FRAMEWORK PARA EL DESARROLLO DE SOFTWARE EN ENTORNOS ACADÉMICOS

JORGE HERNÁN ABAD LONDOÑO

Tesis de maestría presentada como requisito parcial para optar al título de Magister en Ingeniería

Asesor
Rafael David Rincón Bermúdez

UNIVERSIDAD EAFIT
ESCUELA DE INGENIERÍA
MEDELLÍN
2012

Nota de aceptación:

Firma del presidente del jurado

Firma del Jurado

Firma del Jurado

Medellín, 8 de Noviembre del 2012

A mi madre Mariela Londoño Jaramillo, mujer incansable que durante toda mi vida me ha dado su apoyo y amor, siendo un constante ejemplo de perseverancia y dedicación para mí.

A mi hija Mariana, a quién amo entrañablemente.

A mi novia Diana, quien siempre estuvo ahí para darme ánimo y foco en la elaboración de este trabajo

Agradecimientos

Quiero dar gracias, mis gracias más sinceras, a todas las personas que contribuyeron sustancialmente a iniciar y concluir este proyecto de grado.

A mi asesor, amigo y mentor Rafael David Rincón, quien me animó a realizar este proyecto y me ayudó a comprender la necesidad de esta iniciativa.

A los estudiantes que colaboraron con sus trabajos de grado:

- Luis Felipe Tabares, ingeniero incansable, quien hizo grandes aportes a la estructura y contenido de este framework y que casi se deja atrapar por el sub-óptimo.
- A Juan Camilo Villa y a Mario López, por la creación de los ejemplos que ayudan al entendimiento del framework de parte de los estudiantes.

A la Universidad EAFIT, por darme la oportunidad de realizar este importante aporte, que espero sea de gran apoyo a la academia, y en consecuencia, a la industria, local y nacional.

A Dios, el Hacedor, por sus bendiciones incontables en mi vida y por permitirme un recorrido profesional y académico que desemboca en este aporte.

CONTENIDO

pág.

1. LISTA DE TABLAS

pág.

2. LISTA DE FIGURAS

pág.

3. GLOSARIO

Artefacto: Producto tangible resultante del proceso de desarrollo de software (o de una etapa del mismo). 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. Un artefacto puede ser compuesto por otros artefactos.

BPMN (*Business Process Management Notation* – Notación para el Modelado de procesos de negocio): Notación gráfica que presenta un flujo de trabajo sobre un negocio, propuesta por la OMG. Actualmente se encuentra en la versión 2.0.

Disciplina: Una disciplina es una categorización de las tareas que están relacionadas con un área importante de interés y la cooperación. Por ejemplo, se pueden identificar en RUP disciplinas de Modelado Empresaria, Requisitos, Arquitectura, Implementación, etc.

EDT (Estructura de desglose de trabajo): Estructura jerárquica enfocada en los entregables que serán realizados por el equipo del proyecto. Equivalente a WBS (*Work Breakdown Structure*)

Entregable: Es un grupo de productos de trabajo, generalmente artefactos. También es considerado una salida del proceso que tiene valor para el cliente, usuario, u otro *stakeholder*.

EPF Composer: El *Eclipse Process Framework Composer* (*EPF Composer*), es una herramienta gratuita, desarrollada por la comunidad Eclipse, que sirve para

crear y modificar contenido de métodos, procesos o metodologías, y generar automáticamente documentación adecuada en formato para la web.

Esquema: Prototipo de representación simple en donde se emplean trazos, imágenes simples, donde lo que se busca es representar campos de entrada y salida, botones. El foco del esquema no es la riqueza gráfica.

Fase: Una fase es un tipo de actividad que representa un periodo de tiempo en un proyecto. Las fases concluyen típicamente con un hito, o con un grupo de artefactos entregados.

Framework (marco de trabajo): Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

Guía de adaptación: Parametrización de un método, actividad, artefacto o entregable para un determinado escenario.

Hito: Un punto o evento significativo dentro de un proyecto.

Iteración: Es una secuencia de actividades que es realizada más de una vez. Las iteraciones ayudan a organizar el trabajo en varios ciclos.

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

Metodología: Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal.[1]

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

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.

Prototipo: Representación limitada de un sistema, módulo o funcionalidad, permite a las partes entender, probar, o hasta explorar su uso. Puede consistir desde un esquema simple hasta una interfaz enriquecida y con detalles últimos de diseño.

Proyecto: Un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único.[2]

---

[1] REAL ACADEMIA ESPAÑOLA. Diccionario de la Lengua Española - Vigésima Segunda Edición. [en línea]. (Citada: 4 de Julio. 2012) <http://rae.es/rae.html>

[2] PROJECT MANAGEMENT INSTITUTE. PMBoK (Project Management Body of Knowledge), version 4 [en línea]. (Citada: 16 de Junio de 2012) <http://www.pmi.org/PMBOK–Guide–and–Standards.aspx>

Refactoring **(refactorización):** Técnica de Ingeniería de Software para reestructurar el código fuente, modificando su estructura (buscando optimizarla) sin cambiar su comportamiento externo. Esta técnica de modificación de código requiere de dos condiciones: la primera, que exista un acuerdo en donde se consideren los fuentes como propiedad del equipo de trabajo, de forma que la modificación se realice sin solicitar permiso de otro miembro; y la segunda, que exista un repositorio de control de versiones que permita guardar cualquier modificación y devolverse a una versión anterior.

RoI: Es un conjunto bien definido de competencias, funciones y responsabilidades que pueden ser realizadas por una sola o varias personas; igualmente, una persona puede desempeñar varios roles.

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.

SOFTWARE: Según la IEEE, software es "Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación."

SPEM (*Software & Systems Process Engineering Metamodel Specification*): Es un estándar creado por la OMG (*Object Management Group*) para documentar procesos de Ingeniería de Software. Actualmente se encuentra en la versión 2.0.

Stakeholder **(Interesado):** Interesado, involucrado, afectado positiva o negativamente por la existencia del proyecto de software.

Ejemplos de *stakeholders*, son:

- El gerente del proyecto
- El estudiante ejecutor del proyecto
- El cliente
- El Sponsor: Persona que patrocina el proyecto con recursos económicos
- El equipo que construye el proyecto
- Los futuros usuarios del sistema

SUB-OPTIMO: "Hacer lo mejor posible algo que no debería hacerse**" [3] .** Refinar algo demasiado sin ser necesario y sin que sea observable. "Perder" tiempo en tareas o minucias que no agregan valor al resultado deseado.

TDD (*Test Driven Development* - Desarrollo guiado por las pruebas): Estrategia de desarrollo de software en la cual se escriben primero las pruebas unitarias y subsecuentemente el código, de esta forma se garantiza que el código está libre de errores.

UML (*Unified Modeling Language* - Lenguaje Unificado de Modelado): Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema, propuesto y mantenido por la OMG.

[3] VEGA, Jesús. Subóptimo. 2010. [en línea] (Citada: 19 de Junio de 2012) <http://archivo.expansionyempleo.com/2010/09/13/opinion/1284370257.html>

4. RESUMEN

El Desarrollo de software en escenarios académicos, en especial de pregrado, es caracterizado por extremos, durante mi ejercicio de la docencia tanto en pregrado como en posgrado en las universidades de Colombia he notado que este va desde la informalidad de materias en las que se desarrollan ciertos componentes o módulos, hasta la formalidad en materias enfocadas específicamente en Ingeniería de Software donde se buscan desarrollar pequeños, medianos o hasta grandes sistemas, y donde el esfuerzo del docente se centra en lograr que los estudiantes realicen una apropiación de buenas prácticas de ingeniería, como: análisis, diseño, documentación, versionamiento, modelado, descomposición funcional, buenas prácticas de codificación, entre muchas otras. Este trabajo busca proporcionar un framework metodológico con herramientas, plantillas, instructivos, políticas, métodos, entre otros, útil tanto para los desarrollos formales en materias relacionadas con Ingeniería de Software como en los desarrollos “informales” en las materias diferentes la misma, adicionalmente en trabajos dirigidos de grado, tesis y en algunos casos proyectos de investigación.

El framework se construyó con base en cuatro pilares fundamentales: RUP (metodología probada de IBM), CMMI (estándar que propone la incorporación y el uso de las mejores prácticas de Ingeniería de Software), y PMBoK (mejores prácticas en la gerencia de proyectos), y algunas prácticas Agiles, buscando dar al entorno académico elementos similares al de la industria, pero sin la rigurosidad de la misma, con el fin de proporcionar una herramienta útil al entorno académico y que siente unas bases sólidas para el ejercicio profesional.

Esta metodología comprende adaptaciones para proyectos académicos de diferente magnitud, y establece una guía para cada una de los roles implicados en el ciclo de vida del proceso de desarrollo de software. Está orientada principalmente a miembros del equipo de desarrollo que se dedican a las

actividades del ciclo de vida: modelado del negocio, requisitos, análisis y diseño, implementación, pruebas, implantación, gestión de cambios, y gestión del proyecto. Adicionalmente, es útil para que los estudiantes comprendan y aprendan las actividades a realizar por los diferentes roles en un proyecto tanto al interior de la academia como en la industria[4]: analistas (que especifican los requisitos y el comportamiento del sistema), arquitectos (que diseñan los elementos que satisfacen esos requisitos), desarrolladores (que convierten esos diseños en código ejecutable), testers (que verifican y validan la estructura y comportamiento del sistema) y para el líder de proyecto (que coordinan el equipo y se encargan de lograr la ejecución del proyecto).

Para la creación, configuración y publicación del framework metodológico SOFTWARE EDUP (nombrado así debido a que es un framework adaptado del proceso unificado de software para la educación), se empleo el marco de trabajo EPF (*Eclipse Process Framework*), el cual utiliza el estándar para la representación de modelos de procesos de Ingeniería de Software e Ingeniería de Sistemas SPEM (*Software and Systems Process Engineering Metamodel*), versión 2.0, definido por el OMG (*Object Management Group*, Inc.).

[4] van der Duim, L., Andersson, J., & Sinnema, M. (2007). Good Practices for Educational Software Engineering Projects. 29th International Conference on Software Engineering (ICSE'07), 698–707. doi:10.1109/ICSE.2007.40

5. PALABRAS CLAVE

RUP, CMMi, Software EdUP, Scrum, metodologías agiles, proceso software, ingeniería de software, cursos académicos, PMBoK, ingeniería de sistemas, ingeniería informática, ciencias de la computación, SPEM, Eclipse Process Framework, MeRinde, UPEDU, metodología para proyectos académicos.

6. INTRODUCCIÓN

El Desarrollo de software es uno de los procesos más complejos de la industria humana; implica conocimiento, conocimientos técnicos, un buen proceso, un buen grado de control y disciplina por parte de todos los involucrados en su realización.

Por lo tanto, resolver el "cómo" realizar el proceso corresponde a las actividades recurrentes en la Ingeniería de Software, la cual se encuentra en constante maduración y evolución, pues ese "cómo" puede involucrar desde un proceso improvisado de desarrollo de software, que trae consigo malas prácticas y pésimas consecuencias en resultados de tiempo, calidad y dinero; las metodologías robustas, como lo son el IBM Rational Unified Process, con un alto grado de formalismo, protocolo, entregables y documentos, hasta las metodologías Ágiles, que tienen poca documentación, características especiales de contratación y de involucramiento del cliente, y requieren gran disciplina y madurez del equipo de trabajo. Diseñar (o ajustar), enseñar y usar este Proceso Software dentro de una organización puede ser un camino bastante arduo que puede tomar en la adopción de CMMi nivel 3 aproximadamente 25 meses[5], trasladando estos esfuerzos a un escenario académico (objeto de este trabajo) se evidencia la necesidad de esfuerzos adicionales y articulados debido a que los estudiantes en formación no han desempeñado labores profesionales en ingeniería de software.

Es común que en el Proceso Software usado en las prácticas y proyectos académicos se encuentre sin directrices claras (fases, actividades, entregables, roles, tareas, procesos y herramientas), y sin la adopción de un modelo que

[5] Software Engineering Institute. CMMI for Development SCAMPI Class A Appraisal Results, 2011Update . March 2012. [en línea]. (Citada: 22 de Octubre de 2012) < www.sei.cmu.edu/cmmi/casestudies/profiles/pdfs/upload/2012MarCMMI.pdf >

presente las mejores prácticas existentes, que logren producir software de calidad bajo los estándares de documentación y funcionalidad como las que requiere el mercado, trayendo consigo proyectos descontextualizados de su entorno, y estudiantes, grupos de investigación y de trabajo que los ejecutan, sin lineamientos claros que incorporen buenas prácticas y redunden en beneficio de su desempeño profesional, centrándose en un desarrollo heroico que se realiza durante desgastantes horas, pocos días antes de la entrega[6] . Similarmente, profesores expertos en áreas de conocimiento específicas de la ingeniería de sistemas, de software, y ciencias de la computación, ejecutan, asesoran y coordinan dichos proyectos sin la adopción de las mejores prácticas de la industria, haciéndolos propensos al fracaso y deteriorando así una excelentes iniciativa.

Es de notar, que el entono académico cuenta con características que lo diferencian de los proyectos de la industria: menor presión por cumplimiento de cronograma y presupuesto, debido a que se pueden incorporar recursos a muy bajo costo bajo el esquema de prácticas o pasantías; un control de la calidad más laxo, pues siempre habrán recursos para corregir los errores, baja gestión del riesgo y una pésima gestión del alcance y de los cambios. Esto que hace que su control, ejecución, y adherencia a buenas prácticas, tanto de Ingeniería de Software, como de Gerencia de proyectos, llegue a ser más permisivo, implicando que en el proyecto se inviertan recursos de una forma ineficiente (y hasta ilimitada) y aumentando la probabilidad de fracaso del mismo.

Aunado a estos hechos las consecuencias también hacen repercusión en los estudiantes y su “ADN” cognitivo del Proceso Software, donde esas mejores

---

[6] Kruchten, P. (2011). Experience teaching software project management in both industrial and academic settings. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 199–208. doi:10.1109/CSEET.2011.5876087

prácticas de producción de software al no ser suficientemente valoradas y exigidas por el cuerpo docente encargado de la formación traen como consecuencia largos plazos de capacitación en la industria hacia los egresados (incurriendo en costos que debieron haber sido resueltos en la fase de formación profesional), y peor aun viciando las iniciativas de emprendimiento empresarial con malas prácticas que traerán como consecuencia una propensión al fracaso o mayores esfuerzos en alcanzar estándares de la industria.

Con esta tesis se quiere proporcionar al ámbito académico universitario de un framework para la ejecución de proyectos de construcción de software, (esfuerzo similar en otras latitudes: Canadá[7] y la República Bolivariana de Venezuela[8]), empleando las mejores prácticas de desarrollo de software y de gerencia de proyectos, que permita en las diferentes prácticas, tesis y trabajos universitarios generar software con todos sus entregables, actividades, roles y prácticas de calidad, guardando las proporciones y características del entorno, y buscando de una forma sistemática y contextualizada una apropiación del Proceso Software de parte de los estudiantes.

6.1 ANTECEDENTES

“La mayoría de los estudiantes universitarios tienen por lo general muy poca idea sobre lo que significa un proyecto, ven cualquier proyecto de software como un montón de programación - en su mayoría por la noche -, con un gran heroísmo en

[7] Pierre N. Robillard, Philippe Kruchten, Patrick d'Astous, "YOOPEEDOO (UPEDU): A Process for Teaching Software Process," cseet, pp.18, 14th Conference on Software Engineering Education and Training, 2001

[8] CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN. Metodología de la red nacional de integración y desarrollo de software libre (MeRinde): Guía detallada. (Citada: 16 de Junio. 2012) < http://merinde.net. >

depuración y su experiencia máxima en equipos de trabajo, consiste en tratar con 3 ó 4 compañeros de trabajo." [9] Bajo este escenario y la experiencia docente en espacios universitarios (en ingeniería de software y gestión de proyectos informáticos) de estos últimos 8 años, se concluye que las prácticas académicas, proyectos de curso, las tesis del ciclo de pregrado y postgrado, proyectos de investigación relacionados con el desarrollo de software se realizan en general, sin un proceso formal que cuente con una metodología definida y clara que incorpore las mejores prácticas de la industria.

Basado en la bibliografía y en mi experiencia profesional como docente en pregrados y especializaciones de ingeniería de sistemas, en las materias de ingeniería de software, y gestión de proyectos informáticos en las Universidades de Eafit y Medellín de la ciudad de Medellín, y confirmado por La recepción y asignación de los trabajos por parte de los profesores cuenta actualmente con las siguientes características:

- No se trabaja de forma estandarizada (en la misma o diferentes asignaturas)
- En materias donde existe desarrollo de software o componentes de software, los entregables esperados son diferentes en esas asignaturas y en muy pocos casos se solicitan evidencias de un proceso de software organizado.
- En muchos casos lo único que interesa resultado y no el proceso de obtenerlo.[10]

---

[9] Kruchten, P. (2011). Experience teaching software project management in both industrial and academic settings. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 199–208. doi:10.1109/CSEET.2011.5876087

[10] Bernhart, M., Grechenig, T., Hetzl, J., & Zuser, W. (2006). Dimensions of software engineering course design. Proceeding of the 28th international conference on Software engineering - ICSE '06, 667. doi:10.1145/1134285.1134387

- Existen esfuerzos aislados de diferentes profesores para inculcar buenas prácticas de desarrollo de software pero no es un trabajo coordinado por todo el Departamento/Carrera/Equipo.
- Existen casos de profesores que enseñan solo su "mundo" sin mirar donde se encuentra la industria y el mercado local.

De igual forma, desde el punto de vista de los estudiantes, los componentes de software, módulos y/o sistemas producidos por ellos cuentan con las siguientes características:

- Los estudiantes construyen software de la forma que creen o les dicen sus compañeros.
- Se emplean los artefactos y prácticas que les "obligan" unos profesores en unas materias pero dejan de ser empleadas al cambiar de asignatura.
- Los futuros ingenieros de sistemas aprenden a desarrollar software por "voz a voz" y no saben cuál es la mejor forma de hacerlo.
- A los estudiantes les importa solo el resultado, sin percibir lo importante de tener un buen proceso. [11]
- Los estudiantes desarrollan software de forma heroica y no organizada. [12]
- Los estudiantes por lo general no perciben el desarrollo de software como un ejercicio social, que involucra interacciones entre muchos interesados.[13]
- El proceso de software no se adhiere a su "ADN de estudiantes" [14]

---

[11] Kruchten, P. (2011). Experience teaching software project management in both industrial and academic settings. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 199–208. doi:10.1109/CSEET.2011.5876087

[12] Íbidem.

[13] van der Duim, L., Andersson, J., & Sinnema, M. (2007). Good Practices for Educational Software Engineering Projects. 29th International Conference on Software Engineering (ICSE'07), 698–707. doi:10.1109/ICSE.2007.40

Debido a lo anteriormente expuesto tanto por parte de profesores y alumnos, se están presentando los siguientes hechos:

- No se cuenta con un esquema formal de trabajo coherente que sea visualizado por los alumnos, con el cual identifiquen cómo asumir las mejores prácticas para llevar a cabo un proyecto de software exitoso.
- No existe uniformidad en los entregables producidos, tanto en la documentación funcional como en la técnica.
- Carencia de documentación de soporte al ciclo de vida en los proyectos académicos de software.
- Para las materias diferentes a Ingeniería de Software no hay un esquema de control en el cual se asegure que se están adoptando las mejores prácticas de la industria. [15]
- La evaluación de los trabajos académicos se concentra fundamentalmente en los resultados (producto de software) dejando de lado el proceso realizado y los roles involucrados (dicha evaluación debería contar con un componente de proceso, por ejemplo: repositorio, documentación, pruebas unitarias, pruebas funcionales, etc.) [16]
- Falta de uniformidad de criterios para construir y gestionar proyectos de software en las asignaturas que los incluyen.

---

[14] Bernhart, M., Grechenig, T., Hetzl, J., & Zuser, W. (2006). Dimensions of software engineering course design. Proceeding of the 28th international conference on Software engineering - ICSE '06, 667. doi:10.1145/1134285.1134387

[15] Kruchten, P. (2011). Experience teaching software project management in both industrial and academic settings. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 199–208. doi:10.1109/CSEET.2011.5876087

[16] van der Duim, L., Andersson, J., & Sinnema, M. (2007). Good Practices for Educational Software Engineering Projects. 29th International Conference on Software Engineering (ICSE'07), 698–707. doi:10.1109/ICSE.2007.40

- Los frameworks metodológicos son conocidos por los estudiantes como una referencia cognitiva, mas estos no son concebidos como una herramienta clave del desempeño y éxito profesional.[17]
- Ausencia de una metodología formal de desarrollo de los proyectos de software en los cursos relacionados con Programación o Construcción de Software.
- Desconocimiento de tipificación de proyectos para la construcción de software, que permita adaptar la metodología a los diferentes tipos de proyectos.
- Falta de uniformidad de criterios para construir y gestionar proyectos de software en las asignaturas que los incluyen.[18]
- No se cuenta con un esquema formal de trabajo coherente, que sea visualizado por los alumnos, con el cual identifiquen cómo asumir las mejores prácticas para llevar a cabo un proyecto de software exitoso.[19]
- Existe poca o pobre adopción de las mejores prácticas de la industria de software por parte de los alumnos que están realizando sus diferentes prácticas académicas.[20]

---

[17] Pierre N. Robillard, Philippe Kruchten, Patrick d'Astous, "YOOPEEDOO (UPEDU): A Process for Teaching Software Process," cseet, pp.18, 14th Conference on Software Engineering Education and Training, 2001

[18] Tremblay, G., Malenfant, B., Salah, A., & Zentilli, P. (2007). Introducing Students to Professional Software Construction : A " Software Construction and Maintenance " Course and its Maintenance Corpus, 176–180.

[19] Pierre N. Robillard, Philippe Kruchten, Patrick d'Astous, "YOOPEEDOO (UPEDU): A Process for Teaching Software Process," cseet, pp.18, 14th Conference on Software Engineering Education and Training, 2001

[20] Bernhart, M., Grechenig, T., Hetzl, J., & Zuser, W. (2006). Dimensions of software engineering course design. Proceeding of the 28th international conference on Software engineering - ICSE '06, 667. doi:10.1145/1134285.1134387

- La adopción voluntaria y sin seguimiento de buenas prácticas de Ingeniería de Software por parte de los estudiantes, se convierte en un factor de fricción en la productividad, debido a que al llegar al mundo laboral, requieren largos periodos de entrenamiento e inversiones de parte del empleador, que debieron haber sido realizadas y suplidas en el ciclo de formación profesional.
- Los emprendimientos realizados por los estudiantes están viciados por errores metodológicos que influencian de forma directa y posiblemente negativa el éxito de la iniciativa

Ante estos hechos y falencias, se busca con este trabajo, proporcionar al elementos académicos y didácticos para una absorción en el "ADN del estudiante" de un buen Proceso Software como la mejor manera de encarar los proyectos. Y así lograr que sea intrínseco para el futuro ingeniero el conocimiento y uso del proceso, la forma de adaptarlo y de emplearlo, y en consecuencia, lograr el dominio de uno de los vértices del triángulo garantiza un buen producto software. (Un buen producto o servicio depende de tres variables: procesos, personas y herramientas)[21]

[21] CHRISSIS, Mary Beth, KONRAD, Mike y SHRUM, Sandy. CMMI: Guidelines for Process Integration and Product Improvement. Second Edition. Boston, MA: Pearson Education, Inc., 2007.

Figura 1: Vértices que garantizan un buen producto o servicio. (Gráfica adaptada[22])

Se evidencia una necesidad de dar formalidad, uniformidad y coherencia al proceso de desarrollo de soluciones de software elaboradas en escenarios académicos, similar a los esfuerzos hechos en el 2005 y 2006 en la materias de Ingeniería de Software, Taller de Ingeniería de Software en Eafit, o los realizados por la École Polytechnique de Montréal con UPEDU www.upedu.org [23] (el cual apoya la enseñanza de cursos de ingeniería de software, más no a la construcción de todo tipo de software dentro de los diferentes escenarios académicos) , o el Gobierno Bolivariano de Venezuela con el Framework de MeRinde www.merinde.net [24] (el cual acota como audiencia "a cualquier persona implicada en el proceso de desarrollo de software que se lleva a cabo en el Centro Nacional de Tecnologías de Información (CNTI) – de la República Bolivariana de Venezuela

---

[22] Ibidem.

[23] Pierre N. Robillard, Philippe Kruchten, Patrick d'Astous, "YOOPEEDOO (UPEDU): A Process for Teaching Software Process," cseet, pp.18, 14th Conference on Software Engineering Education and Training, 2001

[24] CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN. Metodología de la red nacional de integración y desarrollo de software libre (MeRinde): Guía detallada. (Citada: 16 de Junio. 2012) < http://merinde.net. >

- y también a cualquier individuo, comunidad u organización interesada" [25] no se encuentra diseñado con roles, y entregables específicos para el ámbito académico universitario.) permitiendo que tanto profesores y alumnos perciban un proceso estructurado y coherente, para llevar a cabo proyectos de software basándose en las mejores prácticas de la industria y de esa forma lograr una apropiación del conocimiento (tan largo como ancho pues implican gran cantidad de disciplinas, especialidades y técnicas como lo expresa el SWEBOK[26] [27]) y del proceso por continuo uso (tanto de estudiantes, como profesores y demás involucrados), de forma que este conocimiento del área de Ingeniería de Software redunde en beneficio para la Ingeniería de Sistemas y la Industria de Software en general.

Por lo tanto, este trabajo pretende ser apoyo en la consecución de una industria de software local madura con mejores ingenieros, apoyar la formación de ingenieros con insumos lo mas similares posibles al mundo industrial[28], un empresarismo con bases sólidas y con menores tiempos en ciclos de certificación en estándares internacionales, pues busca la asimilación de un proceso–software por parte de los estudiantes como uno de los elementos clave en la construcción de soluciones informáticas, para luego replicarlo con éxito en el ámbito profesional.

---

[25] Íbidem

[26] Bernhart, M., Grechenig, T., Hetzl, J., & Zuser, W. (2006). Dimensions of software engineering course design. Proceeding of the 28th international conference on Software engineering - ICSE '06, 667. doi:10.1145/1134285.1134387

[27] Abran, A., and Moore, J. W., Guide to the Software Engineering Body of Knowledge: 2004 Edition - SWEBOK, IEEE , 2005.

[28] Bernhart, M., Grechenig, T., Hetzl, J., & Zuser, W. (2006). Dimensions of software engineering course design. Proceeding of the 28th international conference on Software engineering - ICSE '06, 667. doi:10.1145/1134285.1134387

6.2 Objetivos

6.2.1 Objetivo General

Construir un framework metodológico para la ejecución de proyectos académicos que involucren el desarrollo de software, tomando como referentes RUP, PMBoK y CMMI DEV 1.3 y prácticas de metodologías ágiles.

6.2.2 Objetivos Específicos

- Identificar y tipificar diferentes tipos de proyectos académicos que apoyen tanto la construcción de software en materias de ingeniería de software como en otras que requieran de la construcción del mismo para validar conceptos, de forma que se apoye tanto de forma intersemestral como semestral la construcción de software en las carreras de ingeniería de sistemas, ingeniería informática, ciencias de la computación y afines en el Valle de Aburrá (Antioquia – Colombia).
- Definir los procesos, flujos de trabajo, roles, entregables, actividades, para las diferentes disciplinas de Ingeniería de Software en la ejecución de proyectos académicos que involucren la construcción o el desarrollo de soluciones de software.
- Desarrollar un sitio web navegable con el framework propuesto.
- Validar la pertinencia del framework en cursos relacionados con Ingeniería de Software.

7. MARCO TEÓRICO

La Ingeniería de Software ha evolucionado hacia modelos que permiten conocer e implementar la madurez de las organizaciones para desarrollar software CMMi, SPICE, TSP, PSP; de igual forma se han generado grandes bases de conocimiento, como lo son RUP, SWEBoK, MSF, XP, OpenUP, entre otros, donde se consigna desde la mejor forma de construir y gestionar proyectos de software.

Esfuerzos como el de la École Polytechnique de Montréal, con un sitio web adaptado de RUP http://www.upedu.org/upedu en el que presenta a sus estudiantes una forma de aproximarse al desarrollo de software desde el Proceso Unificado de Desarrollo, orientado a soportar su curso de ingeniería de software de 4 año[29]. Otras iniciativas como la del Gobierno Bolivariano de Venezuela, con la Metodología de la Red Nacional de Integración y Desarrollo de Software Libre (MeRinde) http://merinde.net [30]en donde presentan una adaptación del Proceso Unificado de Software para el Desarrollo de Software libre en su país, teniendo como audiencia "a cualquier persona implicada en el proceso de desarrollo de software que se lleva a cabo en el Centro Nacional de Tecnologías de Información (CNTI) – de la República Bolivariana de Venezuela - y también a cualquier individuo, comunidad u organización interesada", teniendo de esta forma una orientación estrictamente industrial, mas no enfocado a proyectos de software dentro de los ámbitos académicos. Cabe anotar, que ninguna de estas dos propuestas tienen características de adaptabilidad, limitando su implementación

[29] Pierre N. Robillard, Philippe Kruchten, Patrick d'Astous, "YOOPEEDOO (UPEDU): A Process for Teaching Software Process," cseet, pp.18, 14th Conference on Software Engineering Education and Training, 2001

[30] CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN. Metodología de la red nacional de integración y desarrollo de software libre (MeRinde): Guía detallada. (Citada: 16 de Junio. 2012) < http://merinde.net. >

solo para un tipo específico de proyecto, dejando escenarios de trabajo sin cubrir, siendo esta su principal debilidad, adicional a las del entorno para el cual fueron desarrollados y el no reflejar las buenas prácticas de Gestión de proyectos proporcionadas por el PMI (*Project Management Institute*) en PMBoK (Cuerpo de conocimiento de la Gerencia de proyectos, o en Inglés, *Project Management Body of Knowledge*).

Se presentará la construcción de un marco de trabajo metodológico para el entorno académico, donde se han considerado los 4 pilares que están impactando y orientando la industria de software nacional y constituyen una tendencia mundial:

- **Proceso Unificado de Desarrollo de Software:** El cual no es sólo una metodología de desarrollo de software adaptable y comprensible, basada en el desarrollo iterativo e incremental, sino que constituye una base de conocimiento de las mejores prácticas de la industria, realizado por los expertos de IBM, que se mantiene en constante evolución y mantenimiento.

- ***Capability Maturity Model Integration* DEV v.1.3:** Constituye un modelo de referencia para la adopción de buenas prácticas de Ingeniería de Software en el área de construcción de software.

- ***Project Management Body of Knowledge*:** El cual es el compendio de las mejores prácticas para la gestión de cualquier tipo de proyectos y cuenta con bastante aceptación por parte de la industria del software.

- ***Metodologías ágiles:*** Basados en el manifiesto ágil se han desarrollado, mejorado, y adaptado metodologías de desarrollo de software que permiten a los equipos construir software de alta calidad con una inversión de esfuerzo

bajo pero con una mejora la adopción de mejores prácticas de ingeniería de software.

Con estos pilares se desarrolló una metodología adaptable a los diferentes tipos de proyectos a los que se ven avocados los estudiantes en su ciclo de formación, y que cuente con las mejores prácticas de la industria. A continuación se profundizará en los conceptos básicos de cada uno de ellos.

7.1 Proceso Unificado de Rational

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 RUP (roles, entregables, flujos de procesos, etc.), de acuerdo con las necesidades, y desplegar un sitio web el cual pueda ser navegable y asequible por todos en la organización.

7.1.1 Principios de desarrollo

El RUP está basado en 6 principios clave, que son:

- **Adaptar el proceso:** El proceso deberá adaptarse a las características propias del proyecto u organización.

- **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.

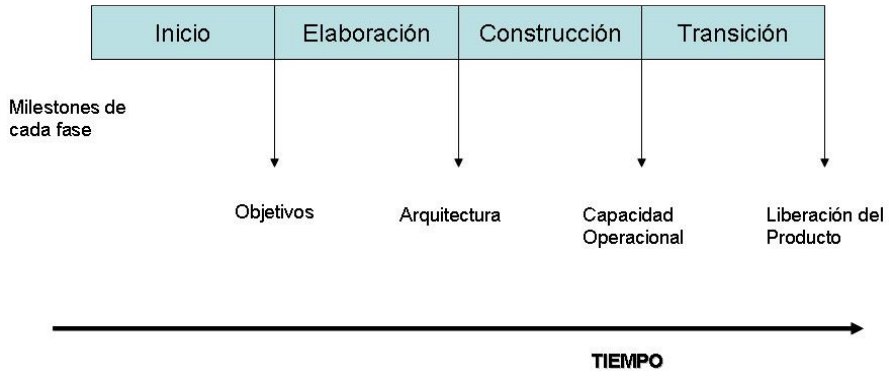
- **Elevar el nivel de abstracción:** Este principio dominante motiva el uso de conceptos reutilizables, tales como patrones de software, lenguajes 4GL o marcos de referencia (frameworks), por nombrar algunos. Esto evita que los Ingenieros de Software vayan directamente de los requisitos a la codificación 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, implicando que el aseguramiento de la calidad forme parte del proceso de desarrollo y no de un grupo independiente al proyecto.

7.1.2 Ciclo de vida

En la metodología RUP, el ciclo de vida de un proyecto se ve gobernado por cuatro fases: Inicio, Elaboración, Construcción y Transición; como se indica en el siguiente esquema (Ver Figura 2: Fases del ciclo de vida del Proceso de Desarrollo de Software de RUP):

Figura 2: Fases del ciclo de vida del Proceso de Desarrollo de Software de RUP[31]



El ciclo de vida del proceso de desarrollo de software desde la perspectiva de Administración del Proyecto se descompone en el tiempo en cuatro fases, que concluyen cada una con un hito. La finalización de cada una de las fases permite

[31] IBM. IBM Rational Unified Process (RUP) para proyectos pequeños [en línea]. (Citada: 16 de Junio 2012) <http://cgrw01.cgr.go.cr/rup/RUP.es/LargeProjects/index.htm#core.base_rup/guidances/supportingmaterials/welcome_2BC5187F.html>

valorar y aprobar el cumplimiento de los objetivos de dicha fase para continuar con la siguiente.

Estas fases se convierten en los indicadores del progreso del proyecto; las fases se detallan a continuación

Fase de Concepción.

Objetivos:

- Especificar la visión final del producto y sus casos de negocios.
- Afinar el alcance del proyecto
- Establecer los servicios de negocio a desarrollar

Fase de Elaboración.

Objetivos:

- Definir la arquitectura del sistema
- Definir y especificar los servicios
- Definir la interfaz gráfica del usuario

Fase de Construcción.

Objetivos:

- Construir el producto, evolucionando la visión del negocio, la arquitectura y los planes hasta que el producto – toda la visión – esté listo para transferencia a la comunidad de usuarios.
- Desarrollar los servicios, casos de uso (funcionalidades), módulos, subsistemas del proyecto.
- Realizar desarrollo por iteraciones, priorizando los servicios, casos de uso, módulos y/o subsistemas más críticos para el negocio.
- Realizar pruebas funcionales de cada uno de los servicios, casos de uso, módulos y/o subsistemas, a medida que se vayan desarrollando.

Fase de Transición.

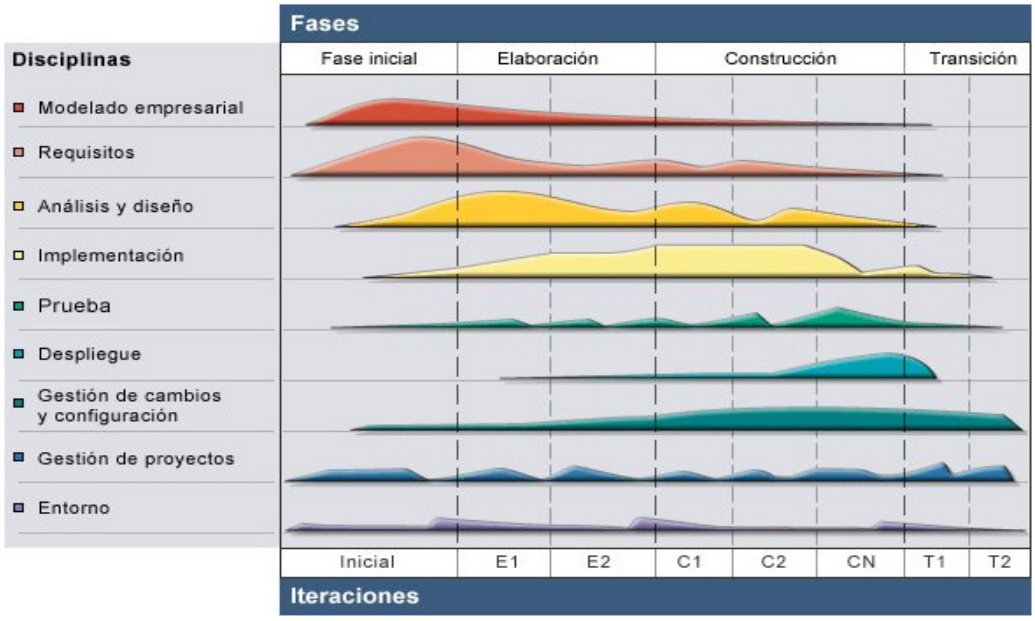
Objetivos:

- Lograr la transición producto software a la comunidad de usuarios la cual incluye: manufactura, entrega, entrenamiento, soporte, mantenimiento del producto.

7.1.3 Disciplinas del ciclo de vida del software

Para llevar a cabo exitosamente cada una de las iteraciones definidas en las fases, se aplican las disciplinas del ciclo de vida del software:

Figura 3: Proceso Unificado de Desarrollo de Software de IBM RATIONAL[32]



[32] IBM. IBM Rational Unified Process (RUP) para proyectos pequeños [en línea]. (Citada: 16 de Junio. 2012) <http://cgrw01.cgr.go.cr/rup/RUP.es/LargeProjects/index.htm#core.base_rup/guidances/supportingmaterials/welcome_2BC5187F.html>

A continuación se explica cómo serán enfocadas las disciplinas que serán usadas en el proyecto y su alcance.

Modelado de negocio

El propósito de esta disciplina es entender el negocio, principalmente la parte de la organización (procesos, roles, cargos) que interactuarán y/o serán afectados por la presencia del futuro sistema.

Su alcance es:

- Identificar los problemas y posibilidades de mejora en la organización que contará con el sistema.
- Evaluar el impacto del cambio organizacional.
- Asegurar que los clientes, usuarios, desarrolladores, y otras partes tienen una comprensión común de la organización.
- Obtener los requisitos del sistema de software necesarios para apoyar la organización en la que estará inmerso el sistema.
- Entender cómo encaja en la organización el sistema de software a ser desplegado.

Gestión de requisitos.

El propósito de la disciplina de Gestión de Requisitos es entender el problema actual en el proyecto e identificar mejoras potenciales, evaluar el impacto de los cambios en la organización, asegurando que todos los interesados en el proyecto manejen un lenguaje unificado para listar los Requisitos. Adicionalmente, establece entre todos los interesados un acuerdo frente a lo que el sistema pueda hacer y a su vez define el alcance total del sistema.

Su alcance es:

- Establecer y mantener un acuerdo sobre cuál es la funcionalidad esperada.
- Entregar a los desarrolladores una descripción clara de los requisitos del sistema.
- Definir los límites del sistema.
- Definir el "*Look and Feel*" de la aplicación.
- Proveer la base para la creación del plan de iteraciones, la estimación de costos y esfuerzos.

Análisis y diseño.

El propósito de la disciplina de Análisis y Diseño es transformar los requisitos de negocio en especificaciones de software, detallar y probar la arquitectura de la solución.

Su alcance es:

- Transformar los requisitos en el diseño del sistema a implementar.
- Definir una arquitectura robusta para el sistema.
- Entregar a los desarrolladores una especificación clara de los requisitos del sistema.
- Adaptar el diseño al ambiente de implementación.

Implementación.

El propósito de la disciplina de Implementación es codificar o configurar los componentes de software que conforman la solución para ensamblarlos y desplegarlos en unidades totalmente funcionales.

Su alcance es:

- Implementar la solución en términos de creación, codificación y personalización de componentes.

- Definir la organización de componentes como subsistemas o módulos.
- Probar los componentes como unidades aisladas.
- Liberar módulos totalmente funcionales.

Pruebas.

El propósito de la disciplina de Pruebas es evaluar y determinar la calidad del producto.

Su alcance es:

- Buscar y documentar defectos encontrados en el software.
- Determinar la calidad del software.
- Validar la especificación de los requisitos frente al sistema liberado.
- Validar el Diseño y Arquitectura del producto de software.
- Validar la implementación de los requisitos.
- Asegurar que todos los defectos fueron corregidos antes de desplegar el producto.
- Asegurar la correcta ejecución del proceso de desarrollo.

Despliegue.

El propósito de la disciplina de Despliegue es asegurar que el producto quede disponible para su uso por parte de los usuarios finales del proyecto.

Su alcance es:

- Especificar la instalación del producto.
- Probar la instalación del producto.
- Crear el material para consulta y capacitación del usuario.
- Validar el diseño del producto de software.
- Asegurar la implementación de los requisitos suplementarios (requisitos no funcionales).

- Entregar el producto funcionando al usuario.

Gestión de la configuración y el cambio (disciplina de apoyo).

El propósito de la disciplina Gestión de la configuración y el cambio es la definición y conducción de los artefactos que deben ser controlados, dado su alto nivel de interacción dentro del proyecto.

Su alcance es:

- Controlar los cambios a los requisitos.
- Controlar los cambios a los artefactos.
- Mantener la integridad de los artefactos.
- Versionar los artefactos.
- Permitir conocer el estado de un artefacto.

Gestión del Proyecto.

El propósito de la disciplina Gestión del Proyecto es proveer un marco de planeación, ejecución, coordinación y control de todas las actividades que hacen parte del proyecto, además de unificar los canales de comunicación dentro del proyecto.

Su alcance es:

- Proveer un framework para proyectos de implementación y desarrollo de software.
- Proveer prácticas para la planeación, ejecución y monitoreo de proyectos.
- Proveer un framework para el manejo de riesgos.

Gestión del Ambiente.

El propósito de la disciplina de Gestión del Ambiente es proveer el soporte necesario para la implementación, describiendo los ambientes, las herramientas y la logística para facilitar la ejecución del proyecto.

Su alcance es:

- Proveer la infraestructura (hardware, software y comunicaciones) para la implementación.
- Proveer las actividades, los artefactos y los lineamientos necesarios para el soporte del proceso.

7.1.4 Elementos de RUP en el framework propuesto

Los elementos de RUP en el framework propuesto se encuentran presentados en el capítulo 8.4.1 Principios adoptados .

7.2 Capability Maturity Model Integration (CMMi)

CMMI (Capability Maturity Model Integration) es un enfoque de mejora de procesos que proporciona a las organizaciones los elementos esenciales de procesos efectivos, lo que mejorará su rendimiento en tres diferentes aspectos: Servicios, Adquisiciones y Desarrollo de software.

Los modelos de CMMI son colecciones de las mejores prácticas que ayudan a las organizaciones para mejorar drásticamente la eficacia, eficiencia y calidad.

Las mejores prácticas CMMI se publican en los documentos llamados modelos. En la actualidad hay tres áreas de interés cubiertas por los modelos de CMMI: Desarrollo, Adquisición y Servicios.

La versión actual de CMMI es la versión 1.3. Hay tres constelaciones de la versión 1.3 disponible:

- CMMI para el Desarrollo (CMMI–DEV o CMMI for Development), En él se tratan procesos de desarrollo de productos y servicios.
- CMMI para la Adquisición (CMMI–ACQ o CMMI for Acquisition). En él se tratan la gestión de la cadena de suministro, adquisición y contratación externa en los procesos del gobierno y la industria.
- CMMI para Servicios (CMMI–SVC o CMMI for Services), está diseñado para cubrir todas las actividades que requieren gestionar, establecer y entregar Servicios.

Dentro de la constelación CMMI–DEV, existen dos modelos:

- CMMI–DEV
- CMMI–DEV + IPPD (Integrated Product and Process Development)

Independientemente de la constelación\modelo que opta una organización, las prácticas CMMI deben adaptarse a cada organización en función de sus objetivos de negocio.

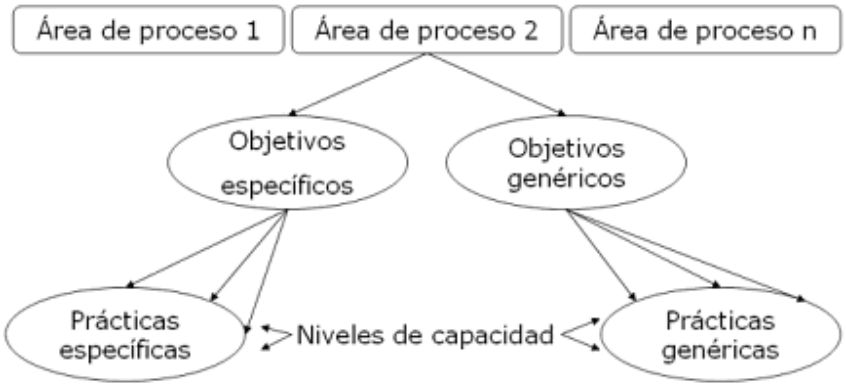
7.2.1 Generalidades

CMMI es la evolución de CMM. CMM Fue desarrollado desde 1987 hasta 1997. En 2002 se lanzó CMMI Version 1.1, luego en agosto de 2006 siguió la versión 1.2 y en el 2011 la versión 1.3. El objetivo del proyecto CMMI es mejorar la usabilidad de modelos de madurez, integrando varios modelos diferentes en un solo marco (framework). Fue creado por miembros de la industria, el gobierno y el SEI (Software Engineering Institute). Entre los principales patrocinadores se incluyen la Oficina del Secretario de Defensa (OSD) y la National Defense Industrial Association.

Se desarrolla sobre el principio de calidad de Jurán: "la calidad del resultado depende principalmente de la calidad de los procesos empleados en su desarrollo"[33].

El Modelo tiene dos representaciones: Representación Continua (Continuous Representation) y Escalonada (Staged Representation):

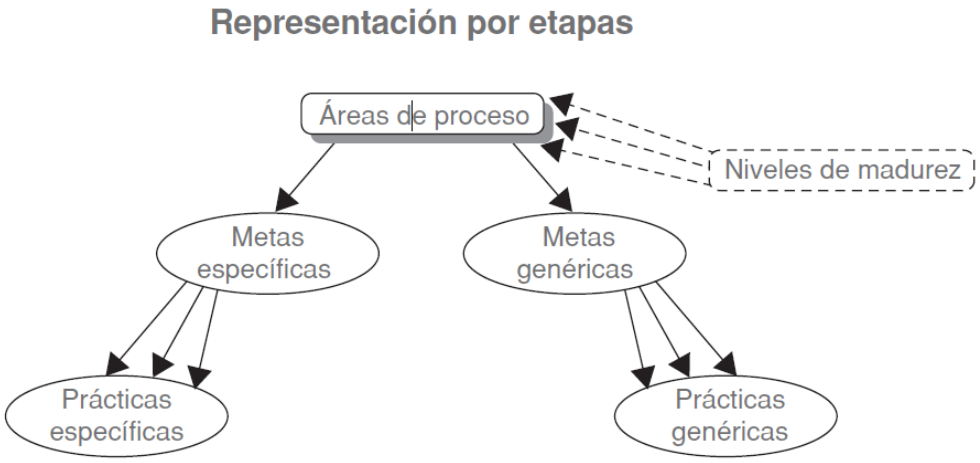
Figura 4. Representación continua del modelo CMMI [34]



[33] SCRUM MANAGER. Glosario [en línea]. (Citada: 5 de Noviembre de 2012) < http://www.scrummanager.net/oks/mod/glossary/view.php?id=158 >

[34] SOFTWARE ENGINEERING INSTITUTE. CMMI® for Development, Version 1.3 CMMI-DEV, V1.3, (November). [en línea]. (Citada: 16 de Junio. 2012) < http://www.sei.cmu.edu/reports/10tr033.pdf >

Figura 5. Representación escalonada del modelo CMMI [35]



Cada organización puede optar por seleccionar el esquema que se adapte a sus características y prioridades de mejora. Existe una "equivalent stagging" el cual establece que un Nivel de Madurez equivale a tener un conjunto de PA´s (Áreas de Proceso – Process Area) un determinado para un Nivel de Capacidad asociado.

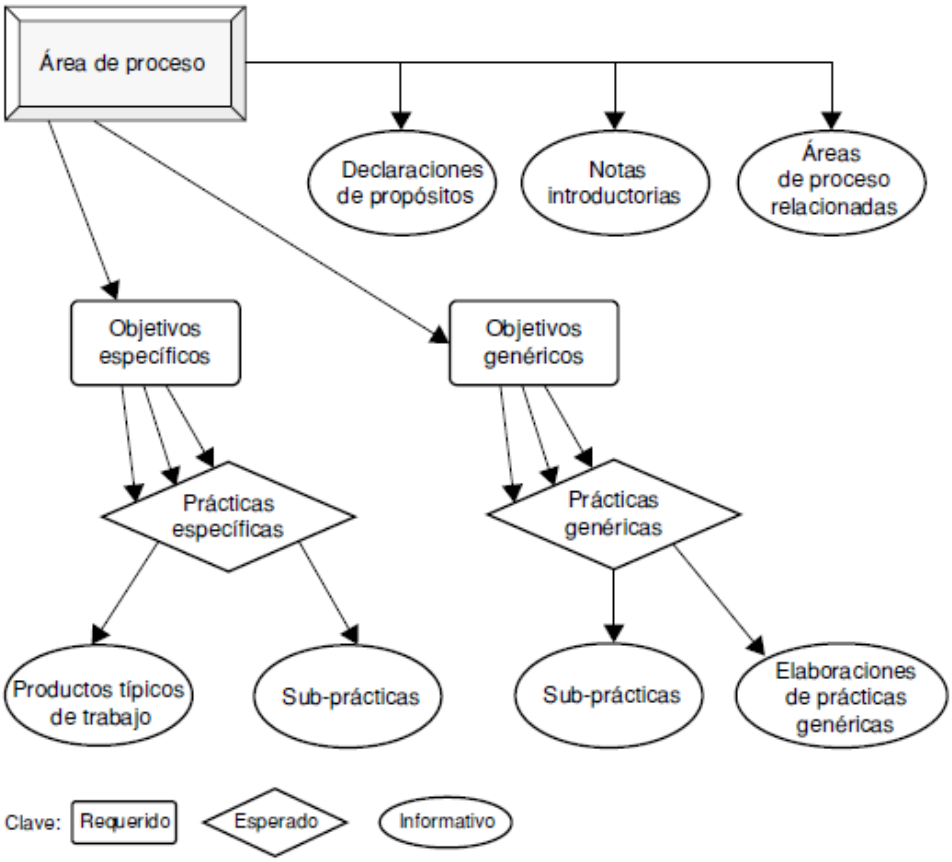
La visión continua proporciona la representación de nivel de capacidad de cada una de las Áreas de Proceso del modelo.

La visión escalonada definirá a la organización dándole en su conjunto un nivel de madurez del 1 al 5.

[35] Ibid.

7.2.2 Componentes del modelo

Figura 6. Componentes del Modelo CMMI[36]



Área de proceso (Process Area – PA): Conjunto de prácticas relacionadas que son ejecutadas de forma conjunta para conseguir un conjunto de objetivos.

En CMMI las Áreas de Proceso son el pilar del modelo y constituyen el elemento clave para definir la madurez de una organización de software. Estas constituyen un grupo de prácticas relacionadas con un área de trabajo del modelo en donde se definen específicamente los “deber ser” de cada una de esas PA´s.

[36] SOFTWARE ENGINEERING INSTITUTE. CMMI® for Development, Version 1.3 CMMI-DEV, V1.3, (November). [en línea]. (Citada: 16 de Junio. 2012) < http://www.sei.cmu.edu/reports/10tr033.pdf >

El modelo CMMI v1.3 (CMMI–DEV) contiene las siguientes 22 Áreas de Proceso:

- Análisis de Causas y Resolución (CAR)
- Gestión de la Configuración (CM)
- Análisis de Decisiones y Resolución (DAR)
- Gestión Integrada de Proyectos (IPM)
- Medición y Análisis (MA)
- Innovación y Despliegue Organizacionales(OID)
- Definición de Procesos Organizacionales (OPD)
- Enfoque Organizacional del Proceso (OPF)
- Rendimiento del Proceso Organizacional (OPP)
- Entrenamiento Organizacional (OT)
- Monitoreo y Control de Proyectos (PMC)
- Planificación de Proyectos (PP)
- Aseguramiento de Calidad de Procesos y Productos (PPQA)
- Integración de Producto (PI)
- Gestión Cuantitativa de Proyectos (QPM)
- Gestión de Requisitos (REQM)
- Desarrollo de Requisitos (RD)
- Gestión de Riesgos (RSKM)
- Gestión de Acuerdos con Proveedores (SAM)
- Solución Técnica (TS)
- Validación (VAL)
- Verificación (VER)

Componentes Requeridos

- **Objetivo genérico:** Los objetivos genéricos asociados a un nivel de capacidad establecen lo que una organización debe alcanzar en ese nivel de capacidad.

- El logro de cada uno de esos objetivos en un Área de Proceso significa mejorar el control en la ejecución del Área de Proceso.
- **Objetivo específico:** Los objetivos específicos se aplican a una única área de proceso y localizan las particularidades que describen qué se debe implementar para satisfacer el propósito del Área de Proceso.

Componentes Esperados

- **Práctica genérica:** Una práctica genérica se aplica a cualquier Área de Proceso porque puede mejorar el funcionamiento y el control de cualquier proceso.

- **Práctica específica:** Una práctica específica es una actividad que se considera importante en la realización del objetivo específico al cual está asociado.

  Las prácticas específicas describen las actividades esperadas para lograr la meta específica de un Área de Proceso.

Componentes Informativos

- Propósito
- Notas introductorias
- Nombres
- Tablas de relaciones práctica – objetivo
- Prácticas
- Productos típicos
- Sub–prácticas: Una sub–práctica es una descripción detallada que sirve como guía para la interpretación de una práctica genérica o específica.

- Ampliaciones de disciplina: Las ampliaciones contienen información relevante de una disciplina particular y relacionada con una práctica específica.
- Elaboraciones de prácticas genéricas: Una elaboración de una práctica genérica es una guía de cómo la práctica genérica debe aplicarse al Área de Proceso.

7.2.3 Capacidad y Madurez

La capacidad del proceso es la habilidad inherente de un proceso para producir los resultados planeados. El principal objetivo de un proceso de software maduro es el de producir productos de calidad que cumplan los requisitos del usuario. Cuando se habla de madurez del proceso se entiende como el crecimiento alcanzado en la capacidad del proceso de software, y se considera como una actividad de largo plazo.

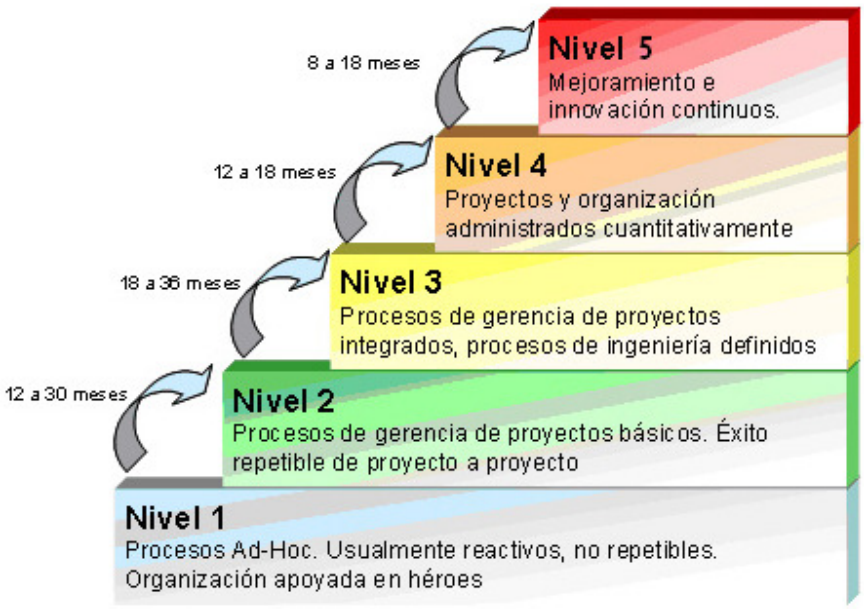
En una compañía de software inmadura, el proceso de software es generalmente improvisado, no existen planes rigurosos, sus actividades se enfocan en resolver las crisis que se presentan, carecen de bases objetivas para evaluar la calidad de los productos o para resolver los problemas que se presentan. De manera contraria, cuando una organización alcanza cierto grado de madurez, posee una gran habilidad para administrar el proceso de desarrollo y mantenimiento de software, se hacen pruebas y análisis costo–beneficio para mejorar el proceso, el director supervisa la calidad del producto y la satisfacción del cliente, se llevan registros, y todos las personas están involucradas en el proceso de desarrollo.

7.2.4 Niveles de Madurez (representación escalonada)

El modelo CMMI consta de cinco niveles (representación escalonada), diseñados de manera que los niveles inferiores proveen las bases para el siguiente nivel,

logrando plasmar un avance progresivo hacia los siguientes niveles. Estas cinco etapas de desarrollo son conocidas como niveles de madurez (ver
Figura 7. Niveles de Madurez), y cada uno la organización alcanza una capacidad superior del proceso. CMMI plantea que una organización puede ubicarse en alguno de los cinco posibles niveles de madurez, dependiendo del grado de perfección de sus procesos.

Figura 7. Niveles de Madurez[37]



- **Nivel de madurez – Inicial: Nivel 1.** Las organizaciones en este nivel no disponen de un proceso estable para el desarrollo y mantenimiento de software. Aunque se utilicen técnicas correctas de ingeniería, los esfuerzos se ven consumidos por falta de planificación y se dependen de esfuerzos heroicos y personales para lograr el éxito en un proyecto. A menudo se presentan fracasos y casi siempre retrasos y sobrecostos. El resultado de

[37] LIDER DE PROYECTO. CMMI [en línea]. (Citada: 16 de Junio. 2012) < http://www.liderdeproyecto.com/manual/cmmi_y_la_administracion_de_proyectos.html>

los proyectos es impredecible. Este nivel no tiene asociadas Áreas de Proceso.

- **Nivel de madurez – Administrado: Nivel 2** (conocido también como **proyecto administrado**). En este nivel las organizaciones disponen de unas prácticas institucionalizadas de gestión de proyectos, existen unas métricas básicas y un razonable seguimiento de la calidad. La relación con subcontratistas y clientes está gestionada sistemáticamente. Las áreas de procesos asociadas con este nivel, son:
    - Medición y Análisis (MA)
    - Monitoreo y Control de Proyectos (PMC)
    - Planificación de Proyectos (PP)
    - Aseguramiento de Calidad de Procesos y Productos (PPQA)
    - Gestión de Requisitos (REQM)
    - Gestión de Acuerdos con Proveedores (SAM)
    - Gestión de la Configuración (CM)

- **Nivel de madurez – Definido: Nivel 3** (conocido también como **proceso definido**). El proceso de software se encuentra documentado, estandarizado e integrado en un proceso de software estándar dentro de la organización, para las actividades administrativas y técnicas. El grupo que trabaja en el proceso enfoca y guía sus esfuerzos al mejoramiento del proceso, facilita la introducción de técnicas y métodos, e informa a la gerencia el estado del proceso. La capacidad del proceso está basada en una amplia compresión, dentro de la organización, de las actividades, roles, y responsabilidades definidas en el proceso de software. Las Áreas de Procesos asociadas con este nivel, son:
    - Gestión Integrada de Proyectos (IPM)
    - Definición de Procesos Organizacionales (OPD)

- Enfoque Organizacional del Proceso (OPF)
- Entrenamiento Organizacional (OT)
- Integración de Producto (PI)
- Desarrollo de Requisitos (RD)
- Gestión de Riesgos (RSKM)
- Solución Técnica (TS)
- Validación (VAL)
- Verificación (VER)
- Análisis de Decisiones y Resolución (DAR)

- **Nivel de madurez – Administrado Cuantitativamente: Nivel 4**. Se caracteriza porque las organizaciones disponen de un conjunto de métricas significativas de calidad y productividad, que se usan de modo sistemático para la toma de decisiones y la gestión de riesgos. El software resultante es de alta calidad. Las Áreas de Procesos asociadas con este nivel, son:
  - Rendimiento del Proceso Organizacional (OPP)
  - Gestión Cuantitativa de Proyectos (QPM)

- **Nivel de madurez – Optimizando: Nivel 5**. La organización completa está volcada en la mejora continua de los procesos. Se hace uso intensivo de las métricas y se gestiona el proceso de innovación. Las Áreas de Procesos asociadas con este nivel, son:
  - Innovación y Despliegue Organizacionales(OID)
  - Análisis de Causas y Resolución (CAR)

Cada nivel de madurez, con excepción del inicial, se caracteriza por un conjunto de Áreas de Proceso que agrupan buenas prácticas, las cuales, al ser ejecutadas colectivamente, permiten cumplir con algún objetivo que es considerado importante para el modelo. En la Tabla 1. Niveles de madurez, Áreas de Proceso y Categorías de Áreas de Proceso, se ilustra las Áreas de Proceso en cada nivel de madurez, correspondientes a una categoría determinada, y se puede observar que antes de introducir prácticas de un nivel determinado, deben estabilizarse las prácticas del nivel anterior.

Tabla 1. Niveles de madurez, Áreas de Proceso y Categorías de Áreas de Proceso[38]

Nivel	Administración de Procesos	Gestión de Proyectos	Ingeniería	Soporte
5	Innovación y Despliegue Organizacional (OID)			Análisis de Causas y Resolución (CAR)
4	Rendimiento del Proceso Organizacional (OPP)	Gestión Cuantitativa de Proyectos (QPM)		
3	Enfoque Organizacional del Proceso (OPF) Definición de Procesos Organizacionales (OPD) Entrenamiento Organizacional (OT)	Gestión Integrada de Proyectos (IPM) Gestión de Riesgos (RSKM)	Desarrollo de Requisitos (RD) Solución Técnica (TS) Verificación (VER) Validación (VAL) Integración de Producto (PI)	Análisis de Decisiones y Resolución (DAR)
2		Planificación de Proyectos (PP) Monitoreo y Control de Proyectos (PMC) Gestión de Acuerdos con Proveedores (SAM)	Gestión de Requisitos (REQM)	Gestión de la Configuración (CM) Aseguramiento de Calidad de Procesos y Productos (PPQA) Medición y Análisis (MA)
1	NA	NA	NA	NA

[38] Elaboración propia

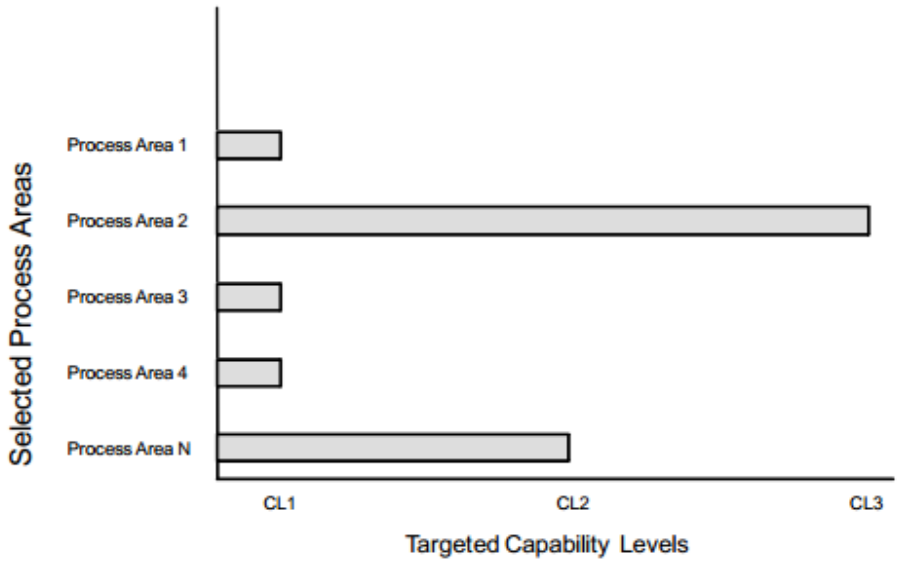
Además de presentar las Áreas de Proceso por nivel de madurez, el modelo propone una vista alternativa (representación continua), en donde las PA´s están agrupadas por categoría, estas categorías reciben el nombre de niveles de capacidad (ver Tabla 1. Niveles de madurez, Áreas de Proceso y Categorías de Áreas de Proceso).

7.2.5 Niveles de Capacidad de los procesos (Representación Continua)

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

- 0.– Incompleto: El proceso no se realiza, o no se consiguen sus objetivos.
- 1.– Ejecutado: El proceso se ejecuta y se logra su objetivo.
- 2.– Gestionado: Además de ejecutarse, el proceso se planifica, se revisa y se evalúa para comprobar que cumple los requisitos.
- 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.
- 4.– Cuantitativamente gestionado: Además de ser un proceso definido, se controla utilizando técnicas cuantitativas.
- 5.– Optimizando: Además de ser un proceso cuantitativamente gestionado, de forma sistemática se revisa y modifica o cambia para adaptarlo a los objetivos del negocio. Mejora continua.

Figura 8. Áreas de Proceso en la Representación Continua y nivel de capacidad (CL: Capability Level).[39]



Por lo tanto, una organización en vez de decidirse por mejorar su capacidad por niveles de madurez en donde tiene que realizar un esfuerzo general por áreas por varias Áreas de Proceso al tiempo, puede elegir Áreas de Proceso en específico y mejorar por nivel de capacidad (CL: Capability Level) según su interés. En consecuencia, una organización orientada interesada en mejorar su Gestión de Proyectos, probablemente elija llevar las PA´s de PP y PMC a un CL3 o superior.

7.2.6 Elementos de CMMi en el framework propuesto

Los elementos de CMMi en el framework propuesto se encuentran presentados en el capítulo 8.4.1 Principios adoptados .

---

[39] SOFTWARE ENGINEERING INSTITUTE. CMMI for development, version 1.3 [en línea]. (Citada: 16 de Junio. 2012) <http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm>

7.3 Project Management Body Of Knowledge (PMBoK)

La Guía del PMBOK® es un estándar en la Administración de proyectos, desarrollado por el *Project Management Institute* (PMI). La misma comprende dos grandes secciones; la primera sobre los procesos y contextos de un proyecto, la segunda sobre las áreas de conocimiento específico para la gestión de un proyecto.

En 1987, el PMI publicó la primera edición del PMBOK® en un intento por documentar y estandarizar información y prácticas generalmente aceptadas en la Gestión de proyectos. Actualmente se encuentra en la cuarta edición, la cual provee de referencias básicas a cualquiera que esté interesado en la gestión de proyectos. Posee un léxico común y una estructura consistente para el campo de la gestión de proyectos.

Consiste en 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, entre otros.

El 'PMBOK' reconoce cinco Grupos de procesos básicos:

- Inicio
- Planeación
- Ejecución
- Monitoreo y control
- Cierre

Estos grupos de procesos se encuentran presentes en el desarrollo de un proyecto o fase del mismo.

Adicionalmente, cuenta con nueve Áreas de Conocimiento comunes a casi todos los proyectos:

- Gestión de la Integración
- Gestión del Alcance
- Gestión del Tiempo
- Gestión de la Calidad
- Gestión de Costos
- Gestión del Riesgo
- Gestión de Recursos Humanos
- Gestión de la Comunicación
- Gestión de las Compras y Adquisiciones

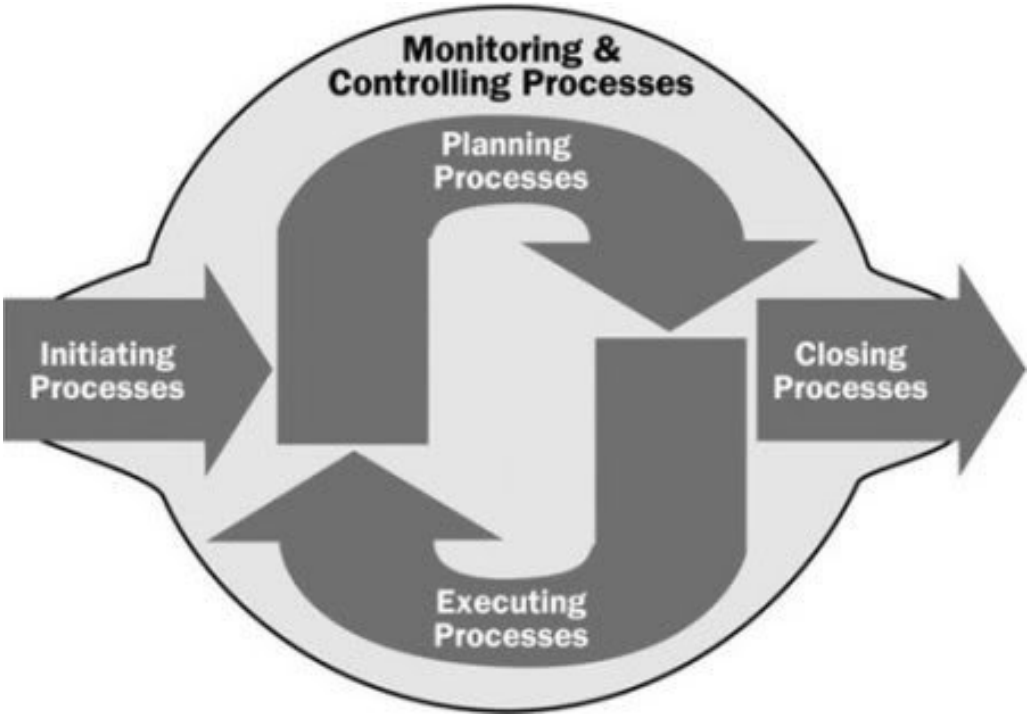
Los procesos se traslapan e interactúan a través de un proyecto o fase y son descritos en términos de:

- Entradas (documentos, planes, diseños, etc.)
- Herramientas y Técnicas (mecanismos aplicados a las entradas)
- Salidas (documentos, productos, etc.).

A continuación se explican los Grupos de proceso y las Áreas de conocimiento.

7.3.1 Grupos de procesos de la Gerencia de proyectos

Figura 9. Grupos de procesos de la Gerencia de proyectos[40]



1. Iniciación:

Define el proyecto en sus aspectos principales, como tiempo, costo, alcance, riesgos, recursos, y autoriza el proyecto o una fase del mismo.

2. Planificación:

Define, refina los objetivos y planifica el curso de acción requerido para lograr los objetivos y el alcance pretendido del proyecto. Se realiza la manufactura de los diferentes planes que servirán para administrar el proyecto. Está formado por veinte procesos.

[40] PROJECT MANAGEMENT INSTITUTE. PMBoK (Project Management Body of Knowledge), version 4 [en línea]. (Citada: 16 de Junio. 2012) < http://www.pmi.org/PMBOK–Guide–and–Standards.aspx>

3. Ejecución:

Compuesto por aquellos procesos realizados para completar el trabajo definido en el plan (plan de planes) 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.

7.3.2 Áreas de Conocimiento.[41]

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 Gestión de proyectos dentro de los Grupos de procesos de Gestión de proyectos. La integración incluye características de unificación, consolidación, articulación, así como las acciones

[41] PROJECT MANAGEMENT INSTITUTE. PMBoK (Project Management Body of Knowledge), version 4 [en línea]. (Citada: 16 de Junio. 2012) <http://www.pmi.org/PMBOK–Guide–and–Standards.aspx>

integradoras que son cruciales para la terminación del proyecto, la gestión exitosa de las expectativas de los interesados y el cumplimiento de los requisitos.

Los procesos de Gestión de la Integración del Proyecto, incluyen:

- **Desarrollar el Acta de Constitución del Proyecto**: Desarrollar el documento que autoriza formalmente un proyecto o una fase y documentar los requisitos iniciales que satisfacen las necesidades y expectativas de los interesados.
- **Desarrollar el Plan para la Dirección del Proyecto:** Consiste en documentar las acciones necesarias para definir, preparar, integrar y coordinar todos los planes subsidiarios. El plan del proyecto es un plan de planes.
- **Dirigir y Gestionar la Ejecución del Proyecto**: Es el proceso que consiste en ejecutar el trabajo definido en el plan para la Gestión del proyecto para cumplir con los objetivos del mismo.
- **Monitorear y Controlar el Trabajo del Proyecto:** Es el proceso que consiste en monitorear, revisar y regular el avance a fin de cumplir con los objetivos de desempeño definidos en el plan para la Gestión del proyecto.
- **Realizar el Control Integrado de Cambios:** Es el proceso que consiste en revisar todas las solicitudes de cambios, aprobar los cambios y gestionar los cambios a los entregables, a los activos de los procesos de la organización, a los documentos del proyecto y al plan para la Gestión del proyecto.
- **Cerrar Proyecto o Fase:** Es el proceso que consiste en finalizar todas las actividades a través de todos los Grupos de procesos de Gestión de proyectos para completar formalmente el proyecto o una fase del mismo.

2. Gestión del Alcance del Proyecto:

Incluye los procesos necesarios para garantizar que el proyecto incluya todo (y únicamente todo) el trabajo requerido para completarlo con éxito. El objetivo principal de la Gestión del Alcance del Proyecto es definir y controlar qué se

incluye y qué no se incluye en el proyecto. Los procesos de Gestión del Alcance del Proyecto incluyen:

- **Recopilar Requisitos**: Es el proceso que consiste en definir y documentar las necesidades de los interesados a fin de cumplir con los objetivos del proyecto.
- **Definir el Alcance:** Es el proceso que consiste en desarrollar una descripción detallada del proyecto y del producto.
- **Crear la EDT**: Es el proceso que consiste en subdividir los entregables y el trabajo del proyecto en componentes más pequeños y más fáciles de manejar.
- **Verificar el Alcance**: Es el proceso que consiste en formalizar la aceptación de los entregables del proyecto que se han completado.
- **Controlar el Alcance:** Es el proceso que consiste en monitorear el estado del alcance del proyecto y del producto, y en gestionar cambios a la línea base del alcance.

3. Gestión del Tiempo del Proyecto:

Incluye los procesos requeridos para administrar la finalización del proyecto a tiempo. Los procesos de Gestión del Tiempo del Proyecto incluyen:

- **Definir las Actividades:** Consiste en identificar las acciones específicas a ser realizadas para elaborar los entregables del proyecto.
- **Secuenciar las Actividades:** Es el proceso que consiste en identificar y documentar las interrelaciones entre las actividades del proyecto apoyándose en diagramas de red u otras herramientas.
- **Estimar los Recursos de las Actividades:** Consiste en estimar el tipo y las cantidades de recursos, tanto humanos como físicos para ejecutar cada actividad.
- **Estimar la Duración de las Actividades**: Es el proceso que consiste en establecer aproximadamente la cantidad de períodos de trabajo necesarios para finalizar cada actividad con los recursos estimados.

- **Desarrollar el Cronograma:** Consiste en analizar el orden de las actividades, su duración, los requisitos de recursos y las restricciones del cronograma para crear el cronograma del proyecto.
- **Controlar el Cronograma**: Es el proceso por el que se da seguimiento al estado del proyecto para actualizar el avance del mismo y gestionar cambios a la línea base del cronograma.

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. Los procesos de Gestión de los Costos del Proyecto incluyen:

- **Estimar los Costos:** Es el proceso que consiste en desarrollar una aproximación de los recursos financieros necesarios para completar las actividades del proyecto.
- **Determinar el Presupuesto:** Es el proceso que consiste en sumar los costos estimados de actividades individuales o paquetes de trabajo para establecer una línea base de costo autorizada.
- **Controlar los Costos**: Es el proceso que consiste en monitorear la situación del proyecto para actualizar el presupuesto del mismo y gestionar cambios a la línea base de costo.

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. Implementa el sistema de gestión de calidad por medio de políticas y procedimientos, con actividades de mejora continua de los procesos llevados a cabo durante todo el proyecto, según corresponda. Los procesos de Gestión de la Calidad del Proyecto incluyen:

- **Planificar la Calidad**: Es el proceso por el cual se identifican los requisitos de calidad y/o normas para el proyecto y el producto, documentando la manera como el proyecto demostrará el cumplimiento con los mismos.
- **Realizar el Aseguramiento de Calidad**: Es el proceso que consiste en auditar los requisitos de calidad y los resultados de las medidas de Control de calidad, para asegurar que se utilicen los estándares, procedimientos, las normas de calidad apropiadas y las definiciones operacionales. En este proceso se realizan las auditorias al proceso y su forma de ejecución
- **Realizar el Control de Calidad**: Es el proceso por el cual se monitorean y registran los resultados de la ejecución de actividades de Control de calidad, a fin de evaluar el desempeño y recomendar cambios necesarios. En este proceso se evalúa el producto.

6. Gestión de los Recursos Humanos del Proyecto:

La Gestión de los Recursos Humanos del Proyecto incluye los procesos que organizan, gestionan y conducen el equipo del proyecto. El equipo del proyecto está conformado por aquellas personas a las que se les han asignado roles y responsabilidades para completar el proyecto. Los procesos de Gestión de los Recursos Humanos del Proyecto incluyen:

- **Desarrollar el Plan de Recursos Humanos:** Es el proceso por el cual se identifican y documentan los roles dentro de un proyecto, las responsabilidades, las habilidades requeridas y las relaciones de comunicación, y se crea el plan para la dirección de personal.
- **Adquirir el Equipo del Proyecto:** Es el proceso por el cual se confirman los recursos humanos disponibles y se forma el equipo necesario para completar las asignaciones del proyecto.

- **Desarrollar el Equipo del Proyecto**: Es el proceso que consiste en mejorar las competencias, la interacción de los miembros del equipo y el ambiente general del equipo para lograr un mejor desempeño del proyecto.
- **Dirigir el Equipo del Proyecto:** Es el proceso que consiste en dar seguimiento al desempeño de los miembros del equipo, proporcionar retroalimentación, resolver problemas y gestionar cambios a fin de optimizar el desempeño 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*). Los procesos de Gestión de los Recursos Humanos del Proyecto incluyen:

- **Identificar a los Interesados:** Es el proceso que consiste en identificar a todas las personas u organizaciones que reciben el impacto del proyecto, y en documentar información relevante relativa a sus intereses, participación e impacto en el éxito del proyecto.
- **Planificar las Comunicaciones:** Es el proceso para determinar las necesidades de información de los interesados en el proyecto y para definir cómo abordar las comunicaciones.
- **Distribuir la Información:** Es el proceso para poner la información relevante a disposición de los interesados en el proyecto, de acuerdo con el plan establecido.
- **Gestionar las Expectativas de los Interesados:** Es el proceso que consiste en comunicarse y trabajar en conjunto con los interesados para satisfacer sus necesidades y abordar los problemas conforme se presentan.

- **Informar el Desempeño:** Es el proceso de recopilación y distribución de la información sobre el desempeño, incluidos los informes de estado, las mediciones del avance y las proyecciones**.**

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. Los objetivos de la Gestión de los Riesgos del Proyecto son aumentar la probabilidad y el impacto de eventos positivos, y disminuir la probabilidad y el impacto de eventos negativos para el proyecto. Los procesos de Gestión de los Riesgos del Proyecto incluyen:

- **Planificar la Gestión de Riesgos:** Es el proceso por el cual se define cómo realizar las actividades de gestión de los riesgos para un proyecto.
- **Identificar los Riesgos:** Es el proceso por el cual se determinan los riesgos que pueden afectar el proyecto y se documentan sus características.
- **Realizar el Análisis Cualitativo de Riesgos:** Es el proceso que consiste en priorizar los riesgos para realizar otros análisis o acciones posteriores, evaluando y combinando la probabilidad de ocurrencia y el impacto de dichos riesgos.
- **Realizar el Análisis Cuantitativo de Riesgos:** Es el proceso que consiste en analizar numéricamente el efecto de los riesgos identificados sobre los objetivos generales del proyecto.
- **Planificar la Respuesta a los Riesgos:** Es el proceso por el cual se desarrollan opciones y acciones para mejorar las oportunidades y reducir las amenazas a los objetivos del proyecto.
- **Monitorear y Controlar los Riesgos:** Es el proceso por el cual se implementan planes de respuesta a los riesgos, se rastrean los riesgos

identificados, se monitorean los riesgos residuales, se identifican nuevos riesgos y se evalúa el proceso de riesgos a través del proyecto.

9. Gestión de las Adquisiciones del Proyecto:

La 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 a fin de realizar el trabajo. La Gestión de las Adquisiciones del Proyecto incluye los procesos de gestión del contrato y de control de cambios necesarios para desarrollar y administrar contratos u órdenes de compra emitidas por miembros autorizados del equipo del proyecto. Los procesos de Gestión de las Adquisiciones del Proyecto incluyen:

- **Planificar las Adquisiciones:** Es el proceso de documentar las decisiones de compra para el proyecto; se especifica el enfoque y se identifican los posibles vendedores.
- **Efectuar las Adquisiciones:** Es el proceso de obtener respuestas de los vendedores, seleccionar un vendedor y adjudicar un contrato.
- **Administrar las Adquisiciones:** Es el proceso de gestionar las relaciones de adquisiciones, monitorear la ejecución de los contratos, y efectuar cambios y correcciones según sea necesario.
- **Cerrar las Adquisiciones:** Es el proceso de finalizar cada adquisición para el proyecto.

7.3.3 Relación entre los Grupos de procesos y las Áreas de conocimiento

A continuación, en la Tabla 2 se presentan los procesos relacionados con cada una de las Áreas de conocimiento y Grupos de procesos:

Tabla 2. Correspondencia entre los Grupos de Procesos y las Áreas de Conocimiento de la Gestión de proyectos[42]

	Grupo de Procesos de Iniciación	Grupo de Procesos de Planificación	Grupo de Procesos de Ejecución	Grupo de Procesos de Seguimiento y Control	Grupo de Procesos de Cierre
Gestión de la Integración del Proyecto	Desarrollar el Acta de Constitución del Proyecto	Desarrollar el Plan para la Dirección del Proyecto	Dirigir y Gestionar la ejecución del Proyecto	Monitorear y Controlar el trabajo del Proyecto Realizar el Control Integrado de Cambios	Cerrar Proyecto o Fase
Gestión del Alcance del Proyecto		Recopilar requisitos Definir el Alcance Crear EDT		Verificar el Alcance Controlar el Alcance	
Gestión del Tiempo del Proyecto		Definir las actividades Secuenciar las actividades Estimar los Recursos de las Actividades Estimar la Duración de las Actividades Desarrollar el Cronograma		Controlar el Cronograma	
Gestión de los Costos del Proyecto		Estimar los Costos		Controlar los Costos	

[42] PROJECT MANAGEMENT INSTITUTE. PMBoK (Project Management Body of Knowledge), version 4 [en línea]. (Citada: 16 de Junio. 2012) <http://www.pmi.org/PMBOK–Guide–and–Standards.aspx>

	Grupo de Procesos de Iniciación	Grupo de Procesos de Planificación	Grupo de Procesos de Ejecución	Grupo de Procesos de Seguimiento y Control	Grupo de Procesos de Cierre
		Determinar el Presupuesto			
Gestión de la Calidad del Proyecto		Planificar la Calidad	Realizar el Aseguramiento de Calidad	Realizar el Control de Calidad	
Gestión de los Recursos Humanos del Proyecto		Desarrollar el Plan de Recursos Humanos	Adquirir el Equipo del Proyecto Desarrollar el Equipo del Proyecto Dirigir el Equipo del Proyecto		
Gestión de las Comunicaciones del Proyecto	Identificar a los Interesados	Planificar las Comunicaciones	Distribuir la Información Gestionar las expectativas de los interesados	Informar el Desempeño	
Gestión de los Riesgos del Proyecto		Planificar la Gestión de Riesgos Identificar los Riesgos Realizar el Análisis Cualitativo de Riesgos Realizar el Análisis Cuantitativo de Riesgos Planificar la Respuesta a los riesgos		Monitorizar y Controlar los Riesgos	
Gestión de las		Planificar las	Efectuar las	Administrar	Cerrar las

	Grupo de Procesos de Iniciación	Grupo de Procesos de Planificación	Grupo de Procesos de Ejecución	Grupo de Procesos de Seguimiento y Control	Grupo de Procesos de Cierre
Adquisiciones del Proyecto		Adquisiciones	Adquisiciones	las Adquisiciones	Adquisiciones

7.3.4 Elementos del PMBoK en el framework propuesto

Los elementos del PMBoK en el framework propuesto se encuentran presentados en el capítulo 8.4.1 Principios adoptados .

7.4 Desarrollo Ágil de Software

El desarrollo ágil de software es un conjunto de métodos y metodologías de desarrollo de software basado en el ciclo de vida de desarrollo de software iterativo e incremental, donde las necesidades y soluciones evolucionan a través de la colaboración entre la auto-organización, equipos multi-funcionales. Promueve la planificación adaptativa, el desarrollo evolutivo, liberaciones útiles y progresivas, enfoque iterativo basado en la restricción del tiempo (timebox = se produce todo lo que más se pueda en una cantidad de tiempo finito, yendo en contraposición de desarrollar un alcance fijo en un tiempo limitado), y alienta rápida y respuesta flexible a los cambios.

Las primeras definiciones de las metodologías ágiles se realizaron en la década de los 90’s como respuesta a las metodologías robustas y pesadas predominantes en esa época, con: Scrum (1995), Crystal Clear, Extreme Programming XP (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995); pero comienzan a tener fuerza

fuertemente con la proclamación de “El Manifiesto Ágil”[43] el cual se realizó en el 2001

7.4.1 El manifiesto Ágil

En febrero de 2001, 17 desarrolladores de software se reunieron en el Snowbird, Utah localidad, para discutir los las metodologías ágiles de desarrollo. De esta reunión ellos publicaron el Manifiesto por el Desarrollo Ágil de Software (En inglés : “Manifesto for Agile Software Development”), definiendo el enfoque ahora se conoce como desarrollo ágil de software. Algunos de los autores del Manifiesto formaron la Agile Alliance, una organización sin fines de lucro que promueve el desarrollo de software de acuerdo con los principios del Manifiesto. El Manifiesto Ágil reza así:

“Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

Individuos e interacciones sobre procesos y herramientas
Software funcionando sobre documentación extensiva
Colaboración con el cliente sobre negociación contractual
Respuesta ante el cambio sobre seguir un plan

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron

[43] Beck, Kent , et al. (2001). "Manifesto for Agile Software Development" . Agile Alliance . (Citada: 21 de Junio de 2012) <http://agilemanifesto.org/iso/es/>

Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas"[44]

De igual forma bajo este manifiesto hay un compromiso a cumplir con los siguientes principios:

1. "Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

---

[44] Beck, Kent , et al. (2001). "Manifesto for Agile Software Development" . Agile Alliance . (Citada: 21 de Junio de 2012) <http://agilemanifesto.org/iso/es/>

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia."[45]

7.4.2 Características

Hay muchas metodologías de desarrollo ágil. La mayoría de promueven el desarrollo, el trabajo en equipo, la colaboración y adaptabilidad en todo el proceso del ciclo de vida del proyecto, y la aceptación del cambio.

Las iteraciones

Otra de las características de las metodologías agiles es dividir las tareas en pequeños incrementos que impliquen un planificación mínima sin involucrar planeaciones a largo plazo. Iteraciones son de plazos cortos fijos (timeboxes) que típicamente duran de una a cuatro semanas. Cada iteración incluye un equipo interdisciplinario de trabajo, funcional en todas las disciplinas de ingeniería de software: planificación, análisis de requisitos, diseño, codificación, pruebas unitarias y pruebas de aceptación, y despliegue. Al final de la iteración un producto de trabajo se presenta a las partes interesadas. Esto minimiza el riesgo global del proyecto y permite adaptarse a los cambios con rapidez. Una iteración no puede adicionar suficiente funcionalidad para justificar una salida al mercado, pero la meta es tener una versión disponible (con un mínimo de errores) al final de cada iteración, la cual sea usable, testeable y valiosa para el cliente, en la que perciba un avance significativo. Múltiples iteraciones podrían ser necesarias para lanzar un producto o nuevas características.

---

[45] Beck, Kent; et al. (2001). "Principles behind the Agile Manifesto". Agile Alliance. Archived from the original on 14 June 2010. Citada el 24 de octubre de 2012. <http://agilemanifesto.org/iso/es/principles.html>

El equipo y la colaboración

La composición del equipo en un proyecto ágil suele ser multi-funcional y la auto-organización, sin considerar jerarquía o roles corporativos de los de los miembros del equipo. Los miembros del equipo deciden individualmente la forma de satisfacer las necesidades y tareas de una iteración.

Las metodologías ágiles enfatizan en la comunicación cara a cara en lugar de documentos escritos (cuando el equipo se encuentra en el mismo lugar). La mayoría de los equipos ágiles trabajar en una sola oficina abierta, que facilita la comunicación. Tamaño del equipo suele ser pequeño (5-9 personas) para simplificar la comunicación del equipo y la colaboración en equipo. Grandes esfuerzos de desarrollo se pueden entregar por varios equipos de trabajo hacia un objetivo común o en diferentes partes de un esfuerzo. Esto puede requerir una coordinación de prioridades entre los equipos.

El cliente

Cada equipo ágil incluirá un representante del cliente. Esta persona es designada por las partes interesadas para que actúe en su nombre y hace un compromiso personal de estar disponible para los desarrolladores, para resolver todas las preguntas requeridas para finalizar la iteración. Al final de cada iteración, los interesados (todos los afectados por la existencia del proyecto, en especial el equipo desarrollador) y el representante del cliente revisan el progreso y re-evalúan prioridades con miras optimizar el retorno de la inversión (ROI) y asegurar la alineación con las necesidades del cliente y los objetivos de la empresa.

Reunión diaria

Algunas metodologías ágiles emplean una reunión diaria cara a cara entre los miembros del equipo. Esto incluye específicamente el representante del cliente y los interesados. En esta breve sesión (de no más de 15 minutos y realizada de

pie), los miembros del equipo informarán mutuamente sobre lo que hicieron el día anterior, lo que van a hacer hoy, y cuáles son sus obstáculos tienen al momento. Esto ayuda a tener el equipo alineados con los objetivos y lograr compromisos de trabajo. En esta sesión no se resuelven temas técnicos, solo se habla tareas realizadas, tareas a realzar e inconvenientes para que el líder del proyecto se encargue de removerlos. Esta técnica hace parte de la metodología de SCRUM.

La métrica del avance

El desarrollo ágil de software enfatiza como la principal medida de progreso el software funcionando. Esto, combinado con la preferencia por la comunicación cara a cara, produce menos documentación escrita que otras metodologías. Las metodologías ágiles alientan a los interesados a dar prioridad a sus "necesidades", basándose exclusivamente en el valor que estas necesidades convertidas en funcionalidades le agregan al negocio.

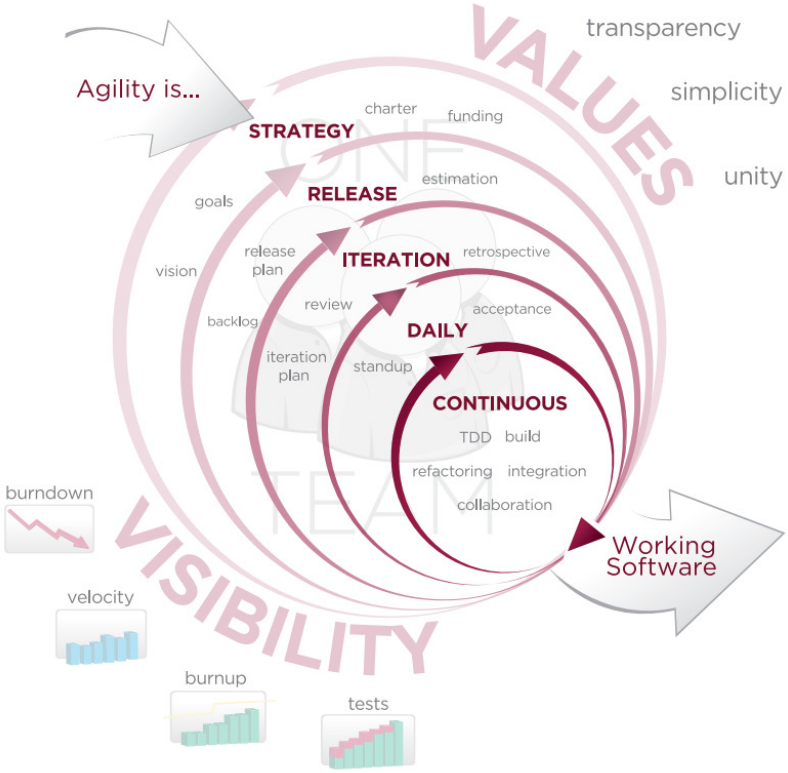
Algunas herramientas y técnicas

Herramientas y técnicas específicas, tales como la integración continua, automatización de pruebas unitarias, programación por pares, el desarrollo basado en pruebas test-driven development), conocimiento y buen uso de patrones de diseño, diseño basado en el dominio (domain-driven design), refactorización de código, pruebas automatizadas de interfaz de usuario, desarrollo basado en comportamientos (behavior driven development) y otras técnicas se utilizan a menudo para mejorar la calidad y aumentar la agilidad del proyecto.

Como lo menciona la página especializada en metodologías ágiles http://versionone.com y lo esquematiza en Figura 10, existen múltiples prácticas

para realizar en un proyecto con "sabor" ágil *THE AGILE CHECKLIST*[46] (El checklist ágil), estas se presentan a continuación:

Figura 10. The Agile Checklist[47]



Las prácticas y sus momentos de uso se presentan en la Tabla 3.

[46] VERSIONONE. The agile checklist [en línea]. (Citada: 21 de Junio de 2012) <http://pm.versionone.com/AgileChecklist.html>. Ver también <http://it-talents.org/agile-development-rhythms-meeting-checklist>

[47] VERSIONONE. Understand agile development methods with the agile development poster. [en línea]. (Citada: 21 de Junio de 2012) <http://pm.versionone.com/AgilePoster.html>

Tabla 3. Prácticas ágiles y su momento de uso (Elaboración Propia)

Momento	Prácticas Ágiles
Prácticas continuas	• TDD – *Test Driven Development*: Desarrollo Dirigido por las Pruebas. • Integración • *Refactoring* • Colaboración entre el equipo
Diarias	• *Standup meeting* • Test de Aceptación
Iteración	• Plan de iteración • Revisión • Reunión retrospectiva
Release	• Backlog: Pila de trabajo • Plan de release • Estimación por parte del equipo
Estrategia	• Visión • Objetivos • Carta de Inicio • Aprobación de presupuesto

7.4.3 Elementos de las metodologías ágiles en el framework propuesto

Los elementos del las metodologías ágiles en el framework propuesto se encuentran presentados en el capítulo 8.4.2 Prácticas adoptadas de las metodologías ágiles.

8. METODOLOGÍA DE TRABAJO

Considerando el Proceso Software tal como lo cita Ruiz[48] es "Un conjunto coherente de políticas, estructura organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software. (Fugetta, 2000)", para la construcción del framework metodológico se determinaron y consideraron estos diferentes elementos a definir y a configurar, para tal fin se realizaron las siguientes actividades:

1. Se eligió una herramienta para plasmar el framework
2. Se identificaron los tipos de proyectos a soportar y la forma de diferenciación
3. Se identificaron cuáles de las mejores prácticas disponibles en el marco teórico basado en RUP, PMBoK, CMMI, deberían quedar plasmadas en el framework.
4. Se identificaron unas prácticas ágiles a incluir en el framework.
5. Se creó la EDT de cada tipo de proyecto
6. Se identificaron y definieron los roles involucrados
7. Se identificaron los entregables / artefactos a generar
8. Se configuro la herramienta para soportar metodologías.

[48] RUIZ, Francisco. Ingeniería de Procesos Software. [en línea]. (Citada: 17 de Junio de 2012) < http://alarcos.inf-cr.uclm.es/doc/psgc/doc/psgc0809_parte2b_ips.pdf>.

8.1 ELECCIÓN DE LA HERRAMIENTA EMPLEADA PARA LA PUBLICACIÓN DEL FRAMEWORK

Al momento de elegir cuál herramienta era la más adecuada para cumplir con el objetivo específico, "Desarrollar un sitio web navegable con el framework propuesto", se evaluaron las siguientes alternativas:

- Crear un sitio web de forma manual empleando una herramienta como *Adobe Dreamweaver*.
- Utilizar un administrador de contenido web (CMS – *Content Management System*)
- Emplear herramientas específicas para Ingeniería de Software como:
    - *Enterprise Architect de Sparxs Systems*
    - *Eclipse Process Framework*
    - *Rational Method Composer*

A continuación se presenta un cuadro comparativo de dichas herramientas. Como criterios de selección se escogieron los siguientes:

- Costo de licenciamiento: Corresponde a la necesidad de adquirir licencia del producto realizando compra de la misma, o si corresponde a alguna iniciativa de *Open Source o freeware*
- Orientación a Ingeniería de Software: En este criterio se evalúa si la herramienta fue concebida para soportar exclusivamente ingeniería de software, o cualquiera de sus procesos.
- Uso de estándares de la industria: En este criterio se evalúa si la herramienta fue concebida para cumplir con estándares de la industria de software.
- Orientación a estándares de Ingeniería de Software: Este criterio evalúa si la herramienta está cumpliendo con estándares de la ingeniería de software
- Facilidad de uso y necesidad de entrenamiento: Este criterio evalúa si la herramienta es fácil de usar y requiere entrenamiento para su empleo.

Tabla 4. Tabla comparativa de herramientas para publicación del framework (Elaboración Propia)

	Costo de Licenciamiento	Orientación a Ingeniería de software	Uso de estándares de la industria	Orientación a estándares de Ingeniería de Software	Facilidad de uso	Observaciones
Adobe Dreamweaver.	Sí	No	Sí	No	No, Requiere entrenamiento	Ninguna
Administrador web de contenido (CMS)	Algunos CMS lo son.	No	Sí	No	Sí (Dependiendo del que se escoja)	Requiere de servidor de aplicaciones para ser usado
Enterprise Architect de Sparxs Systems	Sí	Si	Sí	Sí	Sí	Permite publicar procesos basados en el estándar SPEM[49]
Eclipse Process Framework Composer	No	Si	Sí	Sí	Sí	Diseñada específicamente para soportar SPEM
Rational Method Composer	Si	Si	Sí	Sí	Sí	Diseñada específicamente para soportar SPEM

Dadas las condiciones académicas de esta tesis, enmarcadas en el propósito de libre distribución de este trabajo, adopción de estándares y ser una herramienta específicamente creada para este propósito, se elige el *ECLIPSE PROCESS FRAMEWORK COMPOSER*, en adelante EPF Composer, como la herramienta para la publicación del framework construido en esta tesis.

8.1.1 Características del EPF Composer.

El editor *EPF Composer*, es una herramienta gratuita, desarrollada por la comunidad Eclipse, que sirve para crear y modificar contenido de métodos,

[49] SPEM: (Software & Systems Process Engineering Metamodel Specification). Es un estándar creado por la OMG (Object Management Group) para documentar procesos de Ingeniería de Software. Actualmente se encuentra en la versión 2.0

procesos o metodologías, y generar automáticamente documentación adecuada en formato para la web. La dirección del proyecto es http://www.eclipse.org/epf/.

Es una herramienta para producir frameworks personalizables de procesos de Ingeniería de Software, que contiene además procesos ejemplo que soportan una gran cantidad de herramientas y proyectos con estilos diferentes de desarrollo.

Como se explica en el sitio del proyecto (http://www.eclipse.org/epf/), la herramienta cuenta con dos objetivos primordiales[50]:

- Proporcionar un framework extensible con herramientas para la Ingeniería de procesos de software, - el método, la creación (autoría) de procesos, la gestión de bibliotecas, configuración y publicación de un proceso.
- Proporcionar el contenido de proceso de ejemplo y extensible para una amplia gama de procesos de desarrollos de software, soportando el desarrollo iterativo, ágil, incremental, y aplicable a un amplio conjunto de plataformas de desarrollo y aplicaciones.

Por lo general, existen dos problemas clave que deben abordarse para implementar un proceso

- En primer lugar, los desarrolladores necesitan entender los métodos y las prácticas más importantes de desarrollo de software.

- En segundo lugar, los equipos de desarrollo necesitan definir cómo aplicar sus métodos de desarrollo y sus mejores prácticas, a lo largo del ciclo de vida de desarrollo.

---

[50]ECLIPSE. Eclipse Process Framwork [en línea]. (Citada: 17 de Junio de 2012) <http://www.eclipse.org/epf/>

El proyecto Eclipse Process Framework proporciona soluciones a problemas comunes que los equipos de desarrollo enfrentan en la creación y gestión de sus métodos y procesos[51]. Por ejemplo:

- Los equipos de desarrollo necesitan un acceso fácil y centralizado a la información: por lo general, las organizaciones de desarrollo y en las universidades, no mantienen bases de datos centralizadas para sus prácticas y procesos. Por lo general, los procesos no está documentados en absoluto bajo una misma directriz, o están físicamente distribuidos en diferentes formatos de presentación. Los procesos necesitan desplegarse y accederse en las instalaciones donde se ejecutan los proyectos, para proporcionar la documentación del proceso, mientras que el trabajo se está realizando.

- Es difícil integrar procesos de desarrollo que se implementan en un formato propietario: cada libro y publicación presenta el contenido de método y el proceso utilizando un formato diferente. La falta de estándares ampliamente adoptados y conceptos claramente definidos para representar el contenido del método y los procesos hacen que sea difícil integrar los procesos de diferentes autores y tendencias académicas.

- Los equipos de trabajo que realizan software académico carecen de una base de conocimientos actualizada para capacitarse en métodos y mejores prácticas: antes de utilizar los procesos de desarrollo, los equipos deben ser entrenados.

---

[51] HAUMER, Peter. Eclipse Process Framework Composer. Part 1: Key Concepts. 2007. [en línea] (Citada: 17 de Junio de 2012). <http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf>

- Los equipos de trabajo de proyectos académicos orientados a construcción de módulos, sistemas, proyectos de grado necesitan dimensionar correctamente sus procesos, dichos procesos deben adaptarse no sólo para un proyecto en particular, sino también continuamente durante todo el ciclo de vida del proyecto.

EPF Composer ofrece los siguientes puntos clave:

- Creación de procesos con editores de estructura de desglose y diagramas de flujo de trabajo, utilizando editores de proceso de múltiple presentación, diferentes vistas del proceso, y sincronización de las vistas con los cambios que se realicen en los procesos.

- Reutilización de patrones de mejores prácticas, para un ágil ensamble de procesos con la facilidad de arrastrar y soltar.

- Seleccionar, combinar, adaptar y rápidamente ensamblar configuraciones de proceso a partir de contenidos de método, para los proyectos de desarrollo de una organización.

- Una estructura de gestión y apariencia común, para el contenido de una organización.

- Creación de métodos y procesos con contenidos enriquecidos, como texto, imágenes y multimedia, manteniendo la independencia de la arquitectura del proceso.

- Administración del contenido utilizando interfaces de usuario simples y editores de texto enriquecido para la creación de descripciones del contenido.

- Múltiples vistas del contenido por medio de representaciones de texto y gráficas, y de editores que permiten el uso de estilos, imágenes, tablas, hipervínculos, y edición de HTML.

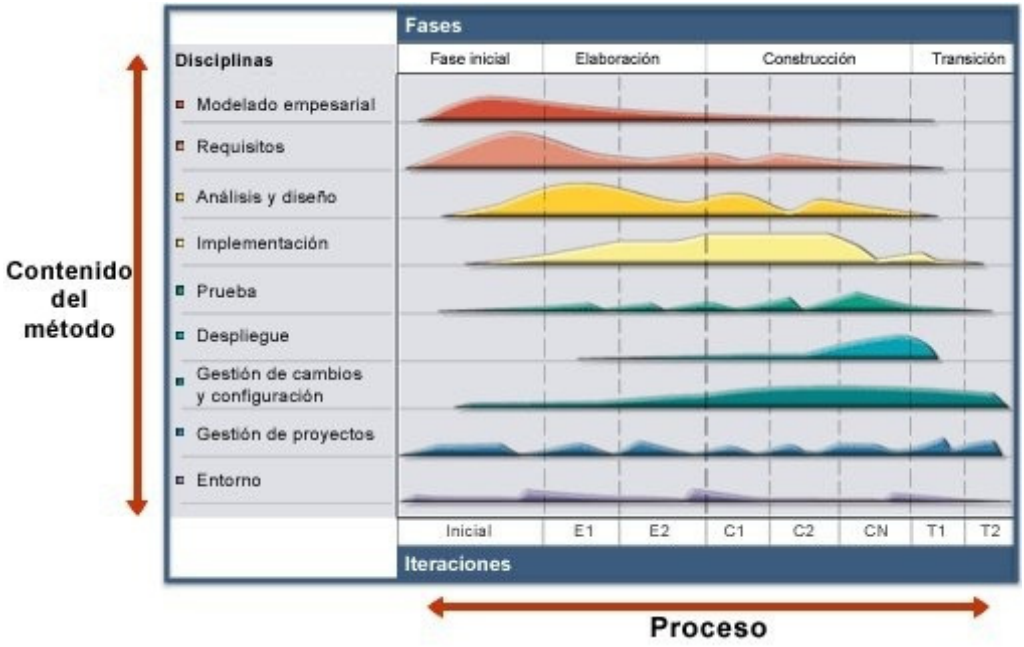
- Publicación de configuraciones de proceso utilizando diferentes formatos, como HTML y PDF.

8.1.2 Terminología y conceptos manejados en el *EPF Composer*

Para un trabajo eficiente con el *EPF Composer* es necesario entender algunos conceptos que se utilizan para organizar el contenido.

El principio más importante en *EPF Composer* es la separación, en dos dimensiones diferentes, del contenido del método y su aplicación en procesos. El contenido (*Method Content*) define lo que se va a producir, las habilidades que se necesitan, y las explicaciones paso a paso que describen cómo alcanzar los objetivos específicos de desarrollo. Las descripciones del contenido son independientes del ciclo de vida del desarrollo. Los procesos (*Processes*) describen el ciclo de vida de desarrollo. Los procesos toman los elementos de contenido y los relacionan en secuencias ordenadas que se adaptan a diferentes tipos de proyecto.

Figura 11. Contenido del método vs Proceso (en ejemplo en RUP)[52]

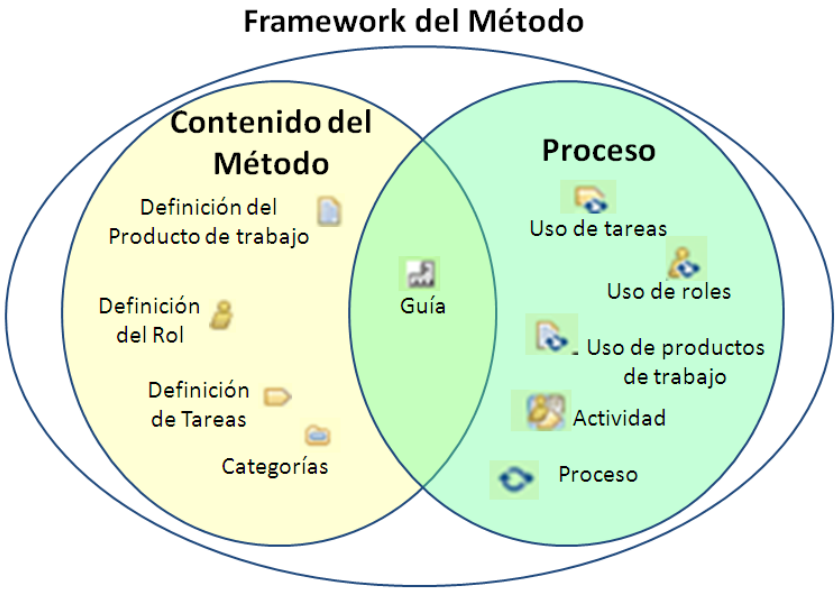


En la Figura 11 se muestra un ejemplo de cómo esta separación se representa en la metodología RUP (Proceso Unificado de Rational, por sus siglas en Español). El contenido del método (*Method Content*) se clasifica en disciplinas, a lo largo del eje Y. El proceso (Process) representa el tiempo, a lo largo del eje X. Este es el ciclo de vida de un proyecto de desarrollo.

EPF Composer permite que el contenido sea estructurado de una forma específica por medio de un esquema predefinido. Este esquema es una evolución de la especificación SPEM (*Software and Systems Process Engineering Metamodel*) versión 2.0, anteriormente UMA (*Unified Method Architecture*). Los contenidos de método y los procesos no se limitan a la Ingeniería de Software, también pueden abarcar otras disciplinas de diseño e ingeniería, tales como la ingeniería mecánica, transformación del negocio, ciclos de ventas, etc.

[52] TABARES, Luis. Personalización de RUP para proyectos académicos de desarrollo de software. Universidad Eafit. 2011.

Figura 12. Un resumen de la terminología propuesta para el Eclipse Process Framework[53]



La Figura 12 ofrece un resumen de los elementos clave que se utilizan en el *EPF Composer*, y sus relaciones con los procesos o el contenido de métodos. Como se puede apreciar, el contenido (*Method Content*) es enunciado principalmente a través de productos de trabajo, roles, tareas, y guías. Las categorías son necesarias para la publicación de las metodologías de proceso en su sitio web. Las guías, como listas de verificación, ejemplos, o planes de trabajo, también se pueden definir para proporcionar tutoriales de un proceso.

En el lado derecho de la Figura 12, se muestran los elementos utilizados para representar los procesos (Process) en *EPF Composer*. El principal elemento de proceso es la actividad, la cual puede ser anidada para definir estructuras de

[53] HAUMER, Peter. Eclipse Process Framework Composer. Part 1: Key Concepts. 2007. [en línea] (Citada: 17 de Junio de 2012). <http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf>

desglose de trabajo, que pueden relacionarse entre sí para definir un flujo de trabajo. Las actividades también contienen descriptores que hacen referencia al contenido. En EPF Composer se utilizan las actividades para definir procesos, y éstos se clasifican en procesos de entrega y patrones de capacidad.

Los procesos de entrega (*Delivery processes*) representa una plantilla de proceso completa e integrada para la realización de un tipo específico de proyecto. Un proceso de entrega describe un ciclo de vida de proyecto completo, de principio a fin, y puede ser utilizado como referencia para la ejecución de proyectos con características similares.

Los patrones de capacidad (*Capability patterns*) son componentes que expresan y comunican el proceso de un área clave, como una disciplina o una práctica. Se utilizan como bloques de construcción para formar procesos de entrega o patrones de capacidad más grandes. Esto permite asegurar una óptima reutilización y aplicación de las principales mejores prácticas en la creación de procesos, en *EPF Composer*.

En caso de requerirse más información acerca de la herramienta, se sugiere visitar "*Getting Started*" del sitio oficial del *EPF Composer*[54].

8.2 Framework Propuesto

El framework metodológico propuesto el cual de ahora en adelante se denominará SOFTWARE EDUP, nombrado así debido a su enfoque educativo (EDU) y su apoyo en el Proceso Unificado (Unified Process – UP). Se encuentra apoyado básicamente en el proceso unificado de Rational al cual se le suprimieron y

[54] ECLIPSE. Eclipse Process Framework - Getting Started. [en línea] (Citada: 17 de Junio de 2012). <http://www.eclipse.org/epf/general/getting_started.php>

adaptaron elementos en búsqueda de la simplificación y facilidad de uso en e escenarios académicos.

Con este framework metodológico se busca proporcionar una documentación consistente que pueda ser empleada a lo largo de la formación profesional, buscando que el estudiante perciba un proceso disciplinado a seguir un proceso disciplinado para producir un componente de software, el cual sea empleado como un referente intrínseco en el estudiante debido a su uso frecuente.

El sector educativo tiene como una de sus principales responsabilidades en los programas Ingeniería de Sistemas, Ingeniería Informática o similares, preparar a los estudiantes para trabajar en la industria del software[55], basados en el cuerpo de conocimiento relacionado con la Ingeniería de software[56]. 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,

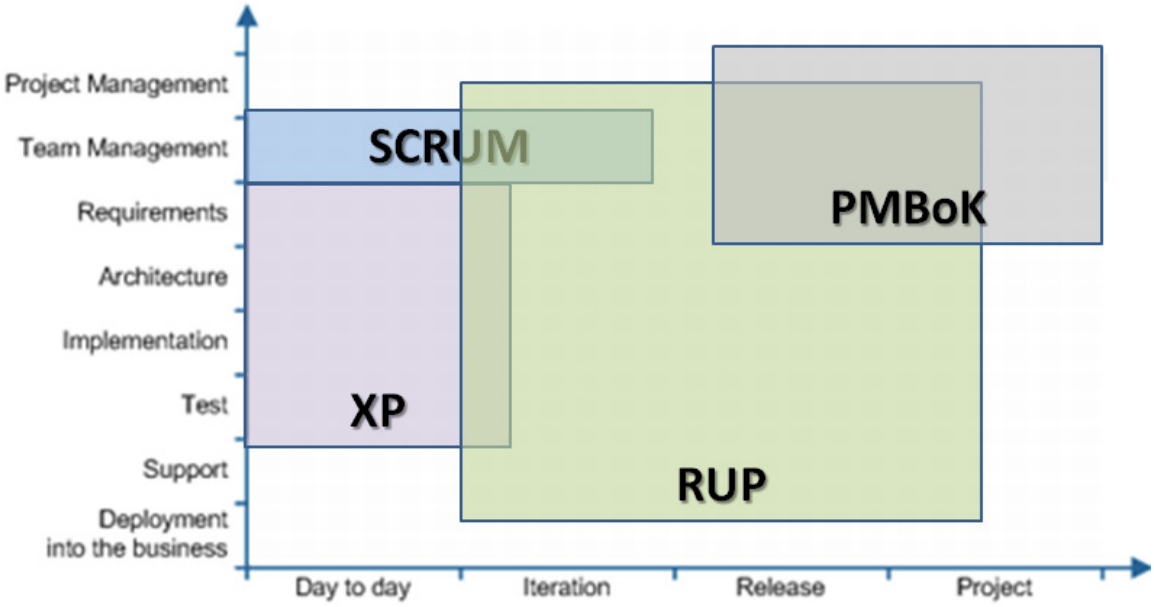
[55] Kruchten, P. (2011). Experience teaching software project management in both industrial and academic settings. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 199–208. doi:10.1109/CSEET.2011.5876087

[56] Bernhart, M., Grechenig, T., Hetzl, J., & Zuser, W. (2006). Dimensions of software engineering course design. Proceeding of the 28th international conference on Software engineering - ICSE '06, 667. doi:10.1145/1134285.1134387

desde la concepción hasta la implementación de la solución. Los escenarios para estas experiencias deben emular 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.

En la Figura 13. Enfoque de RUP, XP y Scrum (Adaptado de Corallis) se presentan para el escenario de tiempo y de gestión e ingeniería, la aplicabilidad de diferentes tipos de metodologías.

Figura 13. Enfoque de RUP, XP y Scrum (Adaptado de Corallis)[57]



[57] COLLARIS, Remi-Armand y DEKKER, Eef. Comparing Methods. 2011.[en línea] (Citada: 17 de Junio de 2012). <http://blog.scrumup.com/2011/05/comparing-methods.html>

De acuerdo con la Figura 13 (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, contando con una leve presencia en el campo de gestión del proyecto, el cual el mismo RUP delega en el PMBoK.

Otras características valoradas para la selección de RUP como la metodología referente, son:

- RUP es una metodología bien definida, que utiliza el ciclo de vida iterativo e incremental y se encuentra muy bien documentada. Estas características son fundamentales para SOFTWARE EDUP, donde los estudiantes con una orientación mínima deben adquirir un conocimiento práctico de los elementos clave del proceso de software y ser productivos en los roles asignados para la ejecución de los proyectos académicos de desarrollo de software.

- RUP promueve la utilización de casos de uso para la descripción de los requisitos de software. Los estudiantes universitarios están familiarizados con el Lenguaje de Modelado Unificado (UML, por sus siglas en Inglés), y los casos de uso, por las asignaturas cursadas en sus programas.

- RUP se enfoca en el desarrollo basado en componentes, como una de sus mejores prácticas, concepto que se debe enseñar a los estudiantes.

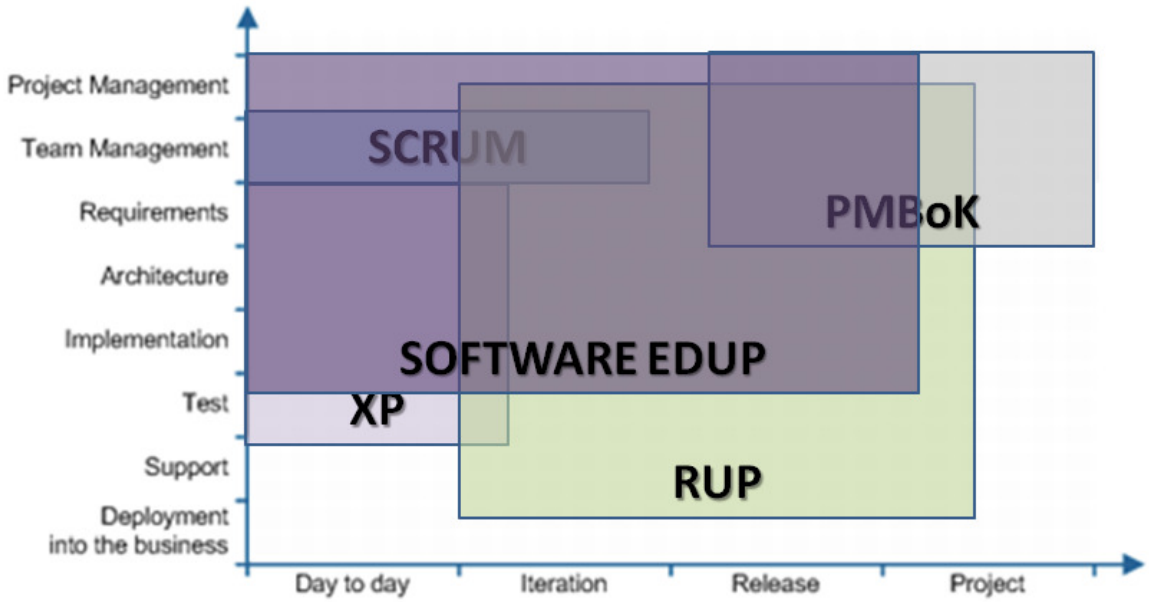
- RUP es un estándar en la industria, que ha sido adaptado por diferentes compañías en el medio local y nacional, permitiéndoles ser más eficientes y rentables en la labor de desarrollar software.

- RUP estructura el desarrollo en cuatro fases, cada una ejecutada en una o más iteraciones, permitiendo una temprana mitigación del riesgo durante todo el proceso. El énfasis en la mitigación del riesgo en proyectos académicos, con equipos de estudiantes desarrollando un producto de software, dentro de un plazo de tres o cuatro meses, es una necesidad.

- RUP es configurable. Esta flexibilidad es fundamental para adaptarse a proyectos de diferente magnitud en el ámbito académico.

El marco del trabajo SOFTWARE EDUP propone una metodología para orientar el desarrollo de software en proyectos académicos de diferente alcance, teniendo como base a RUP (ampliamente utilizada en la industria nacional) pero fortaleciendo el día a día y la gerencia de proyectos. Este enfoque se observa en la Figura 14.

Figura 14. Software EDUP - Disciplinas y alcance del ciclo de vida. (Elaboración Propia)



De acuerdo con lo anterior, el framework metodológico propuesto en SOFTWARE EDUP implementa una personalización de la metodología RUP, e incorpora algunas prácticas de las metodologías OpenUP, XP y Scrum, permitiendo de esta forma establecer una guía para el desarrollo de software en proyectos académicos de diferente magnitud.

A continuación se presentan los elementos constituyentes del framework, enfocándose en prácticas adoptadas de desarrollo de software que serán implementadas y que formarán la línea base para determinar cómo deben ser abordados los proyectos académicos de desarrollo de software al emplear la presente propuesta metodológica.

8.3 Procesos y Proyectos Propuestos

El framework SOFTWARE EDUP se encuentra configurado para dos tipos de procesos que reflejan dos tipos de proyectos académicos de desarrollo de software; estos se clasificaron como:

- **Proyecto Tipo 1 (Proceso Básico)**
- **Proyecto Tipo 2 (Proceso Completo)**

Con base en las áreas de conocimiento de la Gerencia de proyectos (PMBoK), ambos tipos de proyectos se describen en la Tabla 5, presentada continuación:

Tabla 5. Características de los proyectos soportados por el framework (Elaboración Propia)

	Tipo de Proyecto	
Áreas de conocimiento	**Proyecto Tipo 1**	**Proyecto tipo 2**
Tiempo	• Duración máxima 2 meses • Debido a la no existencia de la gestión de costos existe un esfuerzo importante en la gestión del tiempo.	• Duración máxima un semestre. • Debido a la no existencia de la gestión de costos existe un esfuerzo importante en la gestión del tiempo.
Costo	• Los tiempos de las personas involucradas, su costo en el mercado laboral, adicionalmente los recursos físicos involucrados (equipos, licencias, internet, papel, etc.) pueden cuantificados y administrados en el proyecto. Queda a elección del equipo hacerlo, mas no son objeto de esta versión del framework.	• Los tiempos de las personas involucradas, su costo en el mercado laboral, adicionalmente los recursos físicos involucrados (equipos, licencias, internet, papel, etc.) pueden cuantificados y administrados en el proyecto. Queda a elección del equipo hacerlo, mas no son objeto de esta versión del framework.
Alcance	• Desarrollo tipo práctica • Implica el desarrollo de varias funcionalidades, o un módulo. • Probablemente no existan controles de cambio	• Desarrollo tipo práctica del semestre o proyecto de grado. • Implica el desarrollo de un módulo, varios módulos o un sistema • Muy pocos o ningún control de cambio durante el proyecto.
Calidad	• Calidad del proceso o Guiada por el docente a través de revisiones formales • Calidad del producto o Responsabilidad del equipo del proyecto o Verificada por el equipo (para esto se puede hacer uso de estándares como el ISO/IEC 25010).	• Calidad del proceso o Responsabilidad del equipo del proyecto o Guiada por el docente a través de revisiones formales • Calidad del producto o Responsabilidad del equipo del proyecto o Verificada por el equipo (para esto se puede hacer uso de estándares como el ISO/IEC 25010).
Riesgos	• La gestión de riesgos es responsabilidad del equipo	• La gestión de riesgos es responsabilidad del equipo

	Tipo de Proyecto	
Áreas de conocimiento	**Proyecto Tipo 1**	**Proyecto tipo 2**

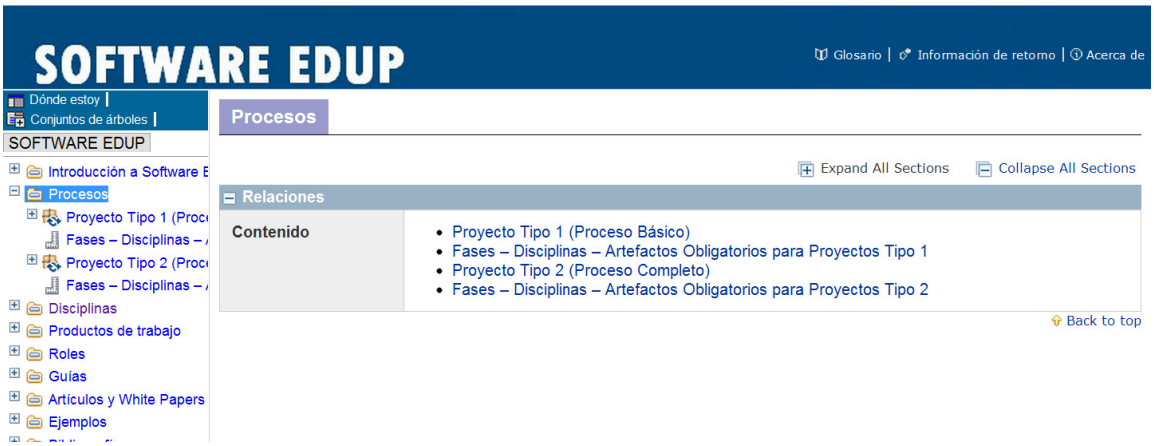
Recursos Humanos	• El docente
	  ○ El papel del docente es relevante, es considerado como el guía en el proceso de desarrollo.
	• El cliente
	  ○ Es representado por el docente que proporciona el enunciado de la práctica y aclara cualquier duda al respecto del enunciado.
	• El equipo
	  ○ El equipo no se encuentra inmerso constantemente en un ambiente de trabajo y se reúne de forma esporádica para resolver el proyecto propuesto. Es posible que esa reunión sea de forma virtual.
	  ○ El equipo no tiene acceso permanente al docente (o cliente)
	  ○ El equipo de construcción es conformado muchas veces por un solo integrante.
		• El docente
		  ○ El papel del docente es considerado como el guía en el proceso de desarrollo.
		• El cliente
		  ○ El cliente algunas veces es representado por el docente, alguien externo o un miembro del equipo que formuló el proyecto a realizar.
		• El equipo
		  ○ El equipo no se encuentra inmerso constantemente en un ambiente de trabajo y se reúne de forma esporádica para resolver el proyecto propuesto. Es posible que esa reunión sea de forma virtual.
		  ○ El equipo no tiene acceso permanente al docente (o cliente)
		  ○ El equipo de construcción es conformado muchas veces por una sola persona. El máximo de integrantes pueden ser de 4 a 5 estudiantes.
		  ○ El equipo posiblemente domina o es experto en la tecnología, y/o se encuentra realizando una inmersión en la misma.

Comunicaciones	• Esfuerzo bajo en la gestión de las comunicaciones • Los informes de avance son recomendados mas no requeridos.	• Esfuerzo bajo en la gestión de las comunicaciones, pero puede incrementarse en la medida que existan instancias diferentes a los estudiantes y al docente que requieran atención. • Los informes de avance son requeridos.
Adquisiciones	• No aplican por lo general para este tipo de proyectos	• No aplican por lo general para este tipo de proyectos

La descripción del ciclo de vida para cada tipo de proyecto puede consultarse en la publicación del framework, la cual es un sitio Web estático y es parte integral de esta tesis, (de la dirección de la cual se puede descargar el framework es https://sites.google.com/site/softwareedup -> versiones, y fue generada con el EPF).

En la Figura 15 se puede observar cómo se ve el framework metodológico al ser desplegado en un navegador web.

Figura 15. Software EDUP - Desplegado



8.4 Fundamentos del Framework

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 este entorno.

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 a los aspectos dinámicos y las disciplinas corresponden a los aspectos estáticos.

8.4.1 Principios adoptados RUP, CMMi y PMBoK

La metodología SOFTWARE EDUP adopta mejores prácticas y principios, dirigidos a facilitar el desarrollo de software de proyectos académicos de diversa magnitud, con el fin de que se desarrollen productos de software con alta calidad, y aprovechando al máximo los recursos disponibles de una forma eficaz y eficiente.

A continuación se presentan las prácticas provenientes del marco teórico, las cuales se consideraron pertinentes incluir en el Framework. Entre corchetes se pondrá la fuente de la cual se adoptó, sea RUP, CMMI o PMBoK, según el caso, por ejemplo: [CMMI / PMBoK] para una práctica proveniente de CMMI y PMBoK respectivamente.

- **Adaptar el proceso [RUP]:** SOFTWARE EDUP es una metodología de desarrollo adaptable, que tiene como objetivo mantener la agilidad durante el

proceso de desarrollo, establecer planes realistas, y estimaciones conforme a las condiciones del proyecto y durante todo el ciclo de vida del proyecto. SOFTWARE EDUP es un marco de trabajo que se presenta como adaptable, y pudiéndose incluir componentes externos (contenidos de método), y a su vez tampoco descarta que sus componentes sirvan para otros marcos de trabajo.

- **Enfocado al nivel de abstracción [RUP]:** SOFTWARE EDUP favorece que se tenga un alto nivel de abstracción para reducir la complejidad y mejorar la comunicación entre los involucrados de los proyectos, a través de la recomendación de emplear herramientas de modelado de alto nivel como UML, el empleo de estándares abiertos, la definición temprana de la arquitectura, y el desarrollo de componentes para permitir la reutilización.

- **Centrado en la arquitectura [RUP]:** SOFTWARE EDUP además de emplear los casos de uso para guiar el proceso de desarrollo, presta especial atención a la definición temprana de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción. La arquitectura de los proyectos es representada a través del modelo de las 4+1 vistas, con el fin de proveer una representación arquitectónica estándar para que todos los *stakeholders* en el desarrollo la puedan comprender, discutir y analizar.

- **Demostrar valor iterativamente [RUP** / **PMBoK]:** En SOFTWARE EDUP las fases se dividen en iteraciones, cuyo resultado es una versión ejecutable. El objetivo de la metodología con cada iteración será mitigar los riesgos de mayor a menor, donde el concepto de riesgo se refiere a ciertos requisitos que son más críticos a la hora de ejecutar el proyecto. Cada iteración debe ser controlada y se debe abordar una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos

requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. La retroalimentación de una iteración finalizada permite reajustar los objetivos para las siguientes iteraciones. Esta dinámica continúa hasta cuando se haya finalizado por completo el ciclo de vida.

- **Dirigido por casos de uso [RUP]:** Para SOFTWARE EDUP, los casos de uso son la herramienta estándar empleada para especificar los requisitos funcionales del sistema, son los que guían el análisis y diseño, implementación y pruebas de todo el sistema, y adicionalmente son los elementos que permiten la trazabilidad.

- **Enfocado en la calidad y las pruebas [RUP** / **CMMI** / **PMBoK]:** SOFTWARE EDUP contiene mecanismos para que la calidad de todos los artefactos se evalúe en varios puntos durante todo el ciclo de vida, especialmente al final de cada iteración. En la metodología SOFTWARE EDUP se pueden encontrar tareas enfocadas a revisar y evaluar, las cuales juegan un papel fundamental para asegurar calidad, no solo al final de los proyectos, sino durante todo su ciclo de vida.

- **Enfocado en los riesgos [RUP** / **CMMI** / **PMBoK]:** La gestión de los riesgos es contemplada en SOFTWARE EDUP desde el inicio del proyecto hasta el final del mismo. El enfoque en los riesgos es el punto de partida para programar las actividades que deben ejecutarse durante las iteraciones.

- **Gestionar los cambios [RUP** / **CMMI** / **PMBoK]:** El cambio es un factor de riesgo crítico en los proyectos de software, ante los cuales SOFTWARE EDUP crea las condiciones necesarias, a través de sus actividades, para gestionarlos con un enfoque ágil lo más tempranamente posible con su proceso iterativo e incremental, con la participación continua de los *stakeholders* y con las

actividades de retroalimentación. Los artefactos de software cambian debido a acciones de refinación o maduración durante el proceso de desarrollo. SOFTWARE EDUP asume que las cosas pueden cambiar y que ningún proyecto está aislado del impacto de estos cambios. Para encarar de manera eficientemente cualquier cambio que se presente, el equipo de proyecto debe mantenerse ágil para reportar y gestionar los cambios, y todos los *stakeholders* deben participar de manera activa para obtener diferentes perspectivas de una posible solución.

- **Habilita la dirección de proyectos [CMMI / PMBoK]:** SOFTWARE EDUP presenta actividades y artefactos que dan visibilidad al a gestión de proyectos, de forma que el equipo ejecutor adquiere y adopta prácticas que les permitirán mitigar riesgos, y aumentar la probabilidad de lograr el tiempo y el alcance formulado.

8.4.2 Prácticas adoptadas de las metodologías ágiles

No todas las prácticas ágiles se adoptaron debido a las características de los proyectos abordados en esta tesis (Ver Tabla 5). Considerando estas estos elementos, se realizó la evaluación de las prácticas ágiles a adoptar para cada proyecto, las cuales se califican en la Tabla 6 presentada a continuación:

Tabla 6. Prácticas ágiles, momento de uso y tipos de proyectos (Elaboración Propia)

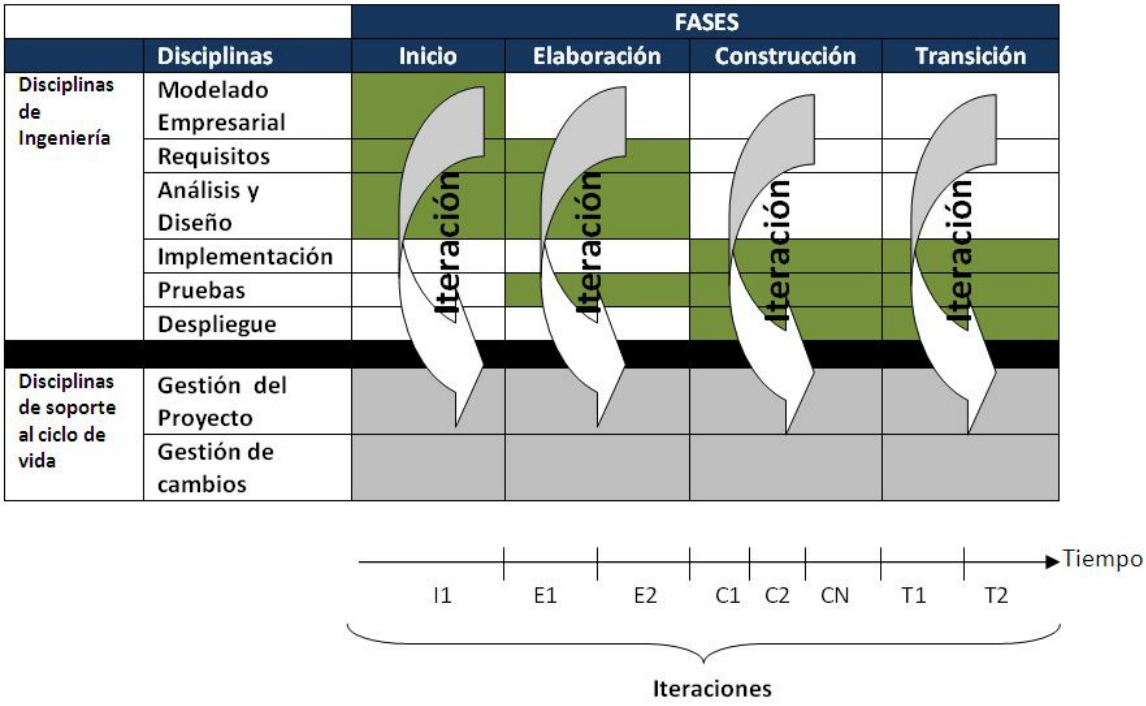
Momento	Prácticas Ágiles	Tipo de proyecto	Adopción de la práctica	Observaciones
Prácticas continuas	TDD – Test Driven Development: Desarrollo dirigido por pruebas.	Tipo 1 /Tipo 2	Completamente adoptada	Ninguna
	Integración continua	Tipo 1 /Tipo 2	No se adoptará,	Debido a las características de los proyectos, se considera una práctica de difícil implantación.
	Refactoring	Tipo 1 /Tipo 2	No se adoptará, se sugerirá	Debido a la característica del equipo de trabajo que inicialmente es inexperto, pero esta práctica es sugerida para el uso en equipos de trabajo avanzados.
	Colaboración entre todos los roles y presencia en un solo ambiente de trabajo	Tipo 1 /Tipo 2	No se adoptará.	Debido a la característica de los proyectos realizados, la mayoría de las veces no se presta para un escenario de estas características.
Diarias	Standup meeting	Tipo 1 /Tipo 2	Parcialmente adoptada	Se sugiere una modificación en la forma de realización, de modo que cuando el equipo de trabajo se reúna, sean las preguntas del Standup las que se realicen; estas son: • ¿Qué se realizó en la última reunión? • ¿Qué se planea hacer hasta la siguiente reunión? • ¿Existe algún impedimento que influya en su progreso normal?
	Test de Aceptación-	Tipo 1 /Tipo 2	No se adoptará,	Debido a la característica de los proyectos realizados, la mayoría de las veces no se presta para un escenario de estas características, pues al final del día no se pueden correr pruebas de aceptación con el cliente.
Iteración	Plan de iteración	Tipo 1	No se adoptará	No se considera necesario debido a la característica corta del proyecto, que puede

Momento	Prácticas Ágiles	Tipo de proyecto	Adopción de la práctica	Observaciones
				verse plasmada en el plan del proyecto
		Tipo 2	Completamente adoptada	Ninguna
	Revisión de la iteración	Tipo 1 /Tipo 2	Completamente adoptada	Las fases o iteraciones en las que son divididos los proyectos son revisados al final de cada una.
	Reunión retrospectiva	Tipo 1 /Tipo 2	Completamente adoptada	Ninguna
Release	Backlog: Pila de trabajo	Tipo 1 /Tipo 2	No se adoptará	No se considera debido a que el proyecto no tendrá varias liberaciones.
	Plan de release	Tipo 1 /Tipo 2	No se adoptará	No se considera debido a que el proyecto no tendrá varias liberaciones.
	Estimación por parte del equipo	Tipo 1 /Tipo 2	No se adoptará, se sugerirá	Debido a la característica del equipo de trabajo que inicialmente es inexperto, esta práctica es sugerida más no obligatoria.
Estrategia	Visión	Tipo 1 /Tipo 2	Completamente adoptada	Ninguna
	Objetivos	Tipo 1 /Tipo 2	Completamente adoptada	Ninguna
	Carta de Inicio	Tipo 1 /Tipo 2	No se adoptará	No aplica
	Aprobación de presupuesto	Tipo 1 /Tipo 2	No se adoptará	No aplica

8.5 Estructura del Proceso

La metodología SOFTWARE EDUP propone una estructura como la de RUP, la cual tiene dos dimensiones (ver **Figura 16**):

Figura 16. Estructura del framework SOFTWARE EDUP



- **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.

8.5.1 Estructura dinámica del framework

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 mismo.

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 (mismas fases correspondientes a las del Proceso Unificado, a RUP, Open UP, MeRinde y UPEDU). 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.

8.5.1.1 Fases

Inicio

El propósito general es establecer los objetivos para el ciclo de vida del producto.
Los objetivos específicos de esta fase son:

- Establecer el alcance 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.

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 soportará 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.

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 casos de uso (funcionalidades), servicios (en caso de ser requeridos), módulos, y/o subsistemas del proyecto.
- Realizar desarrollo por iteraciones, priorizando casos de uso, módulos y/o subsistemas más críticos para la arquitectura y el negocio.
- Realizar pruebas funcionales de cada uno de los casos de uso, servicios, 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.

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.

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

8.5.2 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 ¿Quién?;
- 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?

8.5.2.1 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 del framework (Ver **Figura 17**), (https://sites.google.com/site/softwareedup).

Figura 17. Disciplinas en el framework SOFTWARE EDUP

8.5.2.2 Artefactos

La metodología SOFTWARE EDUP propone un grupo de artefactos que pueden ser generados durante el proceso de desarrollo de software.

Es fundamental que antes del comienzo del proceso de desarrollo se decida cuáles son los artefactos que serán empleados a lo largo del ciclo de vida del desarrollo del proyecto, así como es recomendable que se defina entre las partes los artefactos que deben ser entregados.

Es recomendable que se utilicen la mayoría de los artefactos que se definen en el framework, principalmente si la magnitud del proyecto es grande. Mientras más artefactos detallen en profundidad los elementos de un sistema, mejor será la compresión del proyecto por parte del equipo de trabajo.

La lista de los artefactos que se definen en SOFTWARE EDUP se presenta agrupada por la disciplina a la que pertenecen, en la Tabla 7, la cual se muestra a continuación.

Tabla 7. Disciplinas vs. Artefactos para Proyectos Tipo 1 (Elaboración Propia)

Disciplina	Obligatoriedad	Artefacto
Modelado empresarial	No Aplica	
Requisitos	Requeridos	• Documento de Requisitos del Sistema (DRS) • Casos de Uso
	Opcionales	• Solicitudes de los stakeholders • Glosario • Prototipo de interfaz de usuario

Disciplina	Obligatoriedad	Artefacto
Análisis y Diseño	Requeridos	• Documento de Arquitectura de Software (DAS)[58]
	Opcionales	• Decisiones de arquitectura
Implementación	Requeridos	• Prueba Unitaria • Código fuente del componente • Registro de Pruebas Unitarias • Componente operacional (ejecutable o build)
Pruebas	Requeridos	• Ideas de prueba - casos de prueba - pruebas
Despliegue	Requeridos	• Sistema
	Opcionales	• Manual de instalación • Manual de usuario
Gestión del proyecto	Requeridos	• Visión del sistema • Plan de Proyecto[59] • Lista de elementos de trabajo
	Opcionales	• Acta de Reunión • Acta de Entrega • Registro de Revisión • Lista de riesgos • Gráfica del trabajo pendiente (Burndown chart)
Gestión de configuración y control del cambio	Opcionales	• Repositorio de versiones • Solicitud de cambio (CR)

[58] Para este artefacto existe una versión simplificada denominada “informal”, sugerida para proyectos tipo 1.

[59] Ibídem.

Tabla 8. Disciplinas vs. Artefactos para Proyectos Tipo 2 (Elaboración Propia)

Disciplina	Obligatoriedad	Artefacto
Modelado empresarial	Opcionales	• Modelo del proceso
Requisitos	Requeridos	• Documento de Requisitos del Sistema (DRS) • Modelo de casos de uso • Caso de Uso • Solicitudes de los stakeholders • Glosario • Prototipo de interfaz de usuario
Análisis y Diseño	Requeridos	• Documento de Arquitectura de Software (DAS)[60]
	Opcionales	• Decisiones de arquitectura
Implementación	Requeridos	• Prueba Unitaria • Código fuente del componente • Registro de Pruebas Unitarias • Componente operacional (ejecutable o build)
Pruebas	Requeridos	• Lista de Ideas de prueba • Casos de prueba
Despliegue	Requeridos	• Sistema • Manual de instalación • Manual de usuario
Gestión del proyecto	Requeridos	• Visión del sistema • Plan de Proyecto[61] • Plan de Iteración • Lista de elementos de trabajo • Acta de Reunión • Acta de Entrega • Registro de Revisión • Lista de riesgos • Informe de Avance • Registro de compromisos e incidentes
	Opcionales	• Gráfica del trabajo pendiente (Burndown chart)
Gestión de	Requeridos	• Repositorio de versiones

[60] Para este artefacto existe una versión simplificada denominada “informal”, sugerida para proyectos tipo 1.

[61] Igual que la nota anterior.

Disciplina	Obligatoriedad	Artefacto
configuración y control del cambio		
	Opcionales	• Solicitud de cambio (CR)

Para Proyectos Tipo 1 se tiene un total de:

- 12 artefactos obligatorios
- 13 artefactos opcionales

Para Proyectos Tipo 2 se tiene un total de:

- 27 artefactos obligatorios
- 3 artefactos opcionales

A continuación se presenta, en la Tabla 9 y Tabla 10, se presenta los artefactos distribuidos por cada una de las fases y disciplinas del framework y los correspondientes tipos de proyecto:

Tabla 9. Relación entre Fases – Disciplinas – Artefactos Obligatorios para Proyectos Tipo 1 (Elaboración Propia)

	Inicio	Elaboración	Construcción	Transición
Requisitos	• Documento de requisitos del sistema (DRS) • Casos de uso	• Documento de requisitos del sistema (DRS) • Casos de uso		
Análisis y diseño	• Documento de Arquitectura de Software (DAS)	• Documento de Arquitectura de Software (DAS)		
Implementación			• Prueba Unitaria • Código fuente del componente • Registro de Pruebas Unitarias • Componente operacional (ejecutable o build)	• Prueba Unitaria • Código fuente del componente • Registro de Pruebas Unitarias • Componente operacional (ejecutable o build)
Pruebas	• Ideas de prueba - casos de prueba - pruebas	• Ideas de prueba - casos de prueba - pruebas	• Ideas de prueba - casos de prueba - pruebas	• Ideas de prueba - casos de prueba - pruebas
Despliegue				• Sistema
Gestión de configuración y control del cambio				
Gestión de proyecto	• Visión del sistema • Plan del Proyecto • Lista de elementos de trabajo	• Lista de elementos de trabajo	• Lista de elementos de trabajo	• Lista de elementos de trabajo

Tabla 10. Relación entre Fases – Disciplinas – Artefactos Obligatorios para Proyectos Tipo 2 (Elaboración Propia)

	Inicio	Elaboración	Construcción	Transición
Modelado del negocio				
Requisitos	• Documento de Requisitos del Sistema (DRS) • Modelo de casos de uso • Caso de Uso • Solicitudes de los stakeholders • Glosario • Prototipo de interfaz de usuario	• Documento de Requisitos del Sistema (DRS) • Modelo de casos de uso • Caso de Uso • Solicitudes de los stakeholders • Glosario • Prototipo de interfaz de usuario		
Análisis y diseño	• Documento de Arquitectura de Software (DAS)	• Documento de Arquitectura de Software (DAS)		
Implementación			• Prueba Unitaria • Código fuente del componente • Registro de Pruebas Unitarias • Componente operacional (ejecutable o build)	• Prueba Unitaria • Código fuente del componente • Registro de Pruebas Unitarias • Componente operacional (ejecutable o build)
Pruebas	• Lista de Ideas de prueba	• Lista de Ideas de prueba	• Casos de prueba	• Casos de prueba
Despliegue				• Manual de usuario • Manual de instalación • Sistema
Gestión de	• Repositorio de versiones	• Repositorio de versiones	• Repositorio de	• Repositorio de versiones

	Inicio	Elaboración	Construcción	Transición
configuración			versiones	
Gestión de proyecto	• Visión del sistema • Plan de Proyecto[62] • Plan de Iteración • Lista de elementos de trabajo • Acta de Reunión • Registro de Revisión • Lista de riesgos • Informe de Avance Registro de compromisos e incidentes	• Plan de Iteración • Acta de Reunión • Registro de Revisión • Lista de riesgos • Informe de Avance • Registro de compromisos e incidentes	• Plan de Iteración • Acta de Reunión • Registro de Revisión • Lista de riesgos • Informe de Avance • Registro de compromisos e incidentes	• Plan de Iteración • Acta de Reunión • Registro de Revisión • Lista de riesgos • Informe de Avance • Registro de compromisos e incidentes • Acta de Entrega

La descripción de cada uno de los artefactos puede consultarse en la publicación de la metodología, la cual es un sitio Web estático y es parte integral de esta tesis.

[62] Ibídem.

8.5.2.3 Roles

Una de las razones principales de la adopción de esta metodología para el desarrollo de software consiste en la definición de las tareas que serán llevadas a cabo por los estudiantes que participan en un proyecto. En SOFTWARE EDUP un rol define las responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Los roles se encargan de la realización de tareas, las cuales generan artefactos.

La metodología SOFTWARE EDUP propone ocho (8) roles básicos:

- **Director de proyecto:** El director de proyecto aplica conocimientos, habilidades, herramientas y técnicas de gestión de proyectos a un amplio rango de tareas, para cumplir los requisitos y para proporcionar un resultado final en un proyecto concreto.
- **Analista:** Un analista se encarga de dirigir, coordinar y realizar la recopilación de requisitos y el modelado de casos de uso, esquematizando la funcionalidad del sistema y delimitando el alcance del mismo. Por ejemplo, define los actores y casos de uso y sus interacciones.
- **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:** Este rol es responsable de construir diferentes componentes del sistema, incluyendo comprender y adherirse a las decisiones de arquitectura, posiblemente realizar prototipos de la interfaz de usuario, y entonces implementar, hacer pruebas unitarias e integrar los componentes que son parte de la solución.
- ***Tester*:** Este rol es responsable de las actividades principales del esfuerzo de las pruebas. Estas actividades incluyen identificar, definir, ejecutar y dirigir las pruebas necesarias, como también obtener los resultados de las mismas.

- **Docente:** Este rol es el encargado de la orientación, seguimiento y control formal de los artefactos, que deben elaborarse a través del proceso de desarrollo de software.
- ***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:** Este rol caracteriza las tareas que puede efectuar cualquier miembro del equipo.

La definición completa de los roles con sus definiciones, relaciones, habilidades, propuesta de asignación y factores críticos de éxito, se encuentran en el sitio Web estático generado y que es parte integral de esta tesis y es uno de los productos de este framework.

En la Tabla 11 y Tabla 12, se presenta la matriz RACI (asignación de responsabilidades) de los entregables vs los roles en las que se presentan y su grado de responsabilidad frente al entregable, calificándolos así:

- **R – RESPONSABLE:** Ejecuta la tarea de construcción del entregable. Identifica la(s) función (es) que es asignada para ejecutar una actividad en particular. El grado de responsabilidad es definida por el "*Accountable*". Pueden haber múltiples Responsables para una actividad.
- **A – ACOUNTABLE:** Responsable de que se haga el entregable. Es el responsable final de la conclusión de la actividad y del entregable, y es quien tiene la capacidad de decir "sí" o "no". Debe haber una y sólo una "A" (un responsable) para una decisión, entregable o actividad. Esta responsabilidad no puede ser delegada.

- **C – CONSULTED:** Consultado antes de ejecutar. Identifica las funciones que deben ser “consultadas” antes que una decisión o actividad sea finalizada. Es una comunicación de ida y regreso.
- **I – INFORMED:** Informado. Identifica las funciones que deben ser informadas sobre la conclusión o resultado de la decisión o actividad. Es una comunicación de un sólo sentido.

.

Tabla 11. Matriz RACI para los Artefactos del Framework para Proyectos Tipo 1 (Elaboración Propia)

Disciplina	Artefacto	Director de proyecto	Analista	Arquitecto	Desarrollador	Tester	Docente	Stakeholder	Cualquier Rol
Requisitos	Documento de Requisitos del Sistema (DRS)	I	RA	CI	I	I	CI	I	I
	Casos de Uso	I	RA	CI	I	I	I	I	I
	Solicitudes de los stakeholders	I	RA	I	I	I	I	I	I
	Glosario	I	RA	I	I	I	I	I	I
	Prototipo de interfaz de usuario	I	RA	CI	I	I	CI	I	I
Análisis y Diseño	Documento de Arquitectura de Software (DAS)	C	I	RA	I	I	CI	I	I
	Decisiones de arquitectura	C	I	RA	I	I	CI	I	I
Implementación	Prueba Unitaria	I	I	CI	RA	I	I	I	I
	Código fuente del componente	I	I	CI	RA	I	I	I	I
	Registro de Pruebas Unitarias	I	I	CI	RA	I	I	I	I
	Código ejecutable del componente	I	I	CI	RA	I	I	I	I
Pruebas	Ideas de prueba - casos de prueba - pruebas	C	C	C	C	RA	C	I	I
Despliegue	Manual de instalación	I	I	I	R	RA	I	I	I
	Manual de usuario	I	I	I	R	RA	I	I	I
	Sistema	R	R	RA	R	R	I	I	I
Gestión del proyecto	Visión del sistema	RA	R	I	I	I	I	I	I
	Plan del proyecto	RA	I	C	I	I	I	I	I
	Lista de elementos de trabajo	RA	I	C	I	I	I	I	I
	Lista de riesgos	RA	I	C	I	I	I	I	I
	Acta de Reunión	RA	I	C	I	I	I	I	I
	Acta de Entrega	RA	I	C	I	I	I	I	I
	Registro de Revisión	RA	I	I	I	I	I	I	I
	Lista de Riesgos	RA	I	I	I	I	I	I	I
	Gráfica del Trabajo Pendiente (Burdown chart)	RA	I	C	I	I	I	I	I
Gestión de configuración y control del cambio	Repositorio de versiones	R	R	RA	R	R	I	I	I
	Solicitud de cambio (CR)	I	I	I	I	I	I	I	R

Tabla 12. Matriz RACI para los Artefactos del Framework para Proyectos Tipo 2 (Elaboración Propia)

Disciplina	Artefacto	Director de proyecto	Analista	Arquitecto	Desarrollador	Tester	Docente	Stakeholder	Cualquier Rol
Modelado empresarial	Modelo del Proceso	I	RA	CI	I	I	CI	I	I
Requisitos	Documento de Requisitos del Sistema (DRS)	I	RA	CI	I	I	CI	I	I
	Casos de Uso	I	RA	CI	I	I	I	I	I
	Modelo de casos de uso	I	RA	CI	I	I	I	I	I
	Solicitudes de los stakeholders	I	RA	I	I	I	I	I	I
	Glosario	I	RA	I	I	I	I	I	I
	Prototipo de interfaz de usuario	I	RA	CI	I	I	CI	I	I
Análisis y Diseño	Documento de Arquitectura de Software (DAS)	C	I	RA	I	I	CI	I	I
	Decisiones de arquitectura	C	I	RA	I	I	CI	I	I
Implementación	Prueba Unitaria	I	I	CI	RA	I	I	I	I
	Código fuente del componente	I	I	CI	RA	I	I	I	I
	Registro de Pruebas Unitarias	I	I	CI	RA	I	I	I	I
	Código ejecutable del componente	I	I	CI	RA	I	I	I	I
Pruebas	Lista de ideas de prueba	C	C	C	C	RA	C	I	I
	Casos de prueba	C	C	C	C	RA	C	I	I
Despliegue	Manual de instalación	I	I	I	R	RA	I	I	I
	Manual de usuario	I	I	I	R	RA	I	I	I
	Sistema	R	R	RA	R	R	I	I	I
Gestión del proyecto	Visión del sistema	RA	R	I	I	I	I	I	I
	Plan del proyecto	RA	I	C	I	I	I	I	I
	Plan de Iteración	RA	I	C	I	I	I	I	I
	Lista de elementos de trabajo	RA	I	C	I	I	I	I	I
	Lista de riesgos	RA	I	C	I	I	I	I	I
	Acta de Reunión	RA	I	C	I	I	I	I	I
	Acta de Entrega	RA	I	C	I	I	I	I	I
	Registro de Revisión	RA	I	I	I	I	I	I	I
	Lista de Riesgos	RA	I	I	I	I	I	I	I

Disciplina	Artefacto	Director de proyecto	Analista	Arquitecto	Desarrollador	Tester	Docente	Stakeholder	Cualquier Rol
	Registro de compromisos e incidentes	RA	I	I	I	I	I	I	I
	Gráfica del Trabajo Pendiente (Burdown chart)	RA	I	C	I	I	I	I	I
Gestión de configuración y control del cambio	Repositorio de versiones	R	R	RA	R	R	I	I	I
	Solicitud de cambio (CR)	I	I	I	I	I	I	I	R

8.5.2.4 Recomendaciones y consideraciones frente al framework

A continuación se presentan una serie de restricciones referentes al framework, que constituyen en principios de uso y advertencias, pues no se puede caer en que una metodología es útil para resolver todos los tipos de escenarios y ámbitos de proyectos:

- Es importante adaptar el framework, es posible que todos los artefactos sean útiles, pero también es posible que un pequeño subconjunto de ellos no, o aun más, se requiera de otros para complementarlo. En este caso se requiere que el director de proyecto o docente evalúe responsablemente la adición o supresión de entregables, recuerde este principio: “Los artefactos deben trabajar para usted y no usted para los artefactos, emplee los artefactos que considere indispensables para una correcta ejecución del proyecto”.
- SOFTWARE EDUP no enseñará a los estudiantes cómo convertirse en buenos desarrolladores. SOFTWARE EDUP guiará a los estudiantes en qué hacer, y es probable que en futuras versiones, hasta cómo hacer ciertas cosas, pero es requisito previo tener conocimientos básicos en desarrollo de software.

8.6 Ejemplos Propuestos

Con el objeto de mostrar cómo serían los artefactos generados en cada uno de los tipos de proyectos, se construyeron dos ejemplos, así:

- Una modificación a un sistema de retiro por cajero electrónico de dinero disponible en un CDT, correspondientes a los proyectos Tipo 1.
- Un sistema para la toma de pedidos en restaurantes, correspondientes a los proyectos Tipo 2.

Ambos proyectos hacen parte del proyecto de grado construido por López y Villa[63], los cuales apoyan esta tesis de maestría y se encuentran desplegados en el sitio web estático que contiene el framework.

8.7 Escenarios de Uso

De igual forma, su aplicabilidad estará enmarcada por las siguientes premisas:

- Entornos de desarrollo de software académico en Medellín y su área Metropolitana.
- Prácticas, proyectos, trabajos dirigidos de grado que involucren el desarrollo de software.

El framework no será aplicable, para:

- Entornos de desarrollo de software de misión crítica, en donde las pruebas, tanto funcionales como no funcionales, son de alta rigurosidad.

8.8 Validación del Framework

Para la validación se ejecutaron los siguientes pasos:

1. Convocatoria de profesores voluntarios
2. Presentación del framework y encuesta
3. Recolección y análisis de la encuesta

Para realizar la validación se convocaron a algunos profesores de pregrado y postgrado de la carrera Ingeniería de Sistemas de las siguientes universidades: Universidad Eafit, Universidad de Medellín y el Politécnico Jaime Isaza Cadavid.

---

[63] LÓPEZ, Mario y VILLA, Juan Camilo. Plantillas y artefactos personalización de RUP para proyectos académicos de desarrollo de software. Universidad Eafit 2011.

A estos profesores se les presentó el framework realizando una presentación de sensibilización[64] y uso del mismo.

Durante esta sensibilización se realizó especial énfasis en lo importante que es contar con un entorno de trabajo unificado para la construcción de software en ámbitos académicos (prácticas de pregrado, proyectos de materia, trabajos dirigidos de grado y tesis) y las consecuencias negativas para la vida profesional de los estudiantes ocasionadas por la falta de uso de un proceso de software adherido a sus prácticas académicas.

Paso seguido, se invitó a los profesores a realizar una evaluación y experimentación del framework basado en las siguientes preguntas:

1. Nombre Completo
2. Universidad en que trabaja
3. Materias que dicta en pregrado
4. Materias que dicta en posgrado
5. Su concepto acerca del framework
6. ¿Ha empleado el framework total o parcialmente, un proceso, un entregable, en alguna materia?
7. ¿Qué proceso, actividad, artefacto, o parte del framework empleó con sus estudiantes?
8. ¿En qué materia(s) lo empleó?
9. ¿Qué retroalimentación / reacción recibió de los estudiantes?
10. ¿Cuál es su concepto sobre el uso del framework bajo este escenario en que lo usó (y para el que fue creado)?

Las respuestas a estas preguntas se encuentran en el anexo 13.2.

[64] Ver anexo 13.1 Presentación de Sensibilización

8.9 Componentes físicos del framework

En la Tabla 13. Componentes Físicos del Framework Metodológico se enumeran y referencian los componentes físicos del framework y que hacen parte integral de este trabajo:

Tabla 13. Componentes Físicos del Framework Metodológico (elaboración propia)

Archivo	Descripción
configuracion-biblioteca -2012-09-30.zip	Archivo disponible en : https://sites.google.com/site/softwareedup/home/versiones. Construido con Eclipse Process Framework Composer Versión: 1.5.1.2 y plugins de español. Contiene la configuración de la metodología con sus diferentes contenidos del método y procesos. Para su uso se debe realizar la importación dentro de un ambiente de Eclipse Process Framework Composer Versión: 1.5.1.2.
plugins-2012-09-30.zip	Archivo disponible en : https://sites.google.com/site/softwareedup/home/versiones. Construido con Eclipse Process Framework Composer Versión: 1.5.1.2 y plugins de español. Contiene los diferentes plugins de método del framework metodológico. Para su uso se debe realizar la importación dentro de un ambiente de Eclipse Process Framework Composer Versión: 1.5.1.2.
2012-08-22 V01.zip	Archivo disponible en : https://sites.google.com/site/softwareedup/home/versiones. Construido con Eclipse Process Framework Composer Versión: 1.5.1.2 y plugins de español y publicado en html. Sitio navegable de la metodología, el cual contiene el framework

Archivo	Descripción
	metodológico desplegado, listo para el uso de la comunidad académica. Para su uso se debe descargar e iniciar la navegación en un explorador de internet (preferiblemente Mozilla Firefox Version 14.01.0, Internet Explorer 8 o superiores) , haciendo doble clic en "index.htm"

8.10 Licencia del framework

Todos los productos de esta tesis son de libre distribución y configuración, acogiéndose a Creative Commons V.3.0, significando que los usuarios serán libres y están autorizados para:

- copiar, distribuir y comunicar públicamente la obra
- hacer obras derivadas

Figura 18. Licencia creative commons[65]

Bajo las siguientes condiciones:

[65] Licencia Creative Commons [en línea]. (Citada: 16 de Junio. 2012) <http://creativecommons.org/licenses/by/3.0/deed.es >

- **Reconocimiento** — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).

Entendiendo que:

- **Renuncia** — Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor.
- **Dominio Público** — Donde el trabajo o cualquiera de sus elementos es de dominio público bajo las leyes aplicables, que estas leyes no afecten de alguna manera esta licencia.
- **Otros derechos** — Los derechos siguientes no quedan afectados por la licencia de ninguna manera:
    - Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.
    - Los derechos morales del autor;
    - Derechos que otras personas puedan tener, tanto en la propia obra o en la forma como se utiliza, tanto la publicidad o derechos de privacidad.
- **Aviso** — Al reutilizar o distribuir la obra, se debe dejar bien claro los términos de la licencia de esta obra.

8.11 Valor Agregado del Framework Frente a Iniciativas Existentes

A continuación en la Tabla 14 se presenta un cuadro comparativo de SOFTWARE EDUP versus las iniciativas estudiadas y evaluadas en este trabajo:

Tabla 14. Cuadro comparativo entre SOFTWARE EDUP y las iniciativas existentes (elaboración propia)

Aspecto a evaluar	Software EDUP	RUP	UPEDU	MeRinde
Enfoque estrictamente académico	Sí	No (Enfoque completamente industrial)	Sí	No (Enfoque completamente industrial)
Fuentes disponibles y liberadas para ser modificadas por la comunidad académica	Sí	No	No	No
Licenciamiento	Creative Commons 3.0	Propiedad IBM	Propiedad de École Polytechnique de Montréal	GNU FDL
Uso	Libre	Según licencia de cesión	Estrictamente académico	Libre
Disciplinas adoptadas de la ingeniería de software				
• Modelado Empresarial	Sí	Sí	No	Sí
• Gestión de Requisitos	Sí	Sí	Sí	Sí
• Análisis y Diseño	Sí	Sí	Sí	Sí
• Implementación	Sí	Sí	Sí	Sí
• Pruebas	Sí	Sí	Sí	Sí
• Despliegue	Sí	Sí	No	Sí
• Gestión de la configuración y control del cambio	Sí	Sí	Sí	Sí
• Gestión de Proyectos	Sí	Sí	Sí	Sí
• Gestión de Ambiente	No	Sí	No	Sí
Cumple con la especificación de SPEM 2.0	Sí	Sí	No	Sí
Cuenta con un sitio navegable en html	Sí	Sí	Sí	Sí
Cuenta con diferentes tipos de proyectos y personalizaciones de proyectos de acuerdo a las necesidades requeridas	Sí	Sí	No	No
Cuenta con roles y responsabilidades enfocados a las necesidades académicas	Sí	No	No	No
Cuenta con ejemplos que ilustra su utilización	Sí	Sí	Sí	No
Enfocado en proyectos pequeños que se trabajan en el ámbito académico	Sí	No	Sí	No
Pensado para equipos pequeños de al menos dos personas (donde una de ellas sería el docente)	Sí	No	No	No
Contiene prácticas ágiles	Sí	No	No	Si
Contiene plantillas disponibles para su uso inmediato	Sí	Sí	Parcialmente (estas deben ser compradas)	Sí (pero no todas las mencionadas)

Aspecto a evaluar	Software EDUP	RUP	UPEDU	MeRinde
Contiene ejemplos que muestren aplicabilidad del framework	Sí	Sí	Sí	No
Cuenta con una organización o empresa que lo soporte	No	Sí	Sí	Sí

9. CONCLUSIONES

- SOFTWARE EDUP es un framework metodológico completo, con prácticas y conceptos estandarizados útil en un ambiente académico, que servirá de guía para que los estudiantes en su proceso de desarrollo. El framework cuenta con elementos, pasos y actividades claramente desglosados y detallados, con ejemplos que ilustran su aplicación.

- SOFTWARE EDUP constituye un material de enseñanza de la aplicación de la Ingeniería de Software y su apoyo en los artefactos, permitiendo seleccionar los que más se adapten a los requerimientos de la materia que se dicte o proyecto que se ejecute basándose en los principios allí presentados.

- SOFTWARE EDUP permite la mantenibilidad del framework metodológico debido a que la herramienta en que se construyó (*Eclipse Process Framework*) es un framework vigente, en constante maduración y despliega la metodología en un estándar reconocido, SPEM 2.0, permitiendo una adecuada navegabilidad y documentación, no solo de procesos, actividades, tareas, roles y entregables, sino que permite adherir prácticas, ejemplos, plantillas, guías, artículos (*white papers*) entre otros, que guían el proceso de desarrollo.

- 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 y a desarrollos realizados en otras asignaturas.

- SOFTWARE EDUP es un framework que permite al docente contar con elementos metodológicos para guiar al alumno hacia la producción de

software, de la misma forma como se hace en la industria, pues se apoya en un repositorio metodológico enfocado a su ámbito de trabajo.

- SOFTWARE EDUP siguiere y usa prácticas ágiles que permiten al estudiante y al equipo del proyecto madurar en su forma de construir software.

- SOFTWARE EDUP es un framework que puede ser mantenido y evolucionado en el tiempo por la comunidad académica, debido a su adopción de estándares de la industria como SPEM, y basarse en marcos de trabajo validados y probados como RUP, CMMI y PMBoK.

- SOFTWARE EDUP es un framework que valora la calidad y la gestión del proyecto como valores claves y complementarios a la buena ingeniería en la construcción del producto software, sugiriendo y requiriendo prácticas, tales como pruebas unitarias y revisiones formales, que buscan acompañar el proceso de producción de software por parte del docente o líder del proyecto.

- SOFTWARE EDUP es una herramienta útil en el cierre de la brecha entre la academia y la industria al ser el compilador de estándares como RUP, CMMI y PMBoK, adaptado para los proyectos académicos.

- Con SOFTWARE EDUP, los equipos de trabajo integrados para construir desarrollos de software en las prácticas académicas, tesis de pregrado y proyectos de investigación, cuentan con un repositorio de acceso fácil y centralizado para consulta de las prácticas y procesos a emplear, y con una directriz estándar de documentación y esquematización del trabajo.

- El framework de SOFTWARE EDUP permite que los procesos de desarrollo bajo escenarios académicos queden documentados y puedan ser gestionados bajo un estándar al cual se puedan incorporar información de diferentes fuentes y formatos.

- Los equipos de trabajo que desarrollen software de tipo académico con SOFTWARE EDUP cuentan con una base de conocimientos actualizada en la cual pueden capacitarse en métodos y mejores prácticas (requiriendo un entrenamiento y contextualización previa antes de su uso).

- Debido a las características de su licenciamiento Creative Commons 3.0, SOFTWARE EDUP se convierte en un framework que puede ser extendido, madurado y difundido libremente sin infringir derechos de propiedad intelectual, y proporcionando así material de enseñanza asequible a toda la comunidad académica.

- SOFTWARE EDUP se constituye en una framework válido para esquematizar y difundir el proceso software en escenarios académicos debido a sus características de licenciamiento, orientación al uso académico, plantillas y ejemplos, desarrollo en equipos pequeños, características de extensibilidad para su personalización, superando en el ámbito académico a MeRinde, UPEDU y RUP.

10. TRABAJOS FUTUROS

- A pesar de que se realizó una validación con algunos profesores de cursos relacionados con Ingeniería de software de distintas instituciones educativas, se deben realizar pilotos de implantación en entornos reales para determinar elementos a mejorar, adicionar y/o suprimir, de forma que se logre mejorar el impacto y uso tanto en estudiantes como en los docentes, al igual que su difusión y uso.

- Se deben establecer mecanismos de maduración del framework, de manera que haya un tronco (*trunk* – proyecto matriz) que sea administrado, mantenido y evolucionado por un grupo encargado de madurarlo y modificarlo ya sea de forma única, como por universidad, según las necesidades, tanto académicas como de la industria.

- Se debe realizar un trabajo de identificación de más tipos de proyectos/procesos (diferente duración, especialidad, tipo de gestión) a caracterizar y documentar de forma que se creen nuevas instancias y particularizaciones a partir de la base creada del framework.

- Se sugiere en trabajos futuros apoyar el uso y difusión de framework con la construcción de un tutorial que ayude al entendimiento, uso por parte de los estudiantes y profesores que lo emplearán.

- Se sugiere para los departamentos de ingeniería de sistemas dentro de las universidades en las cuales se imparta el pregrado de de ingeniería de sistemas (o similares) y tomen la decisión de adoptar este framework, realicen previamente un mapeo de los procesos y actividades que tienen aplicabilidad

para cada matera de forma que el estudiante y el cuerpo docente perciban un uso coordinado de SOFTWARE EDUP y del proceso mismo.

- Se pueden realizar herramientas tipo CMS (*Content Management System*) o ECM (*Enterprise Content Management*) que soporten la publicación y gestión de los entregables generados a lo largo del ciclo de vida, apoyadas en las definiciones realizadas en SOFTWARE EDUP, logrando una integralidad entre lo requerido y lo realizado, de igual forma, dando la opción de valorar los entregables por parte del docente encargado de la evaluación y evolución del proyecto.

- El framework se puede enriquecer con más ejemplos de implementación, con diferentes casos, proyectos, áreas de conocimiento, tecnologías, de forma que sirvan de modelo a los estudiantes en sus entregas.

11. DATOS DE CONTACTO

JORGE HERNÁN ABAD LONDOÑO

Email: jorge.abad@gmail.com

Teléfono Móvil: (57) 300 653 2725

12. BIBLIOGRAFÍA

Abran, A., and Moore, J. W., Guide to the Software Engineering Body of Knowledge: 2004 Edition - SWEBOK, IEEE , 2005.

AMBYSOFT. Best practices for software development [en línea]. Citada: 17 de Junio de 2012) <http://www.ambysoft.com/>

Beck, Kent , et al. (2001). "Manifesto for Agile Software Development" . Agile Alliance . (Citada: 21 de Junio de 2012) <http://agilemanifesto.org/iso/es/>

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.

Bernhart, M., Grechenig, T., Hetzl, J., & Zuser, W. (2006). Dimensions of software engineering course design. Proceeding of the 28th international conference on Software engineering - ICSE '06, 667. doi:10.1145/1134285.1134387

CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN. Metodología de la red nacional de integración y desarrollo de software libre (MeRinde): Guía detallada. (Citada: 16 de Junio. 2012) < http://merinde.net . >

Chrissis, Mary Beth & Konrad, Mike & Shrum, Sandy; (2007). CMMI: Guidelines for Process Integration and Product Improvement. Second Edition. Pearson

Education, Inc., Boston, MA, USA, 676 p.

COLLARIS, Remi-Armand y DEKKER, Eef. Comparing Methods. 2011.[en línea] (Citada: 17 de Junio de 2012). <http://blog.scrumup.com/2011/05/comparing-methods.html>

ECLIPSE. Eclipse Process Framework - Getting Started. [en línea] (Citada: 17 de Junio de 2012). http://www.eclipse.org/epf/general/getting_started.php

ECLIPSE. Eclipse Process Framework [en línea]. (Citada: 17 de Junio de 2012) <http://www.eclipse.org/epf/>

ECLIPSE. Introduction to OpenUP (Open Unified Process) [en línea]. (Citada: 16 de Junio. 2012) <http://www.eclipse.org/epf/general/OpenUP.pdf>

ÉCOLE POLYTECHNIQUE DE MONTRÉAL. Unified Process for Education (UPEDU) [en línea]. (Citada: 16 de Junio. 2012) <http://www.upedu.org/>

Guide to the Software Engineering Body of Knowledge (SWEBOK) (Citada: 16 de Junio. 2012) http://www.computer.org/portal/web/swebok .

Haeberer, A. M.; P. A. S. Veloso, G. Baum (1988). Formalización del proceso de desarrollo de software, Ed. preliminar edición (en Español), Buenos Aires: Kapelusz. ISBN 950–13–9880–3.

HAUMER, Peter. Eclipse Process Framework Composer. Part 1: Key Concepts. 2007. [en línea] (Citada: 17 de Junio de 2012). <http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf>

IBM. Arquitectura y Modelado Orientado a Servicios (SOMA). Biblioteca de métodos de la herramienta RATIONAL METHOD COMPOSER sitio web de la herramienta – [en línea]. (Citada: 16 de Junio. 2012) <http://www–01.ibm.com/software/awdtools/rmc/>

IBM. IBM Rational Method Composer [en línea]. (Citada: 16 de Junio. 2012) <http://www–01.ibm.com/software/awdtools/rmc/>

IBM. IBM Rational Unified Process (RUP) [en línea]. (Citada: 16 de Junio. 2012) <http://www–01.ibm.com/software/awdtools/rup/>

IBM. IBM Rational Unified Process (RUP) para proyectos pequeños [en línea]. (Citada: 16 de Junio. 2012) <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]. (Citada: 16 de Junio. 2012) <http://www–01.ibm.com/software/awdtools/rmc/library/>

IBM. Using RUP to manage small projects and teams [en línea]. (Citada: 16 de Junio. 2012) <http://www.ibm.com/developerworks/rational/library/jul05/kohrell/>

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

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

Kruchten, P. (2011). Experience teaching software project management in both industrial and academic settings. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 199–208. doi:10.1109/CSEET.2011.5876087

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

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

Valencia. 2003.

Licencia Creative Commons [en línea]. (Citada: 16 de Junio. 2012) <http://creativecommons.org/licenses/by/3.0/deed.es >

LIDER DE PROYECTO. CMMI [en línea]. (Citada: 16 de Junio. 2012) < http://www.liderdeproyecto.com/manual/cmmi_y_la_administracion_de_proyectos.html>

LÓPEZ, Mario y VILLA, Juan Camilo. Plantillas y artefactos personalización de RUP para proyectos académicos de desarrollo de software. Universidad Eafit 2012

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

Mchale, J., & Chick, T. A. (2010). Implementation Guidance for the Accelerated Improvement Method (AIM), (December).

Método OpenUP para pequeños proyectos. Biblioteca de métodos de la herramienta RATIONAL METHOD COMPOSER sitio web de la herramienta – (Citada: 16 de Junio. 2012) http://www–01.ibm.com/software/awdtools/rmc/ –

Método RUP para grandes proyectos. Biblioteca de métodos de la herramienta

RATIONAL METHOD COMPOSER sitio web de la herramienta – (Citada: 16 de Junio. 2012) http://www–01.ibm.com/software/awdtools/rmc/

Método RUP para pequeños proyectos. Biblioteca de métodos de la herramienta RATIONAL METHOD COMPOSER sitio web de la herramienta –(Citada: 16 de Junio. 2012) http://www–01.ibm.com/software/awdtools/rmc/ –

Metodología de la Red Nacional de Integración y Desarrollo de Software Libre (MeRinde) (Citada: 16 de Junio. 2012) http://merinde.rinde.gob.ve/index.php.

NAVEGAPOLIS. Sinopsis de los modelos CMM y CMMI [en línea]. (Citada: 16 de Junio. 2012) < http://www.navegapolis.net/index.php?option=com_content&task=view&id=330&Itemid=84>

OBJECT MANAGEMENT GROUP (OMG). Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0. [en línea]. (Citada: 17 de Junio de 2012) <http://www.omg.org/spec/SPEM/2.0/PDF>

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

Pierre N. Robillard, Philippe Kruchten, Patrick d'Astous, "YOOPEEDOO (UPEDU): A Process for Teaching Software Process," cseet, pp.18, 14th Conference on

Software Engineering Education and Training, 2001

PROJECT MANAGEMENT INSTITUTE. PMBoK (Project Management Body of Knowledge), version 4 [en línea]. (Citada: 16 de Junio. 2012) <http://www.pmi.org/PMBOK–Guide–and–Standards.aspx>

REAL ACADEMIA ESPAÑOLA. Diccionario de la Lengua Española - Vigésima Segunda Edición. [en línea]. (Citada: 4 de Julio. 2012) <http://rae.es/rae.html>

Rising, L., Janoff, N.S. (2000). *The Scrum Software Development Process for Small Teams* Retrieved –(Citada: 16 de Junio. 2012) http://members.cox.net/risingl1/Articles/IEEEScrum.pdf

RUIZ, Francisco y VERDUGO, Javier. Guía de Uso de SPEM 2 con EPF Composer Versión 3.0. [en línea]. (Citada: 17 de Junio de 2012) < http://alarcos.esi.uclm.es/ipsw/doc/guia-spem2&epf_v30.pdf>. Universidad de Castilla-La Mancha. 2008.

RUIZ, Francisco. Ingeniería de Procesos Software. [en línea]. (Citada: 17 de Junio de 2012) < http://alarcos.inf-cr.uclm.es/doc/psgc/doc/psgc0809_parte2b_ips.pdf>.

SCRUM. (Citada: 16 de Junio. 2012) http://es.wikipedia.org/wiki/Scrum.

SCRUM MANAGER. Glosario [en línea]. (Citada: 5 de Noviembre de 2012) < http://www.scrummanager.net/oks/mod/glossary/view.php?id=158 >

Software Engineering Institute. CMMI for Development SCAMPI Class A Appraisal Results, 2011 Update . March 2012. [en línea]. (Citada: 22 de Octubre de 2012) < www.sei.cmu.edu/cmmi/casestudies/profiles/pdfs/upload/2012MarCMMI.pdf >

SOFTWARE ENGINEERING INSTITUTE. CMMI® for Development, Version 1.3 CMMI-DEV, V1.3, (November). [en línea]. (Citada: 16 de Junio. 2012) < http://www.sei.cmu.edu/reports/10tr033.pdf >

SOFTWARE ENGINEERING INSTITUTE. CMMI overview [en línea]. (Citada: 16 de Junio. 2012) <http://www.sei.cmu.edu/CMMI/>

SOLUCIONES INFORMÁTICAS GLOBALES. CMMI – Capability Maturity Model Integration [en línea]. <http://www.globales.es/imagen/internet/Informaci%C3%B3n%20General%20CMMI.pdf>

TABARES, Luis. Personalización de RUP para proyectos académicos de desarrollo de software. Universidad Eafit. 2011.

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

Tremblay, G., Malenfant, B., Salah, A., & Zentilli, P. (2007). Introducing Students to Professional Software Construction : A " Software Construction and Maintenance " Course and its Maintenance Corpus, 176–180.

UPEDU – http://www.upedu.org/upedu/. École Polytechnique de Montréal.

Van der Duim, L., Andersson, J., & Sinnema, M. (2007). Good Practices for Educational Software Engineering Projects. 29th International Conference on Software Engineering (ICSE'07), 698–707. doi:10.1109/ICSE.2007.40

VEGA, Jesús. Subóptimo. 2010. [en línea] (Citada: 19 de Junio. 2012) <http://archivo.expansionyempleo.com/2010/09/13/opinion/1284370257.html

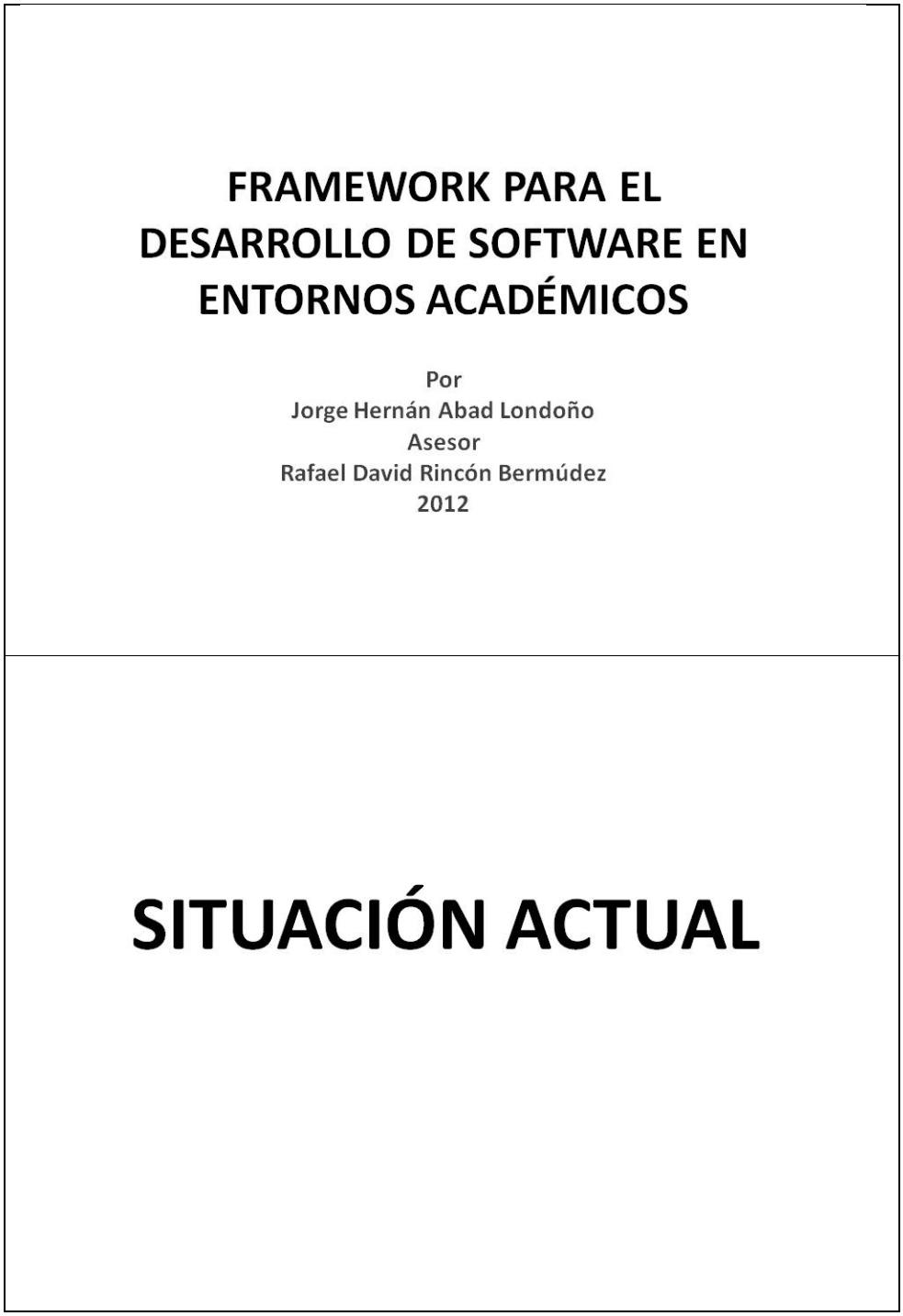
VERSIONONE. The agile checklist [en línea]. (Citada: 21 de Junio de 2012) <http://pm.versionone.com/AgileChecklist.html>. Ver también <http://it-talents.org/agile-development-rhythms-meeting-checklist>

VERSIONONE. Understand agile development methods with the agile development poster. [en línea]. (Citada: 21 de Junio de 2012) <http://pm.versionone.com/AgilePoster.htmlhttp://pm.versionone.com/AgilePoster_ThankYou.html>

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

13. ANEXOS

13.1 Presentación de Sensibilización



¿Cómo asignamos y recibimos actualmente los profesores prácticas y trabajos de desarrollo?

¿Donde estamos los profesores?

- No se trabaja de forma estandarizada (en la misma o diferentes asignaturas)
- En materias donde existe desarrollo de software o componentes, los entregables esperados son diferentes.
- En muchos casos lo único que interesa resultado y no el proceso de obtenerlo.
- Existen esfuerzos aislados de diferentes profesores para inculcar buenas prácticas de desarrollo de software pero no es un trabajo coordinado por todo el Departamento/Carrera/Equipo.
- Existen casos de profesores que enseñan solo su mundo sin mirar donde esta la industria y el mercado local.

¿Cómo construyen los estudiantes el software de las prácticas los estudiantes?

¿Dónde están los estudiantes?

- Construyen software de la forma que creen o les dicen sus compañeros.
- Emplean los artefactos y prácticas que les "obligan" unos profesores, los otros no.
- Aprenden a desarrollar software por "voz a voz" y no saben cual es la mejor forma de hacerlo.
- Les importa solo el resultado, sin percibir lo importante de tener un buen proceso.
- El proceso de software no se adhiere a su "ADN de estudiantes"

CONSECUENCIAS

Consecuencias

- No se cuenta con un esquema formal de trabajo coherente que sea visualizado por los alumnos, con el cual identifiquen cómo asumir las mejores prácticas para llevar a cabo un proyecto de software exitoso.
- No existe uniformidad en los entregables producidos, tanto en la documentación funcional como en la técnica.
- Carencia de documentación de soporte al ciclo de vida en los proyectos académicos de software.
- Para las materias diferentes a Ingeniería de Software no hay un esquema de control en el cual se asegure que se están adoptando las mejores prácticas de la industria.
- La evaluación de los trabajos académicos se concentra fundamentalmente en los resultados (producto de software) dejando de lado el proceso realizado y los roles involucrados (la nota debería tener un componente de proceso: repositorio, documentación, pruebas unitarias, pruebas funcionales, etc)

Consecuencias

- Falta de uniformidad de criterios para construir y gestionar proyectos de software en las asignaturas que los incluyen.
- los frameworks metodológicos son conocidos por los estudiantes como una referencia de cognitiva, mas estos no son concebidos como una herramienta clave del desempeño y éxito profesional.

Consecuencias… para la industria..

- Existe poca o pobre adopción de las mejores prácticas de la industria de software por parte de los alumnos egresados.
- Largos periodos de entrenamiento de los estudiantes en la industria debido a su desconocimiento de las buenas prácticas.
- Emprendimientos realizado por los estudiantes están viciados por errores metodológicos que influencian directamente el fracaso de la iniciativa

COMENZANDO / COMENCEMOS A SOLUCIONAR

UN FRAMEWORK PARA EL PROCESO DE DESARROLLO DE SOFTWARE EN UNIVERSIDADES

Nombre:

SOFTWARE EDUP

Características del Frawework

- Basado Estándares
  - CMMI,
  - RUP,
  - OpenUP,
  - Scrum,
  - PMBoK
  - SPEM 2.0,
- Soportado por una plataforma que evoluciona y de amplio uso académico e industria (EPF – Eclipse Process Framework)
- Evolutivo y Adaptable
- Alineado con la industria y las mejores prácticas de ingeniería de software
- Un paso a seguir por otras iniciativas similares

Beneficios esperados

Beneficios esperados

- Lograr una absorción del proceso como la mejor manera de encarar los proyectos de software.
- Un proceso “quemado” en “ADN de los estudiantes” por el uso coherente y continuado de los profesores.
- Buenas prácticas de desarrollo empleadas por el estudiante en todas las prácticas de la carrera.
- Una documentación uniforme generada tanto horizontal como verticalmente en la carrera.
- Una industria con menos tiempos de capacitación.
- Un empresarismo más fuerte y con menos ciclos de certificación en estándares de la industria

Escenarios de Uso

Escenarios de Uso

- Su aplicabilidad estará enmarcada por las siguientes premisas:
  - Entornos de desarrollo de software académico en Medellín y su área Metropolitana.
  - Prácticas, proyectos, trabajos dirigidos de grado que involucren el desarrollo de software.
- El framework no será aplicable para:
  - Entornos de desarrollo de software de misión crítica, en donde las pruebas, tanto funcionales como no funcionales, son de alta rigurosidad.

Donde descargarlo

http://bit.ly/software_edup

(versiones -> descargar)

Validación

http://bit.ly/software_edup_val

VALIDACIÓN SOFTWARE EDUP

Encuesta de validación de Software EDUP
Recuerde descargarlo del sitio
- https://sites.google.com/site/softwareedup/
- https://sites.google.com/site/softwareedup/home/versiones (link descargar)

--- LA ENCUESTA ESTARÁ ABIERTA HASTA EL 27 DE AGOSTO DE 2012 ---
* Required

1. Nombre Completo *

2. Universidad en que trabaja *

Preguntas

1. Nombre Completo
2. Universidad en que trabaja
3. Materias que dicta en pregrado
4. Materias que dicta en posgrado
5. Su concepto acerca del framework
6. **¿Ha empleado el framework, un proceso, un entregable, en alguna materia?**
7. ¿Qué proceso, actividad, artefacto, o parte del framework empleó con sus estudiantes?
8. ¿En que materia(s) lo empleó?
9. ¿Qué retroalimentación / reacción recibió de los estudiantes?
10. ¿Cuál es su concepto sobre el uso del framework bajo este escenario en que lo uso (y para el que fue creado)?

13.2 Resultado de las Encuestas

1. Nombre Completo
Juan Francisco Cardona McCormick
2. Universidad en que trabaja
Universidad EAFIT
3. Materias que dicta en pregrado
Fundamentos de programación Lenguajes de programación Lenguajes formales y compiladores Organización de computadores Sistemas operativos
4. Materias que dicta en posgrado
No aplica.

5. Su concepto acerca del framework
Es un conjunto de prácticas y conceptos estandarizados, como son ambiente de trabajo para el desarrollo de software.
6. ¿Ha empleado el framework, un proceso, un entregable, en alguna materia?
No.
7. ¿Qué proceso, actividad, artefacto, o parte del framework empleó con sus estudiantes?
No aplica.
8. ¿En qué materia(s) lo empleó?
No aplica.
9. ¿Qué retroalimentación / reacción recibió de los estudiantes?
No aplica.
10. ¿Cuál es su concepto sobre el uso del framework bajo este escenario en que lo usó (y para el que fue creado)?
No aplica.

1. Nombre Completo
Jorge Hernán Suaza Jiménez
2. Universidad en que trabaja
POLITÉCNICO COLOMBIANO JAIME ISAZA CADAVID
3. Materias que dicta en pregrado
DESARROLLO DEL PENSAMIENTO ANALÍTICO Y SISTEMICO CICLO DE VIDA DEL SOFTWARE
4. Materias que dicta en posgrado
No aplica
5. Su concepto acerca del framework
Es un framework muy completo donde se evidencia claramente todos los pasos, actividades, roles, procedimientos en el desarrollo de un sistema de información.

Creo que servirá mucho para que los estudiantes estandaricen todas las actividades y poder tener una homogeneidad en todos los procesos de desarrollo. Lastimosamente aun no lo he utilizado con los estudiantes, pero creo que es un muy buen material de trabajo que me interesa implementar en mis clases.
6. ¿Ha empleado el framework, un proceso, un entregable, en alguna materia?
No.
7. ¿Qué proceso, actividad, artefacto, o parte del framework empleó con sus estudiantes?
No aplica.
8. ¿En qué materia(s) lo empleó?
No aplica.
9. ¿Qué retroalimentación / reacción recibió de los estudiantes?
No aplica.
10. ¿Cuál es su concepto sobre el uso del framework bajo este escenario en que lo usó (y para el que fue creado)?
No aplica.

1. Nombre Completo
Juan Camilo Marín.
2. Universidad en que trabaja
Trabaje en EAFIT en el CINF año y medio y adicionalmente fui docente del Centro de Educación Continua y de la Especialización en Desarrollo de Software dictando el curso Computación Móvil.
3. Materias que dicta en pregrado
No aplica.
4. Materias que dicta en posgrado

Computación Móvil en la Especialización en Desarrollo de Software.
5. Su concepto acerca del framework
En un comienzo me confundí asociando el término framework a un entorno de desarrollo que provee a los desarrolladores con un api que permite elaborar un software determinado. Seguidamente entendí el proceso y el modelo para la definición de sistemas que se piensan construir siguiendo esta metodología. El proyecto tipo 2, completo, está muy bien detallado, tiene los entregables y los procesos claramente desglosados. Pero en entornos más reales y más acordes con el estado actual del desarrollo de software, creo que el modelo básico es de más utilidad. Me parece que la herramienta que usaron para documentar el framework es un punto super importante, ya que permite fácilmente recorrerlo e ir profundizando sobre los tópicos más detenidamente y con una visión global de los objetivos que la metodología plantea. Me parece importante que se contemplen modelos más acordes con los requerimientos actuales de la industria, permitiendo darle una orientación más ágil al desarrollo de los proyectos, igualmente es importante aclarar que se extraen procesos como refactoring y pruebas de unidad de metodologías agiles y se incluyen en esta metodología.
6. ¿Ha empleado el framework, un proceso, un entregable, en alguna materia?
No.
7. ¿Qué proceso, actividad, artefacto, o parte del framework empleó con sus estudiantes?
No aplica.
8. ¿En qué materia(s) lo empleó?

No aplica.
9. ¿Qué retroalimentación / reacción recibió de los estudiantes?
No aplica.
10. ¿Cuál es su concepto sobre el uso del framework bajo este escenario en que lo usó (y para el que fue creado)?
No aplica.

1. Nombre Completo
María Clara Gómez Álvarez
2. Universidad en que trabaja
Universidad de Medellín
3. Materias que dicta en pregrado
Ingeniería de Software I Sistemas de Información y organizaciones I
4. Materias que dicta en posgrado
Ingeniería de Software I y II Tópicos avanzados en ingeniería de requisitos
5. Su concepto acerca del framework
Me parece muy relevante para la incorporación de RUP en procesos educativos y en proyectos de desarrollo de software
6. ¿Ha empleado el framework, un proceso, un entregable, en alguna materia?
Sí.
7. ¿Qué proceso, actividad, artefacto, o parte del framework empleó con sus estudiantes?
Les solicitó a los estudiantes que navegaran el framework y me indicaran las ventajas y desventajas.
8. ¿En qué materia(s) lo empleó?
Ingeniería de Software I.

9. ¿Qué retroalimentación / reacción recibió de los estudiantes?
- Herramienta útil para la aplicación de RUP - La navegabilidad la consideraron un poco plana - No es amigable para personas que no conocen RUP - Muy ilustrativas las plantillas y los ejemplos
10. ¿Cuál es su concepto sobre el uso del framework bajo este escenario en que lo usó (y para el que fue creado)?
Lo considero muy pertinente para apoyar el tema de mejores prácticas y metodologías en la enseñanza de la Ingeniería de Software.

1. Nombre Completo
ADRIANA XIOMARA REYES GAMBAO
2. Universidad en que trabaja
POLITECNICO COLOMBIANO JAIME ISAZA CADAVID
3. Materias que dicta en pregrado
TALLER DE INGENIERIA DE SOFTWARE Y HERRAMIENTAS DE DESARROLLO
4. Materias que dicta en posgrado
No aplica.
5. Su concepto acerca del framework
Es bueno, reúne buenas prácticas del desarrollo del software, las plantillas son apropiadas aunque considero algunas se pueden presentar en forma más didáctica, porque está muy enfocado a RUP, que es un poco pesado en la documentación. Pero en términos generales, me parece apropiado para su utilización en el ámbito académico para la enseñanza y seguimiento de los proyectos de desarrollo de software.
6. ¿Ha empleado el framework, un proceso, un entregable, en alguna materia?
Si.

7. ¿Qué proceso, actividad, artefacto, o parte del framework empleó con sus estudiantes?
EL DOCUMENTO DE LOS REQUISITOS DEL SISTEMA.
8. ¿En qué materia(s) lo empleó?
TALLER DE INGENIERÍA DE SOFTWARE
9. ¿Qué retroalimentación / reacción recibió de los estudiantes?
LA PLANTILLA ERA CLARA Y FACIL DE DILIGENCIAR
10. ¿Cuál es su concepto sobre el uso del framework bajo este escenario en que lo usó (y para el que fue creado)?
Es apropiado para el ámbito académico, considero que sería bueno organizarlo por sesiones; la sesión del Gerente del proyecto para que acceda a lo que corresponde a la gerencia y sesiones para el equipo de desarrollo, para que ingresen a lo que corresponde a la metodología de desarrollo como tal, para evitar confusiones, por parte de los alumnos al acceder a las fases.