECOMENDACIONES

Tras realizar un análisis exhaustivo de un significativo número de elementos que contribuyeron a la elaboración de un conjunto de principios para guiar el proceso productivo de software educativo en la UCI, se hace necesario que se listen un grupo de recomendaciones que permitan la mejora continua de todo el proceso para desarrollar software de alta calidad.

- Capacitación de líderes e integrantes del equipo de trabajo de los proyectos de software educativo sobre todo lo relacionado con los modelos y metodologías de software, a fin de que posean una cultura general sobre el tema y tengan plena conciencia de la necesidad de su aplicación en la elaboración de un producto determinado.
- Profundizar en el estudio de las metodologías de desarrollo de software con el objetivo de plantear otros principios que no hayan sido comprendidos en la propuesta de solución y que puedan contribuir a la minimización de tiempos de desarrollo y aumento en la calidad del producto resultante.
- Analizar y aplicar consecuentemente los principios planteados en el NEPSE en un conjunto de proyectos de software educativo.
- Realizar un seguimiento del trabajo de estos proyectos bajo este nuevo enfoque y velar por los resultados de su aplicación.
- De resultar exitosa la aplicación de NEPSE en estos proyectos productivos, proponer su extensión a todos los proyectos dedicados a esta área en la Universidad.
- Estudiar la posible aplicación de este enfoque en otras empresas que se dediquen al desarrollo de productos educativos.
- Realizar extensión de NEPSE a artefactos y actividades en futuros trabajos.

REFERENCIAS BIBLIOGRÁFICAS

ÁLVAREZ, L. D. M. and L. K. C. RODRÍGUEZ, 2006. [http://www.monografias.com/trabajos31/software-educativo-cuba/software-educativo-cuba.shtml#softeducat]

BATES, T. *Como gestionar el cambio tecnológico. Estrategias para los responsables de centros universitarios*. Primera Edición. Editorial Gedisa. Madrid, España, 2001. p.

BLAYA,I.*Gestión por procesos*, 2006. [http://www.upm.es/innovacion/calidad/documentos/Gestion_Procesos.pp]

CATALDI, I. Z. *Metodología de diseño, desarrollo y evaluación de software educativo*, 2000. p.

Definición de proceso. [http://es.wikipedia.org/wiki/Reingenier%C3%ADa_de_procesos]

DÍAZ-ANTÓN, M. G.; M. A. PÉREZ*, et al. Propuesta de una metodología de desarrollo de software educativo bajo un enfoque de calidad sistémica*, Simón Bolívar, 2004. p. [http://www.infedu.coord.usb.ve/proyectos/proyecto3.html]

ESCOBEDO, M. A. R. B. *La administración de Proyectos*, 2007. [http://www.serempresario.com/muestra.ssp?id=104#]

FERNÁNDEZ, A. *El formador de Formación Profesional y Ocupacional*. Octaedro. Barcelona, 2000. p.

HERNÁN, S. M. *Diseño de una metodología Ágil para el desarrollo de software*. p. [http://64.233.167.104/search?q=cache:huU4z6XXR70J:www.acis.org.co/memorias/JornadasGerencia/IIJNGP/AgileManifesto.pdf+metodologias+pesadas&hl=es&ct=clnk&cd=1&gl=cu]

HEVIA, D. C. I. A. E. C. *EL PAPEL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES (TICs) EN EL PROCESO DE ENSEÑANZA APRENDIZAJE A COMIENZOS DEL SIGLO XXI*.

JACOBSON, I.; G. BOOCH*, et al. El Proceso Unificado de Desarrollo de Software*. Addison - Wesley (Edición en español por la Pearson Educación S.A. traducido de The Unified Software Development Process, 1999). Madrid, 2000. p.

KRUCHTEN, P. *A Rational Development Process*, 2004. [http://www.rational.com/media/whitepapers/xtalk.pdf]

LAMAS, M. R. R. *Introducción a la Informática Educativa* ISPJAE, Ciudad de la Habana, 2000.

LETELIER, P. and E. A. SÁNCHEZ *Metodologías Agiles en el Desarrollo de Software*, 2003.

MANCEBO, D. E. *Proyecto Docente*, 1999. [http://www.infor.uva.es/~descuder/docencia/pd/pd.html]

MARTÍNEZ, I. Y.; I. Y. PIÑERO*, et al. Sistema metodológico para el desarrollo del Software Educativo*. Ciudad de la Habana, Universidad de las Ciencias Informáticas, 2007. p.

PAGE-JONES, M. *Practical Project Management*, 1985.

PALACIO, J. *Gestión y procesos en empresas de software*, 2005. [http://www.navegapolis.net]

PIATTINI, M. *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión*, 1996.

PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edicion Mc Graw-Hill/Interamericana de España, S.A, 2002. p.

TOLEDO *La gestión por procesos*, 2002.

VALENCIA, D. D. S. I. Y. C. U. D. *Proceso de desarrollo de software, ?.*

VELA, L. J. D. J. R. *Administración de proyectos de desarrollo de sistemas de información*, 1997.

[http://www.monografias.com/trabajos15/sist-informacion/sist-informacion.shtml]

ZILBERSTEIN, J.; PORTELA*, et al. Didáctica Integradora de las Ciencias. Experiencia cubana*. en. Editorial Academia, Cuba. 1999. p.

BIBLIOGRAFÍA

AEDO, R. R. F.; P. M. S. GARCÍA*, et al. EL APRENDIZAJE CON EL USO DE LAS NUEVAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES.*

ÁLVAREZ, L. D. M. and L. K. C. RODRÍGUEZ, 2006. [http://www.monografias.com/trabajos31/software-educativo-cuba/software-educativo-cuba.shtml#softeducat]

ANDINO, L. M. R. *Estrategias metodológica para la integración de las TIC en el Proceso de Enseñanza –Aprendizaje de la Carrera de Economía*, 1996.

BATES, T. *Como gestionar el cambio tecnológico. Estrategias para los responsables de centros universitarios*. Primera Edición. Editorial Gedisa. Madrid, España, 2001. p.

Becco, P. G. R. *VIGOTSKY Y LAS TEORÍAS DEL APRENDIZAJE. CONCLUSIONES Y REFLEXIÓN FINAL*. [http://perso.wanadoo.es/angel.saez/pagina_nueva_165.htm]

BLAYA, I. *Gestión por procesos*, 2006. [http://www.upm.es/innovacion/calidad/documentos/Gestion_Procesos.pp]

CATALDI, I. Z. *Metodología de diseño, desarrollo y evaluación de software educativo*, 2000. p.

COMERCIALES, C. D. R. T. Y. *Norma Técnica Peruana NTP-ISO/IEC 12207*, 2006.

DÍAZ-ANTÓN, M. G.; M. A. PÉREZ*, et al. Propuesta de una metodología de desarrollo de software educativo bajo un enfoque de calidad sistémica*, Simón Bolívar, 2004. p. [http://www.infedu.coord.usb.ve/proyectos/proyecto3.html]

ESCOBEDO, M. A. R. B. La administración de Proyectos, 2007. [http://www.serempresario.com/muestra.ssp?id=104#]

FERNÁNDEZ, A. *El formador de Formación Profesional y Ocupacional*. Octaedro. Barcelona, 2000. p.
GRAELLS, D. P. M. *LAS TIC Y SUS APORTACIONES A LA SOCIEDAD*, 2000.

HERNÁN, S. M. *Diseño de una metodología Ágil para el desarrollo de software*. p. [http://64.233.167.104/search?q=cache:huU4z6XXR70J:www.acis.org.co/memorias/JornadasGerencia/IIJNGP/AgileManifesto.pdf+metodologias+pesadas&hl=es&ct=clnk&cd=1&gl=cu]

HEVIA, D. C. I. A. E. C. *EL PAPEL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES (TICs) EN EL PROCESO DE ENSEÑANZA APRENDIZAJE A COMIENZOS DEL SIGLO XXI*.

J. ZILBERSTEIN, P., MACPHERSON. Didáctica Integradora de las Ciencias. Experiencia cubana. en. Editorial Academia, Cuba. 1999.p.

JACOBSON, I.; G. BOOCH*, et al. El Proceso Unificado de Desarrollo de Software*. Addison - Wesley (Edición en español por la Pearson Educación S.A. traducido de The Unified Software Development Process, 1999). Madrid, 2000. p.

KRUCHTEN, P. *A Rational Development Process*, 2004.[http://www.rational.com/media/whitepapers/xtalk.pdf]

LAMAS, M. R. R. *Introducción a la Informática Educativa* ISPJAE, Ciudad de la Habana, 2000.

La informática como Recurso Pedagógico-Didáctico en la Educación, 1997. [http://www.monografias.com/trabajos10/recped/recped.shtml#pea]

LETELIER, P. and E. A. SÁNCHEZ Metodologías Ágiles en el Desarrollo de Software, 2003.

MANCEBO, D. E. *Proyecto Docente*, 1999. [http://www.infor.uva.es/~descuder/docencia/pd/pd.html]

MARTÍNEZ, I. Y.; I. Y. PIÑERO*, et al. Sistema metodológico para el desarrollo del Software Educativo*. Ciudad de la Habana, Universidad de las Ciencias Informáticas, 2007. p.

PAGE-JONES, M. Practical Project Management, 1985.

PALACIO, J. *Gestión y procesos en empresas de software*, 2005. [http://www.navegapolis.net]

PIATTINI, M. Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión, 1996.

PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edición Mc Graw-Hill/Interamericana de España, S.A., 2002. p.

Sanchez, M. A. M. *Metodologías De Desarrollo De Software*, 2004. [http://www.informatizate.net/MetodologíasDeDesarrolloDeSoftware.htm]

Tecnologías de la información y la comunicación, 2006. [http://es.wikipedia.org/w/index.php?title=Tecnolog%C3%ADas_de_la_informaci%C3%B3n&redirect=no]

TORRES, M. A. *Software Educativo*, 2003

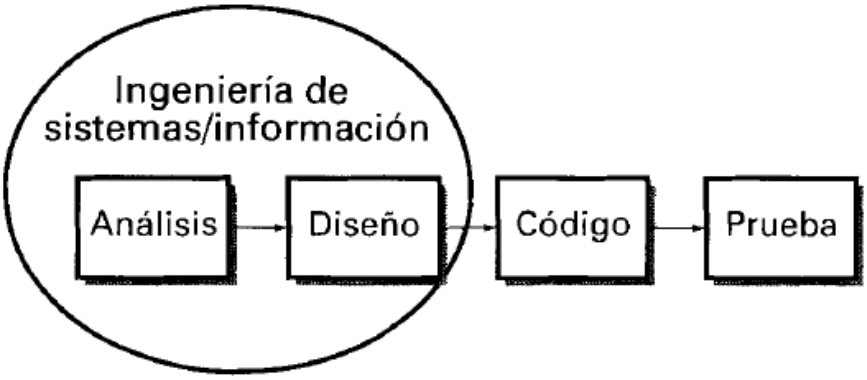
TOLEDO *La gestión por procesos*, 2002.

VALENCIA, D. D. S. I. Y. C. U. D. *Proceso de desarrollo de software.*

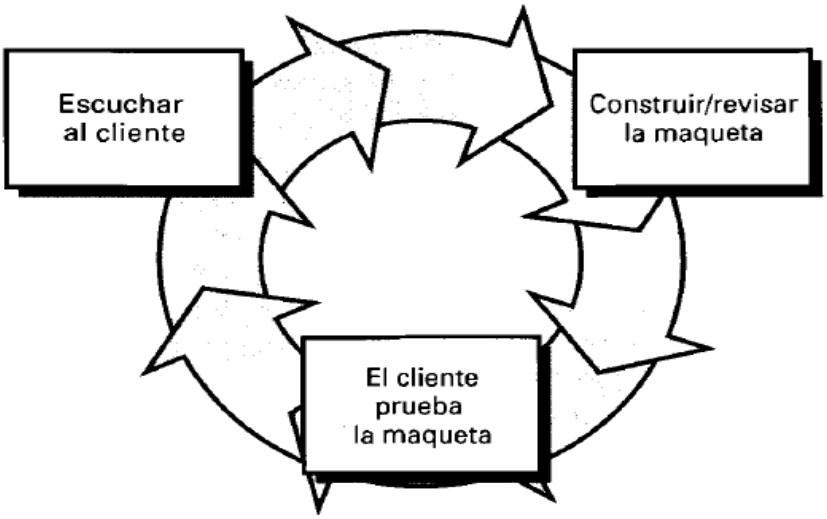
VELA, L. J. D. J. R. *Administración de proyectos de desarrollo de sistemas de información*, 1997. [http://www.monografias.com/trabajos15/sist-informacion/sist-informacion.shtml]

ANEXOS

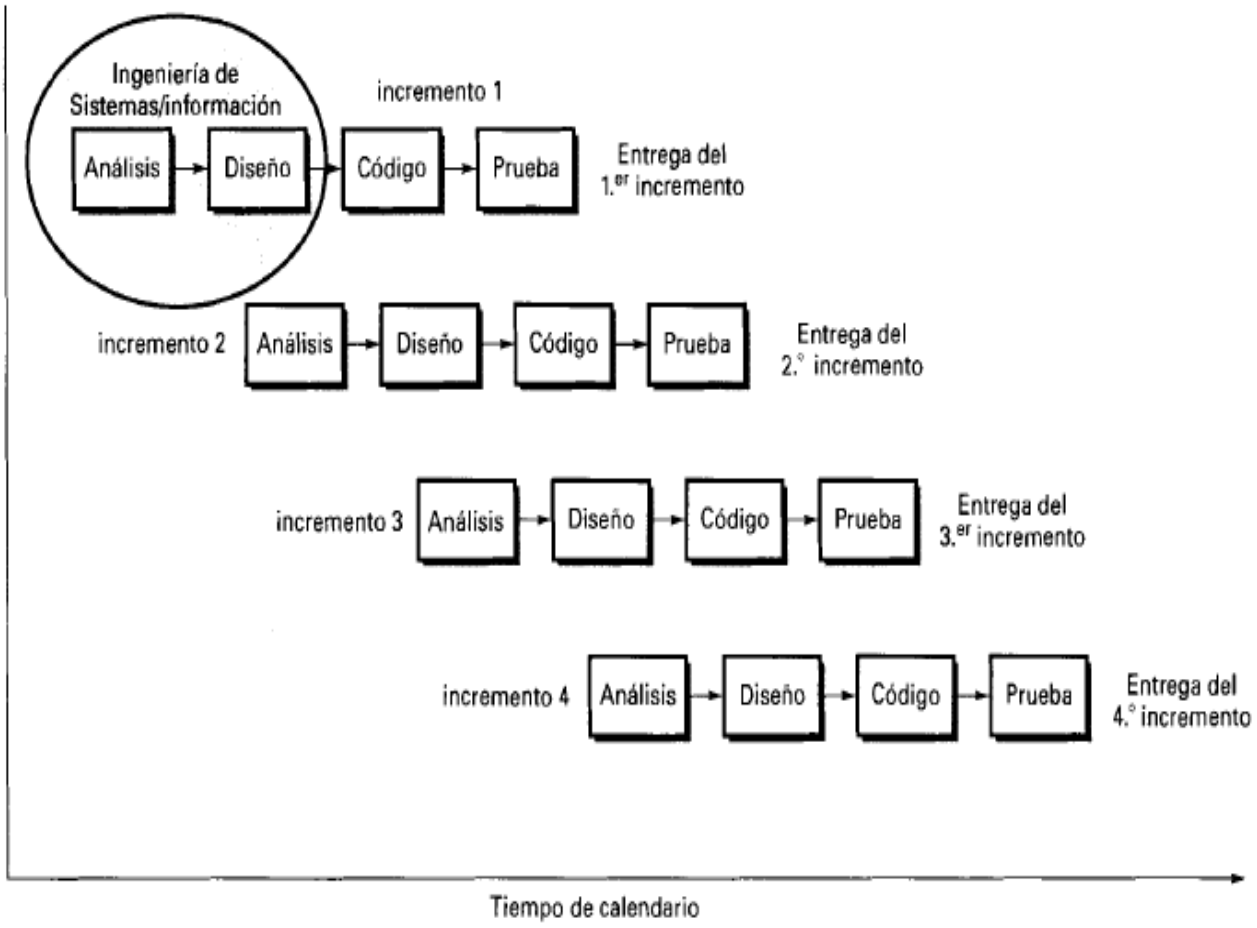
Anexo 1: Modelo lineal secuencial
(Tomado de: PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edición Mc Graw-Hill/Interamericana de España, S.A., 2002.)



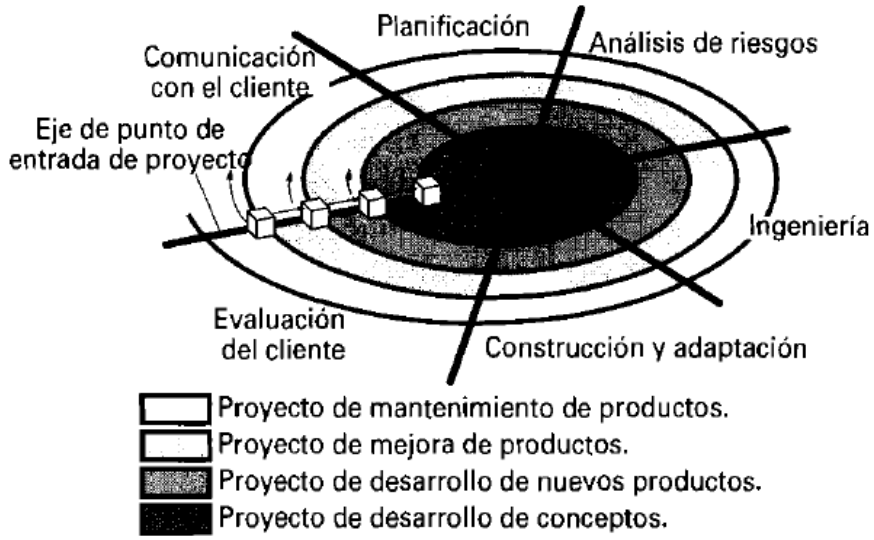
Anexo 2: Paradigma de construcción de prototipos
(Tomado de: PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edición Mc Graw-Hill/Interamericana de España, S.A., 2002.)



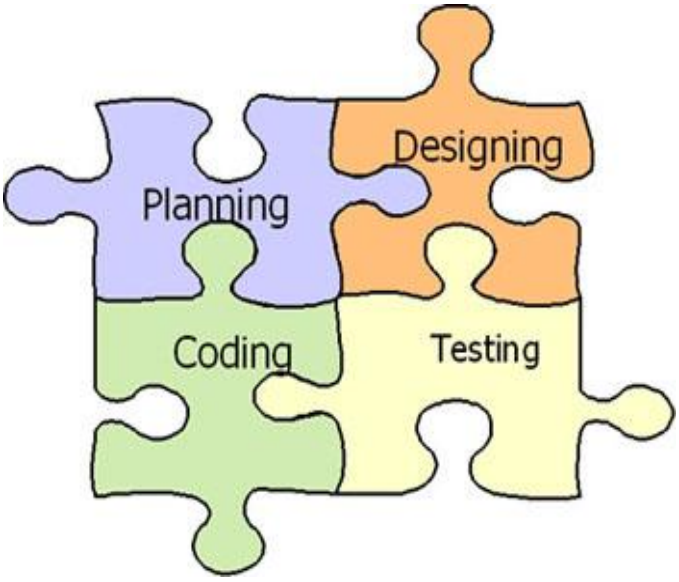
Anexo 3: Modelo incremental
(Tomado de: PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edición Mc Graw-Hill/Interamericana de España, S.A., 2002.)



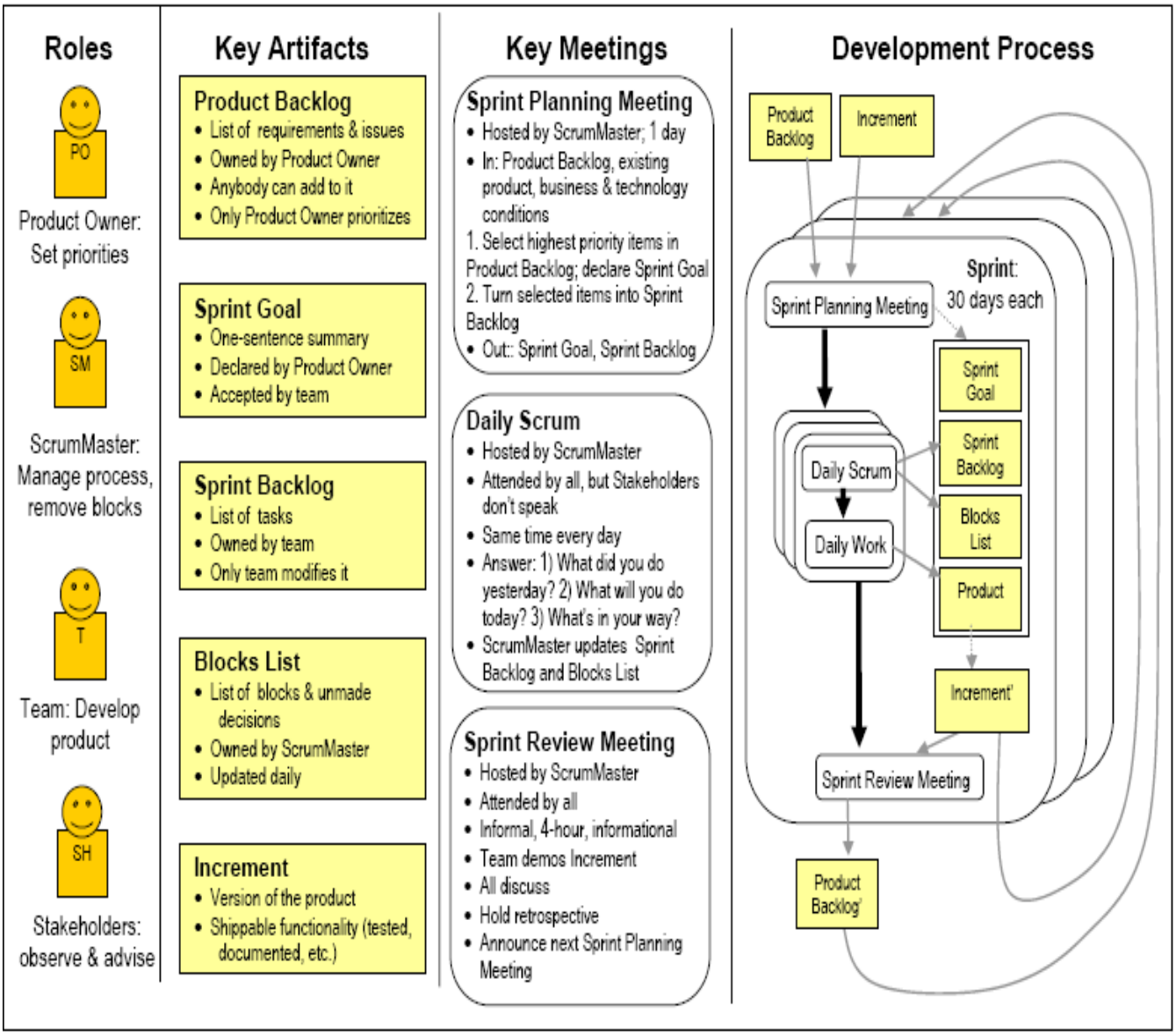
Anexo 4: Modelo espiral típico
(Tomado de: PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edición Mc Graw-Hill/Interamericana de España, S.A., 2002.)



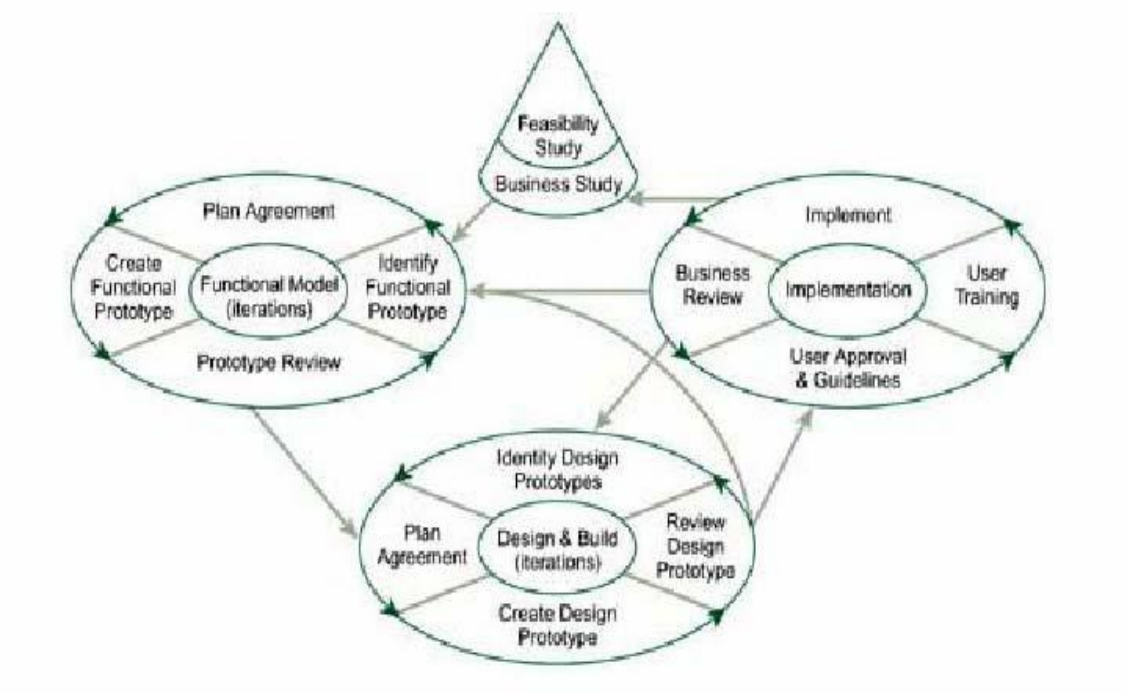
Anexo 5: Metodología Extreme Programming (XP).
(Tomado de: Sanchez, M. A. M.*Metodologías De Desarrollo De Software,* 2004)



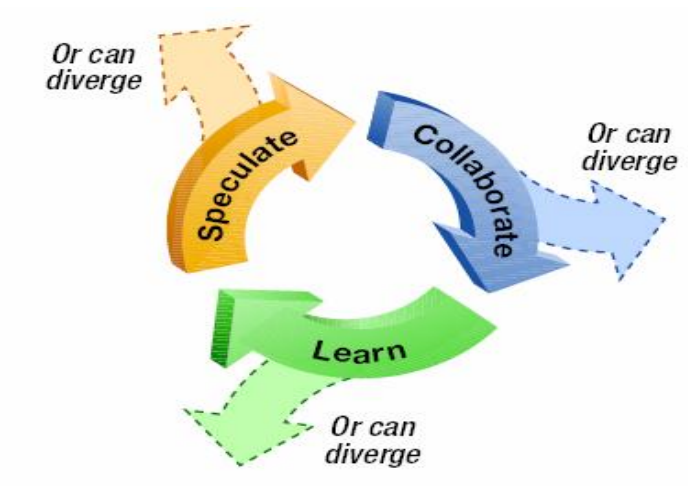
Anexo 6: Roles, artefactos, reuniones y proceso de desarrollo de SCRUM.
(Tomado de: HERNÁN, S. M. *Diseño de una metodología Ágil para el desarrollo de software*.)



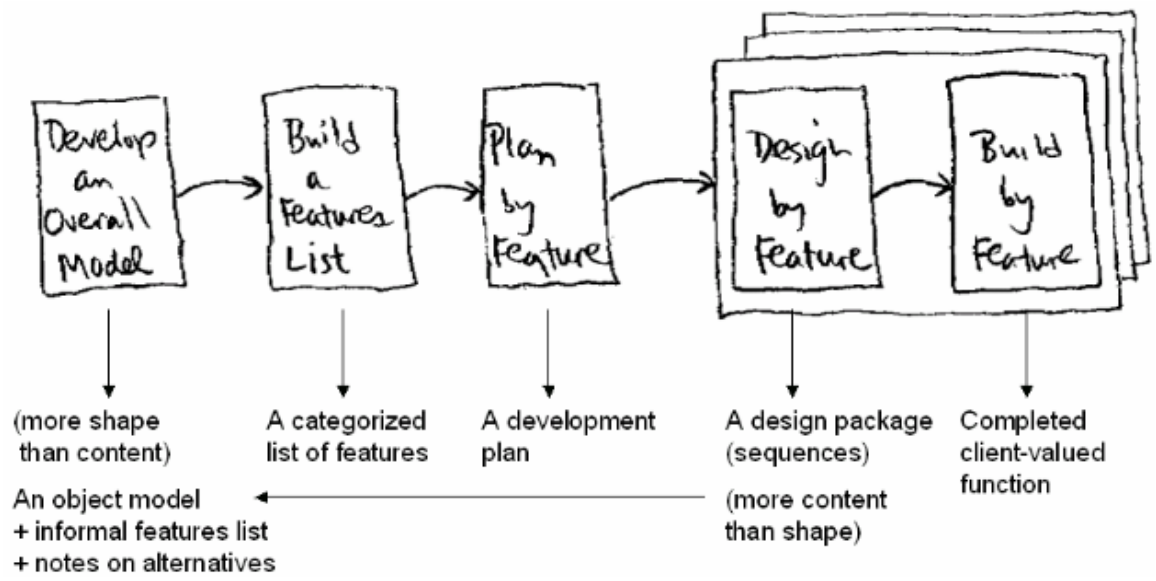
Anexo 7: Fases del proceso de desarrollo de DSDM.
(Tomado de: HERNÁN, S. M. *Diseño de una metodología Ágil para el desarrollo de software*.)



Anexo 8: Ciclo de vida adaptativo
(Tomado de: HERNÁN, S. M. *Diseño de una metodología Ágil para el desarrollo de software*.)



Anexo 9: Procesos de FDD
(Tomado de: HERNÁN, S. M. *Diseño de una metodología Ágil para el desarrollo de software*.)



Anexo 10: Encuesta realizada a líderes de proyecto

Datos Generales

Nombre del encuestado: _____

Nombre del proyecto: _____

1. ¿Utilizan alguna(s) de estas metodologías para el desarrollo de los software?

Rational Unified Process (RUP) ____ RUP (modificado) ____

Extreme Programming (XP) ____ Microsoft Solution Framework (MSF) ____

SCRUM ____ Feature - Driven Development (FDD) ____

Crystal Methodologies ____ Dynamic Systems Development Methods (DSDM) ____

Adaptive Software Development (ASD) ____

Nota: RUP (modificado) se refiere a la variante venezolana para desarrollar software educativo "Metodología de desarrollo de software educativo bajo un enfoque de calidad sistémica".

2. ¿Señale con una cruz los roles que tienen definidos para el desarrollo de un producto software?

Analista	
Diseñador	
Revisor técnico	
Revisor de proyecto	
Programador	
Arquitecto	
Administrador	
Ingeniero	
Revisor de contenido	
Especialista Metodológico	
Guionista de contenido	
Otros (Agregarlos debajo)	

3. ¿Se realiza la planificación basada en algún tiempo y estimación de costos?

Sí ____ No ____

4. ¿En qué medida se cumple la planificación realizada?

Siempre ____ Casi siempre ____ Poco ____

Nunca ____ Casi nunca ____

5. ¿De qué forma se controla el avance del proyecto?

Por el jefe ____ Reuniones Técnicas ____

Tormenta de ideas ____ Otras ____ ¿Cuál (es)? _____

6. Señale en qué etapa se presentan los principales problemas

Obtención de los requisitos de software ____ Análisis ____

Diseño del sistema de software ____ Implementación ____ Pruebas ____

Instalación ____ Mantenimiento ____ Actualización del sistema ____

Otras ____ ¿Cuál (es)? _____

Nota: Adicionar en una oración que cosa es cada uno de los flujos de trabajo.

7. ¿Se realiza la gestión de la propiedad intelectual?

Sí ____ No ____

8. ¿Utilizan métricas en el proceso de desarrollo del software educativo?

Sí ____ No ____

Anexo 11: Entrevista realizada a líderes de proyecto

1. ¿Cómo fueron seleccionados los estudiantes y/o profesores que pertenecen al proyecto?
2. ¿Los integrantes del proyecto tenían preparación previa, se encontraban familiarizados con el lenguaje de programación, las herramientas a utilizar para la elaboración del producto requerido?
3. ¿Una vez en el proyecto, cómo se gestiona la preparación de los integrantes del mismo?
4. ¿Cómo valorarías la preparación y motivación de los integrantes del proyecto?
5. ¿Tiene usted experiencia alguna dirigiendo proyectos de esta índole?
6. ¿Cómo fue seleccionado usted para ocupar el cargo de líder de proyecto?
7. ¿Cuáles son las tareas que usted debería cumplir como líder de proyecto?
8. ¿Cómo esta organizado el equipo de desarrollo de software?
9. ¿Cómo se comporta la comunicación entre los miembros del equipo?
10. ¿Al inicio del proyecto se realizan estimaciones cuantitativas para determinar el tiempo aproximado que les llevaría la construcción del producto?
11. ¿Se concibe un plan organizado para el desarrollo del proyecto?
12. ¿Desde un inicio de establece el ámbito del software?
13. ¿Para la elaboración del software como descomponen el problema?
14. ¿Se realizan estimaciones para determinar el costo del producto?
15. ¿Seleccionan algún modelo de proceso apropiado para la ingeniería del software que debe aplicar el equipo de proyecto?
16. ¿Sigue usted alguna metodología de desarrollo de software para la elaboración del producto?
17. ¿Cómo usted tiene organizado al equipo de desarrollo, Cuáles roles tiene definidos?
18. ¿Cómo se comporta el flujo de trabajo que ustedes siguen para la elaboración del producto?
19. ¿Se realiza al inicio del proyecto una gestión de riesgo del proyecto?
20. ¿Existe una comunicación fluida entre los desarrolladores y el cliente?

21. ¿Existe documentación alguna de lo que se va desarrollar en el proyecto?

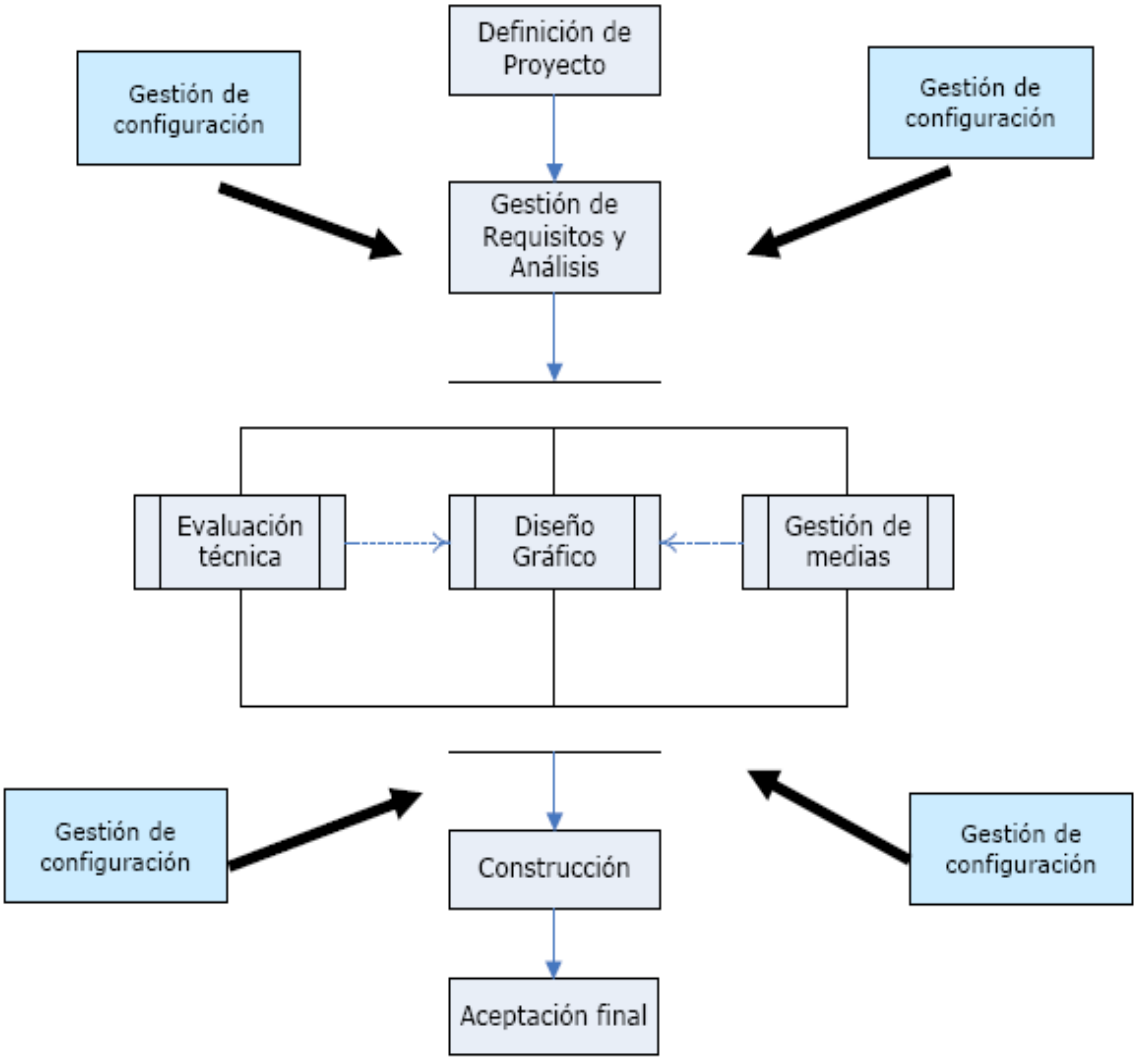
22. ¿Cómo se evalúa la calidad del producto?

23. ¿Cuáles son los principales problemas que se han presentado durante el desarrollo del producto?

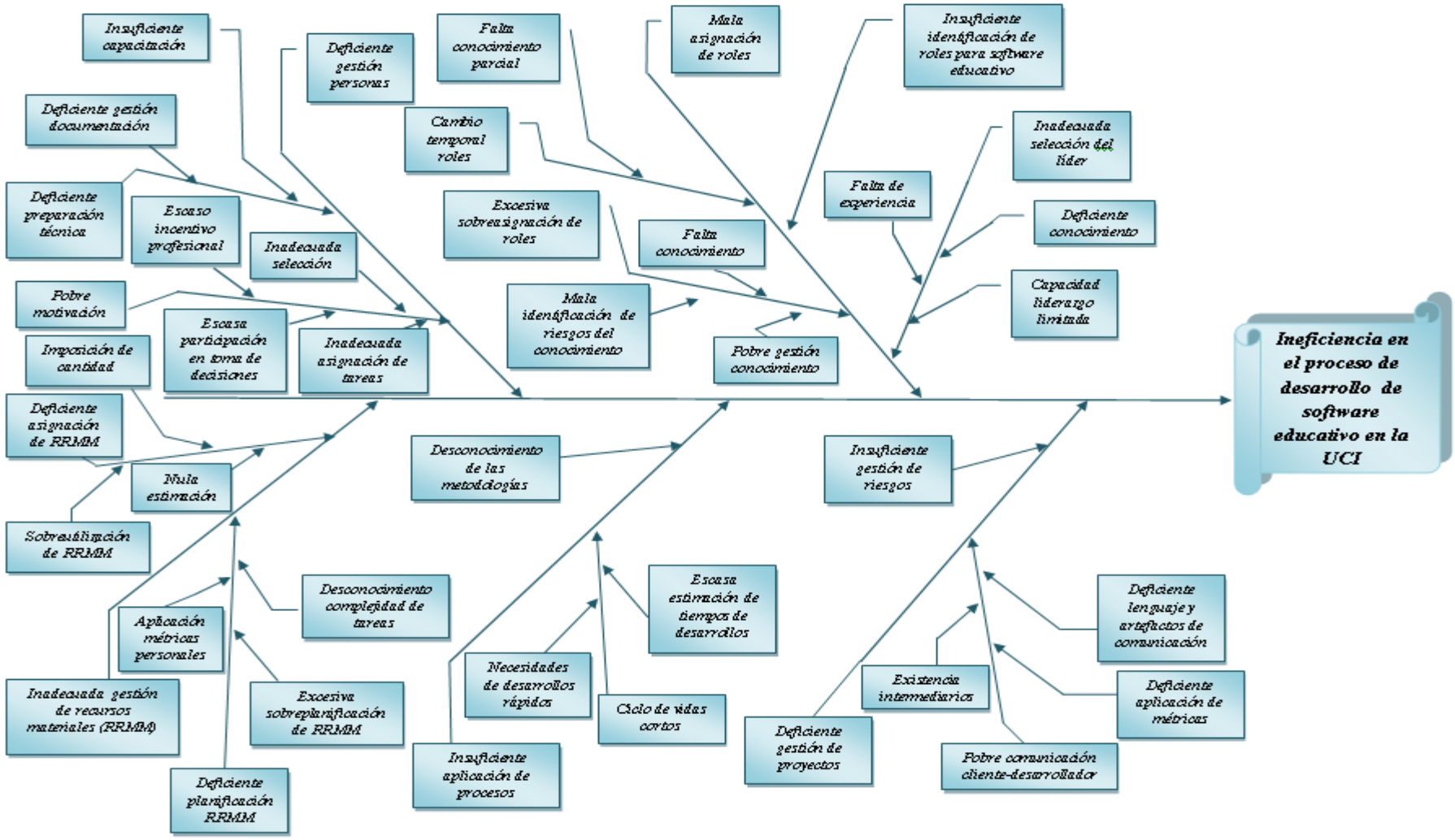
24. ¿Se le ha dado a conocer a usted la existencia de una metodología de desarrollo de software educativo que ha sido creado en la Universidad de las Ciencias Informáticas?

Anexo 12: Flujo general de trabajo definido en el "Sistema Metodológico para el desarrollo de software educativo" en la Universidad de las Ciencias Informáticas

(Tomado de: Piñeiro, Y. Martínez, Y. Lorente, A. López, D. *Sistema Metodológico para el desarrollo de software educativo Pedagógica.* Ciudad de la Habana, 2007)



Anexo 13: Diagrama Causa – Efecto



Anexo 14: Actividades definidas por Pressman para el desarrollo del software

Fase de definición

1. Planificación del proyecto del software

Gestión de proyecto

1. Selección de los jefes de equipo
2. Selección del equipo de software
3. Selección de técnicas para la coordinación de proyectos
4. Determinar ámbito del software
5. Descomponer el problema a resolver
    1. Funcionalidad que debe entregarse
    2. Proceso que se empleara para entregarlo
6. Estimación de la planificación temporal
7. Estimación de costo inicial
8. Selección de un modelo de proceso
9. Definición de un plan de proyecto preliminar
10. Descomposición del proceso
11. Comunicación con el cliente
12. Análisis de riesgo
13. Selección de métricas para el proyecto
14. Seguimiento y control del proyecto

Planificación de proyectos de Software

15. Estimación de recursos humanos, software reutilizables y herramientas hardware/software
16. Estimación de esfuerzo

Análisis y gestión de riesgo

Identificación de riesgo
Estimación de riesgo
Refinamiento de riesgo
Selección, supervisión y gestión de riesgo

2. Análisis de los requisitos

18. Identificación de requisito
20. Refinamiento del ámbito del software
21. Creación de prototipos de software
22. Especificación de los requisitos del software
23. Identificación de casos de uso y actores principales
24. Estructura del modelo de implementación
25. Creación del modelo de datos
26. Creación del modelo funcional
27. Modelado del comportamiento
28. Refinamiento de requisitos de sistema
29. Modelo de análisis
30. Creación del diagrama Entidad-Relación
31. Modelado de contenido de datos (Genera diccionario de datos)

3. Ingeniería de Sistemas o Información

32. Ingeniería de proceso o Negocio
    1. Análisis y diseño de tres arquitecturas
        - Arquitectura de datos
        - Arquitectura de Aplicación
        - Infraestructura de la tecnología

33. Ingeniería de producto
    1. Crear arquitectura e infraestructura

    Arquitectura: 4 componentes (software, hardware, datos (base de datos), personas)
    Infraestructura: Soporte (Incluye la tecnología requerida para unir componentes e información (Documentos, CD-ROM, videos))
34. Ingeniería de requisitos (Identificación, análisis y negociación, especificación, modelización, validación y gestión)
35. Modelado del sistema
    1. Interfaz de usuario
    2. Entrada
    3. Salida
    4. Mantenimiento y comprobación
36. Representación de los subsistemas
37. Elaboración de la especificación del sistema (Documento)

Fase de desarrollo

1. **Diseño del software**

Diseño de Arquitectura

38. Diseño de datos (traduce los objetos de datos definidos en el modelo de análisis en estructuras de datos o arquitectura para base de datos o almacén de datos)
40. Seleccionar estilo arquitectónico y patrones
41. Análisis de diseños arquitectónicos
42. Conversión de requisitos en una arquitectura de software
43. Análisis de transformación o de transacción de Diagrama de flujo de datos (DFD) a un estilo arquitectónico.
44. Refinamiento del diseño arquitectónico

45. Modelo de diseño

Diseño de Interfaz de usuario

46. Identificación de los requisitos de usuario, de las tareas y entorno
47. Análisis y modelado de tareas
48. Actividades del diseño de la interfaz (Se define conjunto de objetos y acciones de la interfaz: pantalla)
49. Creación de un prototipo a partir de la implementación del modelo de diseño
50. Evaluación del diseño

2. Generación de código

Diseño a nivel de componentes

51. Se realiza diseño a nivel de componentes a partir de las representaciones de diseño de datos, arquitectura e interfaz.

3. Pruebas del Software

Técnicas de pruebas del software

52. Determinación de la prueba.
53. Diseño de prueba.
54. Aplicación de pruebas de caja blanca.
55. Aplicación de pruebas de caja negra.
56. Recodificación de errores.

Fase de mantenimiento

57. Corrección de errores
58. Adaptación.

59. Mejoras
60. Prevención

Actividades de protección a lo largo del proceso

1. Seguimiento y control del proyecto
2. Revisiones técnicas formales
3. Garantía de calidad del software
4. Gestión de configuración del software
5. Preparación y producción de documentos
6. Gestión de reutilización
7. Mediciones
8. Gestión de riesgo

GLOSARIO

Actividad: Unidad tangible de trabajo realizada por un trabajador en un flujo de trabajo, de forma que implica una responsabilidad bien definida para el trabajador, produce un resultado bien definido basado en una entrada bien definida , y representa una unidad de trabajo con limites bien definidos. También puede verse como una ejecución de de una operación por un trabajador.

Administración: Conjunto ordenado y sistematizado de principios, técnicas y prácticas que tiene como finalidad apoyar la consecución de los objetivos de una organización a través de la provisión de los medios necesarios para obtener los resultados con la mayor eficiencia, eficacia y congruencia; así como la óptima coordinación y aprovechamiento del personal y los recursos técnicos, materiales y financieros.

Análisis (flujo de trabajo): Flujo de trabajo fundamental cuyo propósito principal es analizar los requisitos descritos en la primera captura de requisitos, mediante su refinamiento y estructuración.

Arquitectura: Conjunto de decisiones significativas acerca de la organización de un sistema software. La arquitectura no solo se interesa por la estructura y el comportamiento, sino también por las restricciones y compromisos de uso, funcionalidad, funcionamiento, flexibilidad al cambio, reutilización, compresión, economía y tecnología, así como por aspectos estéticos del software.

Artefacto: Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades. Un artefacto pude ser un modelo, un elemento de un modelo o un documento.

Caso de prueba: Especificación de un caso para probar el sistema, incluyendo qué probar, con qué entradas y resultados y bajo qué condiciones.

Ciclo de vida del software: Ciclo que cubre 4 fases por las que transita un software.

Cliente: Persona, organización o grupo de personas que encarga la construcción de un sistema, ya sea empezando desde cero, o mediante el refinamiento de versiones sucesivas.

Construcción: Versión ejecutable del sistema, por lo general, una parte especifica del mismo. El desarrollo transcurre a través de una sucesión de construcciones.

Diseño (flujo de trabajo): Flujo de trabajo fundamental cuyo propósito principal es el de formular modelos que se centran en los requisitos no funcionales y el dominio de la solución, y que prepara para la implementación y pruebas del sistema

Fase: Periodo de tiempo entre dos hitos principales de un proceso de desarrollo.

Fase de construcción: Tercera fase del ciclo de vida del software, en la que el software es desarrollado a partir de una línea base de la arquitectura ejecutable, hasta el punto en el que esta listo para ser transmitido a la comunidad de usuarios.

Fase de elaboración: Segunda fase del ciclo de vida en la que se define la arquitectura.

Fase de inicio: Primera fase del ciclo de vida del software.

Fase de transición: Cuarta fase del ciclo de vida del software, en la que este es puesto en manos de la comunidad de usuarios.

Flujo de trabajo: Realización de un caso de uso o parte de él. Puede describirse en términos de diagrama de actividad, que incluye a los trabajadores participantes, las actividades que realizan y los artefactos que producen.

Gestión de la configuración: Tarea de definir y mantener las configuraciones y versiones de los artefactos. Esto incluye la definición de la línea base, el control de versiones de esta y el control del almacenamiento de los artefactos.

Dirigido por caso de uso: En el contexto del ciclo de vida del software, indica que los casos de uso se utilizan como artefacto principal para definir el comportamiento deseado para el sistema, y para comunicar este comportamiento entre las personas involucradas en el sistema.

Implementación (flujo de trabajo): Flujo de trabajo fundamental cuyo propósito esencial es implementar el sistema en términos de componentes, es decir, código fuente, guiones, ficheros binarios, ejecutables etc.

Iteración: Conjunto de actividades llevadas a cabo de acuerdo a un plan (de iteración) y unos criterios de evaluación, que lleva a producir una versión ya sea interna o externa.

Iterativo: En el contexto del ciclo de vida del software, proceso que implica la gestión de una serie de versiones ejecutables.

Prototipo evolutivo: Prototipo que evoluciona y es refinado para convertirse finalmente en parte del sistema en desarrollo. Prototipo que es candidato a ser sujeto de la gestión de configuraciones.

Proyecto: Esfuerzo de desarrollo para llevar un sistema a lo largo de un ciclo de vida.

Requisito: Condición o capacidad que debe cumplir un sistema.

Requisito funcional: Requisito que especifica una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas; requisito que específica comportamiento de entrada/salida de un sistema.

Requisito no funcional: Requisito que especifica propiedades del sistema, como restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma, mantenibilidad, extensibilidad o fiabilidad. Requisito que especifica restricciones físicas sobre un requisito funcional

Riesgo: Variable de un proyecto que pone en peligro o impide su éxito. Los riesgos pueden consistir en que un proyecto experimente sucesos no deseados, como retrasos en la programación, desviaciones de costo o una cancelación definitiva.

Usuario: Individuo u organización que interactúa con un sistema

Versión: Conjunto de artefactos relativamente completo y consistente, que incluye posiblemente una construcción, entregado a un usuario interno o externo; entrega de tal conjunto.