UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 6

Título: “Adaptación de OpenUp/Basic para el Polo Productivo de BioInformática”

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autores:

Rolando Gutiérrez Rosales

Dayana Cabrera Fleites

Tutores:

Ing. Alieski Sarmiento Almenares

Ing. Vilmavis la Rosa Sordo

Junio, 2009

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los _____ días del mes de _____ del año 2009.

_____ _____

Rolando Gutiérrez Rosales Dayana Cabrera Fleites

_____ _____

Ing. Alieski Sarmiento Almenares Ing. Vilmavis la Rosa Sordo

Datos de Contacto

Tutor

Ing. Alieski Sarmiento Almenares

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba.

asarmiento@uci.cu

Tutor

Ing. Vilmavis la Rosa Sordo

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba.

vlarosa@uci.cu

Agradecimientos

A quien con su experiencia nos guió, orientó y aconsejó durante el desarrollo de este trabajo de diploma, quien fue partícipe de cada logro y compartió cada desacierto, quien nos enseño a ser creativos e independientes y enfrentar nuestro trabajo con seriedad y responsabilidad: ***A nuestro tutor y amigo, Ing. Alieski Sarmiento Almenares.***

A quienes se interesaron y nos ayudaron en todo momento, con sus opiniones, consejos y críticas precisas para poder seguir adelante: ***A mis tíos, Manolito y Gaile y a mi amigo Jorge Quiala.***

A quienes siempre han confiado en nosotros, nos han apoyado, nos han dado ánimo y la fuerza necesaria para seguir tras cada obstáculo, tras cada desacierto. A quienes lloran tras cada caída nuestra y se regocijan con cada paso, con cada logro: ***A nuestros queridísimos Padres.***

A quien ha soportado mi mal genio y ha sabido comprenderme, a quien ha escuchado atentamente cuando las cosas no han ido bien, a quien me ha brindado su apoyo incondicional: ***A mi novia Denise.***

A quienes siempre han estado apoyando en cada momento y han hecho que nuestra vida de universitarios quede para siempre en nuestra memoria como una de las etapas que nunca olvidaremos. ***A nuestros compañeros de curso y amigos de la universidad.***

Dedicatoria

Rolando

- *A mi mamá, Marta B. Rosales Lameiro por darme la educación que me ha permitido llegar hasta aquí, por formarme como un hombre de bien, por sus mimos y sus desvelos y por todos los sacrificios que ha tenido que hacer para que nunca me faltara nada y pudiera estudiar siempre sin ninguna preocupación.*
- *A mi Papá, Benito R. Gutiérrez Martínez porque me ha dado el ejemplo de persona que quiero llegar a ser, porque aun estando lejos, nunca ha dejado de preocuparse y de ayudarme en mi carrera, en mis problemas… en mi vida.*
- *A mis tías Mayda y Lourdes, que más que tías, ha sabido ser madre y padre cada vez que las he necesitado.*
- *A mis hermanas Arlet y Daylet, porque con cada sonrisa me dan fuerzas para continuar y me hacen ser mejor ser humano con la esperanza de poder ser su guía, su ejemplo.*
- *A mi novia Denise que se ha pasado toda la carrera a mi lado, por amarme y malcriarme todo este tiempo y por esa paciencia infinita que solo ella posee.*
- *A mis Abuelas que me han llenado de mimos unas veces y otras muchas de enseñanzas.*
- *A Ahmed, porque me ha ayudado a hacerme más fuerte como hombre y ser humano.*

Dedicatoria

Dayana

- *A mis padres Moisés Cabrera García y Aracelis Fleites Fernández por haberme dado la oportunidad de estudiar y formarme en esta escuela del futuro, por haberme apoyado siempre en cada decisión por descabellada que fuese.*
- *A mi hermana Daymara Cabrera Fleites y su esposo Onenci Vega por quererme tanto y hacer que mi vida sea más alegre por contar con ellos.*
- *A mi novio Jianny Hernández por adorarme y hacerme sentir tan enamorada.*

Resumen

El Polo Productivo BioInformática (PPB) necesita adoptar una metodología capaz de adaptarse a las características propias de sus proyectos y cumplir con los lineamentos de calidad que establece la Universidad de las Ciencias Informáticas (UCI). Después de estudiar las características de los procesos de desarrollo de software y hacer un balance comparativo entre las metodologías más utilizadas, el PPB implantó la metodología OpenUp/Basic. Es una metodología ágil, diseñada para proyectos de corta duración y equipos pequeños que puede ser adaptable y extensible a cualquier entorno de desarrollo. Se pretende ampliar dicha metodología de forma tal que mantenga el enfoque de desarrollo ágil y cumpla con los lineamientos de calidad que exige la UCI. Para lograr el objetivo propuesto se realizaron una serie de cambios a las actividades que plantea OpenUp/Basic, con lo que se modificó el expediente de proyecto que propone la UCI, generando así, un nuevo expediente que recoge la información necesaria para mantener documentado el proceso de desarrollo y agilizar el trabajo de dichos proyectos productivos. Este expediente de proyecto fue presentado, junto a la nueva versión de la metodología, nombrada OpenUp/BioInfo, a los principales líderes de los proyectos del PPB y asesores de calidad de la facultad y del Departamento Central de Calidad de la Universidad de las Ciencias Informáticas para medir el nivel de aceptación de la propuesta.

Palabras Claves:

Metodología
OpenUp/Basic
Expediente de Proyecto
CMMI
Proceso de Desarrollo de Software
Polo Productivo BioInformática

Índice

Índice de figuras

Índice de tablas

Introducción

La industria de software crece y se desarrolla a ritmo vertiginoso. Las grandes empresas productoras de software potencian el uso de tecnologías flexibles y adaptables para su desarrollo, donde cada producto posee características propias que lo distinguen del resto. El tiempo de desarrollo, los recursos humanos, las tecnologías, los estándares de calidad, los clientes, y el producto a obtener, son factores influyentes en la selección de las pautas de desarrollo y la metodología o proceso de desarrollo de software.

La Universidad de Ciencias Informáticas (UCI) y la Facultad 6 dentro de ésta, tiene, como proceso fundamental, además de preparar docentemente a sus estudiantes y formarlos para que lleguen a ser Ingenieros Informáticos de excelencia, la producción de software de calidad con pautas o lineamientos bien definidos. Para esto la Facultad se divide en polos productivos que a su vez cuentan con equipos de desarrollo. Los proyectos que se desarrollan en cada polo poseen características propias, por lo que es muy difícil coincidir en que la misma metodología que se emplea para el desarrollo de software en uno, puede ser aplicada en todos con resultados favorables.

La UCI cuenta con una Dirección de Calidad que establece los principios y normas, en un documento nombrado Lineamientos de Calidad de Software (Calisoft, 2008) que deben cumplir todos los proyectos para lograr un producto de calidad. Estos lineamientos abarcan todos los procesos de soporte, gestión e ingeniería que ocurren durante el desarrollo de los proyectos productivos y establecen claramente que debe documentarse todo el proyecto hasta que deriva en producto y es aquí donde la metodología de desarrollo juega un papel esencial sobre todo en la parte ingenieril, pues dice qué se hace, quién lo hace, cómo y cuándo. Esta documentación se recoge en lo que se llama Expediente de Proyecto. Existen muchas metodologías, y cada una de ellas propone una forma particular de documentar.

El PPB está compuesto por proyectos que están a su vez divididos por módulos o mini proyectos con un reducido equipo de desarrollo que tiene como prioridad la entrega de los productos en el menor período de tiempo y con la menor documentación posible y necesita adoptar una metodología que cumpla con los lineamientos de calidad de software, pero que se adapte a las condiciones y características propias del mismo.

Por todo lo anterior se identifica como **problema científico:** ¿Cómo contribuir a que el proceso de desarrollo de software de los proyectos del PPB se ajuste a las características del entorno, se base en los estándares de calidad de la Universidad de las Ciencias Informáticas (UCI) y aproveche

herramientas de configuración de las metodologías de desarrollo? De este modo se define como **objeto de estudio:** Procesos de desarrollo de Software, y como **Campo de acción:** Metodologías de desarrollo de software ágiles basadas en estándares de calidad.

De ahí que el **objetivo general** de la investigación sea: adaptar una metodología de desarrollo de software a las necesidades del PPB.

Se definen los **objetivos específicos**:

- Fundamentar la metodología como la propicia para el desarrollo de software del PPB.
- Adaptar la metodología definiendo trabajadores, actividades y artefactos que cumplan con los estándares de CMMI nivel 2 y con los Lineamientos Mínimos de Calidad de la UCI.
- Montar la metodología adaptada en el Eclipse Process Framework (EPF).

Para resolver el problema planteado y cumplir así los objetivos, se definen las siguientes **tareas:**

- Análisis de las características de los proyectos de software de PPB.
- Análisis de las metodologías de desarrollo de software.
- Análisis de estándares de calidad para procesos de desarrollo de software.
- Definición de artefactos, roles y actividades.
- Análisis del Eclipse Process Framework Composer (EPF).
- Montaje de la propuesta en el EPF.

El documento está estructurado por dos capítulos:

CAPITULO 1. Revisión Bibliográfica.

En el transcurso de este capítulo se exponen las características generales de los procesos de desarrollo de software adentrándose en los modelos genéricos que detallan distintos modos de abordar dicho proceso. Se realiza un estudio de los estándares de calidad vigentes en la actualidad, y finalmente se fundamenta la Metodología OpenUp/Basic partiendo del análisis de las metodologías de desarrollo de software y profundizando en el estudio y comparaciones de las características específicas de los dos enfoques existentes, dígase metodologías ágiles y tradicionales.

CAPÍTULO 2. Programa y Metodología.

En este capítulo se detalla la propuesta para el perfeccionamiento del proceso de desarrollo de software del PPB. Se describen las actividades que propone la metodología utilizada actualmente por el PPB y el expediente de proyecto. Se realizan las modificaciones a la metodología obteniendo un nuevo expediente de proyecto. Se especifica el ciclo de vida a utilizar con las respectivas etapas y disciplinas, los roles, artefactos y actividades que se utilizarán para la obtención de un producto software con calidad y se detalla el montaje de la nueva propuesta en el Eclipse process Framework Composer (EPF Composer).

Capítulo 1. Revisión Bibliográfica.

Introducción.

En el transcurso de este capítulo se realizará una descripción detallada de los procesos de desarrollo de software, para lo que se profundiza en el estudio de varios modelos y metodologías para el desarrollo del software. Se analizan los estándares de calidad vigentes en la actualidad, se fundamenta el uso de la Metodología OpenUp/Basic y se analizan las tendencias actuales de los procesos de desarrollo de Software y las metodologías, que ayudarán a un mejor entendimiento y comprensión de la propuesta de solución.

1.1 Objeto de estudio.

A lo largo de los años, el desarrollo de los proyectos de software causa bastantes confusiones y malas interpretaciones en los requerimientos de los clientes y usuarios, en parte, debido a la abundancia de notaciones, metodologías y conceptos que hace que los desarrolladores de sistemas no se pongan de acuerdo en qué metodología usar o qué es lo que realmente están elaborando.

La Ingeniería de Software ha sido definida como: “La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software; es decir, la aplicación de ingeniería al software” [1].

En cualquier proyecto de ingeniería se requieren de etapas de modelado que permitan experimentar y visualizar el sistema que se construirá.

Entre los principios de modelado se encuentran:

- La forma en que se ve el problema tiene una profunda influencia en el modo de como acometemos el problema y le damos solución al mismo.
- Para modelar un sistema complejo no es suficiente un único modelo, se requieren múltiples modelos donde cada uno representa una vista (aspecto) del sistema; estos modelos se complementan entre sí.
- Cualquier modelo puede ser representado con diferentes grados de precisión.
- Los mejores modelos están ligados a la realidad. [2]

Con el objetivo de hacer más comprensible el fruto de los resultados de la investigación, se propone a continuación la descripción del Objeto de estudio.

1.1.1 Características de los Procesos de Desarrollo de Software

Un Proceso de desarrollo de Software (PDS) es “un conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software” [3]. Tiene como propósito la producción eficaz y eficiente de un producto software que cumpla los requisitos del cliente. Este proceso es intensamente intelectual, afectado por la creatividad y juicio de las personas involucradas. Aunque un proyecto de desarrollo de software es equiparable en muchos aspectos a cualquier otro proyecto de ingeniería, en el desarrollo de software hay una serie de desafíos adicionales, relativos esencialmente a la naturaleza del producto obtenido. [4]

Un producto software en sí es complejo, es prácticamente inviable conseguir un 100% de confiabilidad de un programa por pequeño que sea. Existe una inmensa combinación de factores que impiden una verificación exhaustiva de todas las posibles situaciones de ejecución que se puedan presentar (entradas, valores de variables, datos almacenados, software del sistema, otras aplicaciones que intervienen, el hardware sobre el cual se ejecuta, etc.). Además es intangible y por lo general muy abstracto, esto dificulta la definición del producto y sus requisitos, sobre todo cuando no se tienen precedentes en productos software similar. Esto hace que los requisitos sean difíciles de consolidar tempranamente. Así, los cambios en los requisitos son inevitables, no sólo después de entregado el producto sino también durante el proceso de desarrollo.[4]

El proceso de desarrollo de software no es único. No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo. Debido a esta diversidad, es difícil automatizar todo un proceso de desarrollo de software.

A pesar de la variedad de propuestas de procesos de desarrollo de software, existe un conjunto de actividades fundamentales que se encuentran presentes en todos ellos:

Especificación de software: se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.

Diseño e Implementación: se diseña y construye el software de acuerdo a la especificación.

Validación: el software debe validarse, para asegurar que cumpla con lo que quiere el cliente.

Evolución: el software debe evolucionar, para adaptarse a las necesidades del cliente.

Para determinar los elementos del Proceso de Desarrollo del Software se establecen las relaciones entre elementos que permitan responder **Quién** debe hacer **Qué**, **Cuándo** y **Cómo** debe hacerlo. [4]

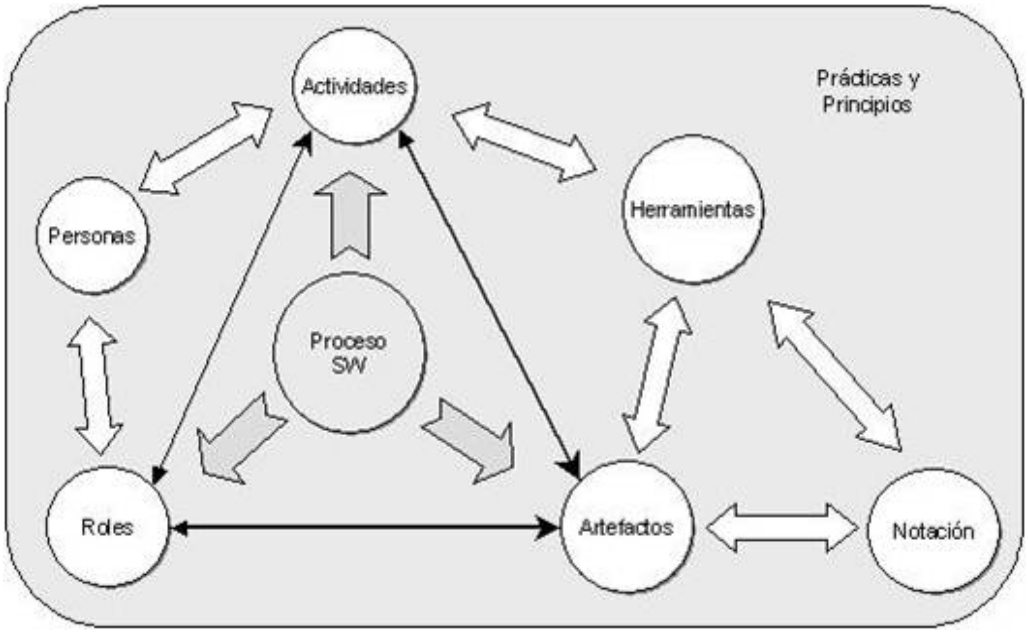


Figura 1 Elementos del Proceso de Desarrollo de Software

Quién: Las personas participantes en el proyecto de desarrollo desempeñando uno o más roles específicos.

Qué: Un artefacto es producido por un rol en una de sus actividades. Los artefactos se especifican utilizando notaciones específicas. Las herramientas apoyan la elaboración de artefactos soportando ciertas notaciones.

Cómo y Cuándo: Las actividades son una serie de pasos que lleva a cabo un rol durante el proceso de desarrollo. El avance del proyecto está controlado mediante hitos que establecen un determinado estado de terminación de ciertos artefactos. [4]

1.1.2 Modelos de desarrollo del software

Existen diversos modelos de desarrollo del Software: Los modelos genéricos no son descripciones definitivas de procesos de software; sin embargo, son abstracciones útiles que pueden ser utilizadas para explicar diferentes enfoques del desarrollo de software.

A continuación se nombran algunos modelos de desarrollo del software:

- Codificar y corregir
- Modelo en cascada
- Desarrollo evolutivo

- Desarrollo formal de sistemas
- Desarrollo basado en reutilización
- Desarrollo incremental
- Desarrollo en espiral[4]

Modelo de desarrollo en cascada

El primer modelo de desarrollo de software que se publicó se derivó de otros procesos de ingeniería. Este toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y las representa como fases separadas del proceso.

El modelo en cascada consta de las siguientes fases:

- **Definición de los requisitos:** los servicios, restricciones y objetivos son establecidos con los usuarios del sistema. Se busca hacer esta definición en detalle.
- **Diseño de software:** se fracciona el sistema en sistemas de software o hardware. Se establece la arquitectura total del sistema. Se identifican y describen las abstracciones y relaciones de los componentes del sistema.
- **Implementación y pruebas unitarias:** Construcción de los módulos y unidades de software. Se realizan pruebas de cada unidad.
- **Integración y pruebas del sistema:** se integran todas las unidades. Se prueban en conjunto. Se entrega el conjunto probado al cliente.
- **Operación y mantenimiento:** generalmente es la fase más larga. El sistema es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación. Se identifican nuevos requisitos.

Cada fase tiene como resultado documentos que deben ser aprobados por el usuario. Una fase no comienza hasta que termine la fase anterior y generalmente se incluye la corrección de los problemas encontrados en fases previas. [4]

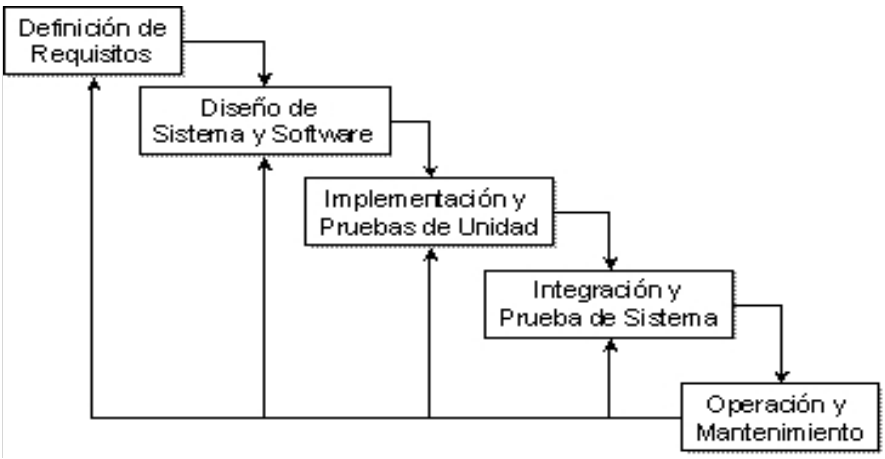


Figura 2 Modelo de desarrollo en cascada

En la práctica, este modelo no es lineal, e involucra varias iteraciones e interacción entre las distintas fases de desarrollo. Algunos problemas que se observan en el modelo de cascada son:

- Las iteraciones son costosas e implican rehacer trabajo debido a la producción y aprobación de documentos.
- Aunque son pocas iteraciones, es normal congelar parte del desarrollo y continuar con las siguientes fases.
- Los problemas se dejan para su posterior resolución, lo que lleva a que estos sean ignorados o corregidos de una forma poco elegante.
- Existe una alta probabilidad de que el software no cumpla con los requisitos del usuario por el largo tiempo de entrega del producto.
- Es inflexible a la hora de evolucionar para incorporar nuevos requisitos. Es difícil responder a cambios en los requisitos.
- Este modelo sólo debe usarse si se entienden a plenitud los requisitos. Aún se utiliza como parte de proyectos grandes. [4]

Desarrollo evolutivo

La idea detrás de este modelo es el desarrollo de una implantación del sistema inicial, exponerla a los comentarios del usuario, refinarla en N versiones hasta que se desarrolle el sistema adecuado. En la

figura 3 se observa cómo las actividades concurrentes: especificación, desarrollo y validación, se realizan durante el desarrollo de las versiones hasta llegar al producto final.

Una ventaja de este modelo es que se obtiene una rápida retroalimentación del usuario, ya que las actividades de especificación, desarrollo y pruebas se ejecutan en cada iteración. [4]

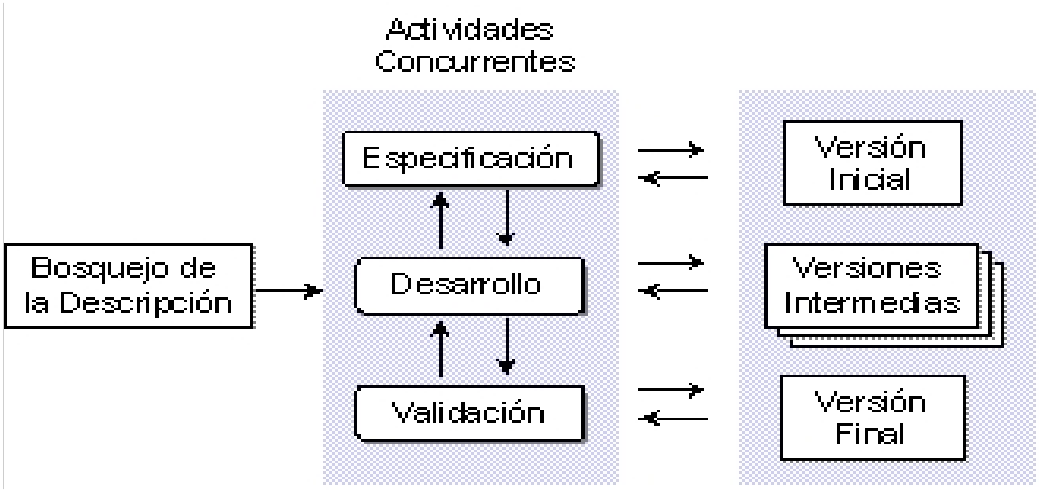


Figura 3 Modelo de desarrollo evolutivo

Existen dos tipos de desarrollo evolutivo:

Desarrollo exploratorio: el objetivo de este enfoque es explorar con el usuario los requisitos hasta llegar a un sistema final. El desarrollo comienza con las partes que se tiene más claras. El sistema evoluciona conforme se añaden nuevas características propuestas por el usuario.

Enfoque utilizando prototipos: el objetivo es entender los requisitos del usuario y trabajar para mejorar la calidad de los mismos. A diferencia del desarrollo exploratorio, se comienza por definir los requisitos que no están claros para el usuario y se utiliza un prototipo para experimentar con ellos. El prototipo ayuda a terminar de definir estos requisitos. [4]

Entre los puntos favorables de este modelo están:

- La especificación puede desarrollarse de forma creciente.
- Los usuarios y desarrolladores logran un mejor entendimiento del sistema. Esto se refleja en una mejora de la calidad del software.

Es más efectivo que el modelo de cascada, ya que cumple con las necesidades inmediatas del cliente. Desde una perspectiva de ingeniería y administración se identifican los siguientes problemas:

- **Proceso no Visible:** los administradores necesitan entregas para medir el progreso. Si se necesita desarrollar rápido el sistema, no es efectivo producir documentos que reflejen cada versión del sistema.
- **Sistemas pobremente estructurados:** los cambios continuos pueden ser perjudiciales para la estructura del software haciendo costoso el mantenimiento.
- **Se requieren técnicas y herramientas:** para el rápido desarrollo se necesitan herramientas que pueden ser incompatibles con otras o que poca gente sabe utilizar.[4]

Desarrollo basado en reutilización

Como su nombre lo indica, es un modelo fuertemente orientado a la reutilización. Este modelo consta de 4 fases. A continuación se describe cada fase:

Análisis de componentes: Se determina qué componentes pueden ser utilizados para el sistema en cuestión. Casi siempre hay que hacer ajustes para adecuarlos.

Modificación de requisitos: Se adaptan (en lo posible) los requisitos para concordar con los componentes de la etapa anterior. Si no se pueden realizar modificaciones en los requisitos, hay que seguir buscando componentes más adecuados.

Diseño del sistema con reutilización: Se diseña o reutiliza el marco de trabajo para el sistema. Se debe tener en cuenta los componentes localizados en la fase 2 para diseñar o determinar este marco.

Desarrollo e integración: El software que no puede comprarse, se desarrolla. Se integran los componentes y subsistemas. La integración es parte del desarrollo en lugar de una actividad separada. [4]

Las ventajas de este modelo son:

- Disminuye el costo y esfuerzo de desarrollo.
- Reduce el tiempo de entrega.
- Disminuye los riesgos durante el desarrollo.

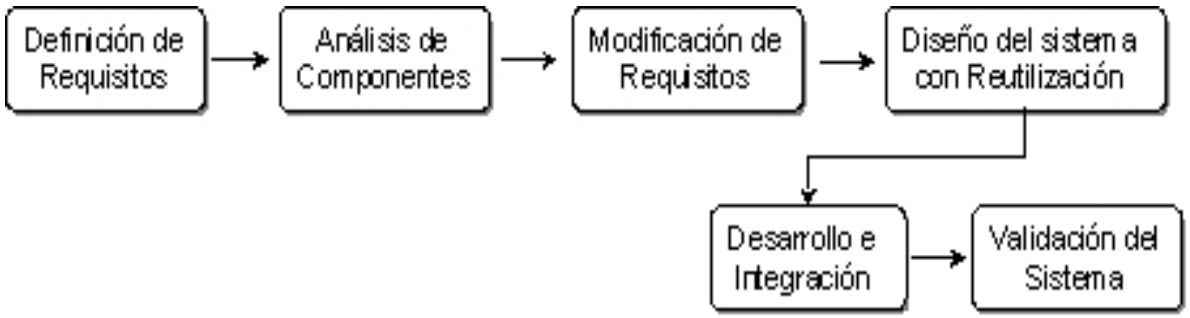


Figura 4 Desarrollo basado en reutilización de componentes

Desventajas de este modelo:

- Los “compromisos” en los requisitos son inevitables, por lo cual puede que el software no cumpla las expectativas del cliente.
- Las actualizaciones de los componentes adquiridos no están en manos de los desarrolladores del sistema.

Desarrollo incremental

Es una combinación del Modelo de Cascada y Modelo Evolutivo. Reduce el rehacer trabajo durante el proceso de desarrollo y da oportunidad para retrasar las decisiones hasta tener experiencia en el sistema. Durante el desarrollo de cada incremento se puede utilizar el modelo de cascada o evolutivo, dependiendo del conocimiento que se tenga sobre los requisitos a implementar. Si se tiene un buen conocimiento, se puede optar por cascada, si es dudoso, evolutivo. [4]

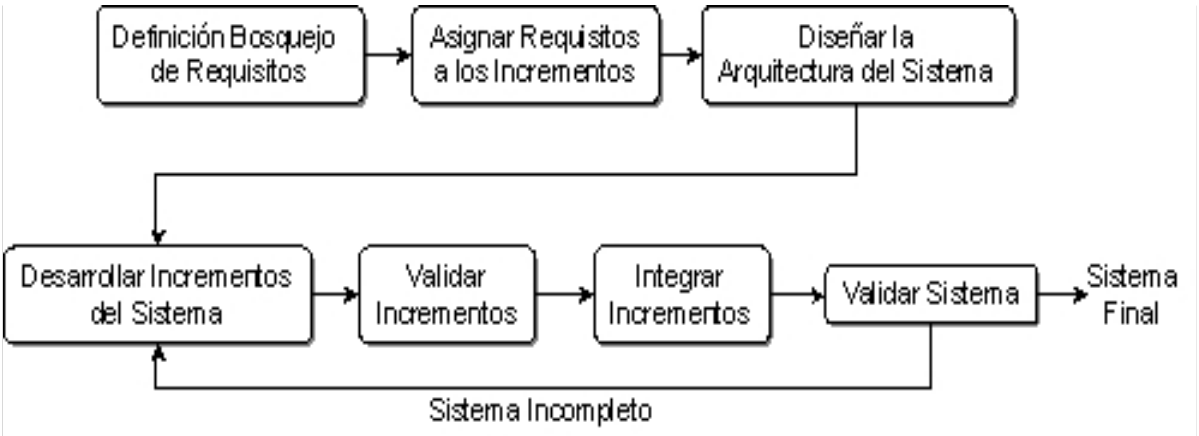


Figura 5 Modelo de desarrollo iterativo incremental

Entre las ventajas del modelo incremental se encuentran:

- Los clientes no esperan hasta el fin del desarrollo para utilizar el sistema. Pueden empezar a usarlo desde el primer incremento.
- Los clientes pueden precisar o los requisitos que no estén refinados en cada iteración, según las entregas del sistema.
- Se disminuye el riesgo de fracaso de todo el proyecto, ya que se puede distribuir en cada incremento.
- Las partes más importantes del sistema son entregadas primero, por lo cual se realizan más pruebas en estos módulos y se disminuye el riesgo de fallos.

Algunas de las desventajas identificadas para este modelo son:

- Cada incremento debe ser pequeño para limitar el riesgo (menos de 20.000 líneas).
- Cada incremento debe aumentar la funcionalidad.
- Es difícil detectar las unidades o servicios genéricos para todo el sistema.
- Es difícil establecer las correspondencias de los requisitos contra los incrementos.[4]

Desarrollo en espiral

El ciclo de desarrollo se representa como una espiral, en lugar de una serie de actividades sucesivas con retrospectiva de una actividad a otra. Cada ciclo de desarrollo se divide en cuatro fases:

- **Definición de objetivos:** se definen los objetivos. Se definen las restricciones del proceso y del producto. Se realiza un diseño detallado del plan administrativo. Se identifican los riesgos y se elaboran estrategias alternativas dependiendo de estos.
- **Evaluación y reducción de riesgos:** se realiza un análisis detallado de cada riesgo identificado. Pueden desarrollarse prototipos para disminuir el riesgo de requisitos dudosos. Se llevan a cabo los pasos para reducir los riesgos.
- **Desarrollo y validación:** se escoge el modelo de desarrollo después de la evaluación del riesgo. El modelo que se utilizará (cascada, sistemas formales, evolutivo, etc.) depende del riesgo identificado para esa fase.
- **Planificación:** se determina si continuar con otro ciclo. Se planea la siguiente fase del proyecto.

Este modelo a diferencia de los otros toma en consideración explícitamente el riesgo, esta es una actividad importante en la administración del proyecto. El ciclo de vida inicia con la definición de los objetivos. De acuerdo a las restricciones se determinan distintas alternativas. Se identifican los riesgos al sopesar los objetivos contra las alternativas. Se evalúan los riesgos con actividades como análisis detallado, simulación, prototipos, etc. Se desarrolla un poco el sistema. [4]

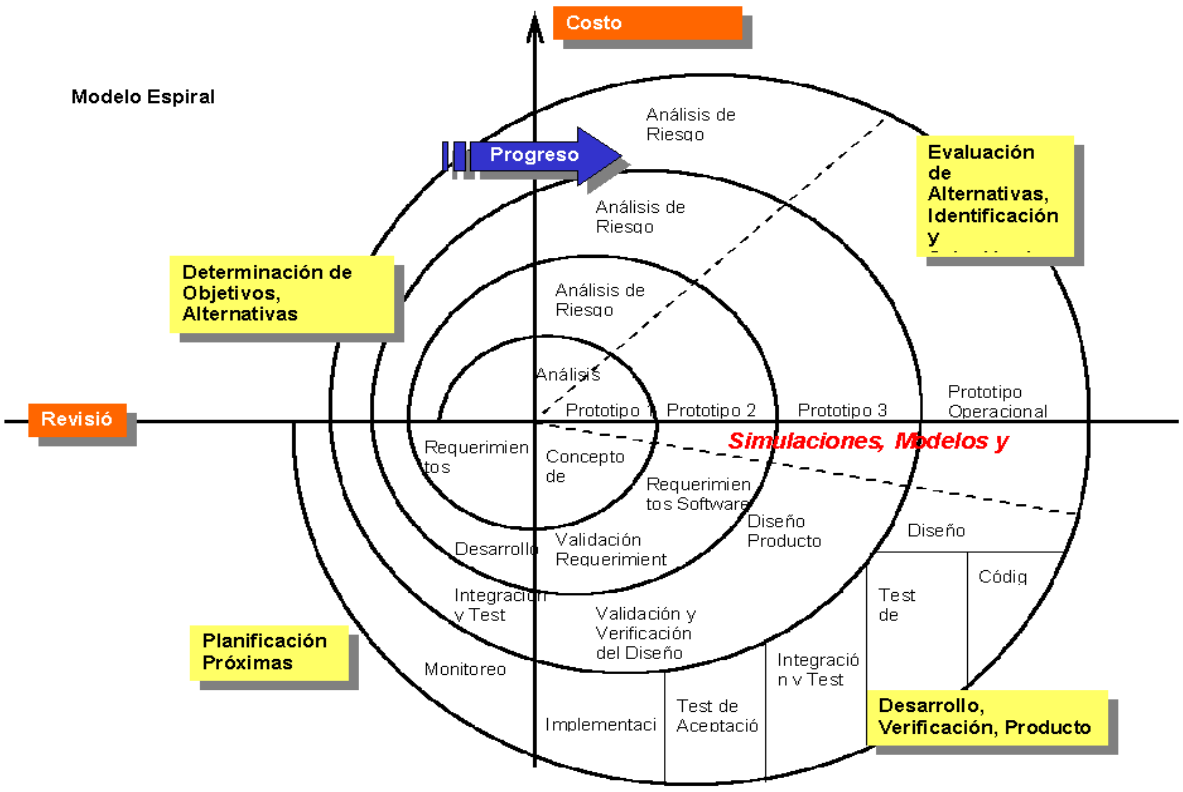


Figura 6 Modelo de desarrollo en Espiral

1.2 Estándares de calidad para desarrollo de software

Actualmente existen diferentes estándares y propuestas internacionales y regionales relacionadas con la mejora de procesos de software (Software Process Improvement SPI) para micro y pequeñas empresas. El enfoque de mejora de procesos surge debido a la necesidad de fortalecer las empresas mediante la utilización de prácticas y guías eficientes de Ingeniería de Software adaptadas a su tamaño y tipo de negocio. En la actualidad los modelos más utilizados son CMMI, ISO 15504 e ISO 9001:2000.

Los modelos de mejora de procesos establecen los diferentes procesos implicados a la hora de desarrollar sistemas informáticos, desde que surge la idea o necesidad de desarrollar las aplicaciones informáticas hasta que éstas se retiran de explotación.

1.2.1 Capability Maturity Model Integration (CMMI)

El Instituto de Ingeniería de Software (SEI Software Engineering Institute) ha desarrollado el Modelo CMMI, el cual proporciona a las organizaciones de software una orientación sobre cómo realizar el control de sus procesos de desarrollo y mantenimiento de software, y cómo evolucionar hacia la implantación de una cultura de ingeniería de software y de gestión por excelencia.

CMMI es un modelo de mejora de procesos de desarrollo que provee orientación para diseñar procesos efectivos (tiempo y coste), en distintos dominios. Ayuda a integrar las funciones de la organización, provee técnicas para conducir la mejora de los procesos, provee una guía de calidad de los procesos, y de puntos de referencia para la evaluación de estos.

CMMI permite a los usuarios elegir entre disciplinas simples o integradas como Ingeniería de software, e Ingeniería de sistemas, además de poder seleccionar una representación por etapas o continua. Incluye información sobre la mejora de procesos e ingeniería, tales como objetivos claros y una extensiva guía en las mejores prácticas para cumplirlos. Presenta un diseño del marco de trabajo bien definido lo que permite incluir disciplinas adicionales para reducir el desarrollo de modelos incompatibles en el futuro. Los modelos de calidad que centran su foco en la madurez de la organización, presentan un modelo de mejora y evaluación “escalonado”. Los que enfocan las actividades de mejora y evaluación en la capacidad de los diferentes procesos presentan un modelo “continuo”. CMMI nació integrando tres modelos diferentes, con representaciones diferentes:

- CMM-SW: representación escalonada.
- SE-CMM: representación continua.
- IPD-CMM: modelo mixto.

En el equipo de desarrollo de CMMI había defensores de ambos tipos de representaciones. El resultado fue la publicación del modelo con dos representaciones: continua y escalonada. Son equivalentes, y cada organización puede optar por adoptar la que se adapte a sus características y prioridades de mejora. [5]

Representaciones de CMMI

CMMI ofrece dos posibles representaciones: la representación Continua y la Escalonada

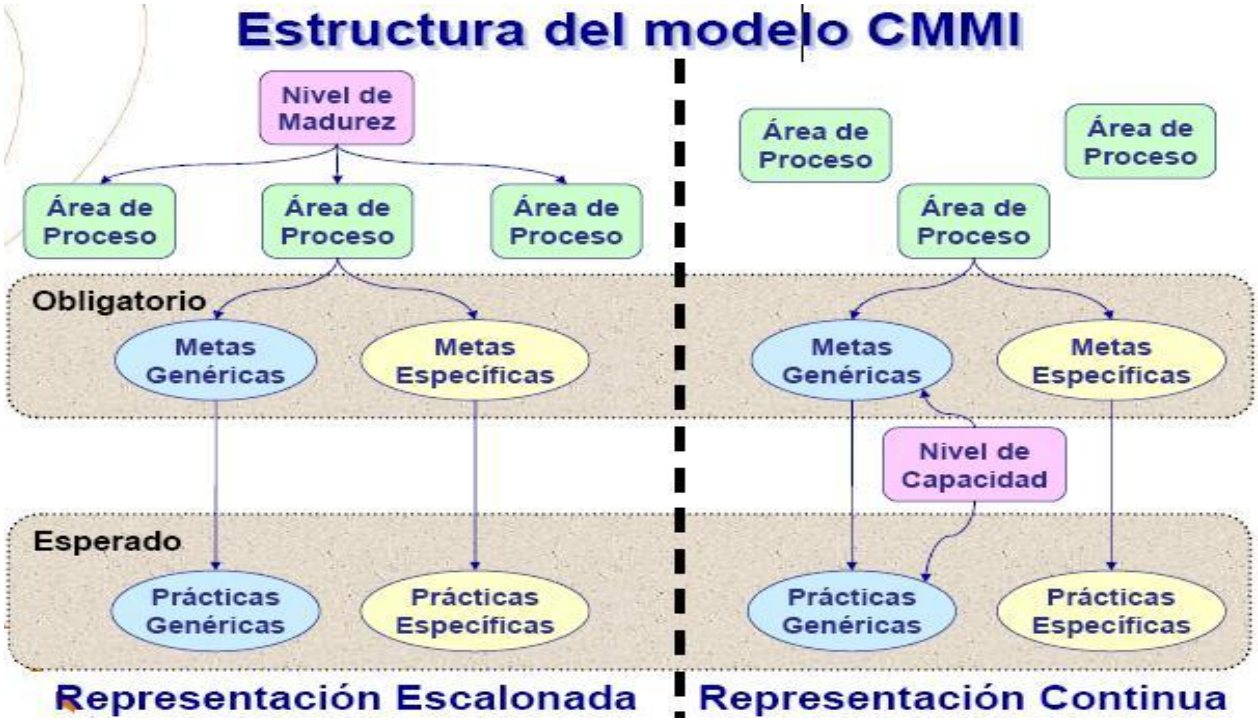


Figura 7 Estructura del modelo CMMI

Representación Escalonada o por Etapas

La representación por etapas organiza las áreas de procesos dentro de cinco niveles de madurez, que soportan y guían la ruta evolutiva del proceso de mejoramiento continuo para toda la organización.

Niveles de madurez

La representación escalonada de CMMI define cinco niveles posibles de madurez para las organizaciones que desarrollan y mantienen software, según las acciones que se realicen dentro de las diferentes instituciones se puede optar por un nivel definido por CMMI:

Nivel 1: Inicial. Los resultados de calidad obtenidos son consecuencia de las personas y de las herramientas que emplean. No de los procesos, porque o no los hay o no se emplean.

Nivel 2: Repetible. Se considera un nivel 2 de madurez cuando se llevan a cabo prácticas básicas de gestión de proyectos, de gestión de requisitos, control de versiones y de los trabajos realizados por subcontratistas. Los equipos de los proyectos pueden aprovechar las prácticas realizadas para aplicarlas en nuevos proyectos.

Nivel 3: Definido. Los procesos comunes para desarrollo y mantenimiento del software están documentados de manera suficiente en una biblioteca accesible a los equipos de desarrollo. Las personas han recibido la formación necesaria para comprender los procesos.

Nivel 4: Gestionado. La organización mide la calidad del producto y del proceso de forma cuantitativa en base a métricas establecidas. La capacidad de los procesos empleados es previsible, y el sistema de medición permite detectar si las variaciones de capacidad exceden los rangos aceptables para adoptar medidas correctivas.

Nivel 5: Optimizado. La mejora continua de los procesos afecta a toda la organización, que cuenta con medios para identificar las debilidades y reforzar la prevención de defectos. Se analizan de forma sistemática datos relativos a la eficacia de los procesos de software para analizar el coste y el beneficio de las adaptaciones y las mejoras. [5]

Representación Continua

La representación continua refleja los niveles de capacidad en su diseño y contenido. Un nivel de capacidad comprende prácticas específicas y genéricas que permiten medir la habilidad de un proceso.

Los 6 niveles definidos en CMMI para medir la capacidad de los procesos son:

Nivel 0: Incompleto. El proceso no se realiza, o no se consiguen sus objetivos.

Nivel 1: Ejecutado. El proceso se ejecuta y se logra su objetivo.

Nivel 2: Gestionado. Además de ejecutarse, el proceso se planifica, se revisa y se evalúa para comprobar que cumplen los requisitos.

Nivel 3: Definido. Además de ser un proceso “gestionado” se ajusta a la política de procesos que existe en la organización, alineada con las directivas de la empresa.

Nivel 4: Cuantitativamente gestionado. Además de ser un proceso definido se controla utilizando técnicas cuantitativas.

Nivel 5: Optimizado. Además de ser un proceso cuantitativamente gestionado, de forma sistemática se revisa y modifica para adaptarlo a los objetivos del negocio. [5]

Categorías de procesos de Software por CMMI

CMMI define una serie de categorías de procesos que agrupan en general 25 áreas de procesos en las cuales actúa este modelo, estas categorías agrupan un grupo de subcategorías que garantizan la realización del proceso en general. Las categorías definidas por CMMI son:

- Gestión de procesos
- Gestión de proyectos
- Ingeniería

- Soporte [5]

1.2.2 ISO 15504

ISO 15504 es un modelo para la mejora y evaluación de los procesos de desarrollo, determinación de capacidad y mantenimiento de sistemas y productos de software, provee de un marco de trabajo uniforme para gestión e ingeniería del software. Su filosofía es desarrollar un conjunto de medidas de capacidad estructuradas para todos los procesos del ciclo de vida y para todos los participantes. Se puede implementar en empresas con departamentos de desarrollo a partir de 10-15 personas. Tiene una difícil implementación en pequeñas empresas de desarrollo de software. Requiere un gran número de herramientas, algunas de ellas con un costo elevado. [6]

La primera, denominada dimensión del proceso, la cual define un conjunto estándar de procesos para el ciclo de vida completo del software. Plantea tres clases básicas de procesos (primaria, soporte y organizativas), que se dividen en cinco categorías de proceso (cliente/suministrador, ingeniería, soporte, administración, organización), veinticuatro procesos de alto nivel y otros dieciséis componentes.

La segunda, dimensión de la capacidad del proceso, se sustenta en un conjunto de atributos que determinan el nivel. Su objetivo es definir la escala de medida para la capacidad del proceso, y para ello se considera una escala de tipo ordinal basada en seis puntos de control:

Incompleto: Hay un fallo generalizado al alcanzar los propósitos del proceso

Realizado: El propósito del proceso es generalmente alcanzado. Este éxito no tiene por qué haber sido rigurosamente planificado ni seguido.

Gestionado: Se liberan productos de acuerdo a procedimientos específicos siendo el proceso planificado y seguido.

Establecido: Se usan procesos definidos basados en principios de la ingeniería del software para alcanzar los objetivos.

Predecible: El proceso definido es ejecutado en consistencia con controles de límites establecidos, para alcanzar los objetivos definidos. Las medidas detalladas del rendimiento son coleccionadas y analizadas.

Optimizado: La realización de los procesos se encuentra optimizada de forma que coincidan con las necesidades actuales y futuras de negocio. Los resultados de los procesos son alcanzados de forma repetida de acuerdo con los objetivos definidos. [6]

Categorías de procesos de software por ISO 15504:

ISO 15504 define una serie de categorías de procesos para el desarrollo de un producto de software, las cuales van a englobar determinadas subcategorías que permitirán la realización de estos procesos definidos. Las categorías de procesos que define son:

- Proceso de Adquisición
- Proceso de Suministro
- Proceso de Ingeniería
- Proceso de Operación
- Proceso de Soporte
- Proceso de Gestión
- Proceso de Mejora
- Proceso de Infraestructura

1.2.3 ISO 9001:2000

ISO 9001:2000 forma parte de la familia de normas ISO 9000 que puede ser aplicado en cualquier tipo de organización, es una metodología de procesos basada en una lista de comprobaciones o requisitos a cumplir, umbral de calidad, valorado apto o no apto, se centra en la eficacia del sistema de gestión de la calidad para dar cumplimiento a los requisitos del cliente. Tiene como objetivo principal construir un Sistema de Gestión de la Calidad que abarque la estructura de la organización, las responsabilidades, los procedimientos, los procesos y los recursos necesarios para implementar una dirección de calidad. Su aplicación proveería a cualquier empresa de una gran gama de beneficios como:

- Reducción de los rechazos e incidencias en la producción o prestación del servicio.
- Aumento de la productividad
- Mayor compromiso con los requisitos del cliente.
- Mejora continua.

El estándar de calidad ISO 9001:2000 presenta una amplia aplicación en cualquier industria y entorno de desarrollo, este afecta la mayoría de las áreas funcionales de una organización, implementa la

libertad de implementación y de interpretación de requisitos. Su aplicación posibilita un incremento en las oportunidades de negocio en diferentes mercados, además que mejora la satisfacción del cliente con los resultados finales.

Este estándar ayuda a implementar la gestión de la calidad acción que propicia que los productos o servicios satisfagan los requisitos de calidad de los clientes, además que posibilita que la organización logre alcanzar una mejora continua de sus resultados, y con esto un aumento de la productividad.

ISO 9001:2000 está basada en ocho principios de gestión de la calidad:

- Orientación al Cliente
- Liderazgo.
- Implicación.
- Enfoque de proceso.
- Enfoque de sistema.
- Mejora continua.
- Toma de decisiones basadas en hechos.
- Relaciones de beneficio mutuo con proveedores. [7]

Secciones de ISO 9001:2000

Las cinco secciones en que se divide ISO 9001:2000 son:

- QMS Sistema de Gestión de la Calidad (SGC) (Requisitos generales y Requisitos de la documentación).
- Responsabilidad de la Gestión (Compromiso de la dirección, Enfoque al cliente, Política de la calidad, Planificación).
- Gestión de los Recursos (Provisión de recursos, Recursos humanos, Infraestructura, Ambiente de trabajo).
- Realización del Producto (Planificación de la realización del producto, Procesos relacionados con los clientes, Diseño y desarrollo, Compras, Prestación del servicio).
- Medición, Análisis y Mejora (Generalidades, Supervisión y Medición, Control de servicio no-conforme, Análisis de datos, Mejora). [8]

Enfoque de procesos de ISO 9001: 2000

ISO 9001: 2000 hace énfasis en la definición, control y mejora de los procesos que permitan garantizar la calidad de los productos y servicios producidos y que se encuentran entre los requerimientos y la satisfacción del cliente. Ha agrupado los procesos en tres grandes categorías, los relativos al soporte de la gestión, los procesos clave que hacen operativa la gestión y los estratégicos que permiten llegar a los objetivos planteados. [9]

El objetivo de la evaluación del proceso es conocer la capacidad de los procesos de una organización. Como resultado de una exitosa implementación de la evaluación de los procesos se determina la información que caracteriza los procesos evaluados y el punto hasta el cual los procesos realizan su propósito.

ISO 9001 pretende fomentar la adopción del enfoque basado en procesos para gestionar una organización. Este tipo de gestión por procesos, cuando se utiliza en el desarrollo, la implementación y la mejora de la eficacia de un SGC, concentra su atención en:

- La comprensión y el cumplimiento de los requisitos de los clientes de cada proceso.
- La necesidad de considerar y de planificar los procesos en términos que aporten valor (el cliente no debe pagar por algo que no le aporte valor).
- El control, la medición y la obtención de resultados del desempeño y de la eficacia de los procesos.
- La mejora continua de los procesos con base en mediciones objetivas.

Con el objetivo de acreditarse entre las empresas de producción de software de excelencia, la Universidad de las Ciencias Informáticas persigue los lineamentos establecidos por estos estándares de Calidad y tiene como meta para el 2012 poder alcanzar un nivel 2 o superior de madurez según la clasificación de CMMI, por lo que la metodología a utilizar en vista de que dichos lineamentos sean alcanzados es fundamental.

1.3 Metodologías para el desarrollo de software:

El desarrollo de software no es una tarea fácil. Desarrollar un buen software depende de un sinnúmero de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo en un determinado proyecto, es trascendental. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. [10]

Por una parte están las metodologías tradicionales, que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben

producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. [10]

Otro enfoque es el de las metodologías ágiles que dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas, centrándose en el factor humano y en el producto software. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Para hacer más comprensible la importancia de la elección correcta de la metodología a emplear, se exponen a continuación las principales características de las metodologías más populares.

1.3.1 Metodologías tradicionales

Las metodologías tradicionales centran su atención en llevar una documentación absoluta de todo el proyecto, están focalizadas en documentación, planificación y procesos. Se caracterizan por exponer procesos basados en planeación exhaustiva. Esta planeación se realiza esperando que el resultado de cada proceso sea determinante y predecible. La experiencia ha mostrado que, como consecuencia de las características del software, los resultados de los procesos no son siempre predecibles y sobre todo, es difícil predecir desde el comienzo del proyecto cada resultado. Para el desarrollo de los productos generalmente existe un contrato prefijado y el cliente interactúa con el equipo de desarrollo mediante reuniones. Otra de las características importantes dentro de este enfoque, son los altos costos al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil.

La solución a algunos de los problemas presentados por las metodologías tradicionales se logra con una gran evolución del modelo espiral. El proceso unificado propone la elaboración de varios ciclos de desarrollo, donde cada uno finaliza con la entrega al cliente de un producto terminado. Este se enmarca entre los conocidos modelos iterativo-incremental.

Rational Unified Process (RUP)

La metodología RUP divide en 4 fases el desarrollo del software:

- **Inicio**: el objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración**: en esta etapa el objetivo es determinar la arquitectura óptima.

- **Construcción**: en esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- **Transición**: el objetivo es llegar a obtener el reléase del proyecto.[11]

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

Disciplinas de Desarrollo

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente. [11]

Disciplina de Soporte

- Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- Administrando el proyecto: Administrando horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.
- Distribución: Hacer todo lo necesario para la salida del proyecto

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

Actividades: son los procesos que se llegan a determinar en cada iteración.

Trabajadores: vienen a ser las personas o entes involucrados en cada proceso.

Artefactos: un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. [11]

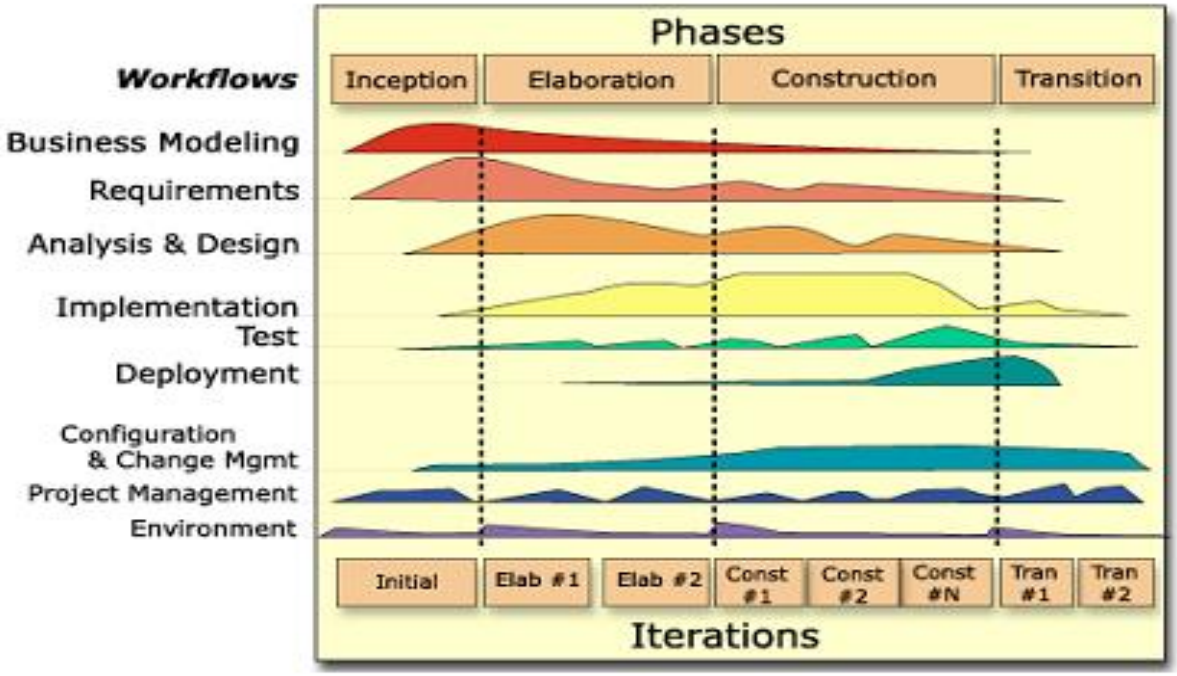


Figura 8 Rational Unified Process (RUP)

Microsoft Solutions Framework (MSF)

MSF es un marco (Framework) que provee una estructura orientada a facilitar el análisis, diseño e implementación de soluciones tecnológicas efectivas. Este marco permite exponer, revelar y manejar riesgos críticos, determinar los criterios de planeación, y establecer las interdependencias necesarias para una ejecución exitosa de los proyectos.

"Como marco, MSF no es una metodología en el sentido estricto, con estructuras de trabajo, tareas y productos predeterminados. En su lugar, MSF provee mecanismos flexibles para aplicar soluciones adecuadas a los problemas tecnológicos y de negocios. MSF no es un marco estático sino que evoluciona respondiendo a los cambios en la tecnología y en los requerimientos de los proyectos".

Microsoft Solutions Framework está basado en un conjunto de modelos, derivados de la experiencia de Microsoft, sus socios tecnológicos y sus clientes en la implementación de tecnologías cliente-servidor y sistemas distribuidos. [11]

Los modelos de MSF incorporan tres factores fundamentales de éxito:

- Un punto de visión, para proveer la guía requerida para tomar decisiones técnicas.
- Un conjunto de puntos de referencia, para realizar un seguimiento efectivo de la marcha de los procesos o proyectos, con énfasis en el manejo de los riesgos durante todo el ciclo de vida.
- Capacidad de reutilización, para tomar ventaja del conocimiento previo en forma estructurada y consistente en un ambiente tecnológico flexible.

MSF tiene las siguientes características:

- **Adaptable**: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- **Escalable:** puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas o más.
- **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología. [11]

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño del Proceso, Modelo de Aplicación. [3]

Figura 9 Microsoft Solutions Framework (MSF)

1.3.2 Metodologías Agiles

Las metodologías ágiles están especialmente orientadas para entornos variables, proyectos pequeños donde los individuos y las interacciones entre ellos, son más importantes que las herramientas y los procesos empleados y en donde se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad. La prioridad de este enfoque es satisfacer al cliente mediante tempranas y continuas entregas que le aporte un valor, por lo que es más importante crear un producto software que funcione sin tener que escribir documentación exhaustiva, tributando con una elevada simplificación pero sin renunciar a las prácticas esenciales para asegurar la calidad del producto. [12]

La colaboración con el cliente debe prevalecer sobre la negociación de contratos por lo que se propone que exista una interacción constante entre el cliente y el equipo de desarrollo.

La planificación no debe ser estricta sino flexible y abierta para poder responder a los cambios que puedan surgir a los largo del proyecto. [12]

Extreme Programming (XP)

Extreme Programming es uno de los procesos de desarrollo de software más exitosos en la actualidad utilizado para proyectos de corto plazo y con equipos pequeños.

Está basado en la simplicidad, la comunicación y la retroalimentación de código.

XP propone lo siguiente:

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierten en miembros del equipo.

XP crea transparencia y un clima de agilidad en la relación entre desarrolladores y clientes. El costo de hora/hombre por cada tipo de recurso es conocido y acordado desde el principio.

Un proyecto de varios meses es dividido en pequeños proyectos de pocas semanas de duración y las metas y cronogramas se van ajustando en tiempo real, de acuerdo al nivel de avance y las dificultades reales que ofrece el proyecto aceptadas en forma conjunta por desarrolladores y clientes. [13]

Lo fundamental de XP es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

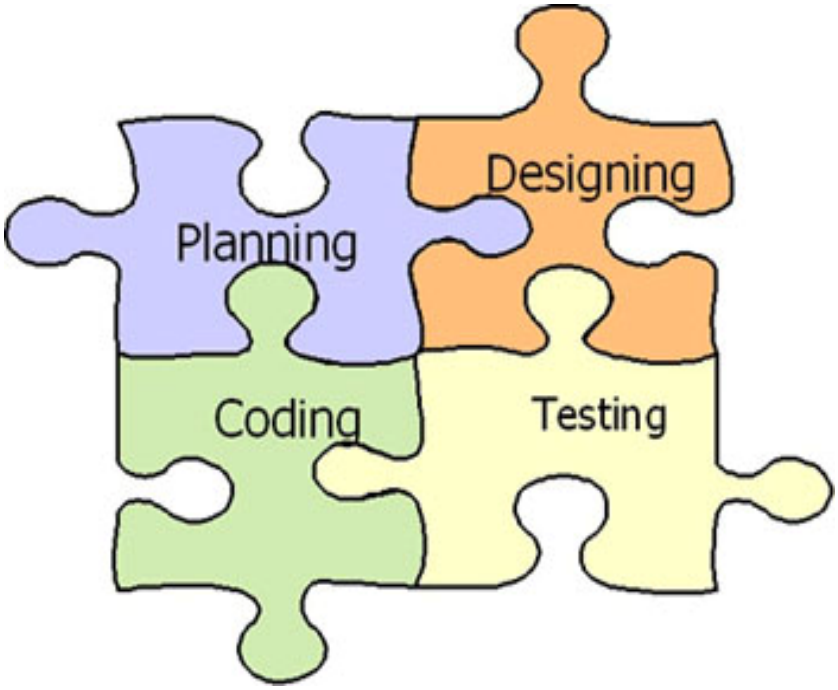


Figura 10 Extreme Programming (XP)

OpenUp/Basic

OpenUp/Basic es un proceso de desarrollo de software de código abierto diseñado para pequeños equipos organizados que quieren tomar una aproximación ágil del desarrollo. Es un proceso iterativo que es Mínimo, Completo, y Extensible. Se valora la colaboración y el aporte de los stakeholders sobre los entregables y la formalidad innecesarios. [14]

OpenUp/Basic está organizado dentro de cuatro áreas principales de contenido: Comunicación y Colaboración, Intención, Solución, y Administración.

OpenUp/Basic se caracteriza por cuatro principios básicos que se soportan mutuamente:

- Colaboración para alinear los intereses y un entendimiento compartido
- Balance para confrontar las prioridades (necesidades y costos técnicos) para maximizar el valor para los stakeholders

- Enfoque en articular la arquitectura para facilitar la colaboración técnica, reducir los riesgos y minimizar excesos y trabajo extra.
- Evolución continua para reducir riesgos, demostrar resultados y obtener retroalimentación de los clientes. [14]

Características Generales:

- Preserva la esencia del Unified Process
- Desarrollo iterativo e incremental
- Desarrollo dirigido por Casos de Uso
- Centrado en la Arquitectura [14]

La mayor fortaleza de OpenUp/Basic es que puede ser extendido en grandes o pequeñas formas para adicionar nuevos contenidos de desarrollo o personalizar el proceso para su entorno específico.

Los practicantes de desarrollo de software pueden encontrar guías sobre lo que se requiere de ellos en los roles definidos por OpenUp/Basic. Cada rol describe un conjunto de actividades y artefactos de los cuales el rol es responsable. También se dan guías sobre cómo estos roles colaboran.

Los ingenieros de procesos de software pueden extender y modificar OpenUp/Basic. Las modificaciones pueden ser tan simples como alterar las plantillas para los productos de trabajo o tan sofisticadas como adicionar actividades necesarias para crear software en su ambiente específico. [14]

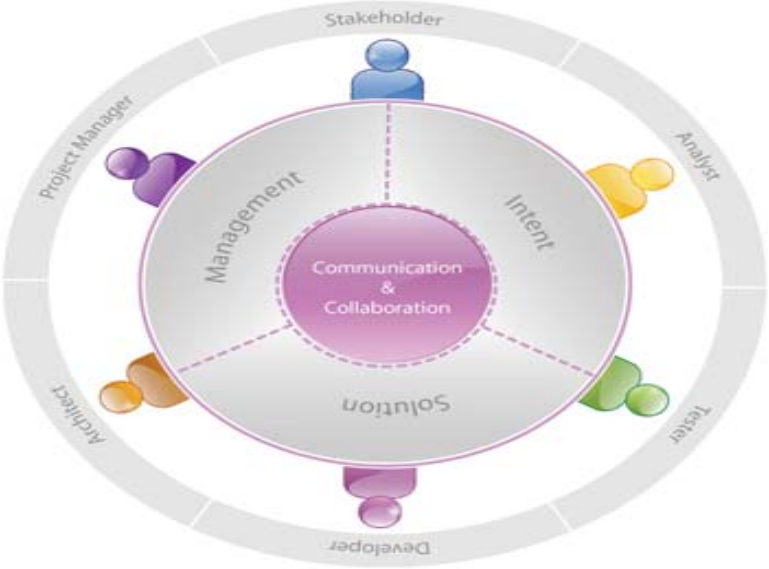


Figura 11 OpenUp/Basic

1.3.3 Diferencias entre las metodologías tradicionales y ágiles

Después de analizar los dos enfoques metodológicos de desarrollo de software y estudiar brevemente algunas metodologías que cumplen con ellos se llega a la siguiente comparación:

Tabla 1 Comparaciones Generales entre Metodologías Agiles y Tradicionales [15]

Metodologías Agiles	Metodologías Tradicionales
Basadas en heurística provenientes de práctica de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparadas para cambios durante el proyecto	Poseen cierta resistencia a los cambios
Proceso menos controlado, con menos principios	Proceso mucho más controlado con numerosas políticas y normas
No existe un contrato formal o al menos es bastante flexible	Existe un contrato prefijado
El cliente generalmente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños, trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Muchos artefactos
Pocos roles	Muchos roles
Menor énfasis en la arquitectura de software	La arquitectura de software es esencial y se expresa mediante modelos
Proyectos con iteraciones cortas y menor tiempo de duración	Proyectos con mayor tiempo de duración

Tabla 2 Diferencias entre Metodologías Agiles y Tradicionales por Flujos de Trabajo

Metodologías Agiles		Metodologías Tradicionales
No se modela el Negocio, (Dominio)	**Análisis de Requerimientos**	Define Modelo de Negocio
Diseño simple; documentación mínima y focalizado en la comunicación	**Diseño**	Diseño flexible y extensible; modelos y documentación exhaustiva
Conocimiento colectivo	**Implementación**	Desarrollo individual con roles y responsabilidades estrictas
El cliente generalmente participa en las pruebas (aporte de los interesados)	**Pruebas**	No hay aporte directo de los interesados en esta etapa
Planificación adaptativa; entregas frecuentes y colaboración del cliente	**Gestión de Proyectos**	Planificación predictiva y aislada

Tabla 3 Comparaciones entre las Metodologías RUP, MSF, XP y OpenUp/Basic

Metodología	Tamaño del Proceso	Tamaño del Equipo	Aprendizaje	Complejidad del Problema
RUP	Medio/Extenso	Medio/Extenso	Medio/Lento	Medio/Alto
MSF	Medio/Extenso	Pequeño/Extenso	Medio/Lento	Medio/Alto
XP	Pequeño/Medio	Pequeño	Rápido	Medio/Alto
OpenUp	Pequeño/Medio	Pequeño	Rápido	Medio/Alto

1.4 Tendencias actuales en cuanto a la utilización de Metodologías de Desarrollo del Software

Las metodologías ingenieriles tradicionales han estado presentes durante mucho tiempo. No se han distinguido precisamente por ser muy exitosas. Aún menos por su popularidad. La crítica más frecuente a estas metodologías es que son burocráticas. Hay tanto que hacer para seguir la metodología que el ritmo entero del desarrollo se retarda.

Como una reacción a estas metodologías, un nuevo grupo de metodologías ha surgido en los últimos años. En el 2001 en Utah-EEUU en la reunión de los 17 expertos de la industria del software, nace el término "ágil" aplicado al desarrollo de software. Durante algún tiempo se conocían como las

metodologías ligeras, pero el término aceptado ahora es metodologías ágiles. Para mucha gente el encanto de estas metodologías ágiles es su reacción a la burocracia de las metodologías tradicionales. Estos nuevos métodos buscan un punto medio entre ninguno y demasiados procesos, proporcionando simplemente los suficientes para que el esfuerzo valga la pena. [16]

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

Los métodos ligeros o agiles son otra opción para el desarrollo. Presentan tanto adeptos como detractores, algunos expertos mencionan que los procesos ágiles son una moda y quedaran ahí, sin embargo existen empresas que desde hace tiempo utilizan y han evolucionado gracias a dichos métodos. Algunas de estas empresas son: [16]

Figura 12 Logos de empresas partidarias de las metodologías ágiles

Aunque las metodologías ligeras se basan en las ideas de los procesos tradicionales estas usan lo más importante para el buen desarrollo del proyecto con lógica y dejando atrás el manejo excesivo de artefactos y burocracia. Desde el surgimiento de estas revolucionarias metodologías los incrementos en adeptos se presentan gradualmente con el tiempo y las tecnologías.

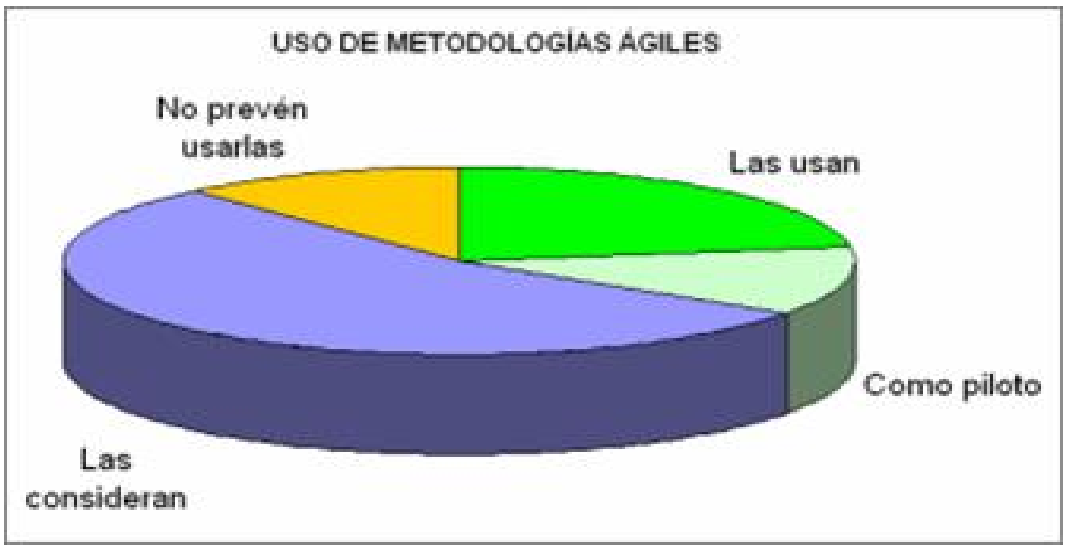


Figura 13 Uso de metodologías ágiles

Y los que las usan, ¿Por qué razones lo hacen?:

- Para reducir el tiempo de desarrollo: 45%
- Para mejorar la calidad: 43%
- Para reducir costes: 23%
- Para alinear el desarrollo con los objetivos de negocio: 39%
- Otras razones: 12%.[16]

1.5 Fundamentación de la Metodología propuesta

Después de haber analizado y de realizar comparaciones entre los dos enfoques metodológicos expuestos, se arriba a la conclusión de que la propuesta de metodología a implantar en el PPB sea la ligera (Metodología Ágil), pues se ajusta más a sus características y cumple con las necesidades y objetivos trazados para el desarrollo de software del los equipos del mismo. *(Ver epígrafe 2.1)*

Ergo, es importante destacar que dentro de esta variante, la metodología que más se ajusta es OpenUp/Basic por varias razones:

- **Preserva la esencia del Proceso Unificado (RUP Rational Unified Process)**, que es la metodología más conocida por los estudiantes y profesores vinculados al PPB debido a que fue la estudiada en los cursos de Ingeniería de Software I y II, asignatura obligatoria de la carrera; y por ser además, la utilizada por la mayoría de los proyectos de la UCI y anteriormente por el propio PPB.

- **Es extensible y adaptable** a las características del equipo de desarrollo, lo que permite modificarla teniendo en cuenta que es necesario realizar cambios a estas metodologías para que además de satisfacer las necesidades en cuanto a simplificación de tiempo, documentación y actividades, cumpla con los lineamientos de calidad y con las metas trazadas por la universidad en cuanto al nivel de madurez de la organización para el proceso de desarrollo de software.
- **Modelo de desarrollo Iterativo Incremental** al Igual que RUP, pero con la diferencia de que al ser un proceso mucho más ligero, permite micro incrementos en breves periodos de tiempo, lo que posibilita ir creando raleases del producto mientras se desarrolla y efectuar modificaciones a los requerimientos sin altos costos en cuanto a tiempo, precio e implementación.

Por lo que finalmente se propone utilizar OpenUp/Basic y realizar las modificaciones necesarias para que se adapte a las características y necesidades concretas del PPB.

1.6 Conclusiones

En este capítulo se han abordado las principales directrices que proponen los estándares de calidad en vista a la necesidad de fortalecer las empresas mediante la utilización de prácticas y guías eficientes de Ingeniería de Software adaptadas a su tamaño y tipo de negocio y cuyos modelos más utilizados en el mercado de software de nuestros días son CMMI, ISO 15504 e ISO 9001:2000. Profundizando en las normas CMMI y particularmente en las áreas de proceso que estas proponen para obtener niveles de capacidad. Pues una de las metas de la UCI es acogerse a las normas CMMI alcanzando un nivel 2 o Gestionado siguiendo la representación continua según la definición que proveen dichos estándares.

Se fundamentó el uso de la Metodología OpenUp/Basic realizando un estudio exhaustivo y comparativo de las metodologías de desarrollo caracterizando los dos enfoques existentes; las metodologías ágiles y las metodologías tradicionales, de manera que se pudieran evaluar las ventajas y desventajas de una sobre otra, y concluyendo en que debido a las características y necesidades del PPB sería favorable la implantación de una metodología ágil siempre y cuando esta fuera extensible y adaptable de modo que pudiera ser modificada para cubrir las especificidades que engloban los lineamientos de calidad de software de la Universidad y las métricas que plantean los estándares de CMMI.

También se ha realizado un análisis detallado de los procesos de desarrollo de software, indagando entre distintos modelos genéricos que pueden ser utilizados para explicar diferentes perspectivas de cómo afrontarlos. Particularmente en el Modelo de Desarrollo Iterativo Incremental, que es el que sigue

la Metodología que se propone adaptar, debido a las ventajas que ofrece y las características que hacen que sea el idóneo para allanar las necesidades del PPB *(ver epígrafes 2.2.1 y 2.2.3)*.

Capítulo 2. Programa y Metodología

Introducción

En este capítulo se detalla el conjunto de tareas realizadas que permite elaborar la propuesta de solución. Se describe el entorno de desarrollo en el que se identificaron las acciones a cometer para adaptar la metodología OpenUp/Basic y el expediente de proyecto propuesto por la UCI a las necesidades de los proyectos del PPB cumpliendo con los estándares de calidad requeridos. Se enuncian las acciones a realizar para ejecutar las fases, además de mencionar los roles, artefactos, y actividades que se generan durante las mismas. Además se evalúa la propuesta realizando entrevistas a los principales líderes de los proyectos del PPB.

2.1 Características del entorno de desarrollo

Para alcanzar los objetivos trazados se hizo necesario, además del estudio del estado del arte de los Procesos de Desarrollo de Software, los estándares de calidad y de las distintas Metodologías de Desarrollo utilizadas en el mundo de la informática actual, estudiar el entorno de desarrollo para lograr su caracterización. Para esto se realizaron entrevistas a los jefes de proyecto, estudiantes y profesores vinculados a los módulos de los proyectos del PPB. Se analizaron las plantillas y demás artefactos con que cuenta actualmente el expediente de proyecto y un estudio del software y de las condiciones del hardware utilizado en los laboratorios. Los resultados arrojados por los análisis y entrevistas permitieron identificar determinadas características propias del entorno.

En cuanto al personal: está compuesto en su gran mayoría, de estudiantes cursando entre el 3er y 5to año de la carrera y recién graduados, ocupando generalmente roles como programadores, pero que aún no están capacitados para el desempeño efectivo de roles claves como Líder de Proyecto o Arquitecto por su corta experiencia en el desarrollo de software.

En cuanto a los proyectos: son grandes y abarcan en su totalidad bastantes recursos y personal, pero se dividen en mini proyectos o módulos pequeños que cuentan con un reducido equipo de desarrollo donde los integrantes van a desempeñar más de un rol durante la creación de la solución, para la cual generalmente se realiza un proceso a ciclo completo, en el que cada módulo obtiene sus propios artefactos (dígase documentos, modelos, ejecutables, código fuente, etc.) de manera independiente, que luego se combinan para llegar a la solución global. El peso de estos módulos está mas bien centrado en un trabajo de investigación profundo después del cual, el desarrollo de la solución ocupa la menor carga y consume menos tiempo.

En cuanto a los procesos: inicialmente se seguían las pautas establecidas por RUP. Una metodología tradicional con grandes volúmenes de actividades y que propone documentación que excede a las necesidades reales del PPB para cada fase de desarrollo, pues la cantidad de artefactos que genera vuelve demasiado lento y complicado el proceso. Para solucionar esta problemática se hizo necesaria la aplicación de otra metodología de desarrollo que permita adaptarse a las necesidades reales que se recogen en este estudio. Actualmente, en el PPB se utiliza OpenUp/Basic, metodología ágil que resuelve los problemas descritos anteriormente, agilizando el proceso de desarrollo, pero ligera en cuanto a la obtención de una documentación adecuada que se ajuste a los estándares de calidad de la UCI, en la que la mayoría de los procesos no están definidos, ni documentados, lo cual trae como consecuencia que se pierda el conocimiento por parte de los módulos cada vez que una persona clave dentro de la organización se marcha.

En cuanto a las tecnologías: el desarrollo de los proyectos del PPB, está basado sobre el sistema Operativo Linux, para aprovechar el uso de herramientas gratuitas, siguiendo la estrategia que propone la Universidad para poder disfrutar de los beneficios que ofrece el software libre. El PPB cuenta con varios laboratorios conectados mediante la red local de la Universidad, que tienen en su haber, numerosas computadoras con una configuración de hardware que posibilita un buen rendimiento y acceso a Internet, que aunque limitado a aproximadamente a 100 MB por estudiante al mes, permite realizar las búsquedas de información necesarias para el avance de los proyectos.

2.2 Descripción de la Situación Actual

2.2.1 Metodología

La adopción de la metodología OpenUp/Basic por el PPB, a pesar de todas las ventajas que brinda, y a las cuales ya se han hecho alusión en este documento, trae como consecuencia, que el nivel de gestión, en cuanto a la descripción y documentación del proceso de desarrollo, sea bajo. Debido al propio enfoque ágil que sigue esta metodología el expediente que propone, aunque cuenta con artefactos elementales para el desarrollo, es liviano y poco explícito, pues no genera toda la documentación necesaria para abordar el proceso de desarrollo de software con las especificaciones requeridas por los lineamientos de calidad, siendo así, que los únicos artefactos que se describen para el proceso de desarrollo de software son:

- Documento de Arquitectura de Software
- Implementación o código fuente
- Build (versión operativa del sistema o subsistema)

- Modelo de Diseño
- Plan de Pruebas
- Lista de Riesgos
- Plan de Iteraciones
- Plan de Desarrollo de Software
- Documento Visión
- Glosario de Términos
- Especificación de Requisitos
- Modelo de casos de Uso
- Casos de Pruebas

2.2.2 Nivel de Madurez

Después del análisis del entorno de desarrollo, y considerando la propuesta actual de la metodología implantada para el desarrollo de los proyectos del PPB, se concluye en que el proceso actual no presenta madurez suficiente para desarrollar software de calidad. Pues el éxito de la organización está supeditado al esfuerzo personal y no al uso de procesos probados, debido a lo que frecuentemente se excede los presupuestos o no se cumplen los planes de entrega.

Siguiendo los lineamientos que plantea CMMI para la evaluación de la madurez de las empresas, y proponiendo utilizar la representación continua que se enfoca en la capacidad de cada área de proceso para establecer una línea a partir de la cual se pueda medir la mejora individual en cada una, y donde la visión continua de una organización mostrará la representación de nivel de capacidad de cada una de las áreas de proceso del modelo, se debe concertar en que actualmente el PPB se encuentra en el nivel 0 o incompleto.

2.2.3 Necesidades del Polo Productivo de Bioinformática

El PPB necesita establecer una metodología que permita:

- **Cumplir con los Lineamientos de Calidad de Software de la Universidad de Las Ciencias Informáticas:** desarrollando todas las actividades y artefactos de carácter obligatorio que constan en este documento.

- **Alcanzar un nivel 2 o mayor de madurez:** a partir del cumplimiento de las actividades que proponen las áreas de proceso que establecen las métricas de calidad de CMMI para esta categoría.
- **Obtener la documentación mínima necesaria:** describiendo y documentando el proceso de manera que quede archivada toda la información necesaria para cumplir con las exigencias de los Lineamientos de Calidad UCI y los estándares de CMMI.
- **Disminuir el tiempo de desarrollo del proceso productivo:** centrándose en realizar solo las actividades necesarias para la culminación de los procesos en dependencia de las características individuales de cada proyecto.
- **Obtener Retroalimentación:** a través de las experiencias en el desarrollo de proyectos productivos y el estudio de los expedientes de proyectos de los mismos, que permita elevar la calidad de los proyectos y los conocimientos de las personas involucradas en dichos módulos.

2.2.4 Propuesta de Solución

La necesidad de mejorar la eficiencia y madurez del PPB para el desarrollo de un producto de calidad, conlleva a adquirir un nivel de capacidad superior al actual, en el cual exista una gestión de los requisitos y donde los procesos estén planeados, medidos, controlados y documentados, de manera que puedan ser gestionados y repetibles. Por lo que se propone adaptar la metodología OpenUp/Basic, adicionando nuevas actividades y artefactos de manera que cumpla con los requerimientos para obtener dicho nivel y alcance a cubrir los lineamientos de calidad planteados por la UCI.

2.3 Proceso de Adaptación de la Metodología OpenUp/Basic

Se propone realizar adaptaciones en cuanto a la manera de definir el proceso de desarrollo de la metodología OpenUp/Basic, conservando los roles y las disciplinas que conforman la metodología, pero creando un expediente de proyecto nuevo que responda a tres categorías: Administración de Proyectos, Ingeniería y Soporte. Este expediente, además de estar basado en los artefactos que definen las prácticas propias de OpenUp/Basic, incluye las prácticas específicas de las áreas de procesos que propone el nivel 2 de capacidad de CMMI y las de los lineamientos de calidad de la Universidad. Para su conformación se tomarán los artefactos iniciales que presenta OpenUp/Basic, agregando las actividades y artefactos necesarios para cumplir con el nivel de exigencias previsto.

2.3.1 Áreas de procesos para alcanzar el nivel 2 de madurez según CMMI

Lo que se pretende con el nivel 2 de CMMI es conseguir que en los proyectos de la organización haya una gestión de los requisitos y que los procesos estén planeados, ejecutados, medidos y controlados, y que el estado de los elementos de trabajo esté visible a la gerencia en puntos definidos de manera que se sepa en cada momento cuánto trabajo está hecho y cuánto queda por hacer. Estas ideas se materializan en las siguientes áreas de proceso: [17]

- Gestión de Requisitos
- Planificación de proyectos
- Monitoreo y Control del Proyecto
- Administración de Acuerdos con Proveedores
- Medición y Análisis
- Aseguramiento de la calidad
- Gestión de la configuración

Donde cada área de proceso no es más que un grupo de prácticas relacionadas que son ejecutadas de forma conjunta para conseguir un conjunto de objetivos. A continuación, se describen las prácticas de cada una de las áreas de procesos, agrupadas por categorías. [18]

Área de Proceso de Ingeniería

Gestión de Requerimientos

El objetivo del área de Proceso Gestión de Requerimientos es gestionar los requerimientos del producto durante todo el ciclo de vida, identificando inconsistencias con los artefactos y planes de proyecto.

Para cumplir con este objetivo, se deben realizar las siguientes prácticas específicas (PS).

- PS 1 Desarrollar entendimiento común con los responsables de entregar los requerimientos sobre el significado y alcance de cada uno de ellos
- PS 2 Obtener compromiso de requerimientos por parte del los participantes del proyecto
- PS 3 Administrar los requerimientos a medida que estos se desarrollan durante el proyecto
- PS 4 Mantener trazabilidad bidireccional entre requerimientos y artefactos

- PS 5: Identificar inconsistencias entre los planes de proyecto y artefactos y los requerimientos

Áreas de Proceso de Gestión de Proyectos

Planificación de proyecto

Los objetivos que persigue el área de proceso Planificación de Proyectos son: Establecer estimaciones de las magnitudes del proyecto, desarrollar y mantener un plan de proyecto que es empleado para poder administrarlo y obtener el compromiso de los interesados de manera que estén formalmente establecidos y sean mantenidos a lo largo del proyecto

Para cumplir con estos objetivos se deben realizar las siguientes prácticas específicas (PS):

- PS 1 Estimar el alcance y definir el ciclo de vida del proyecto.
- PS 2 Estimar esfuerzo y costo del proyecto.
- PS 3 Establecer el cronograma y el presupuesto del proyecto.
- PS 4 Identificar los riesgos del proyecto.
- PS 5 Planificar la administración de datos y recursos necesarios para el proyecto.
- PS 6 Planificar la adquisición de conocimiento y habilidades.
- PS 7 Planificar la participación de los interesados en el proyecto.
- PS 8 Establecer el plan del proyecto.
- SP 9 Revisar todos los planes que puedan afectar al proyecto.
- SP 10 Obtener compromisos respecto al plan.

Monitoreo y Control del Proyecto

Los objetivos que persigue el área de proceso Monitoreo y Control del Proyecto son: Monitorear el proyecto y gestionar acciones correctivas cuando los resultados o la performance del proyecto se desvían significativamente del plan.

Para cumplir con estos objetivos es necesario realizar las siguientes prácticas específicas (PS):

- PS 1 Monitorear los parámetros de planificación del proyecto.
- PS 2 Monitorear los compromisos y los riesgos del proyecto.
- PS 3 Monitorear la administración de datos del proyecto.
- PS 4 Monitorear la participación de los interesados.
- PS 5 Conducir revisiones de avance y de cumplimiento de los hitos.
- PS 6 Administrar acciones correctivas.

Administración de Acuerdos con Proveedores

Los objetivos que persigue el área de proceso Administración de Acuerdos con Proveedores son: Establecer y satisfacer acuerdos con proveedores.

Para cumplir con este objetivo es necesario realizar las siguientes prácticas específicas (PS):

- PS 1 Determinar tipo de adquisición.
- PS 2 Seleccionar proveedores.
- PS 3 Establecer acuerdos con proveedores.
- PS 4 Revisar productos adquiridos.
- PS 5 Ejecutar acuerdos con proveedores.
- PS 6 Aceptar el producto adquirido.
- SP 7 Transición de productos.

Áreas de Proceso de Soporte

Medición y Análisis

Los objetivos que persigue el área de proceso Medición y Análisis son: Alinear las actividades de medición y análisis con los objetivos y necesidades de información y proveer los resultados de la medición.

Para cumplir con estos objetivos es necesario realizar las siguientes prácticas específicas (PS):

- PS 1 Establecer objetivos de las mediciones.
- PS 2 Especificar métricas.
- PS 3 Especificar procedimientos de análisis, recolección y almacenamiento de datos.
- PS 4 Recolectar y analizar datos.
- PS 5 Almacenar datos y resultados.
- PS 6 Comunicar resultados.

Aseguramiento de la Calidad

Los objetivos que persigue el área de proceso Aseguramiento de la Calidad son: Evaluar objetivamente la adhesión de los procesos y artefactos a los estándares y descripciones de proceso vigentes y proveer realimentación de manera que el no cumplimiento de los estándares y descripciones de proceso es objetivamente comunicado y su resolución asegurada.

Para cumplir estos objetivos se deben realizar las siguientes prácticas específicas (PS):

- PS 1 Evaluar procesos objetivamente.
- PS 2 Evaluar productos y servicios objetivamente.
- PS 3 Comunicar y asegurar la resolución de cuestiones de calidad.
- PS 4 Establecer y mantener registros de las actividades de aseguramiento de la calidad.

Administración de la Configuración

Los objetivos que persigue el área de proceso Administración de la Configuración son: Establecer las líneas bases y monitorear y controlar los cambios a los artefactos.

Para cumplir con estos objetivos es necesario realizar una serie de prácticas específicas (PS):

- PS 1 Identificar ítems de configuración.
- PS 2 Establecer un sistema de administración de la configuración.
- PS 3 Crear o liberar líneas base.
- PS 4 Monitorear pedidos de cambio.

- PS 5 Controlar ítems de configuración.

2.3.2 Lineamientos de Calidad de Software

El Aseguramiento de la Calidad de las producciones software es una de las prioridades de la Universidad. Para ello se hace necesario contar con lineamientos, procedimientos y estándares que normen las producciones. En este contexto se suscriben los Lineamientos de Calidad que constituyen una guía a seguir por los proyectos de desarrollo de software atendiendo a cuáles son los artefactos que deben generar y procesos que deben realizar para aspirar a un producto final con calidad.[19]
El cumplimiento de estos lineamientos es obligatorio para todos los proyectos de desarrollo de software de la UCI. Para alcanzar estos lineamientos es necesario conseguir un grupo de objetivos para lo cual se plantean un conjunto de actividades a realizar por cada flujo de trabajo (FT) que quedarán resumidas a continuación.

Tabla 4 Actividades por FT-Lineamientos de Calidad de Software UCI

LINEAMIENTOS DE CALIDAD DE SOFTWARE UCI	
FLUJOS DE TRABAJO	ACTIVIDADES
REQUISITOS	Gestionar requerimientos
	Desarrollar requisitos
ANALISIS Y DISEÑO	Definir arquitectura de software
	Definir arquitectura de información
	Definir modelo de diseño
IMPLEMENTACION	Definir estándares para el desarrollo del proyecto
PRUEBAS	Planificar y ejecutar pruebas del proyecto
	Efectuar seguimiento de las No Conformidades
	Diseñar artefactos de prueba
GESTION DE PROYECTO	Definir equipos de proyecto
	Cuidar los bienes del cliente
	Definir visión del proyecto
	Estimar costo y esfuerzo del proyecto
	Desarrollar plan del proyecto
	Definir plan de resultados del proyecto
	Establecer horarios de trabajo de los miembros del proyecto
	Identificar riesgos del proyecto y establecer plan de mitigación de los mismos
	Gestión de recursos
	Registrar los acuerdos de trabajo y las minutas de las reuniones

SOPORTE	Planificar capacitación para el personal del proyecto
	Establecer registro de resultados de investigación del proyecto
	Definir roles y responsabilidades
	Definir estándares para el desarrollo del proyecto
	Plan de aseguramiento de la calidad
	Definir Plan de Gestión de la Configuración de Software
	Definir glosario de términos
	Definir proyecto técnico para el proyecto

El cumplimiento explícito de estas actividades por parte de todos los proyectos productivos de la Universidad, hace que sean de carácter obligatorio los siguientes artefactos:

- Plan de Capacitación
- Roles y Responsabilidades
- Plan de Gestión de Requerimientos
- Especificación de Requisitos de Software
- Documento de Arquitectura de Software
- Documento de Arquitectura de Información
- Modelo de Diseño
- Modelo del Sistema
- Diseño de Casos de Prueba
- Plan de Pruebas
- No conformidades
- Lista de Chequeos
- Documento Visión
- Plan de Desarrollo de Software
- Presupuesto

- Plan de Mitigación de Riesgos
- Plan de Aseguramiento de la Calidad
- Plan de Gestión de la Configuración
- Minuta de Reunión
- Glosario de Términos

2.3.3 Adaptación de la Metodología OpenUp/Basic

Después de haber establecido las pautas que debe cumplir la metodología utilizada por el PPB, siguiendo las normas que propone CMMI para alcanzar un nivel de capacidad superior al actual y los Lineamientos de Calidad que establece la Universidad de las Ciencias Informáticas, se exponen las adaptaciones que sufrirá la Metodología OpenUp/Basic.

Para hacer más comprensible el resultado de dichos cambios, se analizará cada modificación subdividiendo la propuesta de adaptación de la metodología por Flujos de Trabajo (FT). Agregando finalmente las actividades de Soporte que deben realizarse durante todo el ciclo de vida del proyecto y las que generen las necesidades propias del PPB.

En vista de que las modificaciones que van a aplicarse a la Metodología OpenUp/Basic son debidas a las necesidades específicas del PPB, se propone nombrar la metodología resultante **OpenUp/BioInfo**.

FT: Requisitos

Metodología OpenUp/Basic: Actual

Tabla 5 FT Requisitos-OpenUp Actual

Roles	Actividades	Artefactos
Analista	Buscar y registrar requisitos	Glosario de Términos; Modelo de Casos de Uso; Especificación de Requisitos
Analista	Refinar requisitos	Glosario de Términos; Modelo de Casos de Uso; Especificación de Requisitos
Analista	Definir la visión del proyecto	Documento Visión; Glosario de Términos

Actividades Asociadas: CMMI (nivel 2 de Madurez)

- Desarrollar entendimiento común con los responsables de entregar los requerimientos

- Obtener compromiso de requerimientos por parte de los participantes del proyecto
- Administrar los requerimientos a medida que éstos se desarrollan durante el proyecto

Actividades Asociadas: Lineamientos de Calidad

- Desarrollar requisitos
- Gestionar requisitos

Metodología OpenUp/BioInfo: Propuesta

Tabla 6 FT Requisitos-OpenUp Propuesta

Roles	Actividades	Artefactos
Analista	Comprender el problema	Modelo de Dominio
Analista	Desarrollar entendimiento común	Glosario de Términos
Analista	Desarrollar requisitos	Caso de Uso; Modelo de Casos de Uso; Especificación de Requisitos
Analista	Gestionar requisitos	Plan de Gestión de Requisitos

FT: Análisis y Diseño

Metodología OpenUp/Basic: Actual

Tabla 7 FT Análisis y Diseño-OpenUp Actual

Roles	Actividades	Artefactos
Arquitecto	Analizar los requisitos de arquitectura	Documento de Arquitectura de Software
Arquitecto	Desarrollar la arquitectura	Documento de Arquitectura de Software
Arquitecto	Diseñar la solución	Modelo de Diseño

Actividades Asociadas: Lineamientos de Calidad

- Definir arquitectura de software
- Definir arquitectura de información

- Definir modelo de diseño

Metodología OpenUp/BioInfo: Propuesta

Tabla 8 FT Análisis y Diseño-OpenUp Propuesta

Roles	Actividades	Artefactos
Arquitecto	Analizar los requisitos para la arquitectura	Documento de Arquitectura de Software
Arquitecto	Definir arquitectura de información	Documento de Arquitectura de Información; Informe del levantamiento de Información para la Arquitectura
Arquitecto	Desarrollar y definir la arquitectura	Documento de Arquitectura de Software
Arquitecto	Diseñar la solución	Modelo de Diseño

FT: Implementación

Metodología OpenUp/Basic: Actual

Tabla 9 FT Implementación-OpenUp Actual

Roles	Actividades	Artefactos
Desarrollador	Implementar la solución	Implementación; Build

Actividades Asociadas: Lineamientos de Calidad

- Definir estándares para el desarrollo del proyecto

Metodología OpenUp/BioInfo: Propuesta

Tabla 10 FT Implementación-OpenUp Propuesta

Roles	Actividades	Artefactos
Desarrollador	Implementar la solución	Implementación
Desarrollador	Integrar y crear ejecutable	Build

FT: Pruebas

Metodología OpenUp/Basic: Actual

Tabla 11 FT Pruebas-OpenUp Actual

Roles	Actividades	Artefactos
Probador	Diseñar casos de pruebas	Diseño de Casos de Prueba
Probador	Planificar y ejecutar pruebas	Plan de Prueba

Actividades Asociadas: Lineamientos de Calidad

- Diseñar artefactos de prueba
- Planificar y ejecutar pruebas del proyecto
- Efectuar seguimiento de las No Conformidades

Metodología OpenUp/BioInfo: Propuesta

Tabla 12 FT Pruebas-OpenUp Propuesta

Roles	Actividades	Artefactos
Probador	Diseñar casos de pruebas	Diseño de Casos de Prueba
Probador	Planificar y ejecutar pruebas	Plan de Pruebas; No Conformidades
Probador	Efectuar seguimiento de las No conformidades	Respuesta a No Conformidades

FT: Despliegue

OpenUp/Basic no propone ninguna actividad ni artefacto para el FT de despliegue. Esta área es tratada a nivel organizacional o empresarial. Sin embargo, para el PPB es necesario definir actividades para el despliegue e instalación del software.

Metodología OpenUp/BioInfo: Propuesta

Tabla 13 FT Despliegue OpenUp Propuesta

Roles	Actividades	Artefactos
Todos los roles	Identificar y definir Nodos	Modelo de Despliegue
Todos los roles	Efectuar el Despliegue	Modelo de Despliegue

FT: Gestión de Proyecto

Metodología OpenUp/Basic: Actual

Tabla 14 FT Gestión de Proyecto-OpenUp Actual

Roles	Actividades	Artefactos
Líder de Proyecto	Planificar iteraciones	Plan de Iteraciones
Líder de Proyecto	Gestionar iteraciones	Plan de Iteraciones; Lista de Riesgos
Líder de Proyecto	Planificar proyecto	Plan de Desarrollo de Software; Lista de Riesgos
Líder de Proyecto	Evaluar resultados	Plan de Desarrollo de Software; Lista de Riesgos

Actividades Asociadas: CMMI (nivel 2 de Madurez)

- Estimar el alcance y definir el ciclo de vida del proyecto
- Establecer el cronograma y el presupuesto del proyecto
- Identificar los riesgos del proyecto
- Planificar la administración de datos y recursos necesarios para el proyecto
- Planificar la adquisición de conocimiento y habilidades
- Planificar la participación de los interesados en el proyecto
- Monitorear los compromisos y los riesgos del proyecto y administrar acciones correctivas
- Conducir revisiones de avance y de cumplimiento de los hitos

Actividades Asociadas: Lineamientos de Calidad

- Desarrollar plan del proyecto
- Estimar costo y esfuerzo del proyecto
- Definir visión del proyecto
- Gestión de recursos
- Definir plan de resultados del proyecto
- Definir equipos de proyecto y establecer horarios de trabajo de los miembros
- Identificar riesgos del proyecto y establecer plan de mitigación de los mismos
- Registrar los acuerdos de trabajo y las minutas de las reuniones

Metodología OpenUp BioInfo: Propuesta

Tabla 15 FT Gestión de Proyecto-OpenUp Propuesta

Roles	Actividades	Artefactos
Líder de Proyecto	Definir equipos y horarios de trabajo	Roles y Responsabilidades
Líder de Proyecto	Definir visión del proyecto	Documento Visión
Líder de Proyecto	Establecer plan de mitigación de riesgos	Plan de Mitigación de Riesgos
Líder de Proyecto	Establecer presupuesto del proyecto	Presupuesto
Líder de Proyecto	Identificar riesgos del proyecto	Lista de Riesgos
Líder de Proyecto	Planificar la adquisición de conocimientos y habilidades	Plan de Capacitación
Líder de Proyecto	Planificar iteraciones	Plan de Iteraciones
Líder de Proyecto	Planificar proyecto	Plan de Desarrollo de Software; Lista de Hitos de trabajo
Líder de Proyecto	Registrar acuerdos de trabajo y minutas de reunión	Minuta de Reunión

Actividades de Soporte

- Especificar procedimientos de análisis, recolección y almacenamiento de datos
- Recolectar y analizar datos
- Evaluar procesos objetivamente
- Establecer y mantener registros de las actividades de aseguramiento de la calidad
- Evaluar procesos, productos y servicios objetivamente
- Establecer un sistema de administración de la configuración
- Monitorear pedidos de cambio
- Monitorear pedidos de cambio

Metodología OpenUp/BioInfo: Propuesta

Tabla 16 Actividades de Soporte-OpenUp Propuesta

Roles	Actividades	Artefactos
Cualquier Rol	Definir glosario de términos	Glosario de Términos
Líder de Proyecto	Definir plan para el aseguramiento de la calidad	Plan de Aseguramiento de la Calidad
Líder de Proyecto	Definir plan de gestión de la configuración	Plan de Gestión de la Configuración
Líder de Proyecto	Evaluar los procesos, productos y servicios	Lista de Chequeos
Stakeholder	Solicitar pedidos de cambio	Solicitud de Cambios

2.4 Propuesta del expediente de proyecto

Después de haber adaptado la Metodología OpenUp/Basic a las necesidades concretas del PPB, cumpliendo tanto con los lineamientos de calidad de Software de la Universidad como con las prácticas asociadas a los estándares de CMMI para un nivel 2 de madurez, y quedando conformada la metodología OpenUp/BioInfo, es evidente que las actividades que fueron transformadas o incluidas generan también nuevos artefactos que deben ser archivados. Por consiguiente se propone crear un nuevo expediente de proyecto donde se incluyan todos los artefactos resultantes de la nueva metodología, utilizando para ello, la misma estructura del expediente de proyecto UCI que agrupa los artefactos por categorías que responden a las siguientes definiciones:

- Ingeniería: incorpora todos los artefactos de los flujos de trabajos ingenieriles, dígase requisitos, análisis y diseño, implementación, pruebas y despliegue.
- Gestión de Proyecto: incorpora todos los artefactos relacionados con la gestión del proyecto durante todo el ciclo de vida, relacionándolos por subcategorías tales como planes de proyecto, riesgos, recursos, acuerdos de trabajo y reuniones.
- Soporte: incorpora todos los artefactos resultantes de las actividades de soporte, dígase gestión de la configuración y aseguramiento de la calidad.

2.4.1 Expediente de Proyecto BioInfo

Siguiendo la configuración del expediente UCI el expediente de proyecto del PPB para la metodología OpenUp/BioInfo quedará conformado de la siguiente forma:

1. Ingeniería

1.1. Requisitos

- ❖ Especificación de Requisitos de Software
- ❖ Modelo de Casos de Uso del Sistema
- ❖ Modelo de Dominio
- ❖ Plan de Gestión de Requisitos

1.2. Arquitectura y diseño

- ❖ Documento de Arquitectura de Información
- ❖ Documento de Arquitectura de Software
- ❖ Informe del levantamiento de Información para la Arquitectura
- ❖ Modelo de Diseño

1.3. Implementación y pruebas

1.3.1. Build

1.3.2. Implementación

- ❖ Diseño de Casos de Pruebas
- ❖ No Conformidades
- ❖ Plan de Pruebas

1.4. Despliegue e instalación

- ❖ Modelo de Despliegue

2. Gestión de Proyecto

2.1. Plan de Proyecto

- ❖ Plan de Desarrollo de Software

2.2. Riesgos

- ❖ Plan de Mitigación de Riesgos

2.3. Recursos

- ❖ Plan de Capacitación
- ❖ Roles y Responsabilidades

2.4. Acuerdos de Trabajo

- ❖ Documento Visión

2.5. Reuniones

- ❖ Minuta de Reunión

3. Soporte

3.1. Aseguramiento de la Calidad

- ❖ Glosario de Términos
- ❖ Lista de Chequeos
- ❖ Plan de Aseguramiento de la Calidad

3.2. Gestión de la Configuración

- ❖ Plan de Gestión de Configuración
- ❖ Solicitud de Cambio

2.4.2 Comparaciones entre los expedientes de proyectos OpenUp/Basic, OpenUp/BioInfo y UCI v2.0

Debido a las actividades que se incorporaron en la metodología propuesta, y para cumplir con los requerimientos antes señalados, fue necesario incluir algunos artefactos (12 en total), prácticamente de manera obligatoria, al expediente de proyecto que proponía OpenUp/Basic. Sin embargo, el numero de artefactos resultantes en el nuevo expediente, sigue estando razonablemente por debajo del expediente de proyecto UCI v2.0 (17 artefactos menos). Los resultados se muestran a continuación:

Tabla 17 Comparaciones entre los expedientes de proyecto OpenUp Actual, BioInfo y UCI v2.0

	Expediente de Proyecto			
Criterio	**OpenUp Actual**	**OpenUp Propuesta (BioInfo)**	**UCI v2.0**	**Criterios favorables de la Propuesta**
Cantidad de Artefactos	13	25	42	Reduce la documentación en 17 artefactos con respecto al Expediente UCI.
Cumple con Los Lineamientos de Calidad UCI	no	si	si	Logra cumplir con los lineamientos de calidad de la Universidad con menos documentación(17 artefactos menos)
Se ajusta a las características del PPB	parcialmente	si	parcialmente	Reduce la documentación y cumple con los lineamientos de calidad de la Universidad

Para hacer mayor énfasis en las diferencias entre expediente de proyecto propuesto para el PPB y el de la UCI v2.0 se describe a nivel de artefactos y organizados por las categorías en que estos se agrupan, los cambios perceptibles entre ambos expedientes. (Ver tabla 18).

Tabla 18 Comparación de los expedientes de proyecto BioInfo y UCI v2.0 por categorías

Categorías		Expediente de Proyecto	
		BioInfo	UCI v2.0
Ingeniería	Requisitos	4	10
	Arquitectura y diseño	4	4
	Implementación y Pruebas	5	7
	Despliegue	1	1
Gestión de Proyecto	Plan de Proyecto	1	5
	Riesgos	1	1
	Recursos	2	3
	Acuerdos de Trabajo	1	2
	Información del cliente	0	0
	Informes	0	1
	Reuniones	1	1
Soporte	Aseguramiento de la Calidad	3	4
	Gestión de la Configuración	2	3
	TOTAL	25	42
Totales por Categorías	Ingeniería	14	22
	Gestión de Proyecto	6	13
	Soporte	5	7

2.5 Descripción de la herramienta EPF Composer

En las empresas actuales es bastante importante la correcta definición y documentación de los procesos y prácticas que se llevan a cabo. Puede ocurrir que dicha documentación no esté organizada o no esté escrita en un formato estandarizado, dificultando el acceso y aumentando los tiempos necesarios para comprender la información, haciendo que los equipos de trabajo tarden más en la capacitación de los nuevos miembros. Además, los equipos de trabajo necesitan saber donde aplicar esos conocimientos y prácticas dentro del proceso de desarrollo. Para sobrellevar estas dificultades se han diseñado herramientas que permiten definir, documentar y publicar procesos, dentro de las cuales

se encuentra Eclipse Process Framework Composer (EPF Composer), una herramienta libre soportada por la comunidad Eclipse.

2.5.1 Generalidades

Eclipse Process Framework Composer (EPF Composer) es una herramienta para ingenieros de procesos, líderes y administradores de proyectos, quienes son responsables de mantener e implementar procesos para organizaciones dedicadas al desarrollo o para proyectos individuales. Hay dos problemas típicos que deben ser tratados a la hora de llevar a cabo la implementación de un proceso.

Primero, los desarrolladores deben entender los métodos y las prácticas del desarrollo de software. Ellos deben estar familiarizados con las tareas básicas del desarrollo, tales como el manejo de requisitos, el análisis y el diseño, etc. Para lograr esto muchas empresas se basan en la documentación explícita de los procesos y la enseñanza de tales métodos para establecer prácticas comunes y reguladas.

Segundo, los equipos de desarrollo deben definir cómo se aplican sus métodos de desarrollo y mejores prácticas a través del ciclo de vida de un proyecto, es decir, ellos deben definir o seleccionar un proceso de desarrollo. Además deben entender claramente cómo las diferentes tareas definidas en los métodos se relacionan entre sí: por ejemplo, cómo el método de administración de cambios impacta el método de administración de requisitos. Incluso los equipos de trabajo dentro de una misma organización necesitan definir un procesos que les de alguna guía sobre cómo será delimitado el desarrollo a través del ciclo de vida, cuándo se lograrán y verificarán los hitos, etc. [20]

EPF Componer tiene dos propósitos principales que apuntan a las dos necesidades mencionadas:

- Proveer a los desarrolladores con una base de conocimiento de capital intelectual que les permita buscar, administrar y desplegar contenido. Esta base de conocimiento puede ser usada como referencia y material educativo, y forma la base para el proceso de desarrollo (el segundo propósito). EPF Composer está diseñado para ser un administrador de contenido que provee una estructura de administración y un aspecto comunes para todo el contenido, en vez de ser un sistema administrador de documentos en el cual se almacenan y se acceden documentos difíciles de mantener, que tienen cada uno su propia forma y formato.
- Proveer capacidades de ingeniería de procesos para la selección, adecuación y rápido ensamblado de procesos para proyectos de desarrollo concretos. EPF Composer tiene

catálogos de procesos predefinidos para situaciones típicas de procesos que pueden ser adaptados a necesidades individuales. [20]

Estos dos propósitos se ven reflejados en el principio fundamental de EPF Composer que es la separación del contenido reusable del método de su aplicación en los procesos.

El proyecto de Eclipse Process Framework tiene como objetivo satisfacer algunas necesidades comunes que los líderes y equipos de desarrollo enfrentan cuando están asimilando y administrando métodos y procesos. Por ejemplo:

- Los equipos de desarrollo necesitan un acceso fácil y centralizado a la información.
- Los contenidos de los procesos de desarrollo deben estar en formatos estándar que permitan una fácil integración.
- Los equipos de desarrollo requieren una base de conocimiento actualizada para que ellos mismos aprendan sobre métodos y mejores prácticas.
- Los equipos de desarrollo necesitan soporte para dimensionar correctamente sus procesos.
- Los equipos de desarrollo requieren la habilidad de estandarizar prácticas y procesos dentro de las organizaciones.
- Los equipos de desarrollo necesitan cerrar la brecha entre la ingeniería de procesos y el establecimiento de los mismos en las organizaciones por medio del uso de representaciones y terminologías similares. [20]

2.5.2 Requerimientos e instalación

Para utilizar EPF Composer sólo se requiere tener instalada la máquina virtual de Java (JRE Java Runtime Environment) 1.4.2 o superior. Una vez se tiene instalado JRE, se puede descargar la herramienta desde la página Web del proyecto EPF de Eclipse. Luego de que se descarga el archivo, este se descomprime (preferiblemente en la raíz del disco) y para ejecutar la aplicación se da doble clic sobre el archivo epf.exe. [20]

2.5.3 Escenarios de uso y perspectivas

Existen cuatro escenarios típicos en los cuales se usa EPF Composer.

- **Seleccionar y configurar contenidos de método y procesos existentes:** este es un escenario en el cual se seleccionan los procesos y el contenido del método que se ajustan a unas necesidades específicas por medio de búsquedas en la librería de método de EPF, la cual

contiene todos los contenidos que son comprados (contenidos comerciales) al igual que los contenidos que se descargan gratuitamente de la comunidad de EPF. Una vez se tienen los contenidos que más se ajustan, se empieza a configurar un proceso seleccionando o removiendo paquetes de método.

- **Adecuar un proceso existente:** en este escenario en vez de configurar los procesos existentes para decidir que se muestra y que no, éstos se modifican, por medio del uso de los editores de EPF Composer, para que se ajusten incluso mejor a las necesidades específicas. Es decir, se puede directamente adicionar, remover o reemplazar elementos en el proceso. Por lo tanto, en contraste con cambiar la configuración -lo cual hace cambios globales en el proceso- se pueden definir cambios individuales sólo en los lugares requeridos.
- **Crear un nuevo proceso:** este escenario se presta para crear un proceso completamente nuevo en el cual se reutilicen actividades de uno ó más procesos existentes. En los casos donde no se puede hallar ningún material reusable, también se puede crear un proceso completamente nuevo desde cero.
- **Desarrollar contenido de método y crear o extender procesos:** el último escenario se refiere a la habilidad de no sólo crear o adecuar procesos reutilizando el contenido del método de EPF o de terceros, sino también a la habilidad de desarrollar contenido del método propio y usarlo en la adecuación de procesos existentes o la creación de nuevos procesos.[20]

Una perspectiva define un conjunto de vistas y funciones disponibles para tareas específicas. EPF Componer contiene dos perspectivas de trabajo: una perspectiva para la edición (Authoring) y otra perspectiva para la navegación o exploración (Browsing).

- **Authoring:** la perspectiva Authoring o de edición, provee de vistas y funciones para navegar y editar o crear contenido de métodos y procesos. Para poder crear o editar cualquier tipo de elemento se debe establecer la perspectiva Authoring. La perspectiva Authoring provee dos vistas en paneles separados: La vista de librería y la vista de configuración. Al hacer doble click sobre cualquier elemento de alguna de esas vistas, se abre el panel de edición en la derecha. El panel de edición contiene varios tabs a través de los cuales se puede editar información sobre el elemento que ha seleccionado. Para seleccionar la perspectiva de Authoring, use el menú “Abrir perspectiva” en la barra de tareas principal y seleccione Authoring
- **Browsing:** la perspectiva Browsing (Exploración) le permite hacer una vista previa y navegar a través de una configuración del método sin hacer ningún cambio. La perspectiva Browsing contiene la vista de configuración, que muestra el contenido en la configuración seleccionada.

Haga clic sobre cualquier elemento en el panel de configuración para hacer una vista previa del elemento en la vista de contenido mostrando como dicho elemento aparecerá publicado en un sitio Web. La vista de contenido provee funcionalidades de navegación similares a las de un Explorador o Browser. Haga click sobre cualquier vínculo para ir a la página referenciada. Use los botones en la barra de herramientas de la vista de contenido para realizar funciones similares a las de un Browser, por ejemplo, ir atrás o refrescar. Para seleccionar la perspectiva Browsing, haga clic sobre el botón “Abrir Perspectiva” y seleccione Browsing. [20]

2.5.4 Composición de procesos

El objetivo primordial de la composición de procesos, es dividir el contenido en una parte general y reusable, que puede ser usada a través de los procesos y otra más específica y aplicada al proyecto que se está desarrollando. [20]

Librería de métodos (Method Library)

Una librería de métodos es un contenedor físico de plug-ins de método y definiciones de configuración de métodos. Todos los elementos del método están almacenados en una librería de métodos. Así como una librería tiene libros, una librería de métodos tiene plug-ins de método. Donde una librería de libros está hecha de secciones o capítulos y contenido dentro de esos capítulos, los plug-ins de métodos están compuestos por contenido de método y procesos. El contenido del método contiene paquetes de contenido y categorías, que pueden ser estándares o personalizadas, mientras que los procesos estructuran este contenido en fragmentos de proceso llamados patrones de capacidad y procesos de ciclos de vida completos llamados procesos de entrega.

Una librería de método también tiene una o más configuraciones de método, que filtran la librería y proveen de conjuntos de trabajo más pequeños de contenido de la librería para el usuario final. [20]

Contenido del método (Method Plug-in)

El contenido del método provee explicaciones paso a paso, describiendo como son alcanzadas las metas de desarrollo específicas, independientemente de la ubicación de esos pasos dentro de un ciclo de vida de desarrollo. Los procesos toman esos elementos del método y los relacionan en secuencias semiordenadas que son personalizadas para tipos específicos de proyectos.

Un ingeniero de proceso crea esos elementos, define las relaciones entre ellos, y luego los categoriza. Por ejemplo, un proyecto de desarrollo de software que desarrolle una aplicación desde cero, realiza tareas de desarrollo como “Desarrollar Visión” o “Diseñar casos de uso” similar a un proyecto que extiende un sistema de software existente. De todas formas, los dos proyectos realizarán las

actividades en diferentes puntos del tiempo con un énfasis diferente, por ejemplo, realizan los pasos de esas tareas en puntos diferentes del ciclo de vida del desarrollo y tal vez aplican variaciones y adiciones. [20]

2.5.5 Elementos del contenido del método

Los elementos del contenido del método están organizados dentro de paquetes, los que a su vez están dentro de un plug-in de método. Con el fin de separar el contenido propio del contenido BasicUP original, se debería siempre crear nuevo contenido de método en un plug-in de método que se produzca. Crear contenido de método en un plug-in de método propio, permite además la actualización de la librería propia con nuevas versiones de la librería BasicUP sin afectar el contenido que haya creado.

Cada elemento posee varias vistas donde se puede editar la información de los mismos.

- **Categorías:** el contenido del método está organizado en categorías lógicas. Las categorías pueden aparecer en el sitio publicado como vistas. Hay dos tipos de categoría: estándares y personalizadas.
- **Roles:** un rol define un conjunto de habilidades relacionadas, competencias y responsabilidades de uno o varios individuos.
- **Disciplinas (flujos de trabajo):** una disciplina es una colección de tareas que están relacionadas a un área mayor de interés dentro del ambiente de desarrollo. Por ejemplo, en un proceso de desarrollo de software, es común realizar ciertas tareas de requerimiento de forma coordinada con tareas de análisis y diseño. Separando esas tareas en disciplinas separadas, las hacen más fácil de comprender. Las disciplinas pueden ser organizadas usando Agrupaciones de Disciplinas.
- **Tareas:** una tarea es una unidad asignable de trabajo. Cada tarea es asignada a un rol específico. La granularidad de una tarea se puede determinar por su tiempo de duración, que varía generalmente entre unas cuantas horas y unos pocos días y usualmente afecta uno o sólo un pequeño número de productos de trabajo.
- **Dominios:** un dominio es una jerarquía lógica y refinable de productos de trabajo relacionados, agrupados juntos basándose en los cronogramas, recursos o relación. Mientras que un dominio categoriza muchos productos de trabajo, un producto de trabajo pertenece solo a un único dominio. Los dominios pueden ser subdivididos en subdominios.

- **Productos de trabajo:** un producto de trabajo es un término general para las entradas y salidas de una tarea, descripciones de elementos de contenido que son usados para definir cualquier cosa usada, producida o modificada por una tarea.
- **Guías:** guía es un término general para información complementaria que puede ser agregada a la mayoría de elementos de métodos y procesos. Agregar guías es una forma fácil de tejer información para proyectos específicos. Por ejemplo, con una guía se podría explicar cómo se utiliza un producto de trabajo dentro del proyecto.
- **Proceso:** un proceso describe como una pieza particular de trabajo debería ser hecha. El trabajo puede tener un alcance relativamente pequeño, en donde podría ser descrito como un patrón de capacidad (fase), o podría apuntar al ciclo de vida de un proyecto completo, en donde podría ser descrito como un proceso de entrega (ciclo de vida). Un proceso puede reutilizar elementos de método y combinarlos en una estructura y secuencia para llevar a cabo el trabajo. Hay dos tipos principales de procesos, patrón de capacidad y proceso de entrega. Cada vez que una tarea es incluida en un proceso, un objeto referencia a esa tarea es creado en el contexto del proceso. Este objeto es llamado un descriptor de tarea. La misma tarea puede ser referenciada cualquier número de veces en el mismo proceso. En otras palabras, una tarea puede tener muchos descriptores de tarea. Un descriptor de tarea puede también modificar la tarea base sin realmente cambiar la tarea. Por ejemplo, roles y productos de trabajo pueden ser agregados o eliminados, y pasos pueden ser eliminados o re-secuenciados. Los roles y productos de trabajo pueden ser incluidos en procesos como descriptores de roles y descriptores de productos de trabajo. Los roles y productos de trabajo pueden ser personalizados para encajar en el contenido del proceso en el cual están siendo usados. [20]

2.5.6 Configuración del método

Las configuraciones del método son subconjuntos de elementos contenidos en la librería del método definidos por el usuario.

Una configuración del método define un conjunto de paquetes de trabajo dentro de la librería del método que ayuda a limitar la vista a un conjunto de elementos. Los elementos que hacen parte de una configuración seleccionada son mostrados en la vista de configuración. Las configuraciones del método son usados para la creación de procesos y la publicación, definiendo cuáles elementos son publicados en HTML y cuáles no.

Una configuración de un método se conforma de tres partes:

- Una descripción de la configuración
- Una selección del conjunto de paquetes
- Una selección de vistas que serán publicadas en el sitio Web

En un configuración del método se puede seleccionar y deseleccionar elementos de los paquetes de contenido y los procesos disponibles los conjuntos de plug-ins de la librería del método. Las selecciones que se hagan ayudan a determinar el contenido del sitio Web publicado. A una configuración se le da un nombre y luego se guarda, así ésta puede, posteriormente, ser modificada y publicada nuevamente.

Las vistas que se asocian a la configuración, se refieren generalmente a elementos que aparecen bajo el tipo de “Categorías” dentro de EPF Composer. [20]

2.5.7 Publicación de contenido

EPF Composer permite publicar los procesos y todo su contenido en un sitio Web para que este pueda ser accedido con facilidad. Para publicar los contenidos en EPF Composer se deben crear antes una ó más configuraciones, ya que es con base a dichas configuraciones que la herramienta genera el esquema de presentación y la organización del contenido en el sitio Web. Una vez se tienen las configuraciones, se indica con base en cual se quiere publicar el contenido y luego la herramienta genera automáticamente todos los archivos necesarios para tener un sitio Web listo para montar en algún servidor Web. [8]

2.6 Descripción del montaje de la metodología propuesta en el EPF Composer

La herramienta EPF Composer ofrece la posibilidad de definir y documentar los procesos y prácticas que se llevan a cabo en un proceso de desarrollo de software organizadamente, creando una base de conocimientos que puede ser usada como referencia y material educativo, constituyendo la plataforma para el proceso de desarrollo, garantizando que esta tenga fácil acceso y centralizando la información.

Después de haber detallado la herramienta EPF Composer, solo queda describir el montaje y la configuración de la metodología OpenUp/BioInfo. Para lograr este objetivo, se dividirá la explicación en 4 puntos fundamentales mediante los cuales se caracteriza completamente una metodología de desarrollo de software.

Estos puntos de análisis son:

- Definición de los roles
- Configuración de los flujos de trabajo

- Definición de los artefactos
- Ciclo de vida

La Metodología OpenUp/BioInfo obedece a los mismos principios básicos de OpenUp/Basic

- **Balance:** de las prioridades en contradicción para maximizar el valor de los Stakeholder.
- **Colaboración:** para alinear los intereses y un entendimiento compartido.
- **Enfoque:** en articular la arquitectura.
- **Evolución:** para obtener continuamente retroalimentación y progreso

Es importante antes de continuar, definir que para realizar el montaje de la metodología se parte del escenario del EPF Composer “Desarrollar contenido de método y crear o extender procesos” (ver epígrafe 2.5.3). Donde se comienza creando una librería de métodos y luego un método de contenido al que serán importados los roles, ciclo de vida, conceptos, artefactos y algunas actividades de OpenUp/Basic como punto de partida para comenzar la adaptación. Esta librería y método de contenido es nombrada como la propuesta de metodología: OpenUp/BioInfo.

Es recomendable definir o concebir todos los contenidos que se incluirán en la composición del proceso de acuerdo a los lineamientos presentes en la documentación de la herramienta sobre cómo definir un proceso.

2.6.1 Definición de los roles

La Metodología OpenUp/BioInfo mantiene los mismos roles que OpenUp/Basic, con la diferencia de que van a realizar más tareas y ocupar más responsabilidades en cuanto a la creación o modificación de artefactos. Este proceso está automatizado por la herramienta ya que cuando se crea o se redefine una tarea, al asignar un rol responsable en la vista de asignación de roles, automáticamente se actualiza este rol con la realización de la nueva tarea y la responsabilidad de los artefactos creados o modificados en la misma. También pueden modificarse los artefactos de los que cada rol es responsable en la vista de productos de trabajo del rol.

Los roles de la Metodología OpenUp/BioInfo son:

- **Analista:** las personas en este rol representan al cliente y los usuarios finales involucrados, obteniendo información desde los stakeholders para entender el problema a ser resuelto y capturar y ajustar las prioridades para los requerimientos.

- **Arquitecto:** este rol es el responsable de diseñar la arquitectura del software, la cual incluye tomar las principales decisiones técnicas que condicionan globalmente el diseño y la implementación del proyecto.
- **Desarrollador:** la persona en este rol es responsable por desarrollar una parte del sistema, incluyendo diseñar esta para que se ajuste a la arquitectura, posiblemente crear un prototipo de la interfaz de usuario y entonces implementar, hacer pruebas unitarias e integrar los componentes que son parte de la solución.
- **Probador:** este rol es responsable de las actividades principales del esfuerzo de las pruebas. Estas actividades incluyen identificar, definir, implementar y dirigir las pruebas necesarias, como también verificar los resultados de las pruebas y analizar los resultados.
- **Líder de proyecto:** lidera la planeación del proyecto, coordina interacciones con los stakeholders y conserva el equipo del proyecto enfocado en alcanzar los objetivos del proyecto.
- **Stakeholder:** este rol representa grupos de interés cuyas necesidades deben ser satisfechas por el proyecto. Esto es un rol que podría ser desempeñado por cualquiera que esté (o potencialmente estará) materialmente afectado por el resultado del proyecto.
- **Cualquier rol:** cualquiera en un equipo puede cumplir este rol para llevar a cabo tareas generales.

En la vista de descripción se puede editar información detallada de los roles tales como las características, habilidades o criterios de asignación que deben estar presentes en cualquier individuo para la representación de un rol.

Finalmente en la vista previa se puede apreciar un esquema compuesto por el rol, las actividades que realiza y los artefactos de los que dicho rol es responsable, además de toda la información de la vista de descripción.

2.6.2 Configuración de los flujos de trabajo

Para organizar las tareas por flujos de trabajo, el primer paso es crear tantos paquetes de contenidos (hay 4 paquetes predefinidos dentro de cada paquete de contenido; roles, tareas, productos de trabajo y orientaciones) como flujos de trabajo existentes, modificando o adicionando las tareas de cada flujo de trabajo según la propuesta de adaptación de las actividades de la metodología OpenUp/Basic descrita con anterioridad (ver epígrafe 2.3.3).

- Para Modificar una tarea existente solo es necesario abrir el panel de edición de la tarea que contiene todas las vistas.
- Para crear una nueva tarea, es necesario posicionarse en el paquete de tareas predefinido en el paquete de contenido creado para cada flujo de trabajo, crear una nueva tarea dentro de este, y automáticamente se desplegará el panel de la nueva tarea que contiene todas las vistas de edición.

Los FT de OpenUp/BioInfo son:

- **FT Requisitos:** el objetivo principal de esta disciplina es establecer las funciones que se quiere que satisfaga el sistema a construir. En esta línea los requerimientos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requerimientos que se especifiquen.
- **FT Análisis y Diseño:** el objetivo principal de esta disciplina es transformar los requerimientos a una especificación que describa cómo implementar el sistema. El análisis fundamentalmente consiste en obtener una visión que se preocupa de ver que hace el sistema de software a desarrollar, por tal motivo este se interesa en los requerimientos funcionales. Por otro lado, el diseño es un refinamiento que toma en cuenta los requerimientos no funcionales, por lo cual se centra en como el sistema cumple sus objetivos.
- **FT Implementación:** el objetivo principal de esta disciplina es convertir los elementos del diseño en elementos de implementación, dichos elementos son códigos fuentes, ejecutables, binarios, entre otros.
- **FT Pruebas:** el principal objetivo de esta disciplina es de evaluar la calidad del producto que se está desarrollando a través de las diferentes fases por las cuales este pasa, mediante la aplicación de pruebas concretas para validar que las suposiciones hechas en el diseño y los requerimientos se estén cumpliendo satisfactoriamente, esto quiere decir que se verifica que el producto funcione como se diseñó y que los requerimientos son satisfechos cabalmente. Esta disciplina brinda soporte para encontrar y documentar (y solucionar) defectos en la calidad del sistema a las otras disciplinas. Esta disciplina debe estar presente en todo el ciclo de vida del desarrollo del sistema para ir refinándolo y no al final del mismo.
- **FT Despliegue:** esta disciplina tiene como objetivo distribuir e instalar con éxito el sistema elaborado por el equipo de desarrollo y asegurar la disponibilidad del producto para los usuarios finales.

- **FT Gestión de Proyecto:** el objetivo de la gestión del proyecto es conseguir alcanzar las metas propuestas con el desarrollo del sistema, administrar el riesgo y superar las restricciones para desarrollar un producto que sea acorde a los requerimientos de los clientes y usuarios.
- **Actividades de Soporte:** son las actividades llevadas a cabo durante todo el ciclo de vida del proceso de desarrollo para medir la calidad y controlar que se cumplan los objetivos de las tareas de los flujos de trabajo.

Entre las vistas de edición de las tareas están:

- **Descripción:** donde se define el nombre de la tarea, los propósitos y las consideraciones a tener en cuenta.
- **Roles:** donde se asigna a los roles la responsabilidad de realizar la tarea.
- **Productos de trabajo:** donde se describe qué productos de trabajo se necesitan para ejecutar la tarea y cuales son obtenidos a partir de la realización de la misma.
- **Orientaciones:** donde se puede establecer un link a algún concepto o directriz importante para la realización de la tarea.
- **Categorías:** donde se asigna la tarea a alguna de las categorías definidas en la configuración (para este caso, a un flujo de trabajo específico). Es importante definir a que categoría pertenece cada tarea ya que este proceso es el que las agrupa en flujos de trabajo para su posterior publicación.

Finalmente en la vista previa se puede apreciar una tabla compuesta por los roles encargados de realizar la actividad, los artefactos vinculados con la misma, y él o los flujos de trabajo a que dicha actividad pertenece.

2.6.3 Definición de los artefactos

Al igual que tareas, los artefactos son creados en diferentes fases del ciclo de vida, independientemente de que puedan ser modificados o no desde su surgimiento hasta que el proceso termina completamente. Por esta razón los artefactos también van a estar agrupados por flujos de trabajo dependiendo de en qué etapa surgen, para lo cual se utilizarán los mismos paquetes de contenido (flujos de trabajos) que se crearon para las tareas.

- Para Modificar un artefacto existente solo es necesario abrir el panel de edición del artefacto que contiene todas las vistas que se pueden modificar.

- Para crear un nuevo artefacto, es necesario posicionarse en el paquete de productos de trabajo predefinido en el paquete de contenido creado para cada flujo de trabajo, crear una nueva tarea dentro de este, y automáticamente se desplegará el panel de la nueva tarea que contiene todas las vistas de edición.

Es importante acentuar que no todos los artefactos que se describen en el montaje están obligatoriamente reflejados en un solo documento, ya que pueden formar parte o estar incluidos dentro de otros más generales. A esto se debe que la cantidad de templates (plantillas) y el número real de artefactos en el expediente de proyecto de OpenUp/BioInfo, sea inferior al total de todos los que se nombran en el montaje de la propuesta. La metodología OpenUp/BioInfo cuenta en total con 25 artefactos que están recogidos y debidamente organizados en el expediente de proyecto. Para cada uno de estos artefactos se genero también un template o plantilla que además de recoger las características y la descripción del artefacto, contiene un link a una plantilla descargable en *.doc o *.odt, dependiendo del sistema operativo que se esté utilizando.

Un artefacto está compuesto por las siguientes vistas:

- **Descripción:** donde se define el nombre del artefacto, la descripción y los propósitos del mismo.
- **Orientaciones:** donde se puede establecer un link, generalmente aquí se asocia al template (plantilla).
- **Categorías:** donde se asigna el artefacto a alguna de los dominios definidos en la configuración (para este caso, a un flujo de trabajo específico). Es importante definir a que dominio pertenece cada artefacto, aunque la configuración de la vista para la publicación posterior del expediente de proyecto difiere un poco de la organización por flujos de trabajo.

Finalmente en la vista previa se puede apreciar una tabla compuesta por los roles responsables del artefacto, la tarea que es elaborado, la fase o fase en la que es utilizado o modificado, y el dominio al que pertenece además de toda la información de la vista de descripción y la asociación con el template equivalente.

Un template (plantilla) está compuesto por la vista de descripción en las que se definen las características y objetivos, que deben ser los mismos del artefacto asociado, y un editor que posibilita adjuntar archivos, que en este caso serian las planillas en los formatos antes descritos.

Para la conformación de la vista del expediente de proyecto en el EPF Composer se utilizó la estructura del expediente de proyecto propuesto (ver epígrafe 2.4.1).

2.6.4 Ciclo de Vida

El ciclo de Vida de OpenUp/BioInfo está compuesto por las mismas fases que OpenUp/Basic.

- **Inicio:** el propósito en esta fase es lograr concurrencia entre todos los stakeholders sobre los objetivos del ciclo de vida para el proyecto.
- **Elaboración:** el propósito de esta fase es establecer la línea base de la arquitectura del sistema y proporcionar una base estable para el esfuerzo de desarrollo de la siguiente fase.
- **Construcción:** El propósito de esta fase es completar el desarrollo del sistema basado en la arquitectura.
- **Transición:** El propósito en esta fase es asegurarse que el software está listo para entregarse a los usuarios.

La herramienta EPF Composer tiene definidos dos tipos de procesos: patrones de capacidad y procesos de entrega. Estos procesos son bastante similares a las fases y el ciclo de Vida respectivamente. Para entender el porqué de esta equivalencia se describe el concepto de cada uno de ellos.

- **Patrones de capacidad (capability patterns):** un patrón de capacidad es un proceso especial que describe un conjunto de actividades reusables en procesos de áreas comunes. Los patrones de capacidad expresan y comunican conocimiento de proceso para un área clave de interés como una disciplina y pueden ser directamente usadas por el practicante de un proceso para guiar su trabajo. Los patrones de capacidad también son usados como bloques de construcción para ensamblar procesos de entrega o patrones de capacidad más grandes asegurando reutilización óptima y aplicaciones de prácticas clave que ellos expresan.
- **Proceso de entrega (delivery processes):** un proceso de entrega es el proceso que cubre todo el ciclo de vida del desarrollo desde el principio hasta el fin. Un proceso de entrega puede ser usado como plantilla para planear y correr un proyecto. Provee un modelo de ciclo de vida completo con fases, iteraciones y actividades predefinidas.

Utilizando la estructura de procesos que provee el EPF Composer, se creó un paquete de procesos y dentro de este, cuatro patrones de capacidad respondiendo a cada una de las fases del ciclo de vida de OpenUp/BioInfo. También se creó un proceso de entrega con una estructura de desglose extendida a partir de estos patrones de capacidad que responde al ciclo de vida de OpenUp/BioInfo. Recordemos que los patrones de capacidad también son usados como bloques de construcción para ensamblar procesos de entrega.

Dentro de las vistas de edición de los patrones de capacidad y de los procesos de entrega se encuentran:

- **Descripción:** donde se nombra y se describe el patrón de capacidad
- **Estructura de desglose de trabajo (work breakdown structure):** donde se describe las actividades, tareas e hitos del patrón de capacidad. Esta es la vista principal ya que es donde se edita toda la información referente a la fase que se describa.
  - **Asignación de equipo (team allocation):** donde se muestran los roles responsables de realizar la actividad.
  - **Producto de trabajo usado (work product usage):** donde se muestran los artefactos que son usados en cada actividad
  - **Vista consolidada (consolidated view):** es una visión general de la fase, donde se muestran las actividades, tareas, roles y artefactos.

Estas tres últimas vistas se actualizan de forma automática a partir de la vista estructura de desglose de trabajo.

Vista de estructura de desglose de trabajo (work breakdown structure)

Esta vista es una lista jerárquica de trabajo, que contiene actividades, tareas y pasos, definiendo un proceso. En esta lista son descritos los procesos y pueden ser creadas iteraciones y actividades, y dentro de éstas se pueden agregar tareas desde el contenido de método. Está conformada básicamente por una tabla con varias columnas. Una para nombrar los elementos, una para el indexado, donde a cada elemento se le asigna un número (en lista ascendente), una para indicar predecesores a partir del número indexado a cada elemento, otra para especificar el tipo de elemento y el resto para definir si el elemento es planeado, repetible, opcional etc. Para la construcción de patrones de capacidad y de procesos de entrega es importante conocer varios elementos que esta vista proporciona.

Fase: es un tipo especial de actividad que representa un período de tiempo significante en un proyecto, terminando con un punto clave de gestión, hito o conjunto de entregables.

Iteración: es un grupo de actividades anidadas que son repetidas más de una vez. Representa un elemento estructural importante para organizar el trabajo en ciclos repetitivos.

Actividad: las actividades representan los bloques clave de construcción para los procesos. Las actividades representan un grupo de elementos listados como otras actividades, descriptores de

tareas, descriptores de roles, descriptores de productos de trabajo e hitos. Adicionalmente a las estructuras de listado, las actividades pueden también ser presentadas en diagramas de actividad que gráficamente muestran el flujo de trabajo exponiendo cuales actividades preceden otras actividades. Fases e iteraciones son tipos especiales de actividades que definen propiedades específicas.

Descriptor de Tarea: es un objeto que referencia a una tarea creado cada vez que dicha tarea es incluida en un proceso. La misma tarea puede ser referenciada cualquier número de veces dentro del mismo proceso, es decir, una tarea puede tener muchos descriptores de tarea. Un descriptor de tarea puede también modificar la tarea base sin realmente cambiarla.

Milestone (o hito): describe un evento significativo en un proyecto, como una decisión importante, la terminación de un entregable, o al encontrarse con una dependencia mayor, como la finalización de una fase del proyecto.

2.6.5 Configuración del método y publicación

Las configuraciones del método son subconjuntos de elementos contenidos en la librería del método definidos por el usuario para precisar que contenido se quiere publicar (ver epígrafe 2.5.6 y 2.5.7).

Para la publicación del método, en el menú de la barra de tareas de EPF Composer, se encuentra una opción bajo la etiqueta “Configuration” que posibilita seleccionar la configuración de la vista para mostrar el método y finalmente publicar el contenido como un sitio Web. La Herramienta EPF Composer también facilita la posibilidad de exportar dicho contenido como XML, Microsoft Project, métodos de contenido o librería de configuración.

2.7 Validaciones

Con el objetivo de establecer resultados visibles de la propuesta de adaptación de la Metodología OpenUp/Basic y partiendo de la dificultad temporal que representa implantar la Metodología OpenUp/BioInfo a los proyectos del PPB y probarla mediante el desarrollo a ciclo completo de varios productos, se realizaron entrevistas a los líderes de proyecto y asesores de calidad del PPB, planteándoles la nueva propuesta y exponiendo todo el proceso para llegar a ella (investigación, comparaciones, modificaciones, etc.). Estas entrevistas tienen como meta fundamental obtener las opiniones, criterios, sugerencias y la aprobación de la Metodología OpenUp/BioInfo por parte de los máximos responsables del desarrollo y calidad de los productos obtenidos por los proyectos del PPB. Las entrevistas están dirigidas a los puntos fundamentales:

- Fundamentación del uso de las metodologías ágiles, específicamente OpenUp/Basic.
- Caracterización y necesidades identificadas del PPB.

- Adaptación de la Metodología OpenUp/Basic a las características y necesidades del PPB.
- Cumplimiento de la metodología adaptada (OpenUp/BioInfo) con los Lineamientos de Calidad de Software de la UCI.
- Cumplimiento de la metodología adaptada con las actividades que proponen las áreas de proceso para alcanzar el nivel 2 de madurez según CMMI.
- Presentación de la ayuda de OpenUp/BioInfo, EPF Composer como gestor de modificaciones.
- Caracterización y estructura del expediente de proyecto propuesto por la metodología adaptada.
- Comparaciones entre el expediente de proyecto de OpenUp/BioInfo y el expediente de proyecto UCI v2.0 vigente en la universidad.
- Aceptación de la Metodología OpenUp/BioInfo.

Para lo cual se formularon las siguientes preguntas:

- ¿Se justifica mediante la investigación realizada a los procesos y metodologías de desarrollo de software y las características del PPB el uso de OpenUp/Basic?
- ¿Se considera necesario realizar modificaciones a la metodología OpenUp/Basic usada actualmente para adaptarla a las características específicas del PPB?
- ¿La Metodología OpenUp/BioInfo cumple con los Lineamientos de Calidad de Software de la UCI y las métricas de CMMI para el nivel 2 de madurez y se a las necesidades y características especificas del PPB?
- ¿El expediente de proyecto que propone OpenUp/BioInfo optimiza el trabajo de los proyectos del PPB y posee todos los artefactos de carácter obligatorio que señalan los Lineamientos de Calidad de Software de la UCI?
- ¿El sitio de ayuda de OpenUp/BioInfo resuelve las problemáticas que pudieran plantearse los integrantes del PPB o interesados en la metodología y define con claridad los procesos, roles y artefactos?
- ¿Se valora lo posibilidad de Implantar entonces OpenUp/BioInfo al PPB?

2.7.1 Resultados de las Entrevistas

Las entrevistas realizadas a los líderes de proyecto y los asesores de calidad aunque no garantizan la liberación de la propuesta, arrojaron un resultado favorable con respecto al nivel de aceptación de la

metodología OpenUp/BioInfo y el expediente de proyecto que esta propone. Siendo así la opinión de los entrevistados:

Mónica Teresa LLorente Quesada - Líder de Proyecto – Mapeo: "la aplicación donde se representa la metodología es muy importante pues de esta manera cada uno de los integrantes del proyecto obtiene información acerca de su trabajo, de lo que le corresponde de acuerdo al rol que desempeña, qué tarea realiza cada uno de los roles. Su trabajo de diploma es muy importante para el PPB, pues precisamente el trabajo más embarazoso de un proyecto es la documentación. La propuesta de un expediente de proyecto con menos artefactos que el que propone la UCI es muy buena, siempre que no viole ninguno de los estatutos establecidos para cada proyecto por los Lineamientos de Calidad de la Universidad y se mantengan solo las planillas mínimas necesarias, pues existen planillas que para nuestros proyectos muchas veces no proceden, es decir, no son imperiosas y de todas maneras hay que llenarlas."

Yunet González Mulet - Líder de Proyecto - BioSys: "creo que su propuesta será bien aceptada por el Departamento Central de Calidad de la Universidad siempre que el expediente de proyecto esté regido por los lineamientos que propone la UCI en su versión 2.0, se ajuste a las necesidades que tenemos actualmente en los proyectos del PPB y que además cumpla con las métricas del estándar de calidad CMMI que es el que quiere implantar la Universidad. O sea, OpenUP/BioInfo puede ser una elección acertada siempre y cuando quede documentado todo lo que necesita el proyecto para funcionar correctamente y que a la hora de hacer una revisión todo el proceso conste en algún documento. Opino que está bastante completa y respaldada por el sitio creado en el EPF Composer."

Denys Javier Hernández Peña - Líder de Proyecto - GDEM: "la metodología para estar en su primera versión está muy bien elaborada, y creo que era necesario buscar una opción para hacer viable el desarrollo de los proyectos productivos de la facultad, incluso este era un tema que venía pensándose desde hace algún tiempo, pues la facultad, y en concreto el PPB BioInformática no se encontraba bajo ninguna metodología en específico, pues estaba prácticamente combinando RUP con OpenUp/Basic en dependencia de la documentación y las actividades que se exigen por parte de la Dirección Central de Calidad, donde se usaba OpenUp/Basic prácticamente como pretexto de reducir la documentación pero sin un argumento sólido. Creo que OpenUp/BioInfo es el punto intermedio con el que se logra disminuir considerablemente el expediente de proyecto UCI y respetar los Lineamientos de Calidad de Software y además está respaldado por una web de ayuda bastante consistente creada mediante la herramienta EPF Composer, y que en un futuro pude seguir siendo revisada y modificada para refinar y mejorar tanto los procesos como el propio expediente de proyecto."

Tonyse de la Rosa Martin - Líder de Proyecto - GrafTool: "estoy totalmente de acuerdo con su propuesta, pues los proyectos del PPB necesitaban una metodología que se adaptara a sus características y objetivamente, no lo era OpenUp/Basic tal y como está liberada. OpenUp/BioInfo es una opción válida siempre que se logre disminuir la cantidad de documentación innecesaria que debemos llevar en nuestros proyectos cumpliendo con los Lineamientos de Calidad de Software. Ese es uno de los principales problemas que presentamos y la causa de muchas discusiones con los asesores de la Dirección de Calidad, pues ellos insisten en que debemos documentar cada una de las cosas que plantea el expediente propuesto por la UCI, aunque siempre están abiertos la opción de presentar una propuesta bien planteada donde, sin violar las normas de calidad establecida, nos adaptemos a nuestras características y creo que con OpenUp/BioInfo debemos resolver este problema. Considero además que está bien respaldada con el montaje que realizaron de la propuesta en el EPF Composer, debería publicarse el sitio de OpenUp/BioInfo e incluso difundir el uso de esta herramienta."

Yuneimy Téllez Pérez - Líder de Proyecto - TArenal: "el expediente de proyecto que proponen es muy bueno y ojalá se apruebe, pues el expediente que utilizamos actualmente es muy similar al de los proyectos que utilizan RUP y es muy bueno que logren adaptarlo a las características del nuestros proyectos. Me gusta mucho la propuesta de OpenUp/BioInfo y creo que la idea de crear un sitio donde poder buscar información de esta metodología es acertada, pues es de mucha ayuda para poder guiarnos en el desarrollo. Además, sienta las bases para poder seguirla modificando en el futuro según los intereses del PPB en el EPF Composer."

Sonia González del Sol – Asesor de calidad – GrafTool: "por lo que me han explicado pienso que su propuesta de metodología es atinada siempre que el expediente de proyecto cumpla lo necesario y precise todo lo que debe tener cada proyecto. Sería factible poder probarlo para medir que tanto se ajusta a las necesidades del PPB. Además considero que el sitio de ayuda es una muy buena idea, provee de los conceptos y las especificaciones necesarias para aprender a usar OpenUp/BioInfo."

Noel Moreno Lemus – Líder del Polo Productivo de BioInformática: "la propuesta de adaptación de la Metodología OpenUp/Basic cubre la principal desventaja que existe hoy en los proyectos del PPB, pues la excesiva e innecesaria documentación que hay que llevar en cada uno de ellos obstaculiza el trabajo de los equipos de desarrollo debido a que el tiempo empleado en documentar es excesivo y sustrae demasiado del dedicado al trabajo verdaderamente substancial como el diseño e implementación de algoritmos de cálculo, esto es sin quitarle importancia a la documentación, pero cabe puntualizar que el expediente de proyecto presentado por la UCI es demasiado extenso para nuestros proyectos y justamente necesitábamos una versión del expediente como la que propone

OpenUp/BioInfo, que logra reducir significativamente los artefactos manteniendo únicamente los mínimos necesarios para cumplir con los Lineamientos de Calidad de Software de la UCI, registrar todos los progresos de los procesos de desarrollo y está dirigido a propiciar practicas que eleven los niveles de capacidad y madurez del PPB según propone CMMI. El sitio de ayuda de la Metodología OpenUp/BioInfo creado en el EPF Composer está muy bien diseñado y nos sirve de guía para los lideres de proyecto, pues es una base de estudio y capital intelectual para la preparación de todos los integrantes del PPB y con la utilización del EPF Composer se puede utilizar como base para seguir mejorando las prácticas del proceso de desarrollo de software de nuestros proyectos. Opino que OpenUp/BioInfo es la opción a implantar en el PPB en vista de resolver todos los problemas actuales y debería ser liberado lo antes posible para su utilización."

Roig Calzadilla Díaz – Asesor de calidad – Dirección Central de Calidad UCI: "proponer esta modificación a la Metodología OpenUp/Basic para adaptarla a las necesidades propias del PPB me parece muy interesante, pienso que sería bueno que pudiera implantarse. Este es un trabajo que necesita un análisis bastante profundo y sobre todo un estudio detallado de las actividades que propone CMMI para poder elevar el nivel de capacidad de los proyectos de su facultad."

2.8 Conclusiones

En este capítulo se realizó un análisis de la metodología OpenUp/Basic, utilizada actualmente en los proyectos de desarrollo de software y un estudio exhaustivo de las características del PPB permitió identificar las principales necesidades del mismo y concebir una propuesta de solución que consiste en adaptar OpenUp/Basic a dichas necesidades. Para realizar esta modificación fue necesario realizar previamente un análisis de los Lineamientos de Calidad de Software de la Universidad y profundizar en las actividades que proponen varias áreas de proceso que deben ser cumplidas para optar por el nivel 2 de madurez según plantea CMMI, basados en la representación continua que establece niveles de capacidad que comprenden prácticas específicas y genéricas que permiten medir la habilidad de un proceso.

Se describen por flujos de trabajo las adaptaciones realizadas a OpenUp/Basic derivando en una nueva metodología, OpenUp/BioInfo, nombrada así debido a su aplicación al PPB y que propone un nuevo expediente de proyecto que se ajusta a las necesidades de reducir documentación con respecto al expediente de proyecto UCI v2.0. Se realizan comparaciones entre ambos expedientes arribando a la conclusión de que con el nuevo se optimiza la cantidad de artefactos, brindando solo los indispensables para cumplir con los Lineamientos de Calidad de Software de la UCI y las actividades asociadas a las áreas de procesos que propone CMMI para el nivel 2 de madurez.

Se describe detalladamente la herramienta EPF Composer, utilizada posteriormente para el montaje de la Metodología OpenUp/BioInfo de forma tal que provea a los desarrolladores de una base de conocimiento de capital intelectual.

Finalmente se realizan las validaciones para medir el nivel de aceptación de la metodología OpenUp/Basic y el expediente de proyecto que esta propone basadas en entrevistas con los principales líderes y asesores de calidad de los proyectos del PPB y con asesores del Departamento Central de Calidad de la Universidad de las Ciencias Informáticas.

Conclusiones

Los resultados alcanzados con la realización de este trabajo de diploma han permitido arribar a las siguientes conclusiones:

- Se abordaron las métricas que proponen los estándares de calidad debido a la necesidad de las empresas de utilizar prácticas y guías eficientes de ingeniería de software adaptadas a su tamaño y tipo de negocio. Se profundizó en las normas de calidad de CMMI debido a que la Universidad de las Ciencias Informáticas tiene como meta para el 2012 alcanzar el nivel 2 o Gestionado de madurez según la representación continua del modelo de CMMI, que establece niveles de capacidad que comprenden prácticas específicas y genéricas que permiten medir la habilidad de un proceso.
- Se realizó un análisis detallado de los procesos de desarrollo de software, indagando entre distintos modelos genéricos que pueden ser utilizados para explicar diferentes perspectivas de cómo afrontarlos. Particularmente en el Modelo de Desarrollo Iterativo Incremental, que es el que sigue la Metodología que se propone adaptar, debido a las ventajas que ofrece y las características que hacen que sea el idóneo para allanar las necesidades del PPB.
- Se realizó un estudio del entorno de desarrollo del PPB y del nivel de madurez que presentan sus proyectos, llegando a la conclusión de que el proceso actual no ostentaba madurez suficiente para desarrollar software de calidad lo que permitió desplegar una propuesta de adaptación a la metodología utilizada por los proyectos del PPB con el fin de erradicar estos problemas.
- Se fundamentó el uso de la Metodología OpenUp/Basic como base para cometer la adaptación realizando un estudio exhaustivo y comparativo de las metodologías de desarrollo caracterizando los dos enfoques existentes; las metodologías ágiles y las metodologías tradicionales, de manera que se pudieran evaluar las ventajas y desventajas de una sobre otra, concluyendo en que debido a las características y necesidades del PPB sería favorable la implantación de una metodología ágil siempre y cuando esta fuera extensible y adaptable de modo que pudiera ser modificada para cubrir las especificidades que engloban los Lineamientos de Calidad de Software de la UCI y las métricas que plantean los estándares de CMMI.
- Se concibió la Metodología OpenUp/BioInfo como resultado del proceso de adaptación de OpenUp/Basic a las características y necesidades del PPB.

- Se efectuó el montaje de la Metodología OpenUp/BioInfo en el EPF Composer que provee a los desarrolladores con una base de conocimiento de capital intelectual que les permite buscar, administrar y desplegar contenido.
- Se realizaron entrevistas a los máximos responsables de los procesos de desarrollo de software del PPB para medir los niveles de aceptación de la propuesta.

Recomendaciones

Con el objetivo de mejorar la calidad de los procesos de desarrollo de software en el PPB y la Universidad de las Ciencias Informáticas en general se recomienda:

- Implantar la metodología OpenUp/BioInfo a los Proyectos del PPB.
- Realizar un análisis a la Metodología OpenUp/BioInfo y adaptarla en caso de ser necesario, con el objetivo de hacerla extensible a todos los proyectos de la universidad.
- Realimentarse y ganar en madurez mediante el estudio de la documentación y las experiencias obtenidas con el uso de la Metodología OpenUp/BioInfo.
- Capacitar a los miembros de los módulos del PPB.

Referencias Bibliográficas

1. Ángel, M. Mantenimiento del Software como Actividad de Ingeniería [Disponible en:] http://cnx.org/content/m17401/latest/
2. Soto, L [Disponible en:] http://www.mitecnologico.com/Main/ElModeloComoResultadoDeLaAbstraccion
3. Ruiz, F; Canfora, G. Procesos Software: características, tecnología y entornos [Consultada: 2009 Enero];[Disponible en:] http://www.ati.es/novatica/2004/171/171-5.pdf
4. Proceso de desarrollo de Software. [Consultada: 2009 Febrero]; [Disponible en:] www.dsic.upv.es/asignaturas/facultad/lsi/doc/IntroduccionProcesoSW.doc
5. Palacios, J. Sinopsis de los modelos SW-CMM y CMMI [Consultada: 2009 Febrero]; [Disponible en:] http://www.navegapolis.net/files/articulos/sinopsis_cmm.pdf
6. Metodologías para la gestión y desarrollo de Software [Disponible en:] http://www.scribd.com/doc/8255409/Metodologias-para-la-geston-y-desarrollo-de-Software?__cache_revision=1231652476&__user_id=-1&enable_docview_caching=1
7. ISO 9001 - Norma de Calidad [Disponible en:] http://www.buscarportal.com/articulos/iso_9001_gestion_calidad.html
8. Block, M; Marash, R. Integración de ISO 14001 en un sistema de gestión de la calidad [Disponible en:] http://books.google.com.cu/books?id=R6v7oMSOHD8C
9. Armas, R; Chamorro, A; Montes, M; Gutiérrez, J. Desde ISO 9001 hacia CMMI, pasos para la mejora de los procesos y métricas [Disponible en:] http://www.aemes.org/rpm/descargar.php?volumen=4&numero=1&articulo=3
10. Solis, C; Figueroa, R. Metodologías Tradicionales vs. Metodologías Ágiles [Disponible en:] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.1515
11. Sánchez, M. Metodologías de Desarrollo de Software.2004 [Disponible en:] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html
12. Canos, H; Letelier, P; Penadés, C. Metodologías Ágiles en el Desarrollo de Software [Consultada: 2009 Marzo]; [Disponible en:] http://www.willydev.net/descargas/prev/TodoAgil.pdf
13. Peruserver S.A.C. Desarrollo Metodología Extreme Programing. [En línea: 2008]; [Disponible en:] http://www.peruserver.com/des_metodologia.php.
14. openUP/Basic [En línea: 2007]; [Disponible en:] http://epf.eclipse.org/wikis/openupsp/

15. La Metodología de desarrollo Ágil como una oportunidad para la Ingeniería del software Educativo. [Disponible en:] http://pisis.unalmed.edu.co/avances/archivos/ediciones/Edicion%20Avances%202008%202/21.pdf
16. http://www.monografias.com/trabajos67/metodologia-desarrollo-softwares/metodologia-desarrollo-softwares2.shtml
17. Gracia, J. [En línea: 2005]; [Disponible en:] http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php
18. Teuber, P [Disponible en:] http://www.monografias.com/trabajos57/modelo-calidad-cmmi/modelo-calidad-cmmi2.shtml#xestructcmmi
19. Delgado, R; Neuland, D. Lineamientos de calidad de software [En línea: 2008]
20. EPF. 2008 EclipseProcessFrameworkComposer.pdf [Consultada 10/05/09]

Bibliografía

http://www.ingenierosoftware.com/calidad/cmm-cmmi.php *[Consultada: 20/2/2009]*

http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php *[Consultada: 21/2/2009]*

http://calidad.umh.es/curso/criterio.htm *[Consultada: 2/3/2009]*

www.dsic.upv.es/asignaturas/facultad/lsi/doc/IntroduccionProcesoSW.doc*[Consultada: 2/3/2009]*

http://www.acis.org.co/index.php?id=551 *[Consultada: 13/3/2009]*

http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html *[Consultada: 13/3/2009]*

http://eproano334.blogspot.es/1194376920/ *[Consultada: 20/3/2009]*

http://merinde.rinde.gob.ve/index.php?option=com_search&searchword=requerimientos *[Consultada 20/03/09].*

www.spinec.org/wp-content/metodologiasagiles_01.pdf *[Consultada 20/03/09]*

http://www.programacionextrema.org/articulos/newMethodology.es.html *[Consultada 26/03/09].*

http://teleformacion.uci.cu/course/view.php?id=102 *[Consultada 23/03/09].*

http://www.eclipse.org/epf/ *[Consultada 26/03/09].*

www.sergiovillagra.com/Contenidos/Recursos/WP03%20Una%20Introduccion%20a%20CMMI.pd *[*Consultada 11/04/09*]*

http://www.calidadysoftware.com/otros/introduccion_cmmi.php *[Consultada 13/04/09].*

mmedusa.avansoft.com/terracota/files/produccion/8.2-EclipseProcessFrameworkComposer.pdf *[Consultada 20/04/09].*

http://www.grupoconsultoria.com.co/cmmi.htm *[Consultada 26/04/09].*

http://www.cientec.com/analisis/ana-cmmi.html *[Consultada 29/04/09].*

http://cnx.org/content/m17401/latest/ *[Consultada 06/05/09].*

http://www.mitecnologico.com/Main/ElModeloComoResultadoDeLaAbstraccion *[Consultada 10/05/09].*

http://www.sei.cmu.edu/cmmi/ [Consultada 11/05/09].

http://www.isospice.com/categories/ISO%7B47%7DIEC-15504-Standard [Consultada 16/05/09].

http://wwwhome.cs.utwente.nl/~santanarg/downloads/tesis_rst2003.pdf [Consultada 17/05/09].

http://www.buscarportal.com/articulos/iso_9001_2000_gestion_calidad.html [Consultada 19/05/09].

http://www.scribd.com/doc/41705/Norma-ISO-90012000 [Consultada 20/05/09].

http://www.programacionextrema.org/articulos/newMethodology.es.html [Consultada 21/05/09].

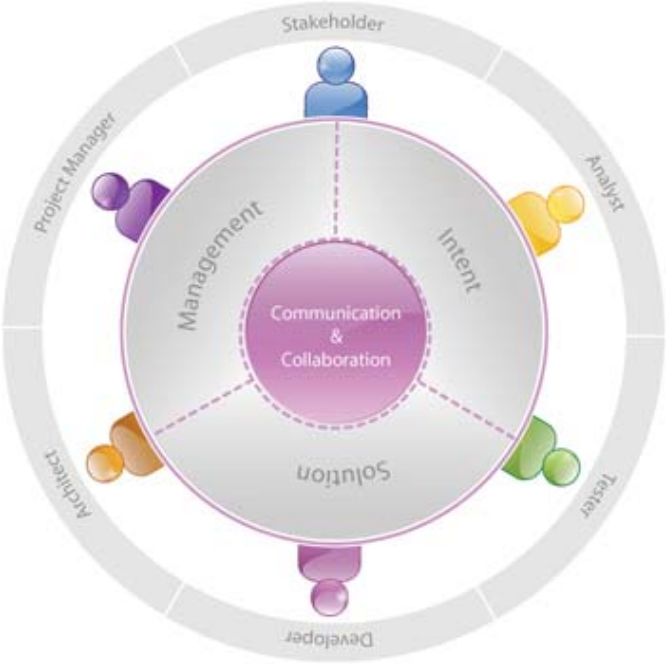
http://www.willydev.net/descargas/prev/TodoAgil.pdf [Consultada 23/05/09].

http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/explicaxp.pdf [Consultada 24/05/09].

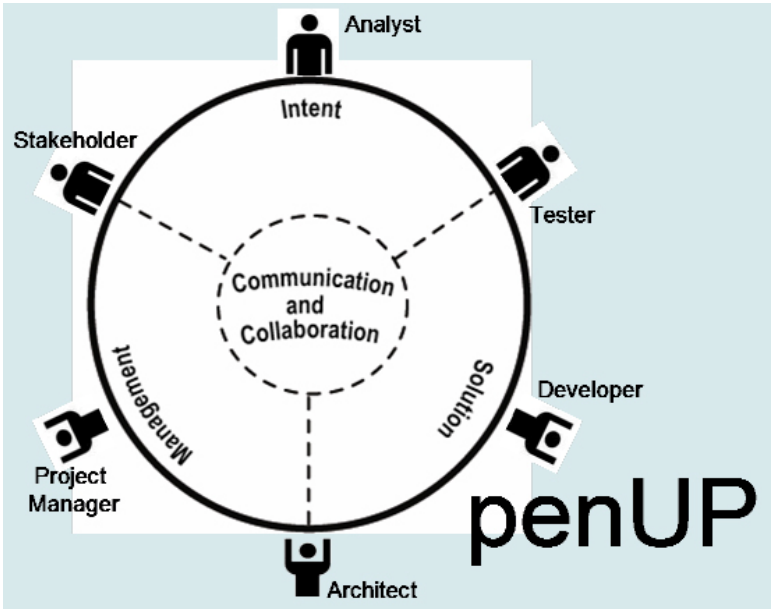
http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf [Consultada 02/06/09].

http://materias.fi.uba.ar/7500/schenone-tesisdegradoingenieriainformatica.pdf [Consultada 02/05/09].

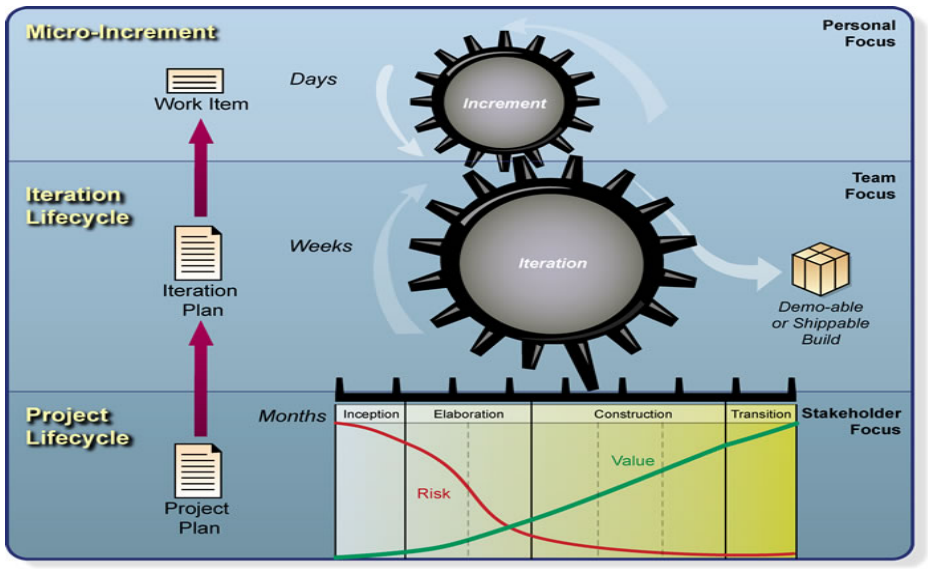
Anexos



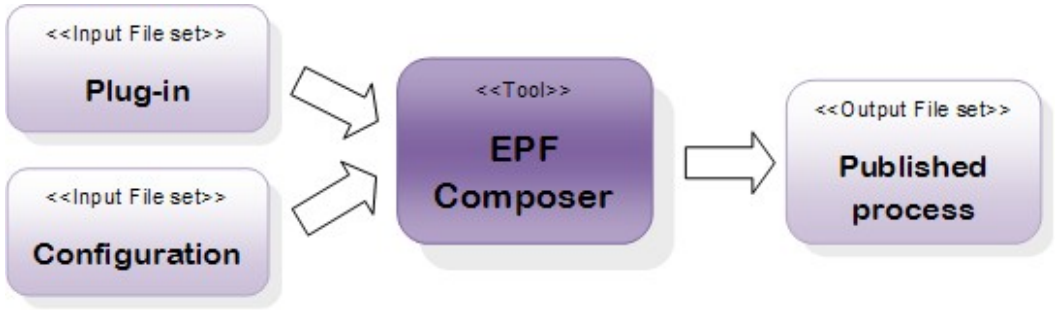
Anexo1: Logo de OpenUp/Basic



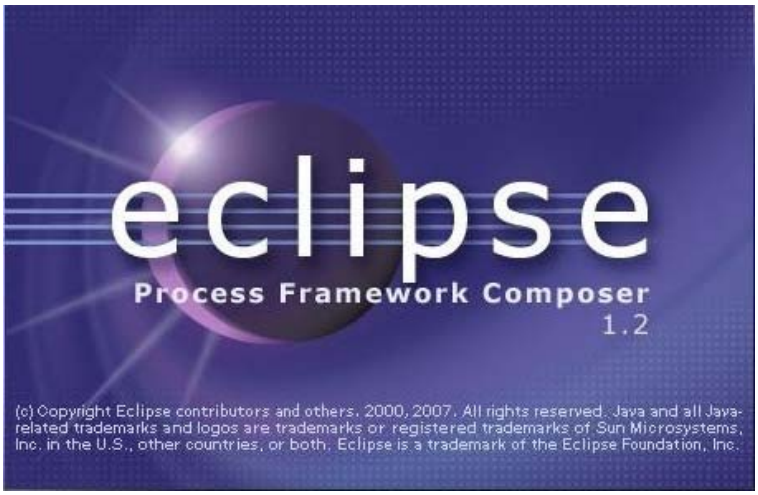
Anexo2: Logo de OpenUp/Basic.



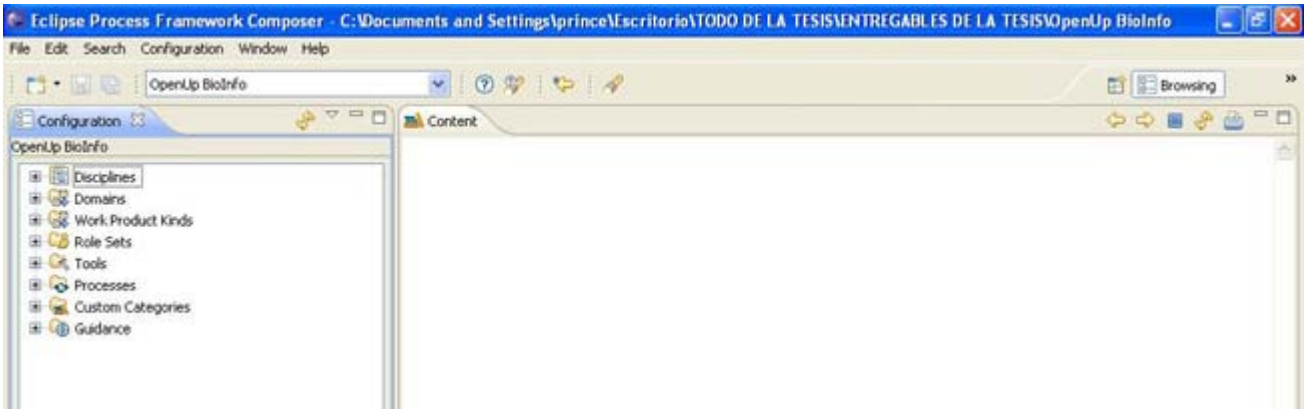
Anexo3: Capas del ciclo de vida de OpenUp/BioInfo (iteraciones).



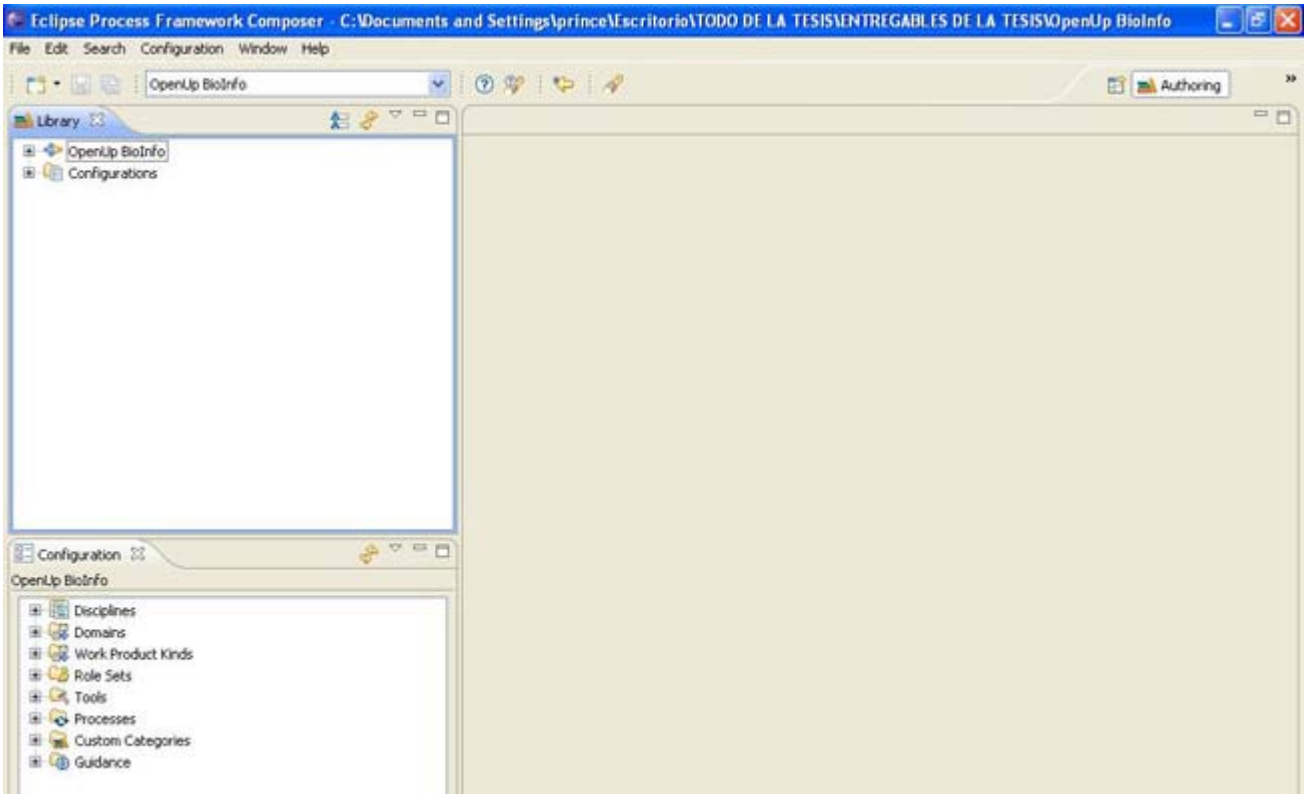
Anexo4: Diagrama del proceso de publicación en el EPF Composer.



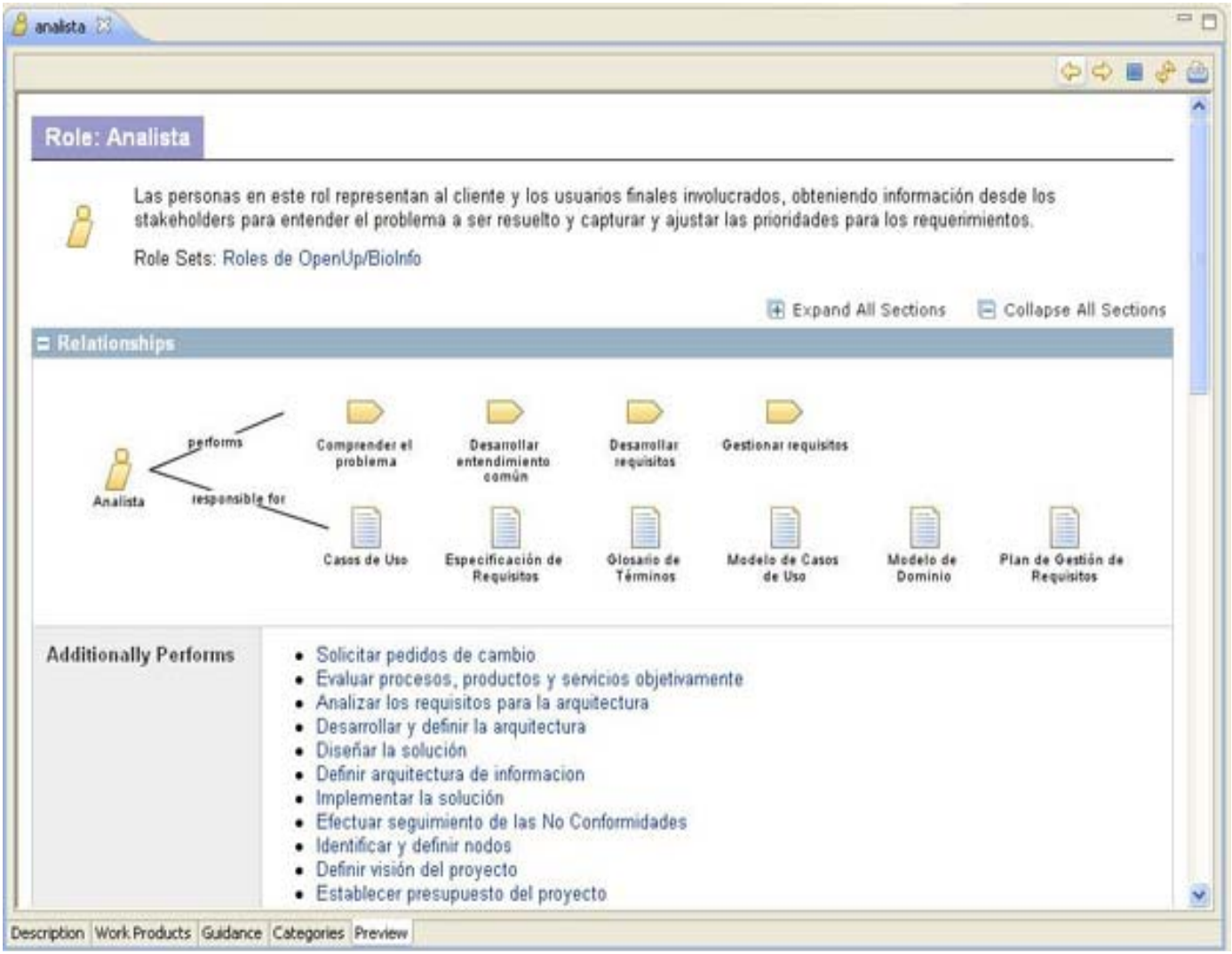
Anexo 5: Logo de la Herramienta EPF Composer v1.2



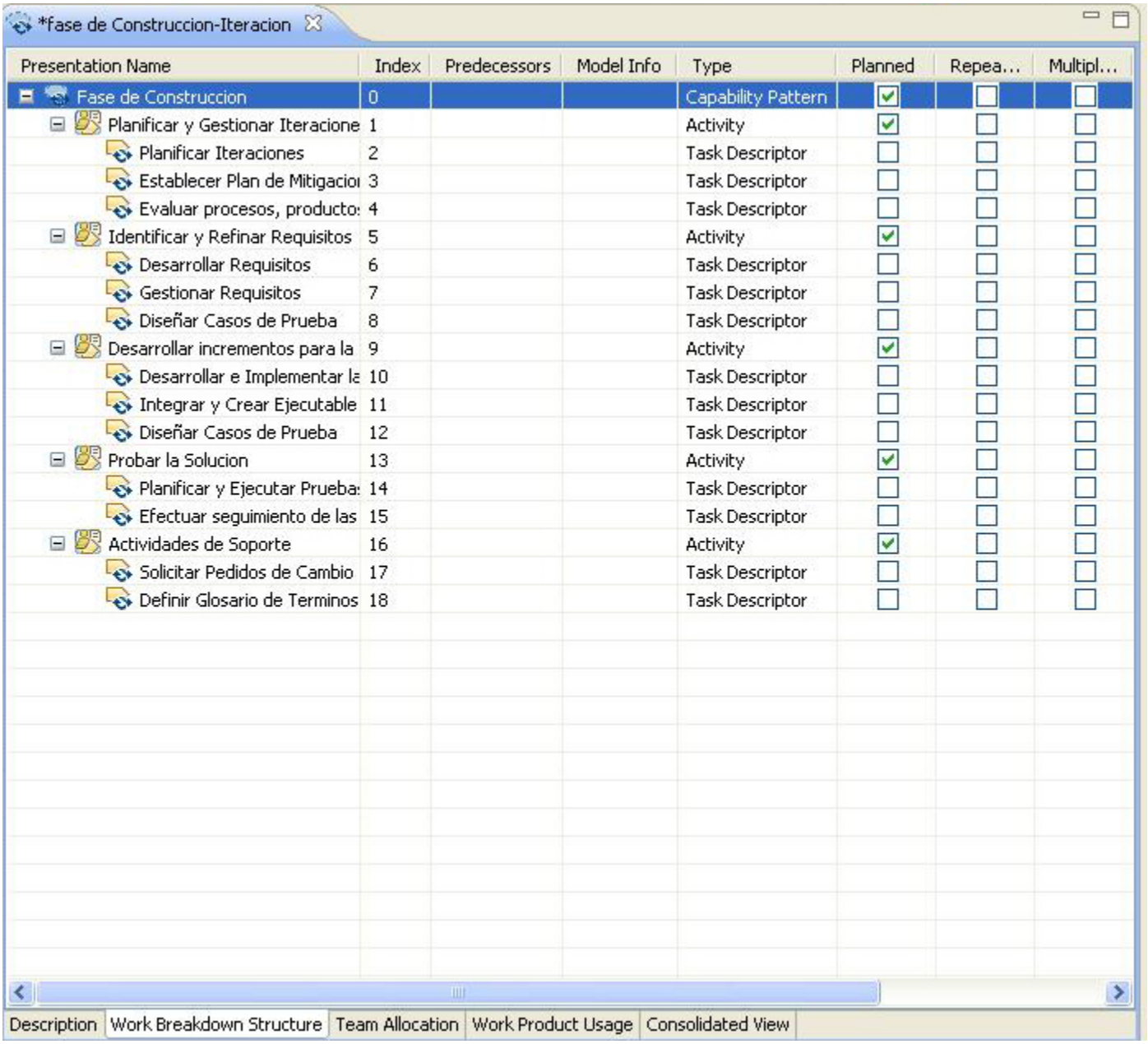
Anexo 6: EPF Composer – perspectiva Browsing.



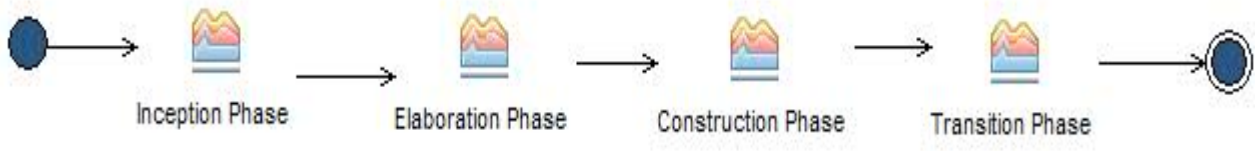
Anexo 7: EPF Composer – perspectiva Authoring.



Anexo 8: Vista previa de roles en el EPF Composer.

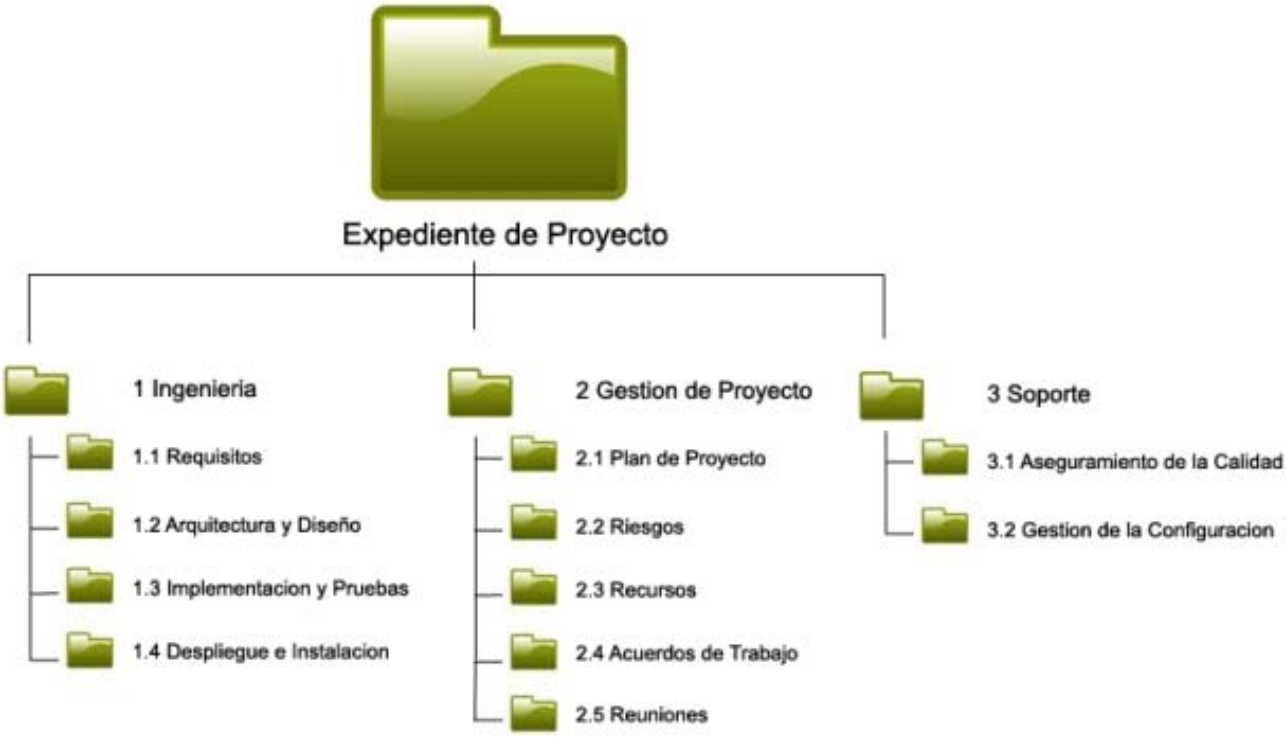


Anexo 9: vista de desglose de estructura de trabajo en el EPF Composer.



Anexo 10: Ciclo de vida de OpenUp/BioInfo.

Anexo 11: Página principal de la Metodología OpenUp/BioInfo, resultado de la publicación del método en el EPF Composer.



Anexo 12: Estructura y conformación del expediente de proyecto de OpenUp/BioInfo.

Representación por etapas

La representación por etapas organiza las áreas de procesos dentro de cinco niveles de madurez, que soportan y guían la ruta evolutiva del proceso de mejoramiento continuo para toda la organización.

Nivel de madurez 1
Ejecutado

Nivel de madurez 2
Administrado

Nivel de madurez 3
Definido

Nivel de madurez 4
Administrado
Cuantitativamente

Nivel de madurez 5
Optimizado

Representación Continua

La representación continua refleja los niveles de capacidad en su diseño y contenido. Un nivel de capacidad comprende prácticas específicas y genéricas que permiten medir la habilidad de un proceso.

Nivel de capacidad 0
Incompleto

Nivel de capacidad 1
Ejecutado

Nivel de capacidad 2
Administrado

Nivel de capacidad 3
Definido

Nivel de capacidad 4
Administrado
Cuantitativamente

Nivel de capacidad 5
Optimizado

Anexo 13: Niveles de madurez de CMMI, representación escalonada y continua.

Nivel de Madurez 2 Administrado

En la organización que se encuentra en este nivel algunas áreas organizacionales y/o proyectos han alcanzado las metas genéricas y específicas establecidas en sus áreas de proceso, es decir planean sus procesos, los ejecutan, los miden y los controlan.

Anexo 14: Nivel 2 de madurez de CMMI.

Nivel de Capacidad 2 Administrado

El proceso tiene las personas con las habilidades y experiencia apropiada; cuenta con los recursos adecuados para la transformación de los insumos en productos finales; es supervisado, controlado y revisado.

Anexo 15: Nivel 2 de capacidad de CMMI.

Glosario de Términos

Polo productivo: Proyectos que se encuentran agrupados por esferas.

Bioinformática: Es la aplicación de tecnología de computadores a la gestión y análisis de datos biológicos.

Metodología de desarrollo de software: Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo de software.

Proceso de desarrollo de software: Conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software.

OpenUp/Basic: Es un FrameWork de procesos de desarrollo de software de código abierto, basado en Rational Unified Process; o sea, es un proceso interactivo de desarrollo de software simplificado, completo y extensible para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y de los involucrados con el resultado del proyecto, por encima de formalidades innecesarias.

Áreas de Proceso: Grupo de prácticas relacionadas que son ejecutadas de forma conjunta para conseguir un conjunto de objetivos.

CMMI (Capability Maturity Model Integration): Modelo de mejora de procesos de desarrollo que provee orientación para diseñar procesos efectivos y medir la calidad de un software.

Expediente de proyecto: Carpeta que contiene toda la documentación del proyecto, organizada por categorías.