PROYECTO DE GRADO

PLANTILLAS Y ARTEFACTOS PERSONALIZACIÓN DE RUP PARA PROYECTOS ACADÉMICOS DE DESARROLLO DE SOFTWARE

Mario Darío López Márquez

Juan Camilo Villa Palacio

Universidad EAFIT

Escuela de ingenierías

2012

PLANTILLAS Y ARTEFACTOS PERSONALIZACIÓN DE RUP PARA PROYECTOS ACADÉMICOS DE DESARROLLO DE SOFTWARE

MARIO DARÍO LÓPEZ MÁRQUEZ

JUAN CAMILO VILLA PALACIO

Trabajo de grado presentado como requisito parcial para optar por el título de ingenieros de sistemas

Asesor: JORGE HERNÁN ABAD LONDOÑO

MEDELLÍN

UNIVERSIDAD EAFIT

FACULTAD DE INGENIERÍA DE SISTEMAS

2012

Nota de aceptación

_____

_____

Presidente del jurado

_____

Jurado

_____

Jurado

_____

Medellín 30 de abril de 2012

CONTENIDO

ANEXOS

Proyecto complejo: Sistema GDP:

- Requisitos:
  - CU - 001 – TOMAR PEDIDO
  - CU - 008 – CONSULTAR ESTADO DEL PEDIDO
  - CU - 011 – CONSULTAR VALOR PEDIDO
  - CU- 002-CONSULTAR MENU
  - Documento de Requisitos del Sistema DRS
  - Solicitudes de los Interesados
  - Visión del Sistema
- Análisis y diseño
  - BPM-GDP
  - Documento de Arquitectura de Software DAS
  - Decisiones de Arquitectura
  - BPM
  - Prototipo
- Implementación
  - registro pruebas unitarias
- Pruebas
  - Casos de prueba tpl
  - Ideas de prueba
- Gestión del proyecto
  - Acta de Entrega del Proyecto
  - Acta de Reunión
  - Lista de Elementos de Trabajo
  - Plan de Gestión de Riesgos
  - Plan del Proyecto
  - Registro de Revisión
- Gestión de cambios
  - Solicitud de cambio

Proyecto Simple: Cajero CDT:

- Requisitos:
  - CU-001-CONSULATR SALDO
  - CU- 003-RETIRAR SALDO
  - Documento de requisitos del sistema DRS
- Análisis y diseño:
  - Documento de arquitectura de software DAS Informal
- Implementación:
  - Registro de pruebas unitarias
- Pruebas:

    - Ideas de pruebas – casos de pruebas - pruebas
- Gestión del proyecto:
    - Acta Entrega del Proyecto
    - Acta de reunión
    - Lista de elementos de trabajo
    - Plan de proyecto informal
    - Plan de gestión de riesgos
    - Registro de revisión
- Gestión de cambios:
    - Solicitud de cambio

GLOSARIO

RUP: Rational Unified Process, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas.

Desarrollo en espiral: Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a ninguna prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior.

Lenguaje Unificado de Modelado: (LUM o UML, por sus siglas en inglés, Unified Modeling Language) Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

Artefacto: Producto tangible resultante del proceso de desarrollo de software. Algunos artefactos como los casos de uso, diagrama de clases u otros modelos UML ayudan a la descripción de la función, la arquitectura o el diseño del software. Otros se enfocan en el proceso de desarrollo en sí mismo, como planes de proyecto, casos de negocios o enfoque de riesgos.

Plan de proyecto: Documento formal usado como guía para la ejecución y el control de un proyecto. El uso primario de un plan de proyecto es documentar los planes y decisiones, facilitar la comunicación entre los stakeholders y documentar el alcance, costos y cronogramas.

Stakeholder: Toda persona interesada o afectada por la ejecución de un proyecto.

Plan de gestión de riesgos: Documento formal que presenta unos supuestos sobre los riesgos que pueden afectar el proyecto y sus planes de contingencia.

Lista de elementos de trabajo (EDT): Documento donde se listas las tareas clasificadas por sus etapas durante el proyecto, los responsables de cada tarea y el tiempo estimado.

Plan de gestión de comunicaciones: Documento formal donde se presentan los canales de comunicación que se tendrán entre todos los actores del proyecto.

BPMN (Modelado de procesos de negocio): Notación gráfica que presenta un flujo de trabajo sobre un negocio.

TABLA DE FIGURAS

RESUMEN

En este documento se encontrará información que presenta un marco teórico del proyecto de grado. Dicho marco proveerá los objetivos generales y específicos, justificaciones, ejemplos y conclusiones que argumentarán el trabajo sobre el tema de metodologías ágiles orientadas a la academia.

El trabajo se realiza con el propósito presentar una serie de plantillas, artefactos y ejemplos cercanos a la vida real de proyectos de desarrollo de software usando la metodología EAFIT-UP (EAFIT Unified Process) basada en RUP (Rational Unified Process), CMMI Dev y PMBOK, demostrando su efectividad en el uso principalmente académico.

El proyecto se apoya en la tesis de maestría de Jorge Hernán Abad Londoño, autor de la metodología y asesor de este trabajo.

Palabras clave: SOFTWARE EDUP, RUP, CMMI Dev, PMBOK (Project Management body of knowledge), Metodologías ágiles, proceso de desarrollo de software

1. INTRODUCCIÓN

Una definición de Ingeniería de Sistemas basada en las ideas de Hall, Wymore, y M'Pherson, es la siguiente: "Ingeniería de Sistemas es un conjunto de metodologías para la resolución de problemas mediante el análisis, diseño y gestión de sistemas". Los sistemas que se referencian son aquellos que incluyen hardware, software y personas.

Partiendo de hechos, primero, que un sistema es un conjunto de componentes interrelacionados, los cuales trabajan conjuntamente para un fin común, y segundo, que la proporción del software en los sistemas sigue creciendo; se puede decir que los problemas de la Ingeniería de Sistemas, son similares a los de la Ingeniería de Software.

Estos problemas tienen que ver con el tiempo, costo, y calidad de los sistemas que se crean. Los productos de software se ubican entre los sistemas más complejos realizados por el hombre, y su naturaleza intangible complica más el problema. La Ingeniería de Software busca dar soluciones a estos problemas.

El objetivo de la Ingeniería de Software es realizar las actividades de desarrollo, operación y mantenimiento de software, de forma sistemática, disciplinada y cuantificable. En otras palabras, la Ingeniería de Software tiene como fin la producción de software de buena calidad, dentro de los costos estimados, fácil de desarrollar y mantener.

El desarrollo de software es un proceso de alta complejidad que consume tiempo, necesita mucho esfuerzo humano y demanda dinero. El éxito de un proyecto de software se mide en función de tres variables fundamentales: costo, tiempo y calidad. Un proyecto exitoso es aquel que se entrega bajo el presupuesto asignado, a tiempo y con la calidad especificada.

Por lo tanto, resolver el "cómo" realizar el proceso de desarrollo de software, corresponde a las actividades recurrentes en la Ingeniería de Software y éste se encuentra en constante maduración y evolución, debido a que ese "cómo", va desde un proceso improvisado de desarrollo de software, que trae consigo malas prácticas y pésimas consecuencias en resultados de dinero, tiempo, y calidad; hasta metodologías formales, como lo son el Rational Unified Process (RUP), el cual es un marco de trabajo genérico, ampliamente utilizado, y que puede adaptarse a diferentes tipos de sistemas, tipos de organizaciones, y diferentes tamaños de proyectos; y metodologías Ágiles, las cuales tienen poca documentación y requieren gran disciplina y madurez del equipo de trabajo.

Es común que el Proceso de Software que se sigue en las prácticas, trabajos y proyectos académicos se encuentren sin directrices claras (roles, entregables, tareas, actividades, procesos y herramientas), y sin la adopción de un modelo que presente las mejores prácticas existentes, que permitan producir software de calidad, bajo los estándares de funcionalidad y documentación que demanda el mercado, trayendo

consigo proyectos descontextualizados de su entorno, y estudiantes que ejecutan sus desarrollos de software sin lineamientos claros, que incorporen buenas prácticas y redunden en beneficio de su desempeño profesional.

El ámbito académico, a diferencia del ámbito empresarial, ejerce menos control en el cumplimiento del cronograma y el presupuesto estimado, prestando solo interés en el entregable final (entregable software final); dejando a las prácticas, trabajos y proyectos sin exigencia en la adherencia de buenas prácticas, elaboración de documentación, utilización eficiente de recursos; y por consiguiente alta probabilidad de fracaso.

Con este proyecto se pretende proporcionar a la Universidad EAFIT, a partir de un marco de trabajo ya definido para la ejecución de prácticas, trabajos y proyectos de desarrollo de software, en el ámbito académico, una serie de plantillas, artefactos y ejemplos que apoyen dicha metodología, esto se ajustara al tiempo planeado, al presupuesto asignado y cumplirá con estándares de calidad.

2. OBJETIVOS

2.1 Objetivo general

Apoyar la metodología personalizada para los proyectos académicos de desarrollo de software, realizando sus respectivas plantillas y artefactos para la ejecución de dicho framework, a través de las disciplinas de Gestión de proyectos, Modelado Empresarial, Requisitos, Análisis y Diseño, Implementación, Pruebas, Despliegue, y la Gestión del cambio; teniendo como referente las metodologías estándar RUP, CMMi DEV, y PMBoK.

2.2 Objetivos específicos

- Hacer uso de los flujos de trabajo de cada disciplina; para la ejecución de cada tipo de proyecto académico, ya definidos en el framework, para realizar sus plantillas y artefactos.
- Extraer los roles y responsabilidades de los encargados del proceso de software, para cada disciplina objeto de este proyecto, de acuerdo a cada tipo de proyecto académico, para apoyar el desarrollo de los artefactos ejemplo.
- Construir las plantillas y artefactos del framework.

3. DEFINICIÓN DEL PROBLEMA

Actualmente la Universidad EAFIT, para los estudiantes de los programas de pregrado y posgrado, no ha definido un proceso de desarrollo de software, formal y flexible, que establezca un marco de trabajo en la ejecución de los desarrollos de software que se determinen para las prácticas, proyectos y tesis.

Ante la ausencia de un proceso de desarrollo, formal y flexible, se presentan los siguientes hechos:

- La evaluación de los diferentes trabajos académicos se centra en el resultado final (el producto software), sin importar el proceso de desarrollo, y otros entregables de igual importancia al código ejecutable.

- Pobre adopción de las mejores prácticas, que la industria ha identificado, para el desarrollo de software; por parte de los estudiantes que están realizando trabajos académicos.

- No existe un esquema de trabajo coherente que este apropiado por los estudiantes, que les defina un ciclo de vida y las mejores prácticas, para llevar un proyecto de desarrollo de software al éxito.

- Falta de uniformidad de criterios para ejecutar y gestionar proyectos de desarrollo de software, en las diferentes asignaturas de la Universidad.

- Carencia de una metodología formal de desarrollo de los proyectos de software, en los cursos relacionados con Desarrollo e Ingeniería de Software.

- Carencia de tipificación en los proyectos de desarrollo de software, que permita adaptar la metodología a los diferentes tipos de proyectos.

- La adopción discrecional y sin seguimiento, por parte de los estudiantes, de un marco de trabajo estándar (con sus buenas prácticas) para el desarrollo de software, se convierte en una brecha entre lo que demanda la industria de desarrollo de software, y los profesionales que egresan de la universidad. Esta brecha, hace necesario que las compañías inviertan en largos periodos de capacitación y entrenamiento en habilidades, que debieron haber sido adquiridas y puestas en práctica en el ciclo de formación profesional.

4. IMPORTANCIA DEL PROYECTO

De acuerdo al Dr. Frederick Brooks (ganador del ACM Turing Award, en 1999), el desarrollo de software es la tarea más compleja que puede iniciar un ser humano; y ésta es una actividad que frecuentemente se repite en la mayoría de las asignaturas que conforman el programa de Ingeniería de Sistemas, en la Universidad EAFIT; así como también es el campo de acción, donde la mayoría de los nuevos profesionales obtiene su primera experiencia laboral.

Un marco de trabajo formal, flexible y basado en las mejores prácticas; para el desarrollo de software, es una guía que define las actividades que se deben realizar, para alcanzar los resultados esperados. Al plantear construir esta guía se pretende hacer énfasis en un proceso de desarrollo de software para proyectos académicos de diferente alcance, el cual busca lograr un producto software de alta calidad, que se ajuste a los requisitos iniciales, dentro de un costo y tiempo determinados.

En el contexto académico, al existir un marco de trabajo genérico para el desarrollo de software, se identifican las siguientes ventajas:

Docente

- Seguir los avances del proyecto.
- Revisar y evaluar los entregables al final de cada etapa, que defina el proceso.
- Realizar correcciones en etapas tempranas.

Estudiante

- Recibir retroalimentación de la interpretación que se hace del proceso.
- Mejorar los entregables, a medida que avance el proceso.
- Corregir el rumbo, desde etapas tempranas, sin esperar hasta el final donde no hay margen para cambios.
- Conocer el proceso de desarrollo, apropiándose de las buenas prácticas y mejorando los resultados en los siguientes proyectos.

De otro lado, un número importante de las compañías que conforman la industria de desarrollo de software local, han establecido procesos de desarrollo de software, la mayoría basados en RUP (IBM Rational Unified Process), buscando ser más eficientes y rentables en la tarea de desarrollar software. Recordemos que RUP es una metodología estándar, ampliamente utilizada, que se basa en la mejores prácticas que se han probado en el campo, y que tiene como objetivo hacer alcanzables tanto pequeños como grandes proyectos de software.

Las compañías que definieron y apropiaron su proceso de desarrollo de software, han decidido conseguir la certificación o evaluación de sus procesos, de acuerdo a estándares internacionales; de tal forma que les permita llegar a nuevos clientes y mercados, los cuales son más exigentes en los requisitos que deben cumplir sus proveedores de servicios.

Por tanto, el estudiante al habituarse al seguimiento de un proceso de desarrollo de software, dentro de su formación profesional, le permitirá adquirir habilidades que facilitarán, agilizarán y aumentarán sus posibilidades de inmersión en el entorno laboral.

En el contexto laboral, las compañías que pertenecen a la industria del desarrollo de software, se verán favorecidas por los profesionales de la Universidad EAFIT, de la siguiente forma:

- Menor inversión en capacitación y entrenamiento de los profesionales egresados de la Universidad EAFIT.
- Mayor rentabilidad económica, en los proyectos donde participen los profesionales egresados de la Universidad EAFIT.
- Capacidad para asumir nuevos proyectos, sin afectar considerablemente los factores de calidad, presupuesto y tiempo.

5. MARCO TEÓRICO

5.1 Metodologías ágiles

Se trata de un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en cortos lapsos de tiempo. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, pero la meta es tener una «demo» (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto.

Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. La mayoría de los equipos ágiles están localizados en una simple oficina abierta, a veces llamadas "plataformas de lanzamiento" (bullpen en inglés). La oficina debe incluir revisores, escritores de documentación y ayuda, diseñadores de iteración y directores de proyecto. Los métodos ágiles también enfatizan que el software funcional es la primera medida del progreso. Combinado con la preferencia por las comunicaciones cara a cara, generalmente los métodos ágiles son criticados y tratados como "indisciplinados" por la falta de documentación técnica.

5.1.1 El manifiesto ágil

El Manifiesto Ágil comienza enumerando los principales valores del desarrollo ágil. Según el Manifiesto se valora:

• Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

• Desarrollar software que funciona más que conseguir una buena documentación. La regla a seguir es “no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante”. Estos documentos deben ser cortos y centrarse en lo fundamental.

• La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

• Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del

proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta. Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

VII. El software que funciona es la medida principal de progreso.

VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

X. La simplicidad es esencial.

XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

5.2 RUP (Rational Unified Process)

El Proceso Unificado de Rational (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

También se conoce por este nombre al software desarrollado por Rational, hoy propiedad de IBM, el cual incluye información entrelazada de diversos artefactos y descripciones de las diversas actividades. Está incluido en el Rational Method Composer (RMC), que permite la personalización de acuerdo con las necesidades.

El RUP está basado en 6 principios clave que son los siguientes:

Adaptar el proceso

El proceso deberá adaptarse a las necesidades del cliente ya que es muy importante interactuar con él. Las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto en un área subformal.

Equilibrar prioridades

Los requisitos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un equilibrio que satisfaga los deseos de todos. Gracias a este equilibrio se podrán corregir desacuerdos que surjan en el futuro.

Demostrar valor iterativamente

Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados

Colaboración entre equipos

El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requisitos, desarrollo, evaluaciones, planes, resultados, etc.

Elevar el nivel de abstracción

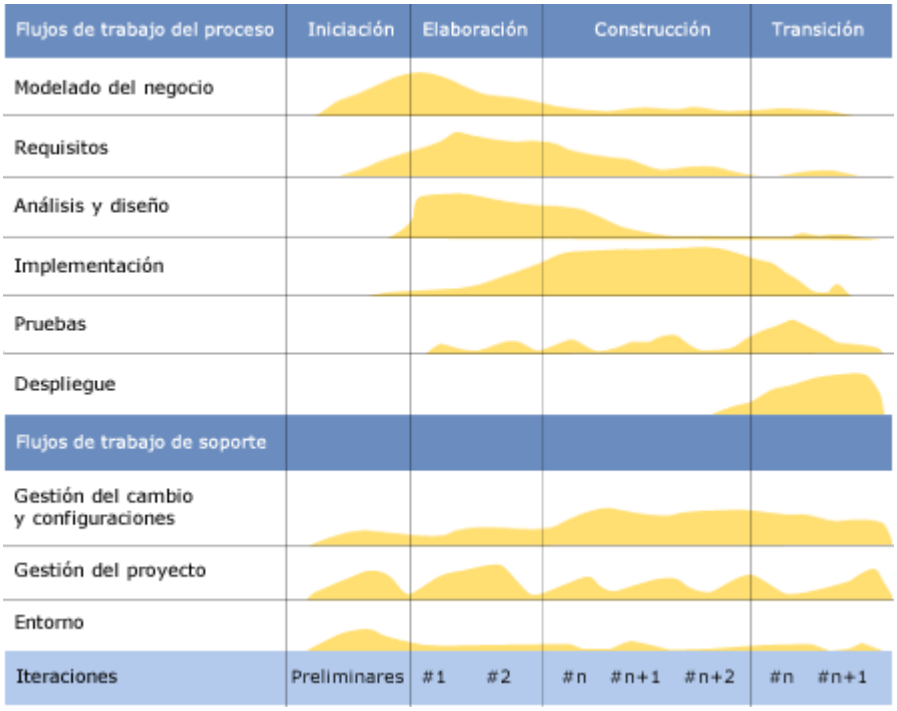
Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o marcos de referencia (frameworks) por

nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requisitos y sin comenzar desde un principio pensando en la reutilización del código. Un alto nivel de abstracción también permite discusiones sobre diversos niveles y soluciones arquitectónicas. Éstas se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con el lenguaje UML.

Enfocarse en la calidad

El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

Figura 1: Proceso de RUP



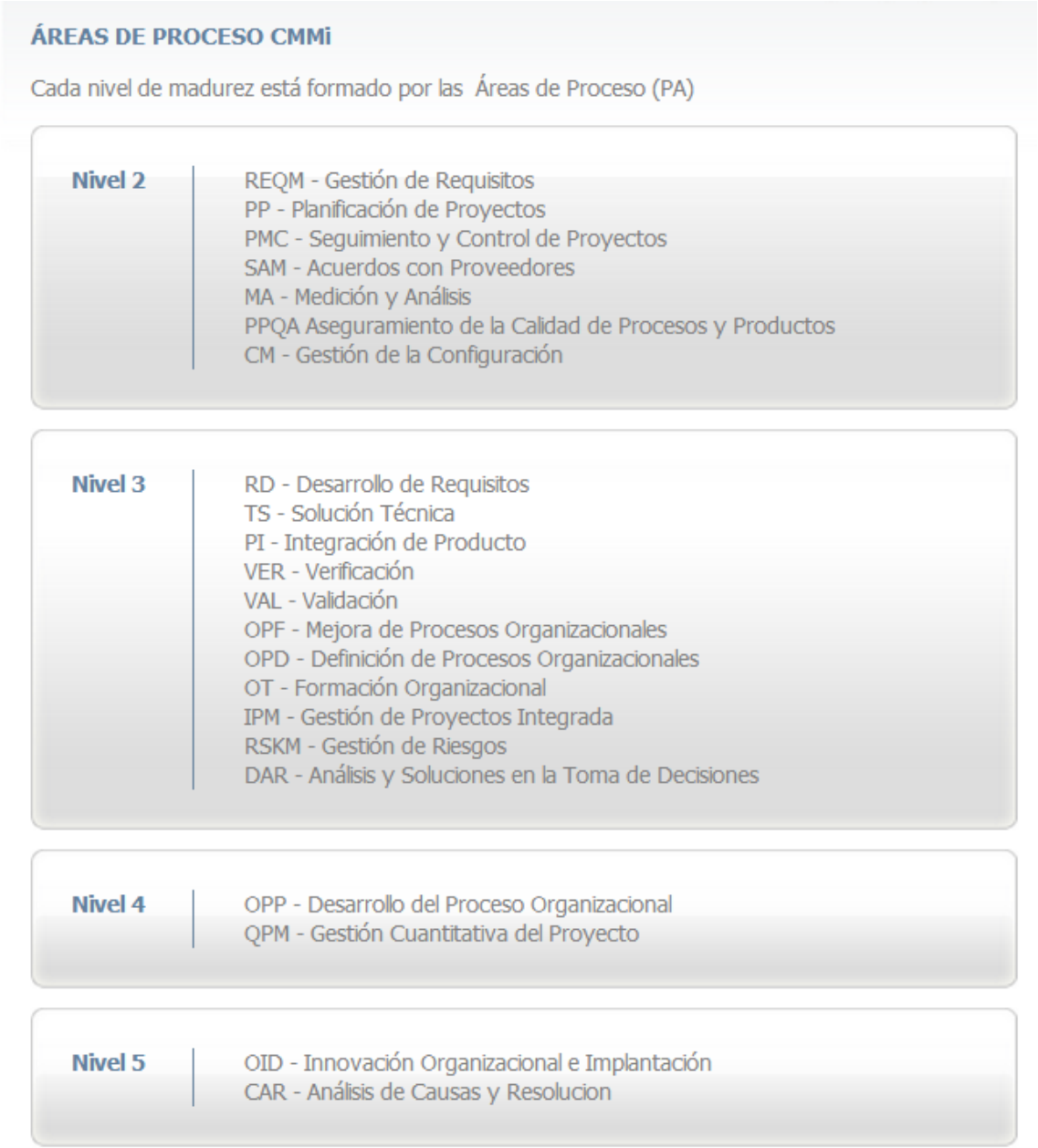
5.3 CMMI Dev

CMMI (Capability Maturity Model Integration) es un modelo de madurez de mejora de los procesos para el desarrollo de productos y de servicios. Consiste en las mejores prácticas que tratan las actividades de desarrollo y de mantenimiento que cubren el ciclo de vida del producto, desde la concepción a la entrega y el mantenimiento.

CMMI para desarrollo es un modelo de referencia que cubre las actividades del desarrollo y del mantenimiento aplicadas tanto a los productos como a los servicios. Las organizaciones de numerosas industrias, incluyendo la aeroespacial, los bancos, la construcción de ordenadores, el software, la defensa, la fabricación del automóvil y las telecomunicaciones, utilizan el CMMI para desarrollo.

Los modelos de la constelación del CMMI para desarrollo contienen prácticas que cubren la gestión de proyectos, la gestión de procesos, la ingeniería de sistemas, la ingeniería del hardware, la ingeniería de software y otros procesos de soporte utilizados en el desarrollo y el mantenimiento. El modelo CMMI para desarrollo + IPPD cubre también la utilización de equipos integrados que están implicados en las actividades de desarrollo y mantenimiento (IPPD).

Figura 2: Áreas de proceso de CMMi



5.4 PMBOK (Project management body of knowledge)

El PMBOK es una colección de procesos y áreas de conocimiento generalmente aceptadas como las mejores prácticas dentro de la gestión de proyectos. El PMBOK es un estándar reconocido internacionalmente (IEEE Std

1490-2003) que provee los fundamentos de la gestión de proyectos que son aplicables a un amplio rango de proyectos, incluyendo construcción, software, ingeniería, etc.

El 'PMBOK' reconoce 5 grupos de procesos básicos y 9 áreas de conocimiento comunes a casi todos los proyectos.

Los procesos se traslapan e interactúan a través de un proyecto o fase y son descritos en términos de:

Entradas (documentos, planes, diseños, etc.)

Herramientas y Técnicas (mecanismos aplicados a las entradas)

Salidas (documentos, productos, etc.)

Los 5 grupos básicos de procesos son:

1. Iniciación:

Define y autoriza el proyecto o una fase del mismo. Está formado por dos procesos.

2. Planificación:

Define, refina los objetivos y planifica el curso de acción requerido para lograr los objetivos y el alcance pretendido del proyecto. Está formado por veinte procesos.

3. Ejecución:

Compuesto por aquellos procesos realizados para completar el trabajo definido en el plan a fin de cumplir con las especificaciones del mismo. Implica coordinar personas y recursos, así como integrar y realizar actividades del proyecto en conformidad con el plan para la dirección del proyecto. Está formado por ocho procesos.

4. Seguimiento y Control:

Mide, supervisa y regula el progreso y desempeño del proyecto, para identificar áreas en las que el plan requiera cambios. Está formado por diez procesos.

5. Cierre:

Formaliza la aceptación del producto, servicio o resultado, y termina ordenadamente el proyecto o una fase del mismo. Está formado por dos procesos.

Las nueve áreas del conocimiento mencionadas en el PMBOK son:

1. Gestión de la Integración del Proyecto:

Incluye los procesos y actividades necesarios para identificar, definir, combinar, unificar y coordinar los diversos procesos y actividades de la dirección de proyectos dentro de los grupos de procesos de dirección de proyectos.

2. Gestión del Alcance del Proyecto:

Incluye los procesos necesarios para garantizar que el proyecto incluya todo (y únicamente todo) el trabajo requerido para completarla con éxito.

3. Gestión del Tiempo del Proyecto:

Incluye los procesos requeridos para administrar la finalización del proyecto a tiempo.

4. Gestión de los Costos del Proyecto:

Incluye los procesos involucrados en estimar, presupuestar y controlar los costos de modo que se complete el proyecto dentro del presupuesto aprobado.

5. Gestión de la Calidad del Proyecto:

Incluye los procesos y actividades de la organización ejecutante que determinan responsabilidades, objetivos y políticas de calidad a fin de que el proyecto satisfaga las necesidades por la cuales fue emprendido.

6. Gestión de los Recursos Humanos del Proyecto:

Incluye los procesos que organizan, gestionan y conducen el equipo del proyecto.

7. Gestión de las Comunicaciones del Proyecto:

Incluye los procesos requeridos para garantizar que la generación, la recopilación, la distribución, el almacenamiento, la recuperación y la disposición final de la información del proyecto sean adecuados, oportunos y entregada a quien corresponda (interesados del proyecto o stakeholders).

8. Gestión de los Riesgos del Proyecto:

Incluye los procesos relacionados con llevar a cabo la planificación de la gestión, identificación, el análisis, la planificación de respuesta a los riesgos, así como su monitoreo y control en un proyecto..

9. Gestión de las Adquisiciones del Proyecto:

Incluye los procesos de compra o adquisición de los productos, servicios o resultados que es necesario obtener fuera del equipo del proyecto.

Figura 3: Grupos de procesos PMBOK

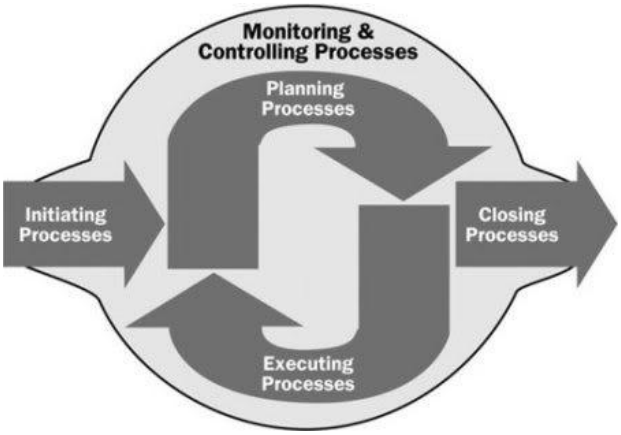
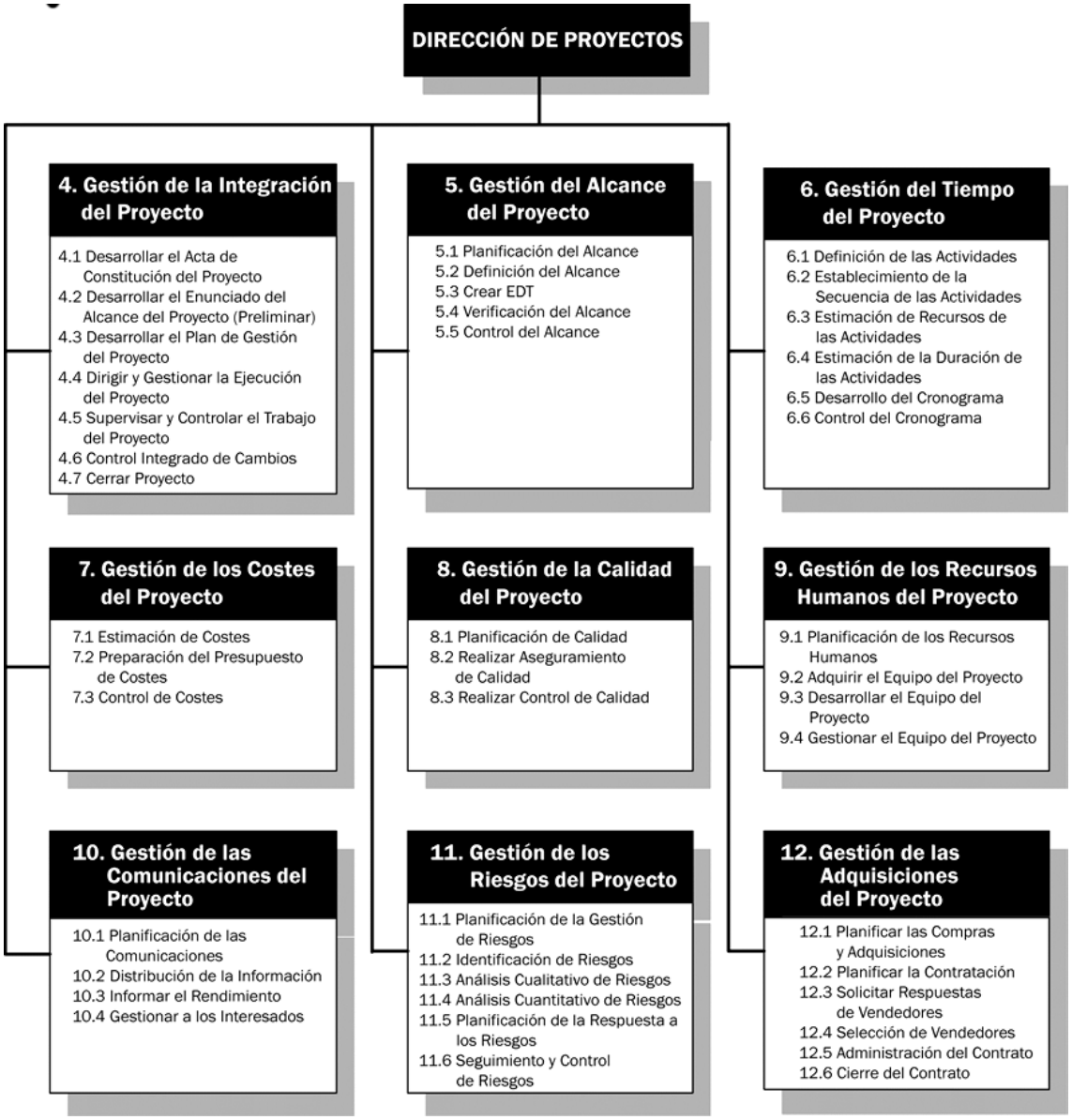


Figura 4: Procesos vs. Áreas de conocimiento PMBOK



6. ESTADO DE LA METODOLOGÍA

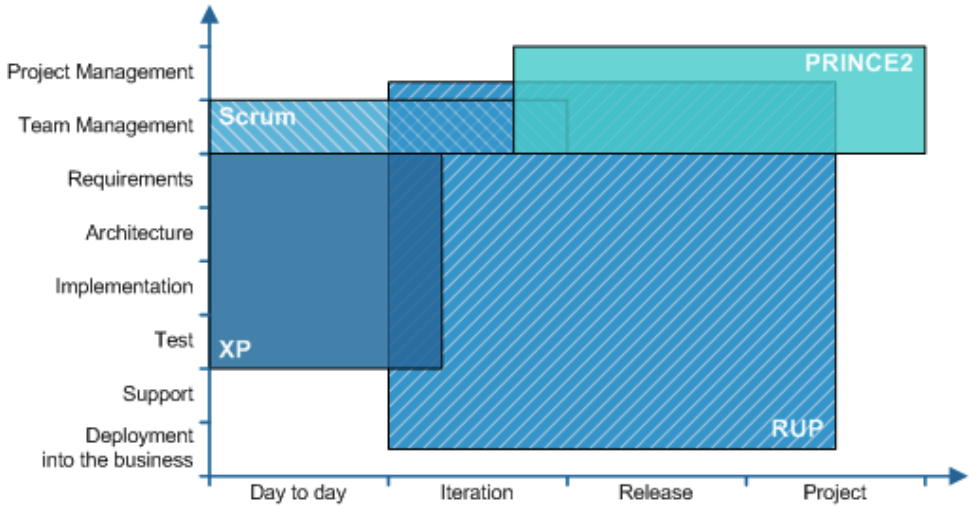
Para la Universidad EAFIT, una de sus principales responsabilidades en los programas, donde la ingeniería de software es un tópico de gran relevancia, debe ser preparar a los estudiantes para trabajar en la industria del software. Esta responsabilidad debe verse reflejada, asegurando que los objetivos y resultados de los programas, se correspondan con la estructura y contenido de los mismos. Los proyectos académicos de desarrollo de software son el punto más crítico en el cumplimiento de esta responsabilidad, al tratar de hacer una comparación, o al menos una analogía, entre la realidad del desarrollo de software en la industria con la realidad actual en la universidad.

Los proyectos académicos de desarrollo de software deben proporcionar a los estudiantes la oportunidad de poner en práctica el conocimiento y las habilidades adquiridas en cursos anteriores, en contextos de problemas de diferente alcance, desde la concepción hasta la implementación de la solución. Los escenarios para estas experiencias deben emular a la realidad de la industria tan cerca como sea posible, utilizando clientes reales (docentes, líderes de investigación, directores de proyecto), procesos, herramientas y criterios de tiempo y calidad. Los escenarios también deben reflejar la realidad de las responsabilidades que se asignan tanto a una persona como a un equipo de trabajo, el compromiso entre el tiempo y la calidad, y la necesidad de adaptarse y hacer frente a los imprevistos. El aprendizaje al practicar y la toma de decisiones críticas, en entornos con algo de incertidumbre, también deben formar parte de esa realidad.

El marco del trabajo SOFTWARE EDUP propone una metodología para orientar el desarrollo de software en proyectos académicos de diferente alcance. La metodología propuesta es una adaptación de metodologías ampliamente utilizadas y probadas en las organizaciones que conforman la industria nacional de software, con una inclinación pronunciada hacia la metodología RUP (Proceso Unificado de Rational, por sus siglas en español).

Esta inclinación hacia la metodología RUP, fue propuesta por el Ingeniero Jorge Abad Londoño, en su trabajo de tesis para la Maestría en Ingeniería. De acuerdo a la figura 5 (eje X, rangos de tiempo, y eje Y, disciplinas de trabajo), se puede apreciar una comparación entre las metodologías descritas en el marco teórico, y se evidencia que la metodología RUP cubre todas las disciplinas que se pretenden poner en práctica, en la ejecución de proyectos académicos de la Universidad EAFIT.

Figura 5: Comparación de metodologías



Estructura del proceso

La metodología SOFTWARE EDUP propone una estructura como la de RUP, la cual tiene dos dimensiones:

Eje horizontal

Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso, en términos de fases, iteraciones e hitos.

Eje vertical

Representa los aspectos estáticos del proceso. Describe el proceso en términos de componentes de proceso, disciplinas, actividades, tareas, artefactos y roles.

Fundamentos de la metodología

SOFTWARE EDUP establece una estructura que cubre todo el ciclo de vida de desarrollo de software, incluyendo fases, roles, actividades, artefactos, disciplinas, flujos de trabajo, gestión de riesgos, control de calidad, gestión del proyecto y control de configuración. En general, esta metodología tiene su base en los contenidos de la metodología RUP, e incorpora algunos elementos de OpenUP, XP, y Scrum.

Cabe destacar que los elementos en SOFTWARE EDUP fueron considerados mediante un análisis de varias metodologías, en la que se compararon las mismas respecto a sus elementos, lo cual permitió la selección de elementos que han tenido éxito en la industria de desarrollo de software, así como también de

elementos que se adaptan a las necesidades de los proyectos académicos que se ejecutan en la Universidad EAFIT.

Especificando los elementos que fueron identificados, analizados y comparados de cada metodología, se puede decir que las mejores prácticas para el desarrollo de software que se recopilan en SOFTWARE EDUP, hacen parte de las metodologías RUP, OpenUP y Scrum.

SOFTWARE EDUP propone una estructura que es una adaptación de la propuesta por metodología RUP, es decir, la conforman aspectos dinámicos y estáticos. Las fases e hitos corresponden los aspectos dinámicos y las disciplinas que corresponde a los aspectos estáticos.

Las metodologías en las que se basa SOFTWARE EDUP son algunas de las más utilizadas, además de que permiten la adaptación, es decir, son marcos de trabajo que permiten seleccionar elementos según las características de cada proyecto. La selección de esos elementos permitió definir la metodología SOFTWARE EDUP para el desarrollo de software, con un enfoque de calidad y cumpliendo con las necesidades de los proyectos académicos de la Universidad EAFIT.

ESTRUCTURA DINÁMICA DE LA METODOLOGÍA

SOFTWARE EDUP se repite a lo largo de una serie de ciclos que constituyen la vida de un proyecto. Cada ciclo concluye con una generación del producto y consta de cuatro fases. Cada fase se subdivide a la vez en iteraciones, el número de iteraciones en cada fase es variable.

Cada fase concluye con un hito bien definido, un punto en el tiempo en el cual se deben tomar ciertas decisiones críticas y alcanzar la meta clave antes de pasar a la siguiente fase, ese hito principal de cada fase se compone de hitos menores que podrían ser los criterios aplicables a cada iteración. Los hitos son puntos de control en los cuales los involucrados en el proyecto revisan el progreso del proyecto.

El ciclo de vida propuesto por la metodología SOFTWARE EDUP para un proyecto de software se compone en el tiempo de cuatro fases secuenciales que son: Inicio, Elaboración, Construcción y Transición. Al final de cada fase el director de proyecto realiza una evaluación para determinar si los objetivos se cumplieron y así pasar a la fase siguiente.

1. Inicio

El propósito general es establecer los objetivos para el ciclo de vida del producto.

Los objetivos específicos de esta fase son:

- Establecer el ámbito del proyecto y sus límites.

- Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Identificar al menos una arquitectura candidata para los escenarios principales.
- Estimar el costo en recursos y tiempo de todo el proyecto.
- Identificar riesgos, y las fuentes de incertidumbre.

El hito en esta fase finaliza con el establecimiento de la viabilidad del proyecto, el acuerdo del alcance del producto, e identificación de los principales riesgos.

2. Elaboración

El propósito general es plantear la arquitectura para el ciclo de vida del producto. Se construye un modelo de la arquitectura, que se desarrolla en iteraciones sucesivas hasta obtener el producto final, este prototipo debe contener los casos de uso críticos que fueron identificados en la fase de inicio. En esta fase se realiza la recopilación de la mayor parte de los requisitos funcionales, y se gestionan los riesgos que interfieran con los objetivos del sistema

Los objetivos específicos de esta fase son:

- Madurar, validar y establecer la arquitectura.
- Demostrar que la arquitectura propuesta soportara la visión, con un costo y tiempo razonables.
- Refinar la estimación del costo de recursos y tiempo del proyecto
- Mitigar los riesgos identificados.

El hito en esta fase finaliza con la obtención de una línea base de la arquitectura del sistema, la recopilación de la mayoría de los requisitos y la mitigación de los riesgos importantes.

3. Construcción

El propósito general es alcanzar la capacidad operacional del producto de forma incremental a través de iteraciones sucesivas. En esta fase todas las características, componentes, y requisitos deben ser integrados, implementados, y probados en su totalidad, obteniendo una versión aceptable del producto; comúnmente llamada versión beta.

Los objetivos específicos de esta fase son:

- Desarrollar servicios, casos de uso (funcionalidades), módulos, y/o subsistemas del proyecto.
- Realizar desarrollo por iteraciones, priorizando los servicios, casos de usos, módulos y/o subsistemas más críticos para el negocio.
- Realizar pruebas funcionales de cada uno de los servicios, casos de usos, módulos y/o subsistemas a medida que se vayan desarrollando.

El hito en esta fase finaliza con el desarrollo del sistema con calidad de producción y la preparación para la entrega al equipo de transición. Toda la funcionalidad debe haber sido implementada y las pruebas para el estado beta de la aplicación completadas. Si el proyecto no cumple con estos criterios de cierre, entonces la transición deberá posponerse una iteración.

4. Transición

El propósito general es entregar el producto funcional en manos de los usuarios finales (docente, comunidad, grupo de investigación), con toda la documentación donde se especifique la instalación, configuración, y usabilidad del producto.

Los objetivos específicos de esta fase son:

- Conseguir un producto final que cumpla los requisitos definidos.
- Garantizar que el usuario está en capacidad de instalar y operar el sistema.

El hito en esta fase se alcanza cuando el cliente revisa y acepta los artefactos que le han sido entregado.

ESTRUCTURA ESTÁTICA DE LA METODOLOGÍA

Una metodología de desarrollo de software define quién hace qué, cómo y cuándo.

La metodología SOFTWARE EDUP define cuatro elementos: los roles, que responden a la pregunta ¿Quien?; las actividades, que responden a la pregunta ¿Cómo?; los artefactos, que responden a la pregunta ¿Qué?; y los flujos de trabajo de las disciplinas que responden a la pregunta ¿Cuándo?

1. Disciplinas

La metodología SOFTWARE EDUP se organiza en disciplinas. Las disciplinas están compuestas por un conjunto de tareas, las cuales tienen como objetivo producir artefactos.

Las disciplinas que conforman esta metodología se dividirán en dos grupos. El primero comprende las disciplinas fundamentales asociadas con las áreas de ingeniería:

- Modelado empresarial
- Requisitos
- Análisis y Diseño
- Implementación
- Pruebas
- Despliegue

El segundo grupo lo integran las disciplinas llamadas de soporte o gestión:

- Gestión del proyecto
- Gestión de cambios

La descripción de cada una de las disciplinas puede consultarse en la publicación de la metodología, la cual es un sitio Web estático y es parte integral de este proyecto de grado.

ARTEFACTOS

La metodología consta de 24 artefactos en total divididos en cada una de sus disciplinas. Cabe anotar, que los artefactos se aplican dependiendo del tamaño y alcance de los proyectos.

Proyectos simples:

- Requisitos (fase de inicio):
    - Casos de uso
    - Documento de requisitos del sistema DRS
    - Glosario
    - Solicitudes de los interesados
    - Visión del sistema
- Análisis y diseño (fase de elaboración):
    - Documento de arquitectura de software DAS
- Implementación (fase de construcción):
    - Registro de pruebas unitarias
- Pruebas (fase de construcción):
    - Ideas de prueba
- Gestión del proyecto (todas las fases):
    - Acta de reunión
    - Gráfica del trabajo pendiente
    - Lista de elementos de trabajo
    - Plan del proyecto
    - Registro de revisión
    - BPMN
- Gestión de cambios (fase de transición):
    - Solicitud de cambio

Proyectos complejos:

- Modelado de negocio (fase de inicio):
    - Organigrama
    - Modelo de procesos
- Requisitos (fase de inicio):
    - Casos de uso
    - Documento de requisitos del sistema DRS
    - Glosario
    - Solicitudes de los interesados

    - Visión del sistema
- Análisis y diseño (fase de elaboración):
    - Decisiones de arquitectura
    - Documento de arquitectura de software DAS
- Implementación (fase de construcción):
    - Registro de pruebas unitarias
- Pruebas (fase de construcción):
    - Casos de pruebas
    - Ideas de prueba
    - Lista de ideas de prueba
- Gestión del proyecto (todas las fases):
    - Plan de gestión de comunicaciones
    - Plan de gestión de riesgos
    - Acta de entrega del proyecto
    - Acta de reunión
    - Gráfica del trabajo pendiente
    - Lista de elementos de trabajo
    - Plan del proyecto
    - Registro de revisión
    - Informe de avance
    - Plan de iteración
    - BPMN
- Gestión de cambios (fase de transición):
    - Solicitud de cambio

7. PROBLEMAS FORMULADOS

Para la implementación de metodología se formularon dos problemas de diferente alcance y tamaño para así demostrar la aplicabilidad de SOFTWARE EDUP:

7.1 Proyecto simple. Cajero CDT

Sistema por el cual se pueden hacer retiros, consignaciones o simplemente para revisar el monto de una cuenta CDT, dichos retiros serán administrados como prestamos, a la cuenta de la cual se ejecutan, posteriormente se restarían del saldo. Así, los usuarios evitarían molestias a la hora de hacer una consignación de una cuenta CDT a la cuenta de ahorros, otro servicio particular contemplado en el sistema, es el de tarjeta con la cual podrás pagar artículos con buenos planes de financiación.

Es un proyecto de dificultad y alcance bajo, cuyo tiempo calculado para ser realizado no pasa las dos semanas de trabajo y puede ser hecho por una o dos personas. Este tipo de proyecto es cercano a la realidad de los proyectos universitarios para estudiantes no expertos y en entrenamiento.

Entregables

Se generaron con base en la metodología los siguientes entregables:

- Requisitos:
  - CU-001-CONSULATR SALDO
  - CU- 003-RETIRAR SALDO
  - Documento de requisitos del sistema DRS
- Análisis y diseño:
  - Documento de arquitectura de software DAS Informal
- Implementación:
  - Registro de pruebas unitarias
- Pruebas:
  - Ideas de pruebas – casos de pruebas - pruebas
- Gestión del proyecto:
  - Acta Entrega del Proyecto
  - Acta de reunión
  - Lista de elementos de trabajo
  - Plan de proyecto informal
  - Plan de gestión de riesgos
  - Registro de revisión
- Gestión de cambios:
  - Solicitud de cambio

7.2 Proyecto complejo. Sistema de gestión de restaurante

Se trata de un sistema que recibe los pedidos por parte de los clientes (a través de un mesero o directamente) y los comunica al área de cocina del restaurante. El sistema deberá ser capaz de listar los menús a los clientes y una vez realizado el pedido, mostrarles el estado del pedido (tiempo, turno, etc.). El sistema deberá también poseer la capacidad de que el cliente añada notas a su pedido (Ej: Sin cebolla, asado a término medio, etc.)

El sistema también podrá atender los domicilios a través de un operador. Recibirá como entradas la dirección y el pedido. Con esta información deberá ser capaz de calcular la ruta más rápida para entregar el domicilio por parte del mensajero. Debido a esta característica, el sistema deberá tener un sistema de mapas de la cuidad y la capacidad de operar en dispositivos móviles (para meseros y mensajeros). A su vez, el sistema deberá comunicar por medio de una página web el estado del domicilio al cliente

El proyecto es de dificultad media-alta, creado para ser realizado entre 6 y 8 semanas de trabajo por un grupo de unos 4 o 5 ingenieros capacitados en las tecnologías requeridas.

Entregables

Se generaron con base en la metodología los siguientes entregables:

- Requisitos:
  - CU - 001 – TOMAR PEDIDO
  - CU - 008 – CONSULTAR ESTADO DEL PEDIDO
  - CU - 011 – CONSULTAR VALOR PEDIDO
  - CU- 002-CONSULTAR MENU
  - Documento de Requisitos del Sistema DRS
  - Solicitudes de los Interesados
  - Visión del Sistema
- Análisis y diseño
  - BPM-GDP
  - Documento de Arquitectura de Software DAS
  - Decisiones de Arquitectura
  - BPM
  - Prototipo
- Implementación
  - registro pruebas unitarias
- Pruebas
  - Casos de prueba tpl
  - Ideas de prueba
- Gestión del proyecto
  - Acta de Entrega del Proyecto
  - Acta de Reunión
  - Lista de Elementos de Trabajo
  - Plan de Gestión de Riesgos

    - Plan del Proyecto
    - Registro de Revisión
- Gestión de cambios
    - Solicitud de cambio

8. CONCLUSIONES

- SOFTWARE EDUP es un marco de trabajo que establece una metodología unificada para ejecutar los proyectos académicos de desarrollo de software, que se proponen en las asignaturas que pertenecen al área de ingeniería de software, de la Universidad EAFIT. Estas asignaturas corresponden tanto a programas de pregrado, como de posgrado.

- SOFTWARE EDUP implementa una metodología de desarrollo de software haciendo una adaptación de la metodología de desarrollo RUP.

- SOFTWARE EDUP es una metodología adaptable a las necesidades del proyecto, prueba de esto, es que actualmente dentro del marco de trabajo se han definido tres procesos que permiten guiar las actividades de desarrollo de software en proyectos académicos de diferente alcance.

- SOFTWARE EDUP es una metodología que genera valor iterativamente, está dirigida por casos de uso, se centra en la definición de una arquitectura, y el desarrollo se basa en componentes.

- El ciclo de vida de un proceso de desarrollo en SOFTWARE EDUP consiste de cuatro fases (inicio, elaboración, construcción y transición), donde cada fase puede incorporar una o varias iteraciones, donde se realizan actividades de aseguramiento de calidad y mitigación de riesgos, de manera continua.

- Existe una marcada diferencia en el conocimiento de la capacidad de adaptación de la metodología de desarrollo RUP para la ejecución de proyectos de software de cualquier tamaño; mientras en el ámbito académico se desconoce o se cataloga a RUP como una metodología pesada, en el ámbito de la industria local y nacional se conoce y valora esta propiedad, lo que ha permitido a muchas compañías (por no decir la mayoría) a definir sus procesos de desarrollo de software como una adaptación de la metodología de desarrollo RUP.

- Es necesario establecer programas de capacitación y entrenamiento para los usuarios (estudiantes, docentes, grupos de investigación, etc.) de la nueva propuesta metodológica, con el propósito de facilitar la adopción de la metodología de desarrollo. Adicionalmente, se deben considerar la creación o modificación de asignaturas que capaciten a los estudiantes en actividades de gestión de proyectos, gestión de riesgos, y gestión de la configuración y el cambio.

- En el momento que SOFTWARE EDUP se establezca como la metodología unificada y de fácil consulta, para orientar los proyectos académicos de desarrollo de software, se deben definir mecanismos para recibir, revisar y valorar la retroalimentación de los usuarios, con el propósito de ajustar y

mejorar la metodología. La metodología de desarrollo debe ser un activo que mejore continuamente.

- La utilización de la metodología SOFTWARE EDUP por parte de los estudiantes, permitirá mejorar la capacidad de adopción y adaptación de un proceso de desarrollo para la ejecución de sus proyectos de software; con el principal propósito de estrechar la brecha existente entre las competencias de los profesionales que egresan y las competencias que demanda la industria.
- La metodología es totalmente adaptable a diferentes tipos de proyectos (alcance y tamaño), lo que la capacita para aplicarse en cualquier etapa de enseñanza e incluso en ambientes laborales y profesionales.

BIBLIOGRAFÍA

Métodologías Ágiles en el Desarrollo de Software. José H. Canós, Patricio Letelier y Mª Carmen Penadés. DSIC -Universidad Politécnica de Valencia [en línea].

http://www.willydev.net/descargas/prev/TodoAgil.Pdf

Universidad Nacional de Trujillo. Facultad de Ciencias Físicas y Matemáticas. Escuela de Informática. Metodologías Ágiles. Amaro Calderón, Sarah Dámaris. Valverde Rebaza. Jorge Carlos [en línea].

http://www.seccperu.org/files/Metodologias%20Agiles.pdf

Manifiesto por el Desarrollo Ágil de Software [en línea].

http://www.agilemanifesto.org/iso/es/manifesto.html

CMMI® Guía para la integración de procesos y la mejora de productos Segunda edición. Mary Beth Chrissis. Mike Konrad. Sandy Shrum [en línea].

http://www.sei.cmu.edu/library/assets/cmmi-dev-v12-spanish.pdf

Risk reduction with the RUP phase plan

Mark Aked, Managing Consultant, Lamri Ltd. [en línea].

http://www.ibm.com/developerworks/rational/library/1826.html#N100E4

IBM Rational Unified Process Best Practices for Software Development Teams [en línea].

http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Guía de los Fundamentos de laDirección de Proyectos Tercera Edición [en línea].

http://gio.uniovi.es/documentos/software/GUIA_PMBok.pdf

Qué es el PMBOK®, y cómo usarlo Por Jaime González, PMP® [en línea].

http://liderdeproyecto.com/manual/que_es_el_pmbok.html

AMBYSOFT. Best practices for software development [en línea]. <http://www.ambysoft.com/>

BERGANDY, Jan. Teaching Software Engineering with Rational Unified Process (RUP). ASEE New England Section 2006 Annual Conference. 2006

BERGSTRÖM, Stefan y RABERG, Lotta. How to adopt RUP in your Project. En: Adopting the rational unified process: Success with the RUP. Addison Wesley, 2003. p. 127-140.

BOOCH, Grady; RUMBAUGH, James y JACOBSON, Ivar. The Unified Modeling Language: User guide. 2 ed. Westford, MA, Addison Wesley, 2005.

CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN. Metodología de la red nacional de integración y desarrollo de software libre (MeRinde): Guía detallada.

COCKBURN, Alistair. Writing effective use cases. Addison Wesley, 2001.

ECLIPSE. Introduction to OpenUP (Open Unified Process) [en línea]. <http://www.eclipse.org/epf/general/OpenUP.pdf>

ÉCOLE POLYTECHNIQUE DE MONTRÉAL. Unified Process for Education (UPEDU) [en línea]. <http://www.upedu.org/>

IBM. IBM Rational Method Composer [en línea]. <http://www-01.ibm.com/software/awdtools/rmc/>

IBM. IBM Rational Unified Process (RUP) [en línea]. <http://www-01.ibm.com/software/awdtools/rup/>

IBM. IBM Rational Unified Process (RUP) para proyectos pequeños [en línea]. <http://cgrw01.cgr.go.cr/rup/RUP.es/LargeProjects/index.htm#core.base_rup/guidances/supportingmaterials/welcome_2BC5187F.html>

IBM. Rational Process Library: The industry's most robust collection of best practices guidance [en línea]. <http://www-01.ibm.com/software/awdtools/rmc/library/>

IBM. Using RUP to manage small projects and teams [en línea]. <http://www.ibm.com/developerworks/rational/library/jul05/kohrell/>

INSTITUTO COLOMBIANO DE NORMALIZACIÓN Y CERTIFICACIÓN. Documentación, presentación de tesis, trabajos de grado y otros trabajos de investigación. NTC 1486. 6 ed. Bogotá D.C., ICONTEC, 2008.

INSTITUTO COLOMBIANO DE NORMALIZACIÓN Y CERTIFICACIÓN. Referencias bibliográficas: Contenido, forma, y estructura. NTC 5613. Bogotá D.C., ICONTEC, 2008.

JACOBSON, Ivar; BOOCH, Grady y RUMBAUGH, James. El Proceso Unificado de desarrollo de software. Madrid, Addison Wesley, 2000.

JOSKOWICZ, José. Reglas y prácticas en eXtreme Programming. Universidad de Vigo. 2008.

KROLL, Per; KRUCHTEN, Philippe y BOOCH, Grady. The rational unified process made easy: A practitioner's guide to the RUP. Addison Wesley, 2003.

KRUCHTEN, Philippe. Architectural Blueprints: The 4+1 view model of software architecture. IEEE Software, 1995.

KRUCHTEN, Philippe. The Rational Unified Process: An Introduction. 3 ed. Addison Wesley, 2003.

LARMAN, Craig. Agile and Iterative development: A manager's guide. Addison Wesley, 2004.

LETELIER, Patricio. Proceso de desarrollo de software. Universidad Politécnica de Valencia. 2003.

LETELIER, Patricio y PENADÉS, María Carmen. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Universidad Politécnica de Valencia. 2003.

MARTÍNEZ, Alejandro y MARTÍNEZ, Raúl. Guía a Rational Unified Process. Albacete, Escuela Politécnica Superior de Albacete, 2000.

PALACIO, Juan y RUATA, Claudia. Scrum Manager: Gestión de proyectos. 2011.

PRESSMAN, Roger. Software Engineering: A Practitioner´s Approach. 5 ed. New York, McGraw-Hill Higher Education, 2000.

PROJECT MANAGEMENT INSTITUTE. PMBoK (Project Management Body of Knowledge), version 4 [en línea]. <http://www.pmi.org/PMBOK-Guide-and-Standards.aspx>

RUMBAUGH, James; JACOBSON, Ivar y BOOCH, Grady. The Unified Modeling Language: Reference Manual. 2 ed. Boston, MA, Addison Wesley, 2004.

SOFTWARE ENGINEERING INSTITUTE. CMMI for development, version 1.3 [en línea]. <http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm>

SOLUCIONES INFORMÁTICAS GLOBALES. CMMI – Capability Maturity Model Integration [en línea].

<http://www.globales.es/imagen/internet/Informaci%C3%B3n%20General%20CMMI.pdf>

TOBARRA, Manuel. CMM y RUP: Una perspectiva común. Universidad de Castilla – La Mancha. 2003

VILLAGRA, Sergio. Una introducción a CMMI. Axentia. 2006