**Universidad del Cauca**
**Facultad de Ingeniería Electrónica y Telecomunicaciones (FIET)**
**Grupo de Tecnologías de Información**
**Línea de Objetos y Agentes**
**Línea de Ingeniería del Software**

# PROYECTO SIMEP-SW[1]
# Trabajo de Investigación: "Hacia una Línea de Procesos Ágiles Agile SPsL"
# Versión 1.0

**Autores:** Julio Ariel Hurtado – Departamento de Sistemas
Universidad del Cauca – Colombia
PhD Cecilia Bastarrica – Departamento de Ciencias de la Computación
Universidad de Chile - Chile

**Fecha**: 8 de mayo de 2005.
**Id. Doc. :** SIMEP-SW-O&A-RT-11-V0.9
**Tarea No:**
**Tipo:** Trabajo de Investigación
**Estado:** Borrador
**Disponibilidad:** Protegido


[1] El proyecto SIMEP-SW(Sistema Integral de mejoramiento de los procesos de desarrollo de software en Colombia) es financiado por la Universidad del Cauca, Colciencias y SITIS Ltda. bajo el contrato 421 de 2003 Colciencias-Unicauca

# Historial de Revisión

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 08-05-2005 | <0.3> | Primera auto revisión del documento, la introducción fue totalmente reemplazada | Julio Ariel Hurtado Alegría |
| 09-05-2005 | <0.55> | Primera reunión física de trabajo con Cecilia Bastarrica. | Cecilia Bastarrica, Julio Ariel Hurtado. |
| | | | |
| | | | |

# Resumen

Este documento presenta la de trabajo de investigación a realizarse entre mayo 9 a Junio 3 de 2005 en el marco de la estancia de investigación de Julio Ariel Hurtado Alegría en el Departamento de Ciencias de la Computación(DCC) de la Universidad de Chile, con la tutoría de la Dra. Cecilia Bastarrica.

# Tabla de Contenido

# Trabajo de Investigación “Hacia una Línea de Procesos Ágiles - Agile SPsL”

## 1. Introducción

Teniendo como premisa que las empresas de desarrollo de software de Latinoamérica deben implementar proyectos de mejoramiento de procesos de desarrollo y de calidad, existen en Latinoamérica diferentes esfuerzos que propenden por el fortalecimiento de la industria de software de cada país. Ese esfuerzo ha ido enfocado hacia la mejora de los procesos de desarrollo, de tal manera que les permita a estas empresas incrementar su competitividad. La certificación de calidad del proceso de desarrollo de software y de los productos de él derivados es un paso que tarde o temprano las empresas productoras de software deben dar como respuesta a dos situaciones: la primera, por imagen, para incursionar y mantenerse en un mercado global; la segunda, por necesidad, para poder hacer de sus proyectos unidades administrativas eficientes y eficaces.

La mayoría de estos esfuerzos, están enfocados a trasladar los requisitos que imponen los modelos como el CMMI [4] e ISO [7] a empresas típicas en Latinoamérica: las micro, pequeña y mediana empresa de software. Los más representativos de estos esfuerzos son mps Br – Mejoramiento del Proceso de Software [8] en Brasil y en México, se ha desarrollado el modelo MoProSoft – Modelo de Procesos para la Industria de Software en México [9] que tiene como propósito fomentar la estandarización de su operación a través de la incorporación de las mejores prácticas en gestión e ingeniería de software. Sin embargo, este tipo de empresas necesita, además de conocer los requisitos que estos modelos brindan, esfuerzos orientados a la implementación de estos requisitos dentro o fuera del marco de un proyecto de mejora.

En el marco del proyecto SIMEP-SW se ha definido una estrategia de mejora, la cual intenta cubrir dos esfuerzos: el de alivianar requisitos y guiar en el proceso de mejora, así como el de generar un conjunto de recomendaciones prácticas para la implementación de los requisitos del proceso software.

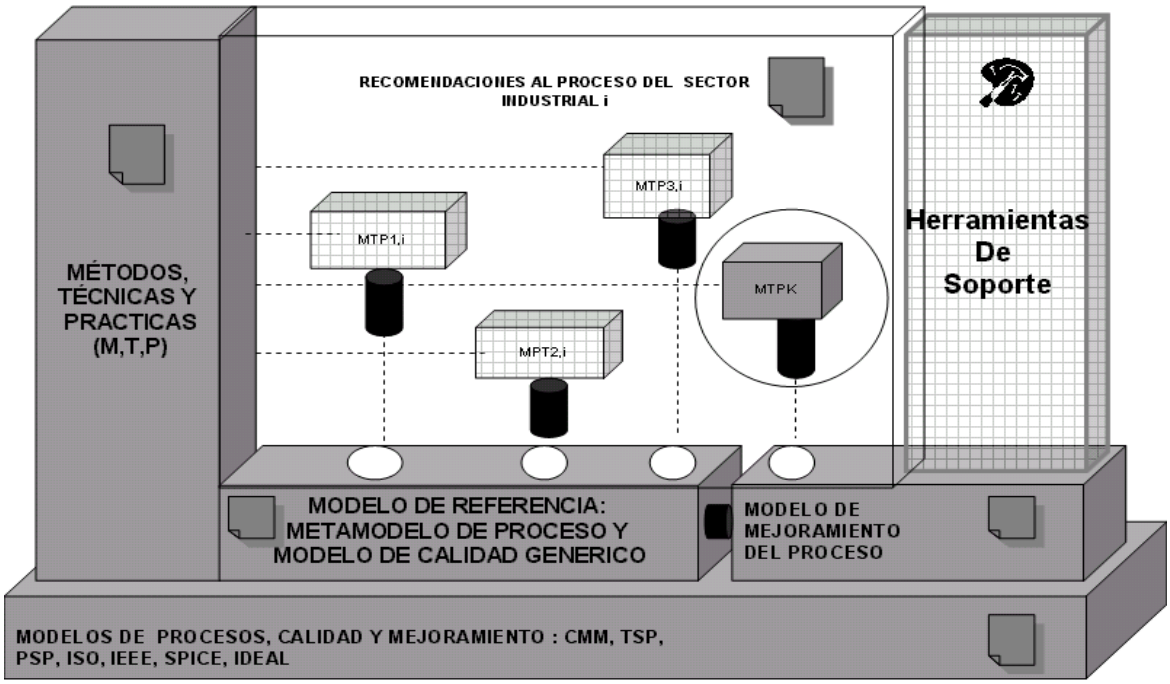



***Figura 1. Arquitectura preliminar para SIMEP-SW***

La figura 1 muestra la Arquitectura preliminar de SIMEP-SW, los bloques grises corresponden a los módulos que están siendo trabajados en el proyecto SIMEP-SW, los módulos rellenos con cuadrícula serán parcialmente desarrollados. El aporte fundamental del proyecto está en el modelo de mejoramiento, en conjunto con los métodos, técnicas y prácticas (M,T,P) que permitan alcanzar la mejora del proceso Software – SPI (Software Process Improvement).

La base de la Arquitectura la componen los modelos de calidad y de mejoramiento más importantes y reconocidos por la industria del software en el mundo, los cuales no pueden desconocerse, puesto que son los modelos a los que en definitiva las empresas deben adecuarse. Sobre este está el modelo de referencia para SIMEP-SW, el cual corresponde al resultado de evaluar los modelos de calidad existentes y al estudio de las prácticas que siguen un conjunto de empresas de desarrollo de software del sur occidente colombiano. El metamodelo de proceso SPEM-Software Process Engineering Metamodel [3] es la base notacional para la especificación y visualización de procesos en SIMEP-SW, y para la unificación de los modelos de calidad y de mejora. Este modelo de calidad debe tener los elementos comunes que le permitan a la organización ir adecuando el proceso para obtener con facilidad una certificación internacional. El modelo de mejoramiento a seguir por parte de las empresas para sacar adelante procesos de mejoramiento y/o certificación es un componente objetivo del proyecto, este modelo de referencia definirá la forma de llevar a cabo un proyecto de mejoramiento dentro de una organización PyME e implementará los elementos de gestión (un proceso, métodos, prácticas y técnicas) necesarios para asegurar su éxito. De igual manera el proyecto busca brindar unas recomendaciones sobre las prácticas, técnicas y métodos a utilizar dentro del proceso de desarrollo de software para las

PyMEs.

En el marco del proyecto y en conjunto con el proyecto [UC – Activos de procesos] , se ha gestado la posibilidad de crear Agile SPsL, una la infraestructura conceptual y tecnológica para la implementación de procesos de software ágiles [1] que cumplan con modelos y/o estándares de calidad [2] asociados a los procesos desarrollo de software.

Agile SPsL permitiría:

- Definir procesos ágiles a partir de un conjunto de activos de procesos reutilizables.
- Facilitar la evolución de estos activos.
- Servir a las agremiaciones de PyMES de Software [Gechs, ParqueSoft] contar con una infraestructura suficiente para: compartir una misma línea de procesos ajustable al tipo de producto que la PyME desarrolle, con el fin de alcanzar las capacidades en diferentes áreas del proceso.

Agile SPsL será una implementación liviana de un proceso organizacional de referencia que cumple con CMMI [4] y con el Agile Manifiesto[1].

Los activos reutilizables podrán ser: la arquitectura de proceso de Agile SPsL, así como los modelos o instancias de ciclo de vida, actividades, roles, productos de trabajo, guías, flujos de trabajo, disciplinas, etc. Los activos reutilizables pueden ser simples o compuestos. Un activo de proceso reutilizable puede ser un paquete o un compuesto de activos de proceso reutilizables.

El objetivo de esta infraestructura es soportar de manera organizada un repositorio de activos de proceso que puedan ser reutilizados por una organización en el proceso de definición, de evaluación y de mejora del proceso de Software. La línea de procesos define una especificación de interfaces flexibles para los activos del proceso, la cual deberá tenerse en cuenta al momento de crear, actualizar e instanciar un activo del proceso.

Esta infraestructura es Agile Software Process Line – Agile SPsL, la cual podrá ser utilizada para la construcción de procesos de software ágiles que cumplan con modelos, estándares o filosofías de calidad asociados a los procesos desarrollo de software, en particular al CMMI.

El alcance de Agile SPsL en este trabajo abarca:

- Un marco conceptual para la definición de procesos Ágiles basado en SPEM y en un conjunto de requisitos extraíbles de CMMI.

- Una arquitectura de referencia candidata para Agile SPsL basada en el marco conceptual y en un modelo de especificación de activos de proceso reutilizables.

- Una instanciación parcial de Agile SPsL en la implementación del Área del Proceso “Administración de Requisitos” en el marco del proceso eXtreme Programming, a manera de ejemplo.

# 2. Estado del Arte y Trabajos Relacionados

## 2.1 Líneas de Productos Software

La reutilización de software a diferentes niveles de abstracción ha demostrado influir significativamente sobre la productividad de los proyectos y la calidad de los productos. Sin embargo el desarrollo de software para reuso debe realizarse en un contexto de forma sistemática y organizada para alcanzar los niveles de productividad y de calidad esperados. Uno de los enfoques para lograr esto ha sido el de la construcción de líneas o familias de productos. Una línea de productos es un conjunto de productos que comparten un conjunto común de requisitos, pero que exhiben una variabilidad significativa en sus requisitos [13]. En la comunidad Estadounidense se utiliza el término líneas de productos, mientras en la comunidad Europea se utiliza el término familias de productos, debido a que las dos comunidades trabajaron independientemente antes de 1996 cuando se celebró la primera reunión en Las Navas, España. Una de las definiciones más aceptadas se expresa a continuación:

> Una línea de productos software es un conjunto de sistemas intensivos en software que comparten un conjunto de características que satisfacen un segmento particular del mercado y que son desarrollados de una manera preestablecida a partir de un conjunto de activos núcleo [11].

Los activos núcleo conforman la base de la línea de productos. Estos activos núcleo pueden ser: la arquitectura de referencia de la línea de productos, componentes de software reutilizables, modelos de dominio, requisitos, documentación, cronogramas, presupuestos, casos de pruebas, descripciones de procesos, entre otros.

Cada producto de la línea de productos, es un sistema software independiente creado a partir de un conjunto de componentes de la base común de activos, adaptándolos a través de mecanismos de variación previamente planificados, adicionando nuevos componentes si es necesario y ensamblando toda esta colección de acuerdo a las reglas de una arquitectura común a toda la línea de productos.

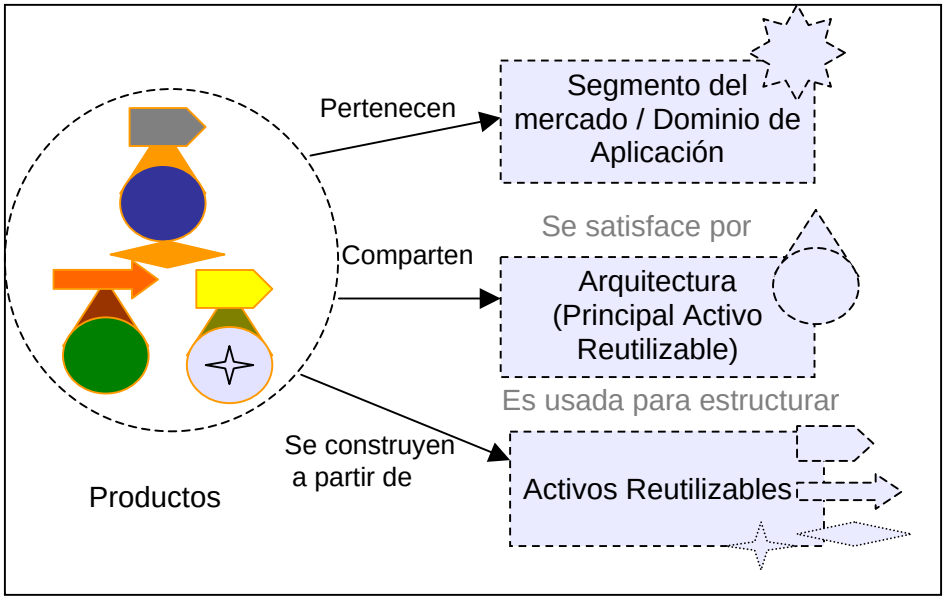



***Figura 2. El concepto de líneas de productos***

El concepto de línea de productos requiere de un proceso de reuso organizado dentro de las empresas desarrolladoras de software y enfoca este proceso a la consecución de activos reutilizables de granularidad gruesa, es decir los que son significativos desde el punto de vista arquitectónico. Desde la perspectiva de la línea de productos, el proceso incluye extiende en dos De acuerdo con Clements et. Al. [6], el proceso de desarrollo incluye tres actividades esenciales: el desarrollo de los activos núcleo reutilizables, los cuales corresponden con el concepto de Ingeniería de Dominio, el desarrollo del producto específico el cual corresponde con la ingeniería de aplicaciones y la gestión del proyecto. Estas actividades pueden visualizarse como tres procesos iterativos, que como tres ruedas dentadas permiten engranar la construcción de la línea de productos, como se muestra en la siguiente figura:

El desarrollo de una línea de productos involucra diferentes aspectos, los cuales Booch los agrupa de la siguiente manera [14]:

- Arquitectura, Componente y Sistema
- Negocio, organización, proceso y tecnología
- Desarrollo instanciación y evolución.

Hay otra agrupación, introducida por Henk Obbink [15] con el acrónimo BAPO – (Bussiness, Architecture, Process and Organization). Esta agrupación se va a tener en cuenta para desarrollar esta parte del estado del arte, debido a que abarca todo el espectro necesario para describir el marco conceptual que rodea la aplicación de líneas de productos a la industria.

### *2.1.1 El negocio*

La idea detrás de una línea de productos es satisfacer bien sea un mismo sector del mercado o un dominio en particular. Esto implica que aplicando un mismo esfuerzo, se podría abarcar un mercado o extenderse en un dominio dado. Pero, ¿hasta donde llega el mercado o hasta dónde se extiende mi dominio?. Es importante hacer esta pregunta por que esto delimita los requisitos, como elementos comunes y variable, que deberá abarcar la línea de productos, a este conjunto de requisitos se le ha denominado el alcance de la línea de productos.

> El alcance de una línea de productos según [15], define un portafolio de productos. El alcance del dominio identifica las fronteras con los dominios relevantes. El alcance de los activos identifica los elementos reutilizables.

Para, el Framework del SEI [6] el alcance es un elemento importante que requiere del acompañamiento de otras prácticas técnicas y de gestión, relacionadas con el cumplimiento de los objetivos del negocio: análisis de mercado, desarrollo del caso de negocio, determinación del alcance, El análisis Construir/Comprar/Buscar/Comisionar componentes software.

### *2.1.2 La Arquitectura*

La arquitectura de una línea de productos es el activo reutilizable más valioso para la organización. Ella se convierte en la "estructura" de referencia para la creación de los productos.

Existen varios puntos de vista de acuerdo a [16] a la hora de definir la arquitectura. Estos son algunos de ellos:

- La arquitectura es un diseño de alto nivel: es verdadero, pero no es una definición suficiente pues el diseño y la arquitectura van de la mano, pero no son intercambiables en el momento de definirlos. Existen tareas asociadas con el diseño que no son arquitecturales como por ejemplo la estructura de datos de la aplicación, el diseño de un algoritmo para mejorar desempeño.

- La arquitectura es toda la estructura del sistema: esta definición implica que el sistema sólo tendrá una sola estructura lo cual no es cierto ya que diferentes estructuras conforman el sistema.

- La arquitectura es la estructura de componentes de un programa o un sistema, sus Interrelaciones, sus principios y guías que gobiernan su diseño y su evolución todo el tiempo: cada sistema tiene una arquitectura independientemente del proceso con la cual esta arquitectura fue diseñada, así que no se hace necesario la información de las guías y los principios.

- La arquitectura son componentes y conectores entre estos componentes: Conectores implican un mecanismo de transferencia y control de datos en tiempo de ejecución, es decir esta definición se centra en las estructuras arquitecturales en tiempo de ejecución. Sería mejor tratar esta definición con el concepto de relaciones entre elementos ya que se podría hablar de relaciones dinámicas (tiempo de ejecución) y estáticas (compilación).

Una de las definiciones más aceptadas de Arquitectura de software es la siguiente:

> Estructura o conjunto de estructuras de un programa o sistema software, las cuales incluyen elementos software, sus propiedades externamente visibles de esos elementos y las relaciones entre estos" [16]

Esta definición no define cuales son los componentes y los conectores exactamente, estos dependen del paradigma arquitectónico o el patrón arquitectural [17] en el que se base para describir la Arquitectura. La siguiente tabla presenta una lista de los paradigmas Arquitecturales más utilizados actualmente para definir la Arquitectura software de un Producto o una línea de productos:

| Paradigma/Patrón | Componente | Relación | Propiedades Externamente Visibles |
|---|---|---|---|
| Orientado a Objetos | Clase/Paquete | Relación | Interface |
| Basado en componentes | Componente | Conector | Puerto |
| Orientado a Aspectos | Aspecto a Componentes o Clases | Weaving (tejido) | Advice |
| Orientado a Servicios | Proveedor/Consumidor de Servicios | Canal de Servicios | Servicio |

***Tabla 1. Arquitectura y Paradigmas***

### *2.1.3 El proceso*

El proceso de desarrollo de software típico involucra un desarrollo separado para cada producto, mientras en el enfoque de líneas de productos se espera en un mismo proceso desarrollar una línea de productos como si fuese un solo producto. El Framework del SEI [6] ha definido tres actividades esenciales y altamente iterativas que envuelven prácticas técnicas y de negocio: el desarrollo de activos núcleo, el desarrollo del producto y la gestión técnica y organizacional, las cuales se representan en la figura 3.

- Desarrollo de activos núcleo: el objetivo de esta actividad es establecer una capacidad de producción de productos software. Recibe como entradas restricciones de productos para establecer las características comunes y variables de cada uno de ellos; estilos, patrones y frameworks para soportar el diseño arquitectónico; restricciones de producción, como por ejemplo normas del sector del mercado y estándares; una estrategia de producción como enfoque general para el desarrollo de los activos núcleo, por ejemplo iniciar con el desarrollo de activos núcleo a productos, o de iniciar con el desarrollo de un conjunto de productos, y un inventario de los activos núcleo existentes. Esta actividad esencial entrega el alcance de la línea de productos, los activos núcleo y un plan de producción, el cual describe la forma en que podrán ser construidos los productos a partir de estos activos.
- El desarrollo del producto: este desarrollo depende de las salidas de la actividad anterior, junto con los requisitos particulares a cada producto.

- La gestión del proyecto: esta es una actividad esencial para garantizar el éxito de una línea de productos, para ello debe haber un compromiso de gestión tanto en el nivel técnico (proyecto), como organizacional. La gestión técnica cubre las actividades asociadas al desarrollo de los activos núcleo y los productos, con el fin de que se siga un proceso definido para la construcción de líneas de productos.

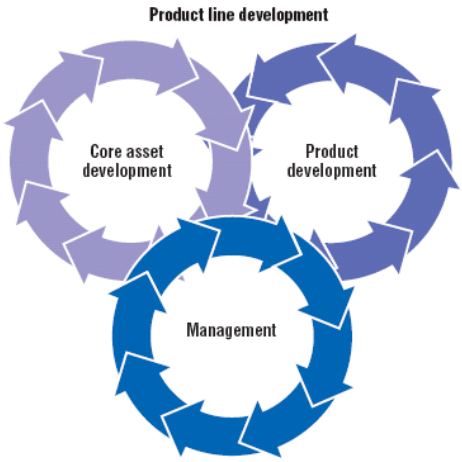



***Figura 3. Actividades esenciales en la construcción de una Línea de Productos***

Del otro lado el instituto europeo de Ingeniería del Software – ESI [15] ha utilizado en el proyecto Prise se liberó un proceso que fue utilizado en los proyectos ESAPS (Engineering Software Architectures, Processes and Platforms for Systems Families) y CAFÉ (Concepts to Application in System-Family Engineering). Este proceso se describe a partir de la figura 4.

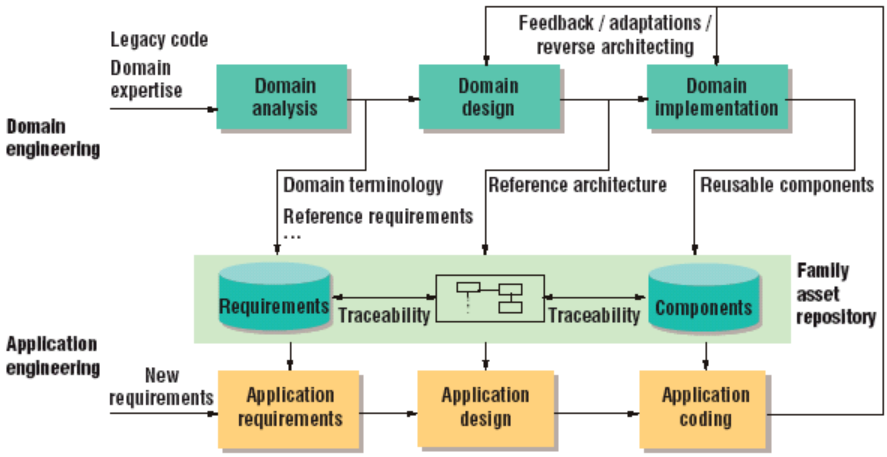



***Figura 4. Actividades esenciales en la construcción de una Línea de Productos***

Este enfoque define dos procesos: un proceso para la ingeniería de dominio y un proceso para la

ingeniería de aplicación.

El proceso de ingeniería de dominio incluye:

- El análisis de dominio para determinar hasta donde llega la familia de productos.
- El diseño de dominio, donde se diseña la Arquitectura de Referencia y demás activos núcleo.
- La implementación del dominio: dónde se construyen y/o compran los componentes que conforman la infraestructura.

El proceso de ingeniería de aplicación incluye:

- Captura de requisitos de la aplicación, los cuales determinan lo que debería ser el producto
- Diseño de la aplicación, que consiste en la selección de los componentes para hacer el producto.
- Codificación de la aplicación: se realiza con base en la Arquitectura, en los componentes seleccionados y posiblemente con código adicional particular al producto (sus elementos variables).

#### 2.1.3.1 Las áreas prácticas de ingeniería de software definidas por el Framework de Líneas de productos definidas por el SEI

Un área práctica es un cuerpo de trabajo o una colección de actividades [6] en que se divide todo el esfuerzo de construcción de una línea de productos, en terminología SPEM corresponde a una Disciplina. Estas áreas son esenciales para el desarrollo de cualquier tipo de producto, sin embargo guardan sus particularidades para la construcción de líneas de productos. El SEI organiza estas áreas prácticas en tres grupos: áreas prácticas de ingeniería de software, áreas prácticas de gestión técnica y la gestión organizacional. La figura 5 muestra las áreas prácticas relacionadas con la ingeniería de software y las relaciones entre estas:

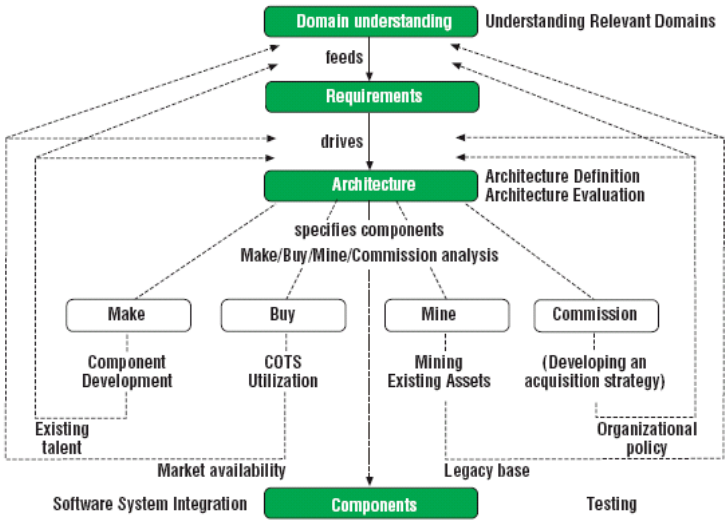



***Figura 5. Relaciones entre las áreas prácticas de Ingeniería del Software [Tomado del SEI]***

Las siguientes tablas describen con un poco de detalle las áreas prácticas de ingeniería de software:

| Áreas prácticas de ingeniería de software | | |
|---|---|---|
| Área | Particularidades para una LP | Aspectos específicos(Soporte) |
| Definición de la Arquitectura | El atributo de modificabilidad [16] es uno de los controladores de la Arquitectura. El soporte a la variación debe tomar muchas formas, con parámetros de configuración de componentes en tiempo de construcción o en tiempo de ejecución. Esto requiere que la variabilidad que debe permitir la Arquitectura, haya sido analizada juiciosamente. En sistemas orientados a objetos la herencia y la delegación son técnicas utilizadas para permitir estas variaciones. La programación orientada a Aspectos permite separar la funcionalidad de tipo crosscut, empaquetada en un Aspecto, la cual permitiría alcanzar una mejor separación de intereses facilitando la variabilidad de los requisitos no funcionales. | • Desarrollo de software basado en la Arquitectura.<br>• Estilos y patrones Arquitecturales.<br>• Desarrollo Orientado a Aspectos.<br>• Guía de construcción de productos.<br>• Mecanismos para lograr variabilidad ( p.e. herencia, extensiones, parametrización, configuración, generación, selección de implementación en tiempo de compilación).<br>• Frameworks |
| Evaluación de la Arquitectura | Se debe evaluar la Arquitectura en dos contextos: (1) En el contexto de la línea de productos se debe evaluar principalmente su robustez y modificabilidad para poder soportar el alcance definido. Igualmente debe ser evaluada en el contexto (2) de las instancias para asegurar el cumplimiento de los requisitos de calidad y funcionalidad esperados. | • Seguir un método de evaluación<br>o ATAM – The Architecture Trade-off Method)<br>o SAAM- The Software Analysis Method.<br>o SPE – Software performance Engineering<br>o ARID – Active Reviews for Intermediate Designs<br>o ADR – Active Design Review |
| Desarrollo de Componentes | Los componentes son tomados como unidades de software que en conjunto permiten armar una línea de productos como un todo. Los componentes deberán ser instalados en un repositorio de activos base, de dónde podrán ser configurados para ser instanciados en la construcción de un producto. Los componentes incluidos en el repositorio deben tener la flexibilidad necesaria para soportar los puntos de variación especificados en los requisitos de la línea de productos. | • Mecanismos de Variabilidad<br>o Herencia<br>o Extensión<br>o Uso<br>o Configuración<br>o Parametrización<br>o Templates<br>o Generación<br>o Reflexión<br>o Programación orientada a aspectos<br>o Patrones de diseño |
| Uso de COTS-Commercial off-the-shelf | Los activos núcleo de una línea de productos puede estar poblada de COTS. Estos COTS deberán soportar la variabilidad exigida para la línea de productos y debe garantizarse la continuidad en el mercado tanto si son producidos o consumidos por la organización. La estrategia de evolución de estos activos es más compleja pues estos soportan un mercado más amplio al cuala se puede afectar. | Prácticas de sistemas basados en COTS para la evaluación de componentes comerciales. Esto incluye : un plan de evaluación, el diseño del instrumento de evaluación y la aplicación del instrumento de evaluación |

| | | |
|---|---|---|
| Minería de Activos existentes | El almacenamiento de los activos debe realizarse pensando en su futuro reuso: debe reunir los requisitos de la línea de productos, deben estar alineados con la arquitectura de referencia y debe permitir relacionar consistentemente los objetivos de calidad con los objetivos de la línea de productos. El enfoque está orientado hacia los activos de grano grueso, aunque esto no excluye la minería de activos de grano fino. | • Evaluar la posibilidad y economía de la minería de componentes existentes para una línea de productos.<br>• Uso de herramientas de reconstrucción de Arquitecturas.<br>• Uso de envolturas o adaptadores. |
| Ingeniería de Requisitos | Los requisitos en la línea de productos definen los productos y las características de los productos en la línea de productos. Los requisitos que atraviesan la familia completa se constituyen en un activo reutilizable fundamental y debe ser mantenido separadamente de los requisitos que son particulares a un producto o a un conjunto de estos. Los requisitos de la línea de productos deben incluir puntos de variación para soportar las extensiones a los requisitos de cada producto, de tal forma que es estos puedan reutilizar este activo. | • Técnicas de análisis de dominio<br>• Modelado de diferentes vistas<br>• Modelado de Características<br>• Modelado de Casos de uso<br>• Modelado de casos de cambio<br>• Guardar la trazabilidad de los requisitos a sus productos de trabajo asociados.<br>• Modelado de escenarios de calidad(No lo tiene explicitamente) |
| Integración del Software | La integración ocurre en dos momentos: el primero al implementar la arquitectura de referencia (activos base) a partir de los activos núcleo y también al construir el producto. | • Definir interfaces estándar (como XML, IDL)<br>• Usar adaptadores<br>• Usar tecnologías de middleware estándar.<br>• Usar generadores de sistemas<br>• Usar generadores FAST – Family – Oriented Abstraction, Specification, and Translation. |
| Prueba de Software | La prueba de software en una línea de producto debe examinar el software de los activos núcleo, el software específico a un producto y las interacciones entre estos. Otros aspectos los expone McGregor [18] | • Planes de prueba<br>• Casos de prueba<br>• Listas de chequeo<br>• Inspecciones de diseño y de código<br>• Evaluación de la Arquitectura<br>• Pruebas de assets |
| Entendimiento de los Dominios relevantes a la línea de productos | Entender los dominios relevantes es el primer paso para entender los elementos comunes y variables de la línea de productos, que corresponden con los productos identificados en el alcance de la línea. Los dominios pueden marcar límites del alcance de la línea de productos, agrupar y gestionar conjunto de activos comunes por dominio, identificar oportunidades de negocio y decidir cuando se hace un nuevo desarrollo, si un activo se debe considerar un activo base o no. Los aspectos a tener en cuenta fueron tomados de [12] y [6]. | • Análisis SCV – Scope, commonaliy, and variability análisis.<br>• DADP(Domain analysis and design process)<br>• FODA (Feature-Oriented Domain Analysis)<br>• OODA (Object Oriented Domain Analysis)<br>• ODM (Organization Domain Modeling)<br>• DAGAR (Domain Architecture-based Generation for Ada Reuse)<br>• FORM (Feature-Oriented Reuse Method)<br>• FeatureRSEB (Feature Reuse- Driven |

| | | |
|---|---|---|
| | | Software Engineering Business) |

***Tabla 2. Áreas prácticas del Framework de líneas de productos del SEI***

### *2.1.4 La organización*

Desde la perspectiva Europea no existe claridad de cómo y por qué una organización de desarrollo puede ser separada en unos departamentos o unidades de desarrollo de familias de productos y productos [15], mientras el SEI aborda la organización a través de dos maneras:

Organiza tres actividades esenciales - desarrollo de los activos base, desarrollo de productos y gestión – las cuales permiten de acuerdo a su abordaje definir los equipos de trabajo.

Organiza el proceso en tres grupos de áreas prácticas las cuales incluyen la gestión técnica y la gestión organizacional. Dentro del grupo de áreas de gestión organizacional el Framework del SEI incluye el área práctica de estructuración de la organización, la cual se describe a continuación.

La organización de acuerdo al framework del SEI, implica la creación de nuevos roles y responsabilidades relacionados con la creación de los activos núcleo y a los productos a partir de esos activos núcleo. El framework define el área práctica de estructuración de la organización para ubicar esos roles en unidades organizacionales adecuadas para soportar efectivamente el enfoque de líneas de productos. En este contexto la estructura organizacional no está centrada en el producto sino que debe incluir: la estructura y fronteras de la empresa, la identificación de los grupos funcionales, la ubicación y asignación de recursos, el monitoreo de la efectividad organizacional, las operaciones de mejora organizacional, establecer las relaciones interorganizcionales y gestionar las transiciones organizacionales. Se debe escoger una estructura que al menos permita definir cual(es) unidad(es) está(n) encargada(s) de:

Producir y mantener la Arquitectura de referencia.

Determinar los requisitos de la línea de productos y sus miembros.

Diseñar, producir y mantener los activos núcleo de la línea de producto.

Asegurar los activos núcleo para su uso y evolución.

Producir productos.

Definir el proceso de desarrollo y su cumplimiento.

Mantener el ambiente de producción en el cuál los productos son producidos.

Los aspectos específicos del área de estructuración de la organización se describen a continuación:

Usar los modelos de Organización de una encuesta a las líneas de productos suecas

Modelo de departamento de desarrollo: en este modelo todo el desarrollo de software es concentrado en una unidad. Este modelo podría ser el adecuado en organizaciones pequeñas.

Modelo de Unidad de negocio: en este modelo cada unidad es responsable de un subconjunto de productos de la línea, los cuales deben ser agrupados por similaridad. Los activos compartidos

son desarrollados por las unidades de negocio que los requieran y son puestos a la disponibilidad de toda la comunidad. Este modelo podría ser el adecuado en organizaciones medianas.

Modelo de unidad de ingeniería de dominio: en este modelo existe una unidad especial que tiene la única responsabilidad de desarrollar y mantener los activos núcleo. Este modelo puede ser aplicado a organizaciones medianas y grandes.

Modelo de unidad de ingeniería de dominio jerárquico: es aplicable cuando el alcance de la línea de productos es muy amplio y existe una infraestructura organizacional grande para soportar su desarrollo. En este caso la línea de producto puede dar lugar a líneas de producto especializadas, en este caso cada línea de productos especializada contaría con una unidad de ingeniería de dominio responsable de su desarrollo y mantenimiento.

Crear una estructura organizacional para negocios basados en reuso: hace referencia a estructuras donde las unidades tienen cierta competencia sobre el negocio, es decir trabajadores con competencias similares y objetos de negocio de su responsabilidad. Estas unidades pueden ser: unidad de captura de requisitos, unidad de diseño, unidad de pruebas, unidad de ingeniería de componente, unidad de Arquitectura y unidad de soporte a componentes.

Crear un grupo pequeño y distribuido con representantes por componente o por producto: la empresa organiza un pequeño grupo para el soporte de actividades comunes tales como la gestión de configuración y el mantenimiento de los activos base, la definición del proceso, entre otros. Este grupo podría incluir un arquitecto para toda la línea de productos. En cada grupo de trabajo hay un representante de componente quién debe asegurar el uso adecuado del componente, así como la creación de los activos base.

Adaptar la estructura de la organización: este enfoque consiste en mantener la estructura que tiene la organización, y crearle paralelamente una estructura lógica de línea de productos superpuesta sobre el esqueleto organizacional de la empresa. Es como si a la estructura de la organización se le colocar un adaptador organizacional, para que asemeje alguna de las estructuras anteriores.

### *2.1.5 Comparando los esfuerzos entre Europa y US*

Basado en los trabajos del SEI [6] y ESI [15] se puede decir que los dos tienen iniciativas que persiguen los mismos objetivos: mejorar los procesos de la industria a través de la introducción del enfoque de familias o líneas de productos. En Europa las compañías unieron esfuerzos a través de trabajos conjuntos que permitieron aprender unos de otros. La iniciativa del SEI es más completa pues cubre todo el espectro negocio – arquitectura – organización – proceso, sobre todo hace bastante énfasis en este último. Mientras el SEI define un único framework que puede ser instanciado a diferente tipo de organizaciones, ESI ha facilitado el desarrollo de diferentes proyectos con los que se espera crear un conjunto de soluciones, dónde la empresa pueda escoger y adaptar la más conveniente. A continuación se presenta la tabla 3 que permite comparar el alcance de los dos trabajos de manera más específica.

| Aspectos de la Iniciativa | | | | |
|---|---|---|---|---|
| Iniciativa | Negocio | Arquitectura | Proceso | Organización |
| ESI Europa | Alcance Análisis de | Arquitecturas de dominio | Ingeniería de requisitos Plataformas Heterogéneas | Grupos de desarrollo de |

| | | | | |
|---|---|---|---|---|
| | negocio y de mercado<br>Adopción y transición a desarrollo de familias. | Arquitectura de familias<br>Análisis de dominio<br>Análisis de Aspectos<br>Requisitos de Familias<br>Glosario de la Arquitectura.<br>Arquitectura de Referencia<br>Plataformas y Componentes<br>Aseguramiento de la Arquitectura<br>Reconstrucción de arquitecturas | Uso de COTS<br>Diseño para calidad.<br>Herramientas de soporte al desarrollo.<br>Modelado de Pruebas<br>Validación<br>Recuperación desde sistemas heredados.<br>Frameworks de procesos para familias<br>Soporte a la evolución<br>Gestión de activos<br>Gestión de configuración<br>Estrategia y metodología de prueba.<br>Validación<br>Derivación de productos | plataforma y de componentes. |
| SEI EU | Desarrollo del caso de negocio<br>Gestión de la interface con el cliente<br>Análisis Desarrollo/ Compra/Extrae/Comisiona<br>Análisis de mercado | Definición de Arquitectura<br>Evaluación de Arquitectura<br>Análisis basado en escenarios de calidad<br>Diseño dirigido por atributos<br>Reconstrucción de arquitecturas | Entendimiento de los dominios<br>Requisitos<br>Arquitectura<br>Análisis H/C/B/C<br>Integración<br>Testing<br>DSBC<br>Estrategia de adquisición<br>Gestión de Configuración<br>Métricas y seguimiento<br>Definición del proceso<br>Planificación técnica<br>Gestión del riesgo técnico<br>Herramientas de soporte | Lanzamiento e institucionalización<br>Operaciones<br>Planificación Organizacional<br>Gestión del riesgo organizacional<br>Estructura de la organización<br>Entrenamiento |

***Tabla 3. Comparación de esfuerzos hacia el enfoque de líneas de productos entre Europa y E.U.***

Del cuadro anterior se puede concluir que la iniciativa del SEI es más completa en este espectro, presenta de manera más organizada la iniciativa a través de áreas prácticas, que como es de esperarse guardan mucha correspondencia con el CMMI [4] apuntando hacia un proceso de líneas de producto maduro y completo para facilitar la adopción por parte de las organizaciones. Adicionalmente, se visualiza como un Framework que puede ser adaptado a las características de las organizaciones, y por ello el SEI ha venido trabajando en tecnologías relacionadas como The Product Line Technical Probe (PLTP)[18]. PLPT es un método utilizado para examinar si una organización está preparada o cuenta con la capacidad para adoptar exitosamente el enfoque de líneas de productos. PLTP utiliza una serie de entrevistas de pequeños grupos de pares al interior de la organización, seguido por un análisis de esta información. El PLTP utiliza el Framework de Líneas de Productos como un modelo de referencia, tanto en la recolección, como en el análisis de la información. Como resultados PLTP incluye un conjunto de hallazgos, las fortalezas y debilidades que caracterizan a la organización para adoptar en enfoque de líneas de productos, así como un conjunto de recomendaciones. Estos hallazgos sirven de entrada, a lo que el SEI ha

llamado el workshop de planeación de líneas de productos, el cual ayuda a desarrollar un plan de acción con el objetivo de incrementar la capacidad de la organización para lograr el éxito de una línea de productos de acuerdo a los objetivos de negocio identificados.

### *2.1.6 Algunos métodos para el desarrollo de líneas de productos*

De acuerdo a la comparación de algunos métodos [20], se pueden nombrar algunos, que sirvan de referencia para como punto de partida para la selección de un método o proceso a seguir para adoptar el enfoque de líneas de productos. En este trabajo los trabajos fueron evaluados desde el punto de vista arquitectónico, con algunas conclusiones en algunos de los otros aspectos.

*COPA - Component Oriented Patform Architecting Method for product family Engineering*

Desarrollado por Philips Research Labs. Es un método orientado a components, pero centrado en la arquitectura que permite desarrollar familias de sistemas intensivos en software. Este método se concentra en el balance entre los enfoques top-down y down-top y cubre todos los aspectos de la ingeniería de líneas de productos: arquitectura, proceso, negocio y organización.

*FAST – Family-Oriented Abstraction, Specification and Translation*

Desarrollado por David Weiss. Es un proceso de desarrollo de software enfocado a la construcción de familias. Incluye una descripción de proceso orientado a familias con actividades, artefactos y roles. No es aplicable como está definido, pero puede ser adaptado.

*FORM – Feature- Oriented Reuse Method*

Propuesto por Kyo C. Kang y sus seguidores en la Universidad de Ciencia y Tecnología de Corea. Es una extensión de método FODA. Este método incluye un análisis de características de dominio y usa estas características para desarrollar artefactos del dominio reutilizables y adaptables; por eso se define como un método orientado a las características (features). Usa las características como una forma de capturar los elementos comunes y variables, y se extiende al diseño arquitectónico y desarrollo de los activos base.

*KobrA – Komponentbasierte Anwendungsentwicklung*

Desarrollado por Fraunhofer IESE. Denota un método práctico para hacer ingeniería de software orientada a líneas de productos y está basado en UML. Se adapta tanto al desarrollo de productos independientes, así como al de líneas de productos.

*QADA – Quality Driven Architecture Design and Análisis*

Desarrollado por el Centro de Investigación Técnica de Finlandia. Permite hacer la trazabilidad de la calidad del producto y el aseguramiento de la calidad en tiempo de diseño. Sugiere que los requisitos de calidad son la fuerza controladora de la Arquitectura, el diseño de la Arquitectura es

combinado con el análisis de calidad. Provee un soporte para el aseguramiento de la calidad paralelo para las arquitecturas de la línea de productos.

## 2.2 Los modelos de Calidad para el proceso Software

Hoy en día existen modelos y estándares de calidad y mejoramiento de procesos reconocidos internacionalmente, entre estos modelos y estándares, cabe destacar los modelos desarrollados por el SEI, como el modelo de procesos CMMI[4], su método de evaluación SCAMPI [27] y el método de mejora asociado IDEAL[28]; y los modelos desarrollados por la International Organization for Standardization – ISO (Organización Internacional para la Estandarización) como el modelo de proceso ISO 15504 [30], el cual se basa en la norma ISO/IEC 12207 [29]y la enmienda I (2002) [31], su método de evaluación ISO 15504 (parte 4)[32], y el método de mejora asociado ISO 15504 (parte 7)[33],; además es importante tener en cuenta la familia de normas ISO 9001:2000 [7]. Los modelos de mejoramiento y calidad más aceptados por la industria del software a nivel mundial son: ISO 9001/2000, CMMI que recoge la integración de los modelo de calidad CMMs y el ISO/IEC 15504 conocido también como SPICE.

Las empresas de software pequeñas también tienen muy serios problemas de madurez en sus procesos de desarrollo de software, de hecho en muchos casos no existe un proceso real conduciendo a menudo a muchos modelos caóticos de operación que afectan toda la empresa[34]. Muchas organizaciones pequeñas planean acreditarse en un modelo de calidad (CMMI, ISO 15504 e ISO 9001-2000) con el fin de poder acceder al mercado de las exportaciones, pero la preparación previa a la certificación es larga y costosa. Los modelos de mejoramiento, proceso y evaluación, de organizaciones como el SEI e ISO, están estructurados para ser aplicables a empresas grandes y difícilmente pueden ser aplicados a empresas pequeñas debido a que un proyecto de mejora supone gran inversión en dinero, tiempo y recursos, además a la complejidad de las recomendaciones y a que el retorno de la inversión se produce a largo plazo [35].

### *2.2.1 Los modelos del SEI*

Hasta un poco, la estrategia de calidad, al menos en las organizaciones industriales, fue basado en la mayoría de casos basado en la prueba. Los equipos típicamente establecidos en departamentos de calidad eran los encargados de encontrar y remover problemas después de que los productos eran liberados. No fue sino hasta 1970 y 1980 que W. Eduardo Deming y J.M.Juran convencieron a la industria norteamericana de enfocarse sobre el mejoramiento en la forma en que la gente hacía sus trabajos [21,22]. En los años siguientes, este enfoque se hizo sobre el proceso productivo, el cual ha sido responsable de mejoras en la calidad de automóviles, electrónica, entre otros productos. La estrategia tradicional de probar y remover problemas al producto terminado se concibe como una estrategia complicada, lenta (alto consumo de tiempo), e inefectiva para el trabajo de manufactura e ingeniería.

Hoy en día, que la mayoría de los sectores industriales han adoptado principios de calidad mientras la comunidad de software ha continuado reposando sobre la prueba como método principal de aseguramiento de calidad. Para la comunidad de software, el primer paso fue liderado por Deming y Juran y fue dado por Michael Fagan cuando en 1976 introdujo las inspecciones de software [23]. Usando inspecciones, las organizaciones han mejorado notablemente la calidad del producto software. Otro paso significativo en el mejoramiento de la calidad de software fue dado

con la introducción inicial del Capability Maturity Model – CMM (Modelo de Madurez de la Capacidad del Proceso) para software[42]. El foco principal de CMM está sobre la gestión del sistema y el soporte y asistencia provista a los ingenieros de desarrollo. El CMM ha permitido obtener resultados positivos en el desempeño de las empresas desarrolladoras de software.

Otro paso significativo en el mejoramiento de la calidad del software dado por el SEI fue dado al crear el Personal Software Process – PSP (Proceso de Software para Personas) [25]. El PSP extiende el mejoramiento del proceso a las personas, ingenieros que son los que realmente hacen las cosas prácticas. El PSP se concentra sobre las prácticas de trabajo individuales de los ingenieros. El principio detrás del PSP es producir sistemas software de calidad, cada ingeniero que trabaja en el desarrollo del sistema debe hacer el trabajo de calidad.

El desarrollo del Team Software Process – TSP (Proceso de Software para Equipos) [26] sigue la estrategia de calidad orientada al proceso siguiendo al PSP. El TSP provee un contexto disciplinado para el trabajo de ingeniería. La principal motivación para el desarrollo del TSP fue la convicción de que los grupos de ingeniería pueden hacer un extraordinario trabajo, pero sólo si está adecuadamente organizado, entrenado, distribuido y eficazmente conducido. El objetivo del TSP fue construir una guías para equipos de trabajo efectivos.

El SEI se identifica a menudo con el trabajo de CMM. A través del tiempo, el SEI ha desarrollado seis productos del modelo de la madurez de la capacidad. Algunos de los nuevos productos fueron desarrollados a partir de los más antiguos. Los CMMs que el SEI actualmente tiene por evolucionar, ampliar, o mantener son:

CMMI - Capability Maturity Model Integration (Modelo de Madurez de Capacidad Integrado)[4]

P-CMM - People Capability Maturity Model (Modelo de madurez de capacidad del Talento Humano)[44]

SA-CMM - Software Acquisition Capability Maturity Model (Modelo de Madurez de capacidad de Adquisición de Software.[46]

Los modelos CMMs heredados se han incorporado en el modelo CMMI, y que por lo tanto no serán mantenidos o evolucionados son:

SW – CMM – CMM for Software (CMM para ingeniería de software)[42]

SE – CMM – Systems Enginnering CMM (CMM para Ingeniería de Sistemas)[ 43]

IPD-CMM - Integrated Product Development CMM (CMM para desarrollo integrado de producto).[45]

Las metas de SEI en desarrollar los modelos CMMs incluyen:

Tratar la ingeniería del software y otras disciplinas que afectan el desarrollo y mantenimiento del software.

Proveer modelos integrados de referencia de la mejora de proceso que constituyan un amplio consenso de la comunidad

Armonizar con los estándares relacionados

Permitir la mejora eficiente a través de las disciplinas relevantes al desarrollo y al mantenimiento del software.

El mejoramiento del proceso de desarrollo de software ha probado incrementar la calidad de los

productos y servicios cuando las aplicaciones lo aplican para alcanzar sus objetivos de negocio. Hoy, la suite de productos CMMI (Capability Maturity Model Integration) es uno de los más importantes modelos de referencia para los procesos de mejoramiento, proveyendo las últimas mejores prácticas para el desarrollo y mantenimiento de productos y servicios.

Existen varios modelos CMMI disponibles. Para ello, se necesita estar preparado para decidir cuales de los modelos CMMI mejor se ajusta a las necesidades para el mejoramiento del proceso para su organización. El punto de inicio es seleccionar cuales disciplinas se desea incluir en su programa de mejoramiento del proceso y seleccionar un modelo de representación de los dos que presenta el modelo: escalonado o continuo.

### *2.2.2 Las normas de ISO*

#### 2.2.2.1 ISO/IEC 12207:1995/Amd 1:2002

Esta norma establece un marco de referencia común para los procesos del ciclo de vida del software, con una terminología bien definida a la que puede hacer referencia la industria del software. Contiene procesos, actividades y tareas para aplicar durante la adquisición de un sistema que contiene software, un producto software puro o un servicio software, y durante el suministro desarrollo operación y mantenimiento de productos software. La norma incluye un proceso que puede emplearse para definir, controlar y mejorar los procesos de ciclo de vida del software. La norma ha sido escrita para quienes adquieren sistemas, productos y servicios software, y para los proveedores, desarrolladores, operadores, responsables de mantenimiento, administradores, responsables de aseguramiento de calidad y usuarios de productos software.

#### 2.2.2.2 ISO/IEC TR 15504 (1998)

El estándar ISO/IEC 15504 (1998) es un estándar internacional para la evaluación y mejora de procesos software. En este estándar se desarrolla un conjunto de medidas de capacidad estructuradas con el objetivo de evaluar el proceso de ciclo de vida del software. ISO, en conjunto con IEC - Internacional Electrotechnical Comission crearon un estándar de certificación y estandarización denominado ISO/IEC 15504, que provee un modelo conceptual y marco para la evaluación, validación, optimización y certificación del proceso de desarrollo o construcción de software. Su primera publicación es realizada en Julio de 1998 y en Mayo de 1999 se le dio carácter de Reporte técnico. Este marco puede ser utilizado por organizaciones que se vean involucradas en las diferentes etapas del proceso de construcción y/o selección de software o proveedores del mismo, así como el planeamiento, gerenciamiento, monitoreo, control y mejoras en la adquisición, desarrollo, operación y soporte.

Con la premisa de que la gestión y administración de la calidad se ha convertido en algo absolutamente importante para dejarlo al azar, el estándar ISO/IEC15504 intenta ser el elemento diferenciador que por medio de la definición de un modelo y marco de referencia permiten evaluar el proceso de desarrollo de software de acuerdo a niveles definidos por la norma. Al mismo tiempo, permite asegurar la gestión de calidad en el proceso de desarrollo de software, identificando áreas de mejora y potenciales riesgos. De esta forma, la metodología propuesta por el estándar ISO/IEC 15504 permite satisfacer diferentes objetivos de acuerdo a quien sea su ejecutor o asesor y ellos cubren las necesidades de las empresas u organizaciones en aspectos

como: Determinar el estado de su propio proceso de desarrollo de software, Definir el grado de cumplimiento del proceso de desarrollo de software de acuerdo a los requisitos específicos, Determinar el grado de madurez del proceso de desarrollo de software de sus contratistas.

La parte 2 del estándar ISO/IEC 15504 define el Modelo de Referencia de la norma. A este modelo se lo puede considerar como la guía conceptual que cubriendo y evaluando las características de rendimiento y cualidades de los procesos para el desarrollo del software, permiten determinar su grado de madurez.

La arquitectura abarcada en el Modelo de Referencia se extiende en dos dimensiones:

La dimensión de los procesos, la cual se caracteriza por focalizarse en las características y propósitos de un proceso específico dentro del modelo de negocio estudiado, siendo ellos los elementos mensurables de los objetivos del mismo. Sus indicadores se encuentran definidos en cinco categorías de procesos.

La dimensión de las capacidades y habilidades de los procesos, caracterizada por un conjunto de atributos propios a todo proceso, y aplicables a cualquiera en estudio que presenta características mensurables necesarias para administrar un proceso y mejorar su capacidad. Esta dimensión define una serie de atributos agrupados en niveles de capacidad, suministrando éstos atributos las características mensurables de un proceso.

#### 2.2.2.3 ISO/IEC TR 15504 (2003)

La norma ISO/IEC 15504 (2003), se enmarca bajo el nombre Tecnologías de Información: proceso de evaluación, y esta constituida por cinco partes:

Parte 1. Conceptos y vocabulario. (En preparación). Introduce los conceptos y el vocabulario de términos relacionados con evaluación de procesos.

Parte 2. Realización de la evaluación. Define las bases para evaluar procesos.

Parte 3. Guía para la realización de la evaluación. Proporciona una guía para la interpretación de los requerimientos para la realización de una evaluación.

Parte 4. Guía para usar en la determinación de la capacidad del proceso y mejora de procesos.

Parte 5. Un ejemplo de un modelo de evaluación de procesos. (En preparación). Contiene un ejemplo de un modelo de evaluación de proceso que esta basado sobre el modelo de proceso de referencia ISO/IEC 12207:1995/Amd 1:2002.

## 2.3 Los modelos de mejora

Cada uno de los modelos de calidad antes mencionados por sí mismo tiene un gran valor. Cualquiera que sea su elección, se recomienda a la organización realizar un análisis del camino que tendrá que recorrer para lograr su objetivo, es decir, deberá establecer un programa de mejora:

> "Un programa de mejora es un proyecto continuo que conduce el mejoramiento de los procesos de software de una organización y es responsabilidad directa de la Alta Dirección. Se dice que es un proyecto continuo porque tiene un inicio pero no tiene un fin. Esto es debido a que un programa de mejora está constituido por ciclos de mejora y cada ciclo por fases, es decir, es un proyecto con ciclo de vida iterativo e incremental"

El mejoramiento de un proceso es el esfuerzo continuo para saber acerca del sistema de causas en un proceso y para usar este conocimiento en el cambio y mejora del proceso y de esa manera reducir su variación, complejidad y mejorar la satisfacción del cliente" [36].

### *2.3.1 Importancia de los SPI- Software Process Improvement*

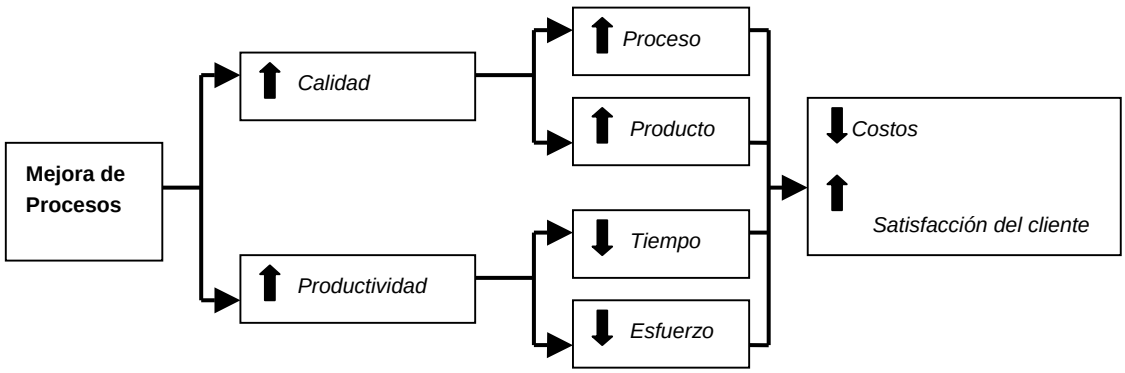



***Figura 6. Reducción de Costos y Aumento en la satisfacción del Cliente, como indicadores primarios de la mejora del proceso***

En los años noventa, la mejora de proceso de software se promovió principalmente bajo los auspicios de lograr los requisitos de varios modelos y estándares. Tantara cree que los modelos y estándares tienen un papel importante en la mejora del proceso de software pero cree que estos no son necesarios como requisito previo para la excelencia comercial [37] ya que la importancia de los SPI se centra no solo en la elevación de la calidad del producto, sino también en aumentar:

La reducción de costos y tiempo

La posibilidad de reproducir éxitos en proyectos

El control sobre los riesgos de procesos

Aumentar la confianza y satisfacción del cliente.

A continuación se relaciona de manera resumida los métodos y/o modelos de mejora de procesos de software que se han adelantado.

### *2.3.2 IDEAL*

IDEAL [28] Es un modelo para un programa SPI. Consta de cinco fases de un programa SPI que

le dan el nombre: (I) Iniciación, (D) Diagnóstico, (E) Establecimiento, (A) Ejecución (Acting) y (L) Aprendizaje (Learning). Estas fases proveen un ciclo infinito a través de los pasos necesarios para un SPI. El tiempo para cada ciclo y cada fase depende de la organización. Habrá organizaciones comprometidas con programa SPI, adicionalmente, las fases o partes de estas pueden ser llevadas en paralelo dependiendo de la infraestructura organizacional y de los equipos de trabajo dentro del programa. Es importante notar que la infraestructura necesaria para programa SPI juega un papel importante para alcanzar el éxito del proyecto. Entender los compromiso, roles y responsabilidades es de gran importancia.

### *2.3.3 Estándar ISO/IEC 15504 parte 7.*

En su parte 7[33] reside este estándar internacional desarrollado en el proyecto SPiCE (Software Process Improvement Capability dEtermination) para la evaluación de proceso del software. El estándar ISO/IEC 15504 proporciona un marco para todos los aspectos de una evaluación de proceso que se puede utilizar para evaluar la capacidad de los procesos de su organización. El marco precisa los requerimientos para la realización de una evaluación conforme a la ISO/IEC 15504. Esta norma está constituida por cinco partes: Parte 1, Conceptos y vocabulario, parte 2, Realización de la evaluación, parte 3, Guía para la realización de la evaluación, parte 4, Guía para usar en la determinación de la capacidad del proceso y mejora de procesos y parte 5, Un ejemplo de un modelo de evaluación de procesos.

### *2.3.4 EL PDCA*

Este ciclo es un modelo para aprender. Se hace una deducción (predicción) basada en alguna teoría, se recogen observaciones (colección de datos), se hace una comparación de los datos con las consecuencias predichas, y se hace una modificación de la teoría (aprendizaje) cuando las consecuencias y los datos no concuerdan. El ciclo de mejoramiento tiene cuatro fases: Plan-Do-Check-Act (Planear-Hacer-Chequear-Actuar). [39]

### *2.3.5 El framework IMPACT*

Impact es un framework que propone un paradigma liviano para SPI[40], el cual consta de los siguientes estados: Entender-Mejorar-Aplicar-Medir, los cuales pueden ser aplicados incrementalmente a lo largo de varios proyectos. Por otra parte, al implementar iteraciones sucesivas, las metas de mejoramiento serán alcanzadas rápidamente (usualmente dentro de unos pocos meses) y son factibles de medirse a través de los proyectos en los cuales el enfoque de mejoramiento ha sido aplicado. Éste framework diferencia dos niveles: el nivel de proyecto y el nivel de proceso. En el nivel de proyecto, muchos proyectos se desarrollan de acuerdo a buenas prácticas de gestión de proyectos (Por ejemplo Plan-Do-Check-Act (Planear-Hacer-Chequear-Actuar)). En el nivel de proceso, la experiencia y el entendimiento de muchos proyectos se usan para entender y mejorar el modelo de procesos genérico, el cual es usado después para guiar proyectos futuros, este es el ciclo que se llama el ciclo del proceso. Estos ciclos interactúan muy de cerca – el ciclo de proceso conduce la ejecución de los proyectos y los proyectos conducen las mejoras para el proceso [38].

### *2.3.6 Tesis Reidar Conradi y Alfonso Fuggetta*

A través de la experiencia obtenida en varias iniciativas SPI, los autores [41] presentan seis tesis de cómo mejorar y aplicar mejor los frameworks SPI. Esas tesis son las siguientes:

**Tesis1:** Los frameworks SPI deben apoyar estrategias de mejora que se enfocan en la orientación de la meta e innovación del producto.

**Tesis2:** Los diseñadores son motivados para el cambio; si es posible, comiencen desde el fondo hacia arriba con iniciativas concretas.

**Tesis 3:** El soporte de proceso software ha sido sobre-enfatizado.

**Tesis 4:** El análisis costo-beneficio requiere a los nuevos modelos de amortización.

**Tesis 5:** SPI asume los cambios culturales, para que nosotros necesitemos especialización de las sociologías. Tesis 6: SPI es acerca de aprendizaje – no control como en QA.

## 2.4 El CMMI

### *2.4.1 Introducción*

Los modelos de CMMI [4] involucran el concepto de CMM (Capability Maturity Model), establecido por el Modelo de Madurez de la Capacidad para Software (SW-CMM) [42], a un nuevo nivel que promete el crecimiento y expansión continua del concepto CMM a múltiples disciplinas o cuerpos de conocimiento. Así como el SW-CMM [42], EIA/IS 731, IPD-CMM[45], SA-CMM[46], entre otros, los modelos CMMI son herramientas para ayudar a las organizaciones a mejorar sus procesos de desarrollo, adquisición y mantenimiento de sus productos y servicios. Los conceptos cubiertos por este modelo incluyen: Ingeniería de Sistemas, Ingeniería de Software, Desarrollo Integrado de productos y procesos, fuentes de suministro tales como los conceptos tradicionales de CMM, por ejemplo Gestión del proceso y gestión de proyectos.

Cada modelo CMMI ha sido diseñado para ser usado en concierto con otros modelos CMMI, facilitando a las organizaciones el mejoramiento del proceso. El modelo CMMI tiene dos tipos de representaciones, la versión escalonada (staged), la cuál se enfoca en la medición del proceso usando niveles de madurez, y la versión contínua la cual enfoca la medición del proceso para cada una de las áreas del proceso, usando niveles de capacidad

La suite de productos CMMI contiene y es producida desde un framework que provee la habilidad para generar múltiples modelos y los materiales asociados para el entrenamiento y evaluación. Estos modelos reflejan contenidos de los cuerpos de conocimiento (por ejemplo Ingeniería de Sistemas, Ingeniería de Software, Integración de proceso de desarrollo y producto) en las combinaciones necesarias. (CMMI-SE/SW, CMMI-SE/SW/IPDDS/SS).

Las organizaciones pueden usar un modelo CMMI para ayudar a fijar los objetivos de mejoramiento del proceso, mejorar los procesos y para proveer una guía para asegurar un proceso estable, capaz y maduro. Un modelo CMMI seleccionado promete servir como guía para el mejoramiento del proceso organizacional.

Dado que existen múltiples modelos CMMI disponibles generados a través del framework CMMI. Consecuentemente, debe existir la preparación para poder decidir cual es el más adecuado a las necesidades de mejoramiento de la organización. Se debe seleccionar una

representación, contínua o escalonada, y debe determinarse los cuerpos de conocimiento que desea incluir en el modelo que la organización utilizaría. Existen muchas razones válidas para seleccionar una representación u otra, el CMMI recomienda que la organización debe usar la que le sea más familiar con ella.

La representación contínua usa niveles de capacidad para medir el mejoramiento del proceso alcanzado, mientras la representación escalonada usa niveles de madurez. La principal diferencia entre niveles de madurez y de capacidad es la forma de abarcar la representación y cómo ellos son aplicados.

Los niveles de capacidad, los cuales corresponden a la representación contínua, aplican al proceso de la organización para cada una de las áreas del proceso. Hay seis niveles de capacidad numerados de 0 hasta el 5. Cada nivel de capacidad corresponde a un objetivo genérico y un conjunto de prácticas genéricas y específicas. Fácil migración de EIA/IS 731-1[47] y fácil comparación con ISO/IEC 15504 parte 2[30].

Los niveles de madurez, los cuales corresponden a la representación escalonada, aplican a la madurez de toda la organización. Hay cinco niveles de madurez nombrados de 1 hasta 5. Cada nivel de madurez comprende un conjunto predefinido de áreas del proceso. Permite la comparación con una gran cantidad de organizaciones y provee una fácil migración de SW-CMM.

La representación contínua tiene más prácticas específicas que la representación escalonada debido a que la representación contínua tiene dos tipos de prácticas específicas: básica y avanzada, mientras la representación escalonada tiene solamente un tipo de prácticas específica.

### *2.4.2 Disciplinas o cuerpos del conocimiento definidos por el CMMI*

Dependiendo de la disciplina que se seleccione para el modelo CMMI, este cubre unos propósitos particulares, esas disciplinas (las cuales son ampliadas en el modelo dado por el SEI) son:

***Ingeniería de Sistemas:*** cubre el desarrollo total de sistemas, los cuales pueden o no incluir software. Los ingenieros de sistemas se enfocan en transformar las necesidades de los clientes, las expectativas y las restricciones en productos de solución y soportan la solución de ese producto a través de su ciclo de vida. Cuando se selecciona ingeniería de sistemas el modelo CMMI deberá contener gestión de proceso, gestión de proyecto, soporte y áreas de proceso de ingeniería.

***Ingeniería de Software:*** cubre el desarrollo de sistemas software. Los ingenieros de software se enfocan sobre la aplicación sistemática, disciplinada y cuantificable orientada al desarrollo, operación, y mantenimiento de software. Cuando se selecciona esta disciplina en el modelo, este debe contener gestión de proceso, gestión de proyecto, soporte y áreas de proceso de ingeniería.

**Integración de Proceso de desarrollo y Producto:** es un enfoque sistemático para lograr una colaboración periódica de los participantes relevantes a lo largo de la vida del producto para satisfacer de las necesidades del cliente, las expectativas y los requisitos.

**Fuentes de suministro:** cuando el trabajo empieza a ser más complejo, los proyectos requieren suministros para garantizar algunas funciones o adicionar modificaciones a los productos que son específicamente necesarios para el proyecto. Cuando estas actividades son críticas los proyectos requieren de un análisis de las fuentes y del monitoreo de las actividades de suministros antes de liberar el producto. Esta disciplina cubre la adquisión de productos bajo estas circunstancias.

### 2.4.3 Estructura

La estructura del CMMI viene dada los componentes del modelo que las describen y las organizan, tanto en la representación contínua como escalonada, son: áreas de proceso, objetivos específicos, prácticas específicas, prácticas genéricas, productos de trabajo típicos, subprácticas, notas, amplificaciones de disciplina, elaboraciones de prácticas genéricas y referencias

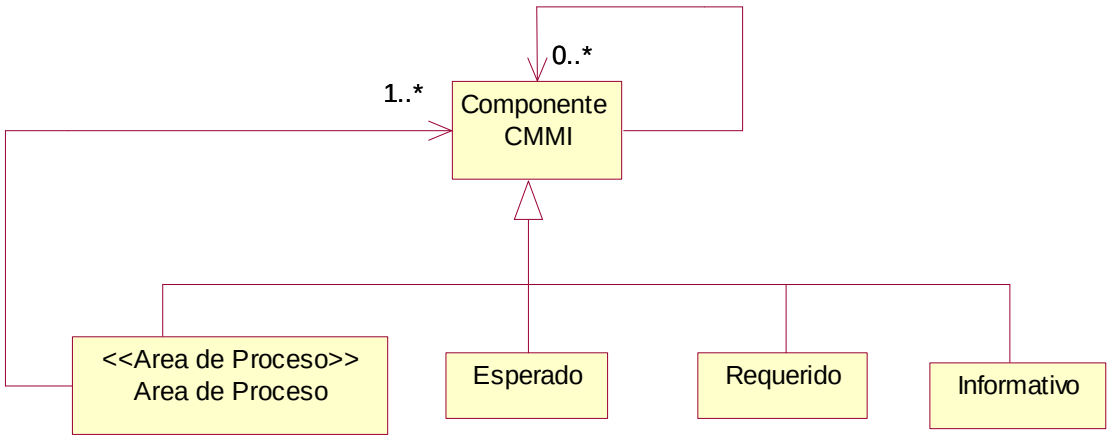



***Figura 7. Tipos y relaciones de componentes de CMMI (Abstracción del modelo CMMI)***

El componente primario del modelo es el Área de proceso (process area), un área de proceso es una agrupación de prácticas relacionadas en un área que cuando son implementadas colectivamente, satisfacen un conjunto de de objetivos considerados importantes para realizar mejoras significativas al área.

Los componentes de CMMI define en el marco de un área de proceso son:

Componente Requerido: describe los requisitos que la organización debe lograr para satisfacer un área de proceso. Este logro debe estar visiblemente implementado en el proceso de la organización. Los componentes requeridos en CMMI son los objetivos generales y específicos. La satisfacción de estos objetivos es utilizada en las evaluaciones para saber si el área de proceso ha alcanzado y satisfecho los logros.

Componente esperado: Un conjunto de componentes esperado describen lo que una organización debería tener implementado típicamente para alcanzar un componente requerido. Estos componentes sirven de guía a quienes evalúan o mejoran un proceso. Estos incluyen prácticas generales o específicas. Es importante saber que antes de que los objetivos se consideren satisfechos si se presentan estas prácticas descritas o prácticas alternativas en los planes y aplicaciones del proceso.

Componente Informativo: proveen detalle que ayuda a las organizaciones a iniciar a pensar en la forma de alcanzar los componentes requeridos y esperados. Son ejemplos de componentes

informativos: sub. prácticas, productos de trabajo típico, amplificaciones de disciplina, elaboraciones de prácticas genéricas, títulos de objetivos y prácticas, notas de objetivos y prácticas, y referencias.

El siguiente esquema utiliza los tipos de componentes como estereotipos UML para describir gráficamente la relación puntual de estos componentes del modelo:

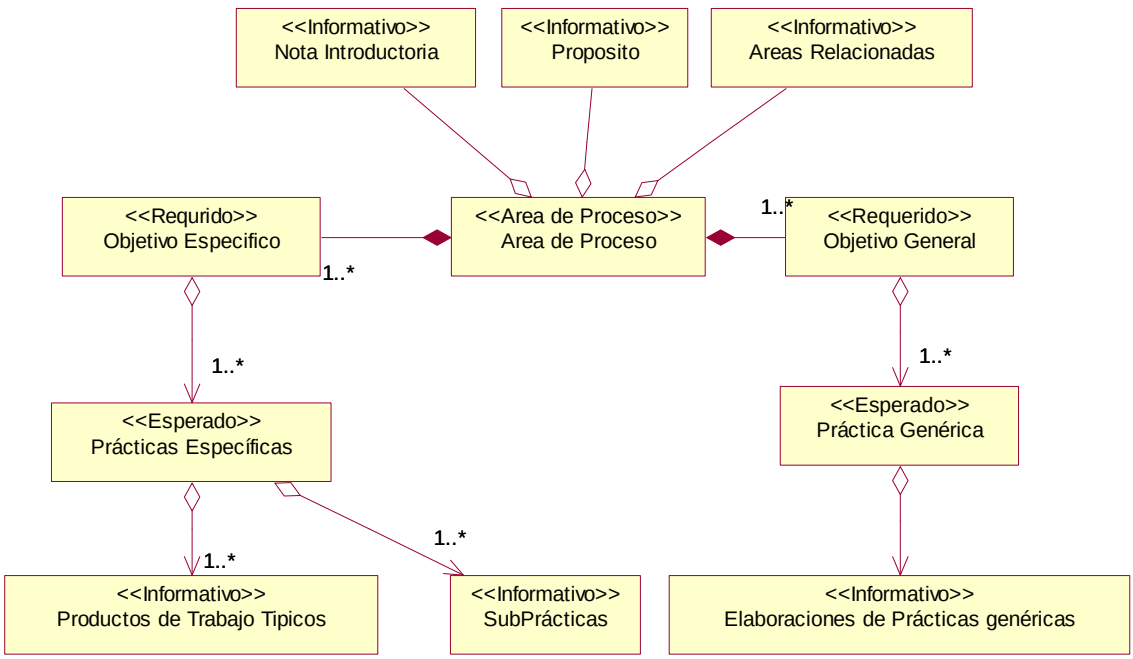



***Figura 8. Componentes del modelo CMMI***

### *2.4.4 Representaciones*

Las representaciones son formas de presentar los componentes CMMI, o de otro modo presentar las mejores prácticas que promueve. Esta representación puede ser de dos tipos: contínua o escalonada.

#### 2.4.4.1 La representación contínua

En esta representación se organiza en seis niveles de ***capacidad, perfiles de capacidad, nivel objetivo y nivel equivalente*** como principios organizacionales de los componentes del modelo. Esta representación agrupa las áreas del proceso por categorías afines y niveles de capacidad diseñados para mejorar el proceso dentro de cada área de proceso. Los perfiles de calidad representan caminos de mejoramiento del proceso para ilustrar la evolución del mejoramiento de cada una de las áreas del proceso. El nivel equivalente es usado para relacionar los niveles de

capacidad de las áreas del proceso a los niveles de madurez de la representación escalonada.

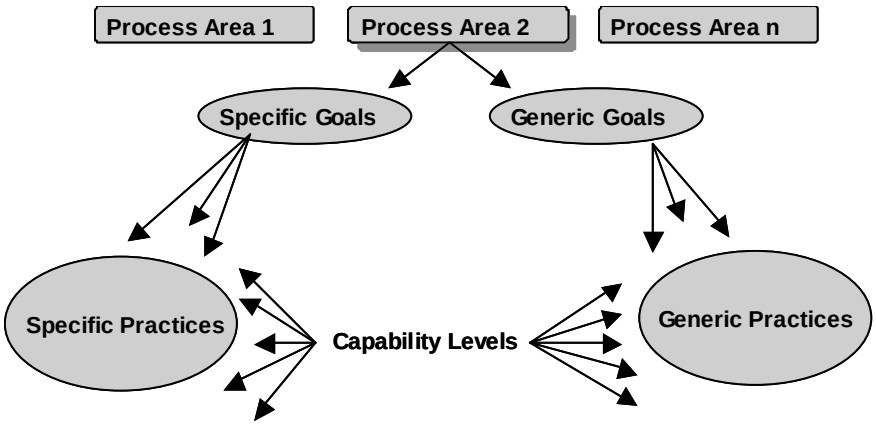



***Figura 9. Estructura de los componentes del modelo CMMI representación contínua.***

La capacidad hace referencia al logro de los genéricas y específicas que una organización ha alcanzado en un área de proceso específica. De acuerdo a este logro se obtiene un nivel de capacidad y de acuerdo a ese nivel, la organización puede aprovechar los beneficios de la mejora alcanzada. CMMI, define una escala jerárquica de 6 niveles que representan el incremento en las capacidades de las áreas de proceso en mejoramiento. De esta forma, el escalón mas bajo de la escala denota que la ejecución del proceso no cumple con el propósito del mismo, poniendo ello en riesgo la calidad del producto por él generado, mientras que el nivel más alto indica que su ejecución cumple ampliamente los objetivos del negocio, asegurando así la satisfacción y calidad del producto de software generado. La escala, queda definida por los siguientes 6 niveles expresados desde el de menor capacidad (nivel 0) al de mayor capacidad (nivel 5):

| Nivel de Capacidad | Nivel de capacidad en representación contínua |
|---|---|
| 0 | Incompleto |
| 1 | Realizado |
| 2 | Gestionado |
| 3 | Definido |
| 4 | Gestionado cuantitativamente |
| 5 | Optimizado |

***Tabla 4. Niveles de capacidad en el CMMI***

**Nivel 0:** Incompleto. Un proceso incompleto es un proceso no logrado o parcialmente logrado. Uno o más de los objetivos específicos del área del proceso no han sido satisfechos y no existen objetivos genéricos en este nivel y no hay razones para institucionalizar un proceso parcialmente logrado.

**Nivel 1:** Logrado. Un proceso logrado es un proceso que satisface los objetivos específicos del área del proceso. Este soporta y guía el trabajo necesario para poder construir los productos de trabajo.

**Nivel 2:** Administrado. Un proceso administrado es un proceso logrado que cuenta con la infraestructura básica para soportar el proceso. Este es planeado y ejecutado de acuerdo a políticas. Los empleados se adecuan a los recursos disponibles para lograr salidas de productos controladas, involucra stakeholders relevantes al proceso, es monitoreado, controlado y revisado; y es evaluado por su adherencia a la descripción de proceso.

**Nivel 3:** Definido. Un proceso definido es un proceso Administrado que es instanciado (Tailoring) de un conjunto de procesos estándares de la organización de acuerdo a unas guías de instanciación a la organización y contribuye con productos de trabajo, medidas, y otra información de mejoramiento a los activos de proceso organizacional.

**Nivel 4:** Administrado cuantitativamente. Un proceso administrado cuantitativamente es un proceso definido y controlado usando técnicas de estadísticas y otras técnicas cuantitativas. Se establecen objetivos cuantitativos de calidad y desempeño del proceso son usados como criterios para administrar el proceso. La calidad y el desempeño del proceso es entendido en términos estadísticos y son gestionados a través de la vida de los procesos.

**Nivel 5:** Optimizado. Un proceso optimizado es un proceso administrado cuantitativamente que es mejorado de acuerdo al entendimiento de las causas comunes de variación inherentes al proceso. El foco de un proceso optimizado es el mejoramiento contínuo del rango de desempeño del proceso a través de mejoras innovativas e incrementales.

#### 2.4.4.2 La representación escalonada

Esta representación organiza las áreas de proceso en cinco niveles de madurez para soportar y guiar el mejoramiento del proceso. La representación escalonadas agrupa las áreas del proceso por nivel de madurez, indicando cuales áreas del proceso implementar para alcanzar cada nivel de madurez. Los niveles de madurez representan un camino que ilustra la evolución de la organización completa de un trabajo de mejoramiento del proceso. Para cada área de proceso, los objetivos y prácticas específicos son listados primero, a partir de los objetivos y prácticas genéricas. Mientras la representación contínua usa objetivos genéricos para organizar las prácticas genéricas, la representación escalonada usa cuatro características comunes para organizar las prácticas genéricas: compromisos de la alta gerencia, habilidades a desarrollar, orientación de la implementación y verificación de la implementación.

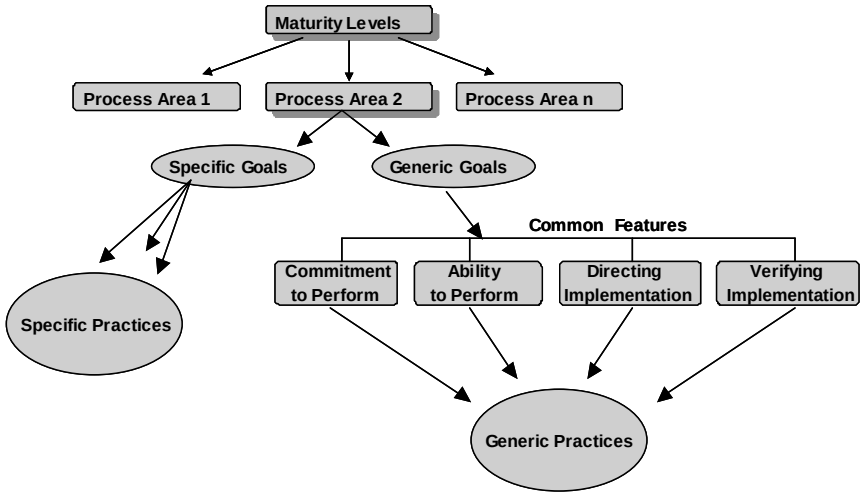



***Figura 10. Estructura de los componentes del modelo CMMI representación escalonada.***

Un nivel de madurez consiste de unas prácticas genéricas y específicas relacionadas para un conjunto de áreas de proceso predefinidas para mejorar el desempeño de toda la organización. El nivel de madurez de organización provee una forma de predecir el desempeño de la organización en una o varias disciplinas dadas. La experiencia, según el SEI, muestra que hay mejores resultados cuando la organización se concentra sus esfuerzos de mejora en un número manejable de áreas de proceso. Cada nivel de madurez establece una parte importante del proceso organizacional, y prepara a la organización para ir hacia el siguiente nivel de madurez. Los niveles de madurez son medidos por el logro de los objetivos específicos y los objetivos genéricos asociados con cada conjunto de áreas predefinidas.

| Nivel de Madurez | Nivel de Madurez en representación escalonada |
|---|---|
| 1 | Inicial |
| 2 | Gestionado |
| 3 | Definido |
| 4 | Gestionado cuantitativamente |
| 5 | Optimizado |

***Tabla 5. Niveles de Madurez del CMMI***

**Nivel de madurez 1: Inicial.** Los procesos son usualmente ad-hoc y caóticos. La organización usualmente no provee un ambiente estable para soportar los procesos. El éxito en la organización depende de las competencias y actos heroicos de la gente y no del uso de procesos probados. En medio de este caos, las empresas producen productos y servicios que trabajan, sin embargo, la producción se excede en sus costos y no cumple con los cronogramas. Una empresa en nivel 1 se caracteriza por la tendencia a sobrecargarse, abandonar procesos en tiempo de crisis y con muy baja capacidad de repetir sus éxitos.

**Nivel de madurez 2. Administrado**. Los proyectos de la organización aseguran que los requisitos son gestionados, y que los procesos son planeados, logrados, medidos y controlados. La disciplina del proceso en el nivel 2 ayuda asegurar la existencia de prácticas en los tiempos de estrés. Cuando las prácticas son las adecuadas, los proyectos son logrados y administrados de acuerdo a unos planes documentados. En el nivel de madurez 2, el estado de los productos de trabajo y los servicios es visible a la administración en puntos definidos (p.e. los hitos principales y la terminación de tareas principales). Se establecen los compromisos entre los participantes y son revisados cuando sea necesario. Los productos de trabajo son apropiadamente controlados. Los productos de trabajo y servicios satisfacen las descripciones, estándares y procedimientos especificados en su proceso.

**Nivel de madurez 3. Definido.** En este nivel, los procesos están bien caracterizados y entendidos, y son descritos en estándares, procedimientos, herramientas y métodos. El conjunto de procesos estándar de la organización, los cuales son la base para alcanzar el nivel de madurez 3, es establecido y mejorado en el tiempo. Estos procesos estándar son usados para proveer consistencia en la organización. Los proyectos establecen sus procesos definidos, instanciándolos (tailoring) a partir del conjunto de procesos estándar de la organización de acuerdo a unas guías de de instanciación (tailoring guidelines). Una distinción crítica entre los niveles de madurez 2 y 3 es el alcance de los estándares, descripciones de los procesos y procedimientos. En el nivel de madurez 2, estos pueden ser diferentes en cada instancia específica del proceso, en el nivel 3 estos, obligatoriamente, deberán ser instanciados a partir de los procesos estándares de la organización, excepto por las diferencias que permita la guía de instanciación. Otra diferencia significativa es que los procesos en el nivel 3 son definidos más rigurosamente. Un proceso definido claramente presenta el propósito, entradas, criterios de éxito, actividades, roles, medidas, pasos de verificación, salidas y criterios de éxito. En este nivel los procesos son gestionados más proactivamente usando la compresión de las interrelaciones entre las diferentes actividades del proceso y sus mediciones y los productos de trabajo y los servicios. En conclusión, el proceso es institucionalizado.

**Nivel de madurez 4. Cuantitativamente Administrado**. Se establecen para la empresa y para los proyectos objetivos cuantitativos de calidad y desempeño de los procesos y se utilizan esos objetivos como criterios de gestión del proceso. Estos objetivos están basados en las necesidades del cliente, usuarios finales, la organización y quienes implementan el proceso. La calidad y el desempeño son entendidos en términos estadísticos a través del ciclo de vida de los procesos. Esta información es incorporada en el repositorio de medidas de la organización para la toma de decisiones basada en hechos prácticos. A través de la información se pueden identificar causas y prevenir futuras ocurrencias. Una distinción crítica entre los niveles de madurez 3 y 4 es la de poder predecir el desempeño del proceso. El nivel 3 es tipicamente controlado solo de manera cualitativa.

**Nivel de Madurez 5. Optimizado.** La organización mejora continuamente mejora sus procesos basados en el conocimiento cuantitativo de las causas comunes de variación inherente en los procesos (debido a las interacciones normales entre procesos). El foco del nivel de madurez es el mejoramiento continuo del desempeño del proceso, a través de un proceso incremental e innovativo y de mejoras tecnológicas. Una distinción importante con respecto al nivel anterior es

la orientación del tipo de variación del proceso. En el nivel 4, la organización se orienta por las causas especiales de variación del proceso (son específicas a unas circunstancias transitorias y no a una parte inherente del proceso) y predice estadísticamente los resultados. Aunque los procesos pueden producir resultados predecibles, los resultados pueden resultar insuficientes para lograr los objetivos establecidos. En el nivel 5, la organización se interesa por orientarse a las causas comunes de variación del proceso y cambia los procesos para mejorar desempeño y lograr los objetivos de mejora del proceso establecidos de manera cuantitativamente.

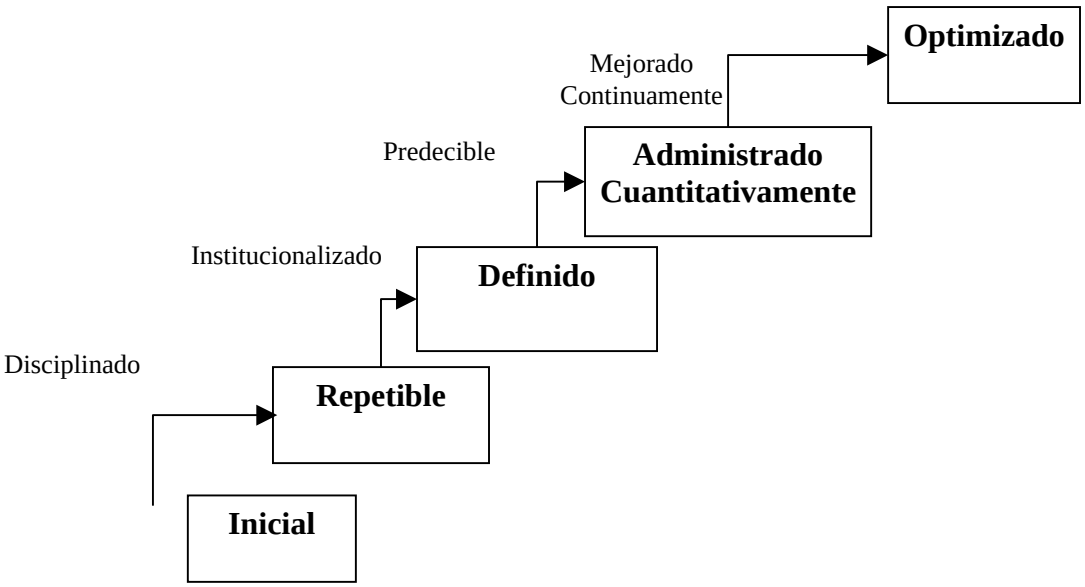



***Figura 11. Niveles de madurez del proceso***

### 2.4.4.3 Representación continua vs. representación escalonada

El SEI sugiere que para que la transición hacia el modelo sea lo más fácil posible a la organización, se seleccione la que sea más similar al modelo que han venido trabajando, si no se ha iniciado con ninguna cualquiera de las dos será una buena alternativa. La siguiente tabla compara las ventajas de cada una de las representaciones, las cuales pueden ayudar a tomar la decisión con respecto a cual seleccionar.

| Representación continua | Representación Escalonada |
|---|---|
| Da la libertad de seleccionar el orden de mejora que más se adecue a los objetivos de negocio de la empresa y a mitigar los riesgos en ciertas áreas de la organización. | Define a la organización un camino de mejora probado. |
| Da visibilidad de la capacidad lograda en cada área. | Se enfoca sobre un conjunto de procesos que proveen a una organización con una capacidad específica caracterizada por cada nivel de madurez. |
| Provee un puntaje del nivel de capacidad que es usada normalmente en la comunicación interna de la | Provee un puntaje del nivel de madurez que es frecuentemente usado en la comunicación interna |

| | |
|---|---|
| organización, pero que rara vez es comunicada externamente. | de la organización, y a nivel externo para poder calificar en licitaciones. |
| Permite mejoras de diferentes procesos a diferentes niveles de capacidad. | En resumen, la mejora se evalúa con un número – el nivel de madurez. |
| Es un nuevo enfoque que aún no tiene información suficiente de Retorno a la inversión (ROI). | Es un enfoque respaldado por un gran historial de uso, lo que incluye casos de estudio e información que demuestran un probado ROI. |
| Provee una fácil migración de SECM (EIA/IS 731) | Provee una fácil migración de CMM a CMMI |
| Permite una fácil comparación con el modelo de mejora de ISO/IEC 15504 debido a que la organización de las áreas del proceso es derivada de este modelo. | Permite la comparación a ISO/IEC 15504, pero la organización de las áreas del proceso no corresponde con ISO/IEC 15504. |

***Tabla 6. Ventajas de las dos representaciones***

La guía de uso del CMMI [48] describe tres factores que pueden influenciar la decisión de seleccionar una representación: el negocio, la cultura y los modelos heredados.

**Negocio:** En una organización con un conocimiento maduro de sus objetivos de negocio, es posible que tenga un mapeo entre estos y sus áreas de procesos. Una organización de este tipo puede resultarle útil usar la representación continua para evaluar sus procesos y determinar que tan bien estos soportan los objetivos de negocio. En una organización con enfoque de líneas de productos se ajusta más a la representación escalonada, pues esta le ayudará a seleccionar los procesos críticos para enfocar su mejoramiento. La misma organización puede optar por mejorar los procesos por cada línea de productos y obtener diferentes niveles de capacidad por cada línea de productos. El trabajo es conocer los objetivos de negocio y mirar su alineación con cada una de las representaciones y con esto decidir cuál es la mejor opción.

**Cultura:** los factores culturales tienen que ver con la forma en que será instalado el programa de mejora. Por ejemplo, se escogería la representación escalonada cuando una empresa tiene experiencia en la mejora de sus procesos o cuando tiene un proceso específico que necesita ser mejorado rápidamente. Una organización con poca experiencia debería escoger la representación escalonada, pues establece un camino de mejoramiento.

**Modelos heredados:** si una empresa tiene experiencia con alguna de las representaciones, lo mejor es que siga utilizando la misma representación.

### *2.4.5 Institucionalización del proceso*

La institucionalización es un concepto importante en el proceso de mejora. Cuando se menciona en los objetivos y prácticas generales (GG y GP), la institucionalización está asociada a la forma en que el trabajo es desarrollado y si cumple y es consistente al proceso ejecutado. El grado de institucionalización es capturado en los objetivos genéricos así:

Proceso logrado – GG1, Proceso Gestionado – GG2, Proceso Definido GG3, Proceso Gestionado cuantitativamente – GG4 y Optimizado GG5. Las descripciones de estos procesos corresponden con los cinco niveles de madurez definidos anteriormente.

A continuación se listan los objetivos y prácticas genéricas del CMMI

GG1 Alcanzar los objetivos específicos

"El proceso soporta y permite alcanzar los objetivos de las áreas de proceso a través de la transformación de productos de trabajo identificados de entrada para producir unos productos de trabajo identificados de salida"

GP 1.1 Aplicar las prácticas base

"Aplicar las prácticas base del área del proceso para desarrollar los productos de trabajo y servicios para lograr los objetivos específicos del área"

GG2 Institucionalizar un proceso gestionado

"El proceso es institucionalizado como un proceso gestionado"

GP. 2.1. Establecer una política organizacional

"Establecer y mantener una política organizacional para planear y ejecutar el proceso"

GP. 2.2. Planear el proceso

"Establecer y mantener un plan para la ejecución del proceso"

GP. 2.3. Proveer recursos

"Proveer los recursos adecuados para ejecutar el proceso, desarrollar los productos de trabajo y proveer los servicios del proceso"

GP. 2.4. Asignar responsabilidad

"Asignar responsabilidad y autoridad para ejecutar los procesos, desarrollar los productos de trabajo y proveer los servicios del proceso."

GP. 2.5. Entrenar el personal

"Entrenar el personal que participará en o soportará los procesos cuando sea necesario"

GP. 2.6. Gestionar Configuraciones

"Asignarle a los productos de trabajo designados del proceso los niveles apropiados de gestión de configuración"

GP. 2.7. Identificar e involucrar los participantes relevantes

"Identificar e involucrar los participantes relevantes de acuerdo a lo planeado"

GP. 2.8. Seguir y controlar el proceso

"Seguir y controlar el proceso de acuerdo al plan de ejecución del proceso y tomar las acciones correctivas apropiadas"

GP. 2.9. Evaluar objetivamente la adherencia

"Evaluar objetivamente la adherencia de los procesos con respecto a su descripción, estándares y procedimientos y orientar las no conformidades"

GP. 2.10. Revisar el estado con el más alto nivel de gestión

Revisar las actividades, el estado, y los resultados del proceso con el más alto nivel de gestión para resolver inconvenientes.

GG 3. Institucionalizar un proceso definido

"El proceso es institucionalizado como un proceso definido"

GP. 3.1. Establecer un proceso definido

"Establecer y mantener la descripción de un proceso definido"

GP. 3.2. Recolectar información de mejoramiento

"Recolectar información de productos de trabajo, mediciones, resultados de mediciones y de mejoramiento derivada de la planeación y ejecución de procesos para soportar el uso futuro y el mejoramiento de activos de procesos y procesos de la organización"

GG4. Institucionalizar un proceso gestionado cuantitativamente

"El proceso es institucionalizado como un proceso gestionado cuantitativamente"

GP. 4.1. Establecer objetivos cuantitativos para el proyecto

"Establecer y mantener objetivos cuantitativos para el proceso que se orienten hacia la calidad y desempeño de los procesos basados en las necesidades de los clientes y en los objetivos de negocio"

GP. 4.2. Estabilizar el desempeño de los subprocesos

"Estabilizar el desempeño de uno o más subprocesos para determinar la habilidad del proceso para lograr los objetivos de calidad y de desempeño del proceso establecidos"

GG5. Institucionalizar un proceso optimizado

"El proceso es institucionalizado como un proceso optimizado"

GP. 5.1. Asegurar el mejoramiento continuo

"Asegurar el mejoramiento continuo de los procesos en total cumplimiento con los objetivos de negocio de la organización"

GP. 5.2. Corregir de raíz las causas de los problemas

"Identificar y corregir de raíz las causas de los defectos y otros problemas de los procesos"

Las prácticas genéricas GP, son implementadas a través de una o un conjunto de áreas de proceso. Por ejemplo la práctica genérica GP 2.6. Gestionar configuraciones es resuelta completamente con las prácticas específicas del área de proceso Gestión de la Configuración.

### *2.4.6 Terminología de CMMI*

En cualquier modelo CMM, la terminología usada y cómo es definida es importante para entender el contenido. A continuación se presentan los principales conceptos del CMMI:

**Suite de productos CMMI:** es el conjunto completo de productos desarrollados alrededor del concepto de CMMI. Estos productos incluyen el Framework mismo, los modelos, los métodos de evaluación, los materiales de evaluación y varios tipos de entrenamiento que son producidos desde el Framework CMMI.

**Framework CMMI:** es la estructura básica que organiza los componentes CMMI, incluyendo los elementos comunes a los modelos CMMI actuales, así como las reglas y métodos para generar

los modelos, los métodos de evaluación(incluyendo los artefactos asociados) y los materiales de entrenamiento. El Framework está habilitado para que nuevas disciplinas se adicionen al CMMI, así como la integración de estas nuevas disciplinas a las existentes.

**Modelo CMMI:** Son los generados a través del Framework, son múltiples modelos. El modelo es un conjunto de información que representa la base de referencia para el mejoramiento del proceso. La frase "Modelos CMMI" se refiere a uno, algunos, o a una colección entera de modelos posibles que pueden ser mejorados desde el Framework.

**Revisión de pares:** este término es utilizado en la suite de productos CMMI en lugar del término inspección de producto de trabajo. Esencialmente, estos términos significan la misma cosa. Una revisión de pares es la revisión de un producto de trabajo realizado por pares durante el desarrollo de los productos de trabajo para identificar y remover defectos.

**Conjunto de procesos estándar de la organización:** contiene las definiciones de los procesos que guían todas las actividades de la organización. Un proceso estándar brinda unas actividades consistentes de desarrollo y mantenimiento a través de la organización y los elementos esenciales para el mejoramiento y la estabilidad a largo plazo.

**Proceso:** consta de actividades que pueden ser reconocidas como implementaciones de prácticas en un modelo CMMI. Estas actividades pueden ser mapeadas a una o más prácticas en las áreas de proceso del modelo dado para ser usadas por un evaluador del proceso y de su mejora.

**Proceso gestionado:** es un proceso realizado que es planeado y ejecutado de acuerdo con una política; los empleados tienen los suficientes recursos para producir las salidas controladas, envuelve los participantes relevantes, es monitoreado, controlado y revisado y es evaluado a sus descriptores.

**Proceso Definido:** es un proceso gestionado que es delimitado desde un conjunto de procesos estándar de acuerdo las líneas guías definidas para la organización. Un proceso definido para u proyecto provee una base para planear, realizar y mejorar las actividades y tareas del proyecto.

**Activos del proceso organizacional:** son artefactos relacionados a la descripción, implementación y mejoramiento del proceso (por ejemplo políticas, mediciones, descripciones de procesos, y herramientas de soporte a la implementación del proceso). Incluyen: conjunto de procesos estándares (incluso la arquitectura del proceso y los elementos del proceso), descripciones de modelos de ciclo de vida apropiados a la organización, líneas guías y criterios para utilizar los procesos, el repositorio de métricas de la organización y la librería de activos de los procesos. Las líneas base del proceso y los modelos de desempeño del proceso también son activos del proceso.

**Arquitectura de proceso:** describe la organización, interfaces, interdependencias y otras relaciones entre elementos del proceso dentro de un proceso estándar, así como con otros procesos externos.

**Ciclo de vida del producto:** es el período de tiempo, consistente en fases que inicia cuando el producto es concebido y termina cuando el producto no está disponible para su uso (muerte). Dentro de la organización deben ser producidos múltiples productos para múltiples clientes, una descripción de ciclo de vida de un producto no es adecuada, por ello la organización debe definir un conjunto de modelos de ciclo de vida apropiados a la empresa.

**Repositorio de métricas de la organización:** es utilizado para coleccionar y brindar información de medida del proceso y de los productos de trabajo, particularmente los relacionados a los

procesos estándar de la organización.

**Librería de activos del proceso de la organización**: es una librería de información para almacenar y disponer de los activos de proceso de la organización. Esta librería contiene activos de procesos tales como documentos, fragmento de documentos, implementaciones de procesos, entre otros.

**Documento:** colección de información que tiene permanencia y que puede ser leído por humanos y máquinas, esto incluye documentos electrónicos e impresos.

### *2.4.7 Áreas de Proceso*

Las áreas de proceso se pueden agrupar de diferentes maneras, por ejemplo se pueden organizar por la afinidad entre estas, por el nivel de madurez en el que se ubican o de la forma en que una empresa lo considere útil de acuerdo a sus objetivos de negocio.

Las áreas de proceso agrupadas por la afinidad se pueden clasificar de la siguiente forma (representación continua):

Áreas de gestión de proceso

Enfoque hacia el proceso de la organización (OPF)

Definición del proceso organizacional(OPD)

Entrenamiento organizacional(OT)

Desempeño del proceso organizacional(OPP)

Innovación y despliegue organizacional(OID)

Áreas de gestión de Proyectos

Planificación de proyectos(PP)

Seguimiento y Control de proyectos(PMC)

Gestión de la subcontratación (SAM)

Gestión integrada de proyectos(IPM)

Gestión del riesgo(RSKM)

Equipo Integrado(IT)

Gestión integrada de suministros(ISM)

Gestión cuantitativa del proyecto (QPM)

Áreas de Ingeniería

Desarrollo de requisitos(RD)

Administración de requisitos(REQM)

Solución Técnica(TS)

Integración de producto(PI)

Verificación(VER)

Validación(VAL)
Áreas de soporte
Gestión de la Configuración (CM)
Aseguramiento de calidad de producto y proceso (PPQA)
Medición y Análisis (MA)
Ambiente Organizacional para la integración (OEI)
Análisis y resolución de decisiones (DAR)
Análisis y resolución causal(CAR)

Agrupadas por niveles las áreas son (Representación escalonada):
Nivel de madurez 1 Inicial. Ninguna
Nivel de madurez 2. Logrado
Planificación de proyectos(PP)
Seguimiento y Control de Proyectos(PMC)
Administración de requisitos(REQM)
Gestión de la Configuración (CM)
Aseguramiento de calidad de producto y proceso (PPQA)
Gestión de la subcontratación (SAM)
Medición y Análisis (MA)
Nivel de madurez 3. Definido
Desarrollo de requisitos(RD)
Solución Técnica(TS)
Integración de producto(PI)
Verificación(VER)
Validación(VAL)
Gestión del riesgo(RSKM)
Enfoque hacia el proceso de la organización (OPF)
Definición del proceso organizacional(OPD)
Entrenamiento organizacional(OT)
Análisis y resolución de decisiones (DAR)
Gestión integrada de proyectos(IPM)
Equipo Integrado(IT)

Gestión integrada de suministros(ISM)

Ambiente Organizacional para la integración (OEI)

Nivel de madurez 4. Administrado cuantitativamente

Desempeño del proceso organizacional (OPP)

Gestión cuantitativa del proyecto (QPM)

Nivel de madurez 5. A

Innovación y despliegue organizacional(OID)

Análisis y resolución causal (CAR)

#### 2.4.7.1 Áreas del proceso relacionadas con la gestión de proyectos

Estas áreas cubren las actividades de gestión de proyectos relacionadas con la planeación, monitoreo y control del proyecto. Estas se describen a través de la siguiente tabla 7.

| **Áreas del proceso relacionadas con la gestión de proyectos** | | |
|---|---|---|
| **Área** | **Propósito** | **Descripción** |
| Planificación de proyectos (PP) | Establecer y mantener planes que definan las actividades del proyecto. | La planificación empieza con los requisitos del proyecto e incluye estimar atributos del producto y de las tareas, determinar los recursos necesarios, negociar los compromisos, realizar cronogramas e identificar y analizar los riesgos del proyecto. El plan provee la base para ejecutar y controlar las actividades del proyecto para cumplir con el compromiso establecido con el cliente. Esto involucra desarrollar un plan, teniendo en cuenta la interacción con todos los participantes, obtener los compromisos del plan y mantenerlo a lo largo del proyecto. |
| Seguimiento y Control de Proyectos (PMC) | Proveer la información acerca del progreso del proyecto para poder tomar las acciones correctivas apropiadas cuando la ejecución del proyecto se desvía significativamente del plan. | La base del seguimiento y control, es un plan de proyecto documentado, a partir del cual comunica el estado y se toman planes correctivos. El progreso se determina básicamente comparando el trabajo real a partir de los atributos de productos y tareas con respecto al trabajo ideal definido en el cronograma o el diagrama de seguimiento seleccionado. Una desviación significativa (si no es resuelta amenaza los objetivos del proyecto) deberá ser detectada y corregida oportunamente. Normalmente se requiere revisar los planes y adecuarlos al desarrollo del proyecto |
| Gestión de la Subcontratación (SAM) | Gestionar la adquisición de productos desde proveedores con los cuales existe un contrato formal. | Esta área aplica principalmente a la adquisición de productos y componentes que son liberados al cliente del proyecto. Para minimizar los riesgos del proyecto, esta área también debe aplicarse cuando los productos no son liberados al cliente (procesos, ambientes de desarrollo, herramientas de pruebas, etc.). Esta área no aplica cuando el proveedor hace parte del proyecto. |
| Gestión integrada | Establecer y mantener el | Involucra establecer el proceso definido del proyecto a partir |

| | | |
|---|---|---|
| de proyectos<br>IPM | proyecto y el contexto de los participantes relevantes de acuerdo a un proceso definido e integrado que es instanciado (tailored) desde un conjunto de procesos estándares de la organización. | de los procesos estándares de la empresa, gestionar el proyecto usando el proceso definido para el proyecto, usar y contribuir a los activos del proceso organizacional, permitir a los intereses relevantes de cada participante: ser identificados, considerados y si es apropiado, orientarlos durante el desarrollo del producto (Vistas por roles). También involucra asegurar que los participantes realizan sus tareas de manera coordinada y oportuna para: (1) lograr las metas del producto y del proyecto, (2) Cumplir los compromisos y (3) Identificar, seguir y resolver inconvenientes. |
| Gestión del riesgo(RSKM) | Identificar problemas potenciales antes de que estos ocurran con el fin de planear y realizar actividades de manejo de riesgo. | Se aplica un enfoque continuo de gestión del riesgo para anticipar y mitigar efectivamente los riesgos que puedan tener un impacto crítico sobre el proyecto. Esta gestión incluye una identificación temprana y agresiva del riesgo con la participación de todos los participantes involucrados, para lo cual el líder deberá proveer un ambiente abierto y libre para su discusión. La gestión del riesgo puede ser dividida en tres partes: definir la estrategia; identificar y analizar riesgos, y manejar los riesgos identificados, incluyendo los planes de mitigación del riesgo cuando estos lo ameriten. |
| Equipo Integrado(IT) | Formar y solventar un grupo integrado para el desarrollo de los productos de trabajo. | Un equipo integrado está compuesto de participantes quienes generan e implementan decisiones para el producto de trabajo en desarrollo. Los miembros del equipo son colectivamente responsables de la liberación del producto de trabajo. Esta área busca que los miembros tengan las habilidades y experticia necesarias, puedan seguir todo el ciclo de vida del producto, se colaboren interna y externamente, se entiendan con unos objetivos y tareas comunes, se conduzcan a sí mismos(independencia) de acuerdo a unos principios operativos y reglas fundamentales. |
| Gestión integrada de suministros(ISM) | Identificar proactivamente fuentes de productos que puedan ser usadas para satisfacer los requisitos del proyecto y gestionar los proveedores seleccionados mientras se mantiene una relación cooperativa proyecto-proveedor. | Esta área se construye sobre el área de Gestión de la Subcontratación, pero adicionando las prácticas que enfatizan en la relación cooperativa entre los proveedores. Está diseñada para las situaciones en las cuales los proyectos usan proveedores para realizar las funciones que son críticas para el éxito del proyecto. Esta área involucra actividades tales como: identificar, analizar y seleccionar fuentes potenciales de productos; evaluar y determinar las fuentes usadas para la adquisición de productos, monitorear y analizar los proveedores seleccionados, evaluar estos productos y revisar los contratos o las relaciones con los proveedores cuando sea necesario. |
| Gestión cuantitativa del proyecto (QPM) | Gestionar cuantitativamente el proceso definido para el proyecto para lograr los objetivos establecidos de calidad y desempeño del | Involucra establecer estos objetivos, identificar los subprocesos que componen el proceso del proyecto basado en la estabilidad y capacidad histórica de la información encontrada en las líneas base o modelos de desempeño del proceso. Se deben seleccionar estos subprocesos de manera estadística. Se hace seguimiento y control cuantitativo, para lo |

| | | |
|---|---|---|
| | proceso. | cual se selecciona técnicas de medida y análisis para la gestión de los subprocesos. Se definen las variaciones de estos procesos, se monitorea el desempeño de los procesos en la satisfacción de los objetivos cuantitativos. Se registra la información de gestión de calidad y las estadísticas en el repositorio de medida de la organización. |

***Tabla 7. Áreas de proceso relacionadas con la gestión de proyectos***

Las relaciones entre estas áreas y el resto de áreas del proceso se visualizan a través de las figuras 12 y 13, donde se han separado las áreas en dos grupos: en la figura 12 se muestran las áreas básicas (actividades básicas de gestión), y en la figura 13 las áreas avanzadas (establecer un proceso definido a partir de la instanciación del proceso de la organización).

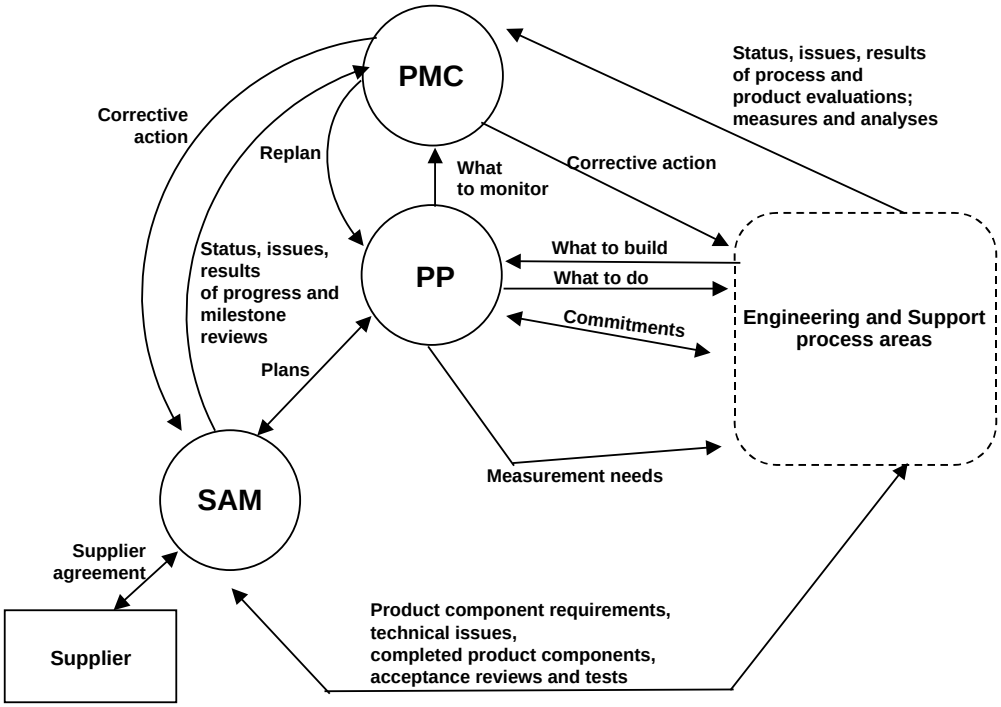



***Figura 12. Relaciones entre las áreas básicas de gestión de proyectos y el resto de áreas***

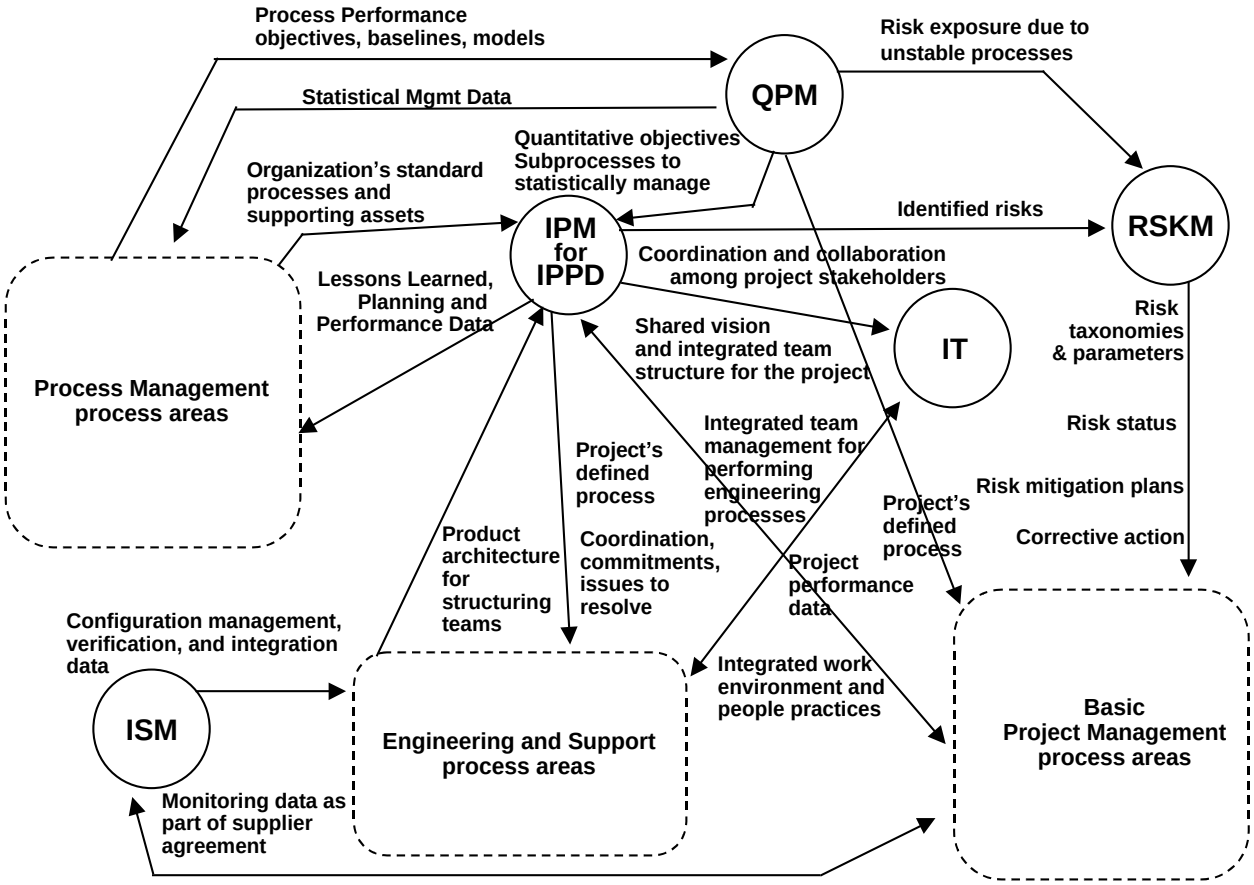



***Figura 13. Relaciones entre las áreas avanzadas de gestión de proyectos y el resto de áreas.***

### 2.4.7.2 Áreas del proceso relacionadas con el proceso de ingeniería

Estas áreas cubren las actividades de desarrollo y mantenimiento que son compartidas a través de todas las disciplinas de ingeniería. Estas áreas integran Ingeniería del Software e Ingeniería de Sistemas en un solo proceso de desarrollo de productos, soportando una estrategia de desarrollo orientada al producto. Estas áreas aplican a cualquier producto o servicio de tipo ingenieril (software, hardware, servicios o procesos).

| Áreas del proceso relacionadas con la gestión de proyectos | | |
|---|---|---|
| **Área** | **Propósito** | **Descripción** |
| Desarrollo de | Producir y analizar los | Los requisitos deben ser elicitados, analizados, validados a |

| | | |
|---|---|---|
| Requisitos<br>(RD) | requisitos del cliente, del producto y de componentes del producto. | partir de las necesidades, expectativas y restricciones del cliente. Se debe recolectar y coordinar las necesidades de los participantes, desarrollar el ciclo de vida de los requisitos del producto. Se deben establecer los requisitos iniciales del producto y de los componentes de manera consistente con los requisitos del cliente. Esta área se orienta a los requisitos del cliente antes de los requisitos a nivel de productos. Los requisitos del producto son refinados a partir de los requisitos del cliente y derivados de la solución de diseño seleccionada. Los requisitos son identificados , definidos y actualizados a través de las fases del ciclo de vida del producto. |
| Administración de requisitos(REQM) | Administrar los requisitos de los productos y componentes de productos del proyecto e identificar consistencias entre los requisitos y los planes del proyecto y los productos de trabajo. | Está área es la encargada de gestionar todos los requisitos recibidos o generados por el proyecto, incluyendo requisitos técnicos y no técnicos. Cuando el área de desarrollo genera requisitos de producto o de componente de producto, estos también deberán ser administrados por esta área. El proyecto sigue los pasos apropiados para asegurar que hay una adecuada gestión de requisitos para soportar las necesidades de planeación y ejecución del proyecto. Cuando un proyecto recibe los requisitos desde un proveedor de requisitos, los requisitos son revisados con el proveedor para resolver problemas y malos entendidos antes de ser incorporados a los planes del proyecto. Una vez son incorporados el proyecto deberá gestionar el cambio en estos cuando se identifiquen inconsistencias entre planes, productos de trabajo y requisitos. Parte de la gestión de requisitos es documentar sus cambios y sus razones, y mantener la trazabilidad bidireccional entre fuentes de requisitos y todos los requisitos del producto y componentes de producto. |
| Solución Técnica(TS) | Diseñar, desarrollar e implementar soluciones de acuerdo a los requisitos. Las soluciones, diseños e implementaciones se refieren a componentes de producto, productos, procesos relacionados con el ciclo de vida del producto o combinaciones apropiadas de estos. | Es un área aplicable a cualquier nivel: Arquitectura de producto, componente de producto, producto, proceso relacionado al ciclo de vida del producto y servicio. Esto incluye evaluar y seleccionar soluciones que satisfagan potencialmente los requisitos definidos, desarrollar diseños detallados de las soluciones seleccionadas e implementar los diseños como productos o componentes de producto. Normalmente estas actividades se soportar interactivamente unas a otras. Se pueden utilizar prototipos para ir adquiriendo conocimiento de la solución técnica o de un paquete completo de requisitos. |
| Integración de producto(PI) | Ensamblar el producto desde los componentes del producto, asegurar que el producto integrado funciona adecuadamente para ser entregable. | Está orientada a la integración de componentes en componentes más complejos o en un producto completo. El alcance de esta área es lograr la integración del producto a través de un ensamble progresivo e iterativo de componentes, de una manera incremental de acuerdo a una secuencia y unos procedimientos de integración definidos. Un aspecto crítico en el área es la gestión de las interfaces internas y externas de los productos y sus componentes para asegurar su |

| | | |
|---|---|---|
| | | compatibilidad. |
| Verificación(VER) | Asegurar que los productos de trabajo seleccionados cumplen con los requisitos especificados. | Incluye la preparación y ejecución de la verificación, así como la identificación de acciones correctivas. Esta se aplica tanto al producto como a los productos de trabajo intermedios de acuerdo a los requisitos del cliente, del producto y de los componentes del producto. La verificación es un proceso incremental debido a que ocurre a través de todo el proceso de desarrollo, pues se aplica a los requisitos, productos de trabajo, componentes de productos y productos. |
| Validación(VAL) | Demostrar que el producto o componente del producto cumple completamente con su intensión de uso cuando es instalado en su ambiente de trabajo. | Es aplicable a todos los aspectos del producto en cualquiera de sus ambientes de trabajo, tales como operación, entrenamiento, manufactura, mantenimiento y servicio de soporte. Los métodos de validación deben ser aplicados tanto a los productos de trabajo como a los productos y a los componentes de productos. El ambiente de validación debe representar el ambiente de trabajo sobre el cual se quiere probar. Mientras la verificación se orienta a que los productos de trabajo reflejen los requisitos especificados, la verificación demuestra que este cumple completamente con la intensión de uso. Las actividades de Validación son similares a las de verificación, pero en estas, frecuentemente son involucrados los usuarios finales. Puede ejecutarse en conjunto con la verificación y usar el mismo ambiente de prueba. |

***Tabla 8. Áreas de proceso relacionadas con el proceso de ingeniería***

Las relaciones entre estas áreas y el resto de áreas del proceso se visualizan en la figura 14.

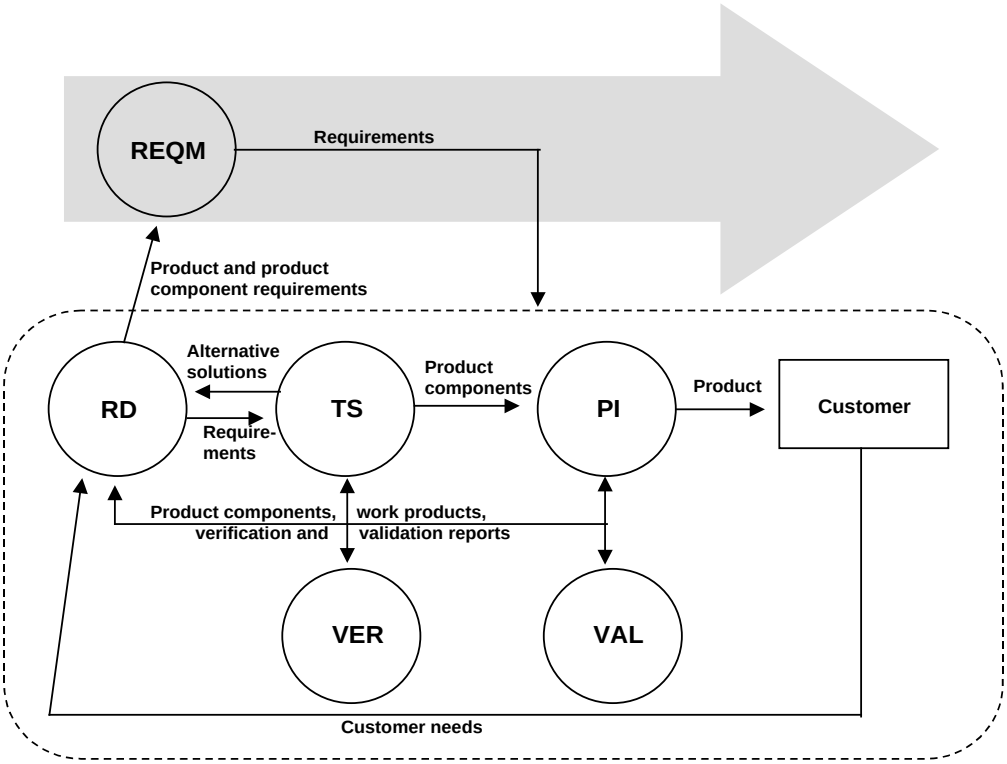



***Figura 14. Relaciones entre las áreas de ingeniería y el resto de áreas.***

2.4.7.3 Áreas del proceso relacionadas con los procesos de Soporte

Estas áreas cubren las actividades que soportan el desarrollo y mantenimiento del producto. Se orientan a procesos que son utilizados en el contexto de ejecución de otros procesos. La tabla 9 las presenta de manera resumida:

| **Áreas del proceso relacionadas con la gestión de proyectos** | | |
|---|---|---|
| **Área** | **Propósito** | **Descripción** |
| Gestión de la Configuración (CM) | Establecer y mantener la integralidad de los productos de trabajo usando identificación de la configuración, control de la configuración, estado de cuenta de la configuración, y las auditorias de configuración. | El área involucra identificar la configuración de los productos de trabajo seleccionados para realizar la línea base en ciertos momentos. Controlar los cambios de los ítems de configuración, construir o proveer especificaciones para productos de trabajo desde un sistema de gestión de la configuración. Mantener la integridad de las líneas base y proveer el estado de cuenta y la información de la configuración a los desarrolladores, usuarios finales y los clientes. |
| Aseguramiento de calidad de producto y proceso (PPQA) | Proveer la gestión y el poder a un grupo de trabajo para revisar internamente procesos y productos de trabajo. | Esta área involucra evaluar objetivamente los procesos, productos de trabajo, servicios. Identificar y documentar no conformidades. Proveer retroalimentación al grupo del proyecto y a los directores sobre los resultados de las actividades de calidad y asegurar que las no conformidades sean resueltas. Mientras el área de Verificación asegura que los requisitos especificados son satisfechos, ésta área asegura que los procesos planeados son realizados. |
| Medición y Análisis (MA) | Desarrollar y sostener una capacidad de medición que se utiliza para soportar las necesidades de información de | Se debe especificar los objetivos de la medición y el análisis de acuerdo a los objetivos y necesidades de información identificados. Se debe especificar las medidas, recolección de datos, mecanismos de almacenamiento, técnicas de análisis, y |

| | | |
|---|---|---|
| | la gestión. | mecanismo de reporte y de retroalimentación. Implementar la recolección, almacenamiento, análisis y reporte de la información. Proveer resultados objetivos que pueden ser usados para tomar las acciones correctivas apropiadas. |
| Ambiente Organizacional para la integración (OEI) | Proveer una infraestructura y personal de gestión para la integración del producto y el proceso de desarrollo (IPPD). | La organización es un sistema integrado capaz de proveer y sostener las personas, los productos y procesos necesarios para una ejecución efectiva y eficiente de sus proyectos. La organización debe proveer los estímulos necesarios para promover la excelencia personal y de los grupos. Una característica importante de ambientes efectivos para la integración incluye gente entrenada para explotar el ambiente colaborativo; un lugar de trabajo que provea los recursos para maximizar la productividad de la gente y facilitar los grupos integrados; y un conjunto de activos de procesos estándar y organizacionales que culturalmente dispongan un ambiente IPPD que promueva y premie la excelencia de grupos, así como la individual. |
| Análisis y resolución de decisiones (DAR) | Analizar posibles decisiones utilizando un proceso de evaluación formal que evalúa alternativas identificadas a partir de unos criterios establecidos. | Involucra establecer guías para determinar a qué se le puede aplicar, así como la aplicación de, un proceso de evaluación formal. Un proceso de evaluación formal es un enfoque estructurado de evaluar alternativas de solución de acuerdo a unos criterios establecidos, para determinar una solución recomendada. Esta área incluye establecer los criterios, identificar las alternativas de solución, seleccionar los métodos de evaluación, evaluar las alternativas usando los criterios y métodos establecidos y seleccionar las soluciones recomendadas de acuerdo a los resultados de la evaluación. |
| Análisis y resolución causal(CAR) | Identificar las causas de defectos y otros problemas y tomar la acción de prevenir su ocurrencia en el futuro. | Esta involucra identificar y analizar las causas de defectos y otros problemas, así como tomar las acciones específicas para eliminar las causas y prevenir la ocurrencia de este tipo de defectos o de problemas en el futuro. El análisis y la resolución causal mejora la calidad y la productividad previniendo la introducción de defectos en el producto, estos son un mecanismo de comunicar lecciones aprendidas entre los proyectos. Esta área nos solo aplica a defectos sino a otros problemas, por ejemplo puede ser utilizado para mejorar atributos de calidad (desempeñeo). |

***Tabla 9. Áreas de Soporte***

Las relaciones entre estas áreas y el resto de áreas del proceso se visualizan en las figura 15 y 16.

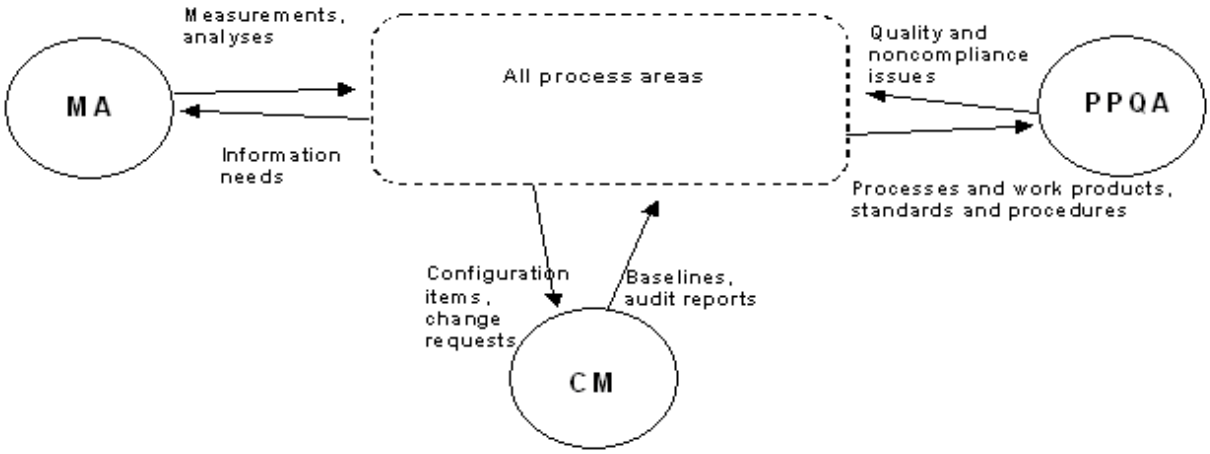



***Figura 15. Relaciones entre las áreas básicas de soporte y el resto de áreas.***

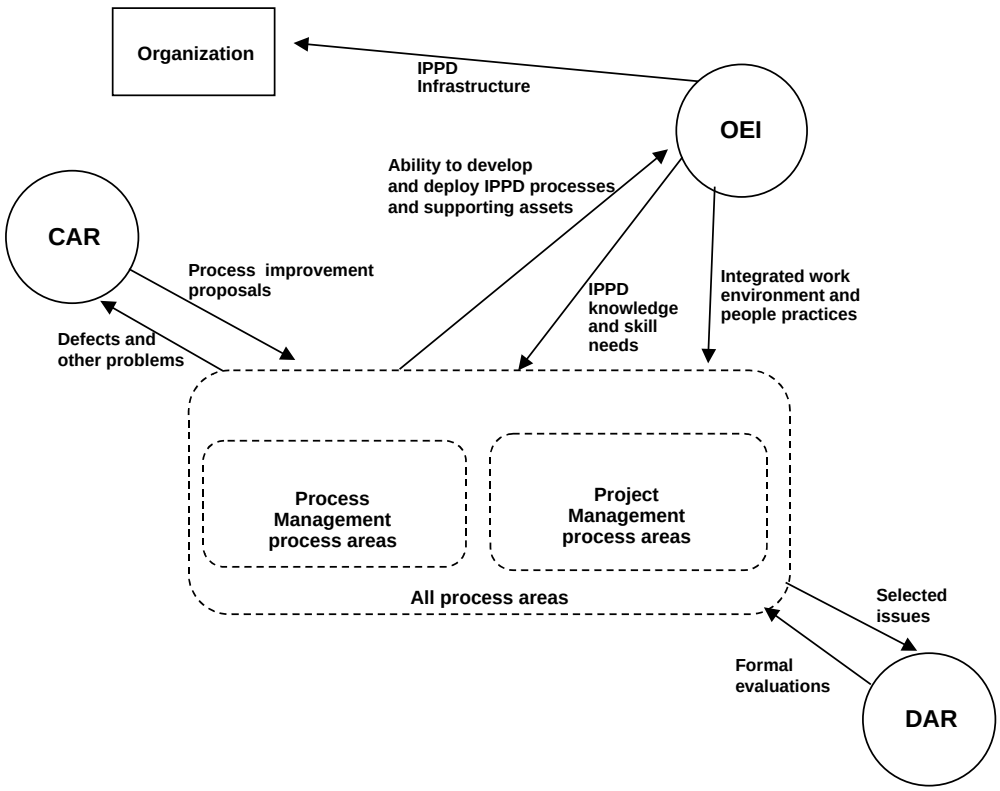



***Figura 16. Relaciones entre las áreas avanzadas de soporte y el resto de áreas.***

#### 2.4.7.4 Áreas del proceso relacionadas con la gestión de procesos

Estas áreas cubren las actividades a través de todos los proyectos relacionadas con la definición, planificación, despliegue, implementación, monitoreo, control, evaluación, medida y mejora de procesos. La tabla 10 las presenta de manera resumida:

| **Áreas del proceso relacionadas con la gestión de proyectos** | | |
|---|---|---|
| **Área** | **Propósito** | **Descripción** |
| Enfoque hacia el proceso de la organización (OPF) | Planear e implementar el mejoramiento del proceso organizacional basado en las fortalezas y debilidades actuales de los activos de proceso y procesos de la organización. | El mejoramiento ocurre en el contexto de las necesidades de la organización y es usado para orientarse hacia los objetivos de la organización. La organización define un grupo de proceso, independiente de quienes participan en los procesos para que se encargue de coordinar y liderar las actividades evaluación y de mejora. Esto involucra desarrollar un plan de mejora (p.e. IDEAL), el cual debe incluir planes de evaluación, de acciones, de pilotos y de despliegue. |
| Definición del proceso organizacional(OPD) | Establecer y mantener un conjunto de activos de proceso organizacional usables. | Los activos de proceso de la organización posibilitan una ejecución consistente del proceso a través d e la organización y provee una base acumulativa a la organización con beneficios a largo plazo. La librería de de activos de proceso de la organización es una colección de ítems mantenidos por la organización para ser usados por los proyectos y las personas dentro de la organización. Esta colección de ítems incluye descripciones de procesos, elementos de procesos, descripciones de modelos de ciclo de vida, guías de instanciación (tailoring), documentación relacionada con el proyecto e información. La librería de activos de procesos soporta el aprendizaje organizacional permitiendo compartir las mejores prácticas a través de la organización. Un proceso estándar está compuesto de otros procesos o elementos de procesos (Unidad fundamental de definición de proceso, descrita por actividades y tareas para realizar un trabajo). La arquitectura del proceso provee las reglas para la conexión de los elementos del proceso de un proceso estándar. |
| Entrenamiento organizacional(OT) | Desarrollar las habilidades y el conocimiento de las personas para que puedan desempeñar sus roles efectiva y eficientemente. | Incluye el entrenamiento necesario para soportar los objetivos estratégicos de la organización y las necesidades tácticas que son comunes a los grupos de proyectos y de soporte. Las necesidades individuales y específicas de entrenamiento por cada proyecto y grupos de soporte se salen del alcance de ésta área. Un programa de entrenamiento organizacional involucra: identificar las necesidades de entrenamiento de la organización, obtener y proveer entrenamiento dirigido a estas necesidades, establecer y mantener la capacidad de entrenamiento, establecer y mantener los registros de entrenamiento, y asegurar la efectividad del entrenamiento. |

| | | |
|---|---|---|
| Desempeño del proceso organizacional(OPP) | Establecer y mantener una comprensión cuantitativa del desempeño del conjunto de activos de proceso estándar de la organización para soportar objetivos de calidad y desempeño del proceso, y para proveer información del desempeño del proceso, líneas base y modelos cuantitativamente gestionados de los proyectos de la organización. | El desempeño del proceso es una medida de los resultados reales logrados al seguir un proceso. Este desempeño es caracterizado tanto por medidas del proceso (esfuerzo, tiempo, etc.), como por medidas del producto (confiabilidad, densidad de defectos, etc.). Esta información es analizada para establecer la distribución y el rango de resultados, los cuales caracterizarán el rango esperado de desempeño cuando sea utilizado por un proyecto individual en la organización. |
| Innovación y despliegue organizacional(OID) | Seleccionar y desplegar mejoras incrementales e innovativas que mejoren de manera medible el proceso organizacional y las tecnologías. El soporte al mejoramiento de los objetivos de calidad y desempeño del proceso de la organización, así como los derivados de los objetivos de negocio de la organización. | Permite la selección e instalación de mejoras a la organización que permiten reunir las capacidades necesarias para alcanzar sus objetivos de calidad y de desempeño del proceso. Las mejoras en esta área se refieren a ideas probadas y por ser probadas que pueden cambiar los procesos y tecnologías de la organización para cumplir de una mejor forma con sus objetivos de calidad y desempeño. Esos objetivos son: mejorar la calidad del producto, incrementar la productividad, disminuir el tiempo de los ciclos, alcanzar una gran satisfacción del cliente y del usuario final, acortar tiempos de desarrollo o de producción para cambiar funcionalidad, adicionar características o adaptarse a nuevas tecnologías. |
| Enfoque hacia el proceso de la organización (OPF) | | |

***Tabla 10. Áreas de Gestión de procesos***

Las relaciones entre estas áreas y el resto de áreas del proceso se visualizan en las figura 17 y 18.

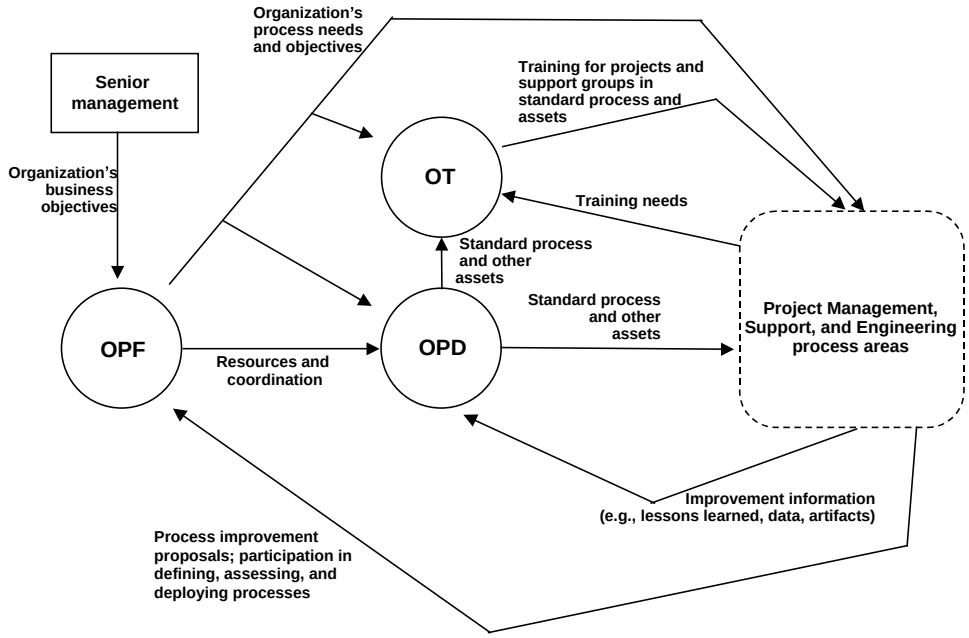



***Figura 17. Relaciones entre las áreas básicas de gestión de procesos y el resto de áreas.***

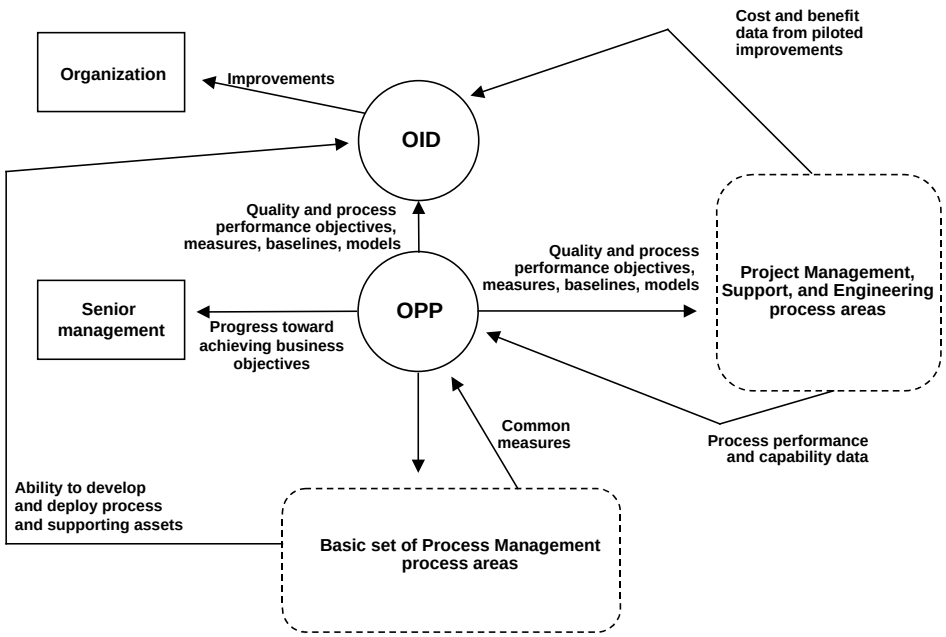



***Figura 16. Relaciones entre las áreas avanzadas de gestión de procesos y el resto de áreas.***

### *2.4.8 CMMI como una SPL – Una línea de modelos de referencia*

Si bien el modelo CMMI no es una línea de procesos, ni de productos software; si está estructurado y ha sido concebido como una línea de modelos de calidad con un conjunto de activos base, modelos de la línea de modelos y activos de instanciación de los modelos a partir de los activos base. Vista como una línea de modelos CMMI tiene asociados los siguientes elementos:

**El dominio:** el dominio de CMMI son los modelos CMMs heredados, con las disciplinas de Ingeniería del Software, Ingeniería de sistemas, desarrollo integrado de proceso y producto y fuentes de suministro. De igual forma se alimenta de los modelos desarrollados por ISO 15504 y por IEC/IA 731-1.

**Arquitectura:** la arquitectura viene determinada por un Framework de objetivos y prácticas genéricas, con un conjunto de áreas de procesos, guías de instanciación y un equivalente escalonado (para medir el nivel de madurez a partir de los niveles de capacidad), los cuales corresponden a los elementos comunes. La variabilidad viene dada en las disciplinas en la cual se puede extender y las dos representaciones. Estos elementos comunes forman los activos núcleo. La figura 17 muestra esta arquitectura.

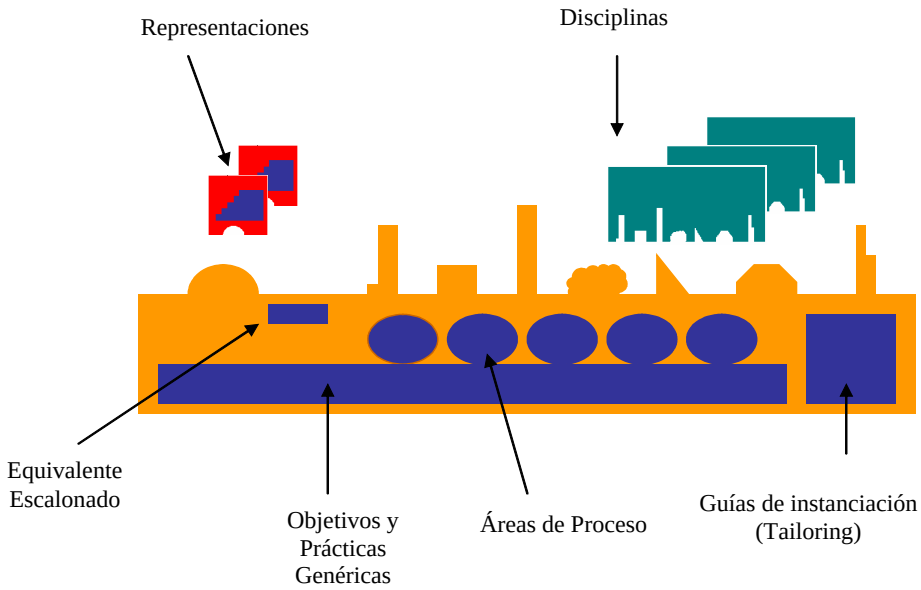



***Figura 17. Arquitectura de CMMI***

**Los activos de la línea de modelos son:** el framework CMMI, las áreas de proceso, las guías de instanciación, las representaciones (frameworks de medida) y las disciplinas.

**Los productos de la línea de modelos pueden ser:** CMMI-SW representación escalonada, CMM-SW representación continua, CMM-SW/IPPD representación contínua, etc. La figura 18 muestra la extensión del Framework para formar CMMI-SW.

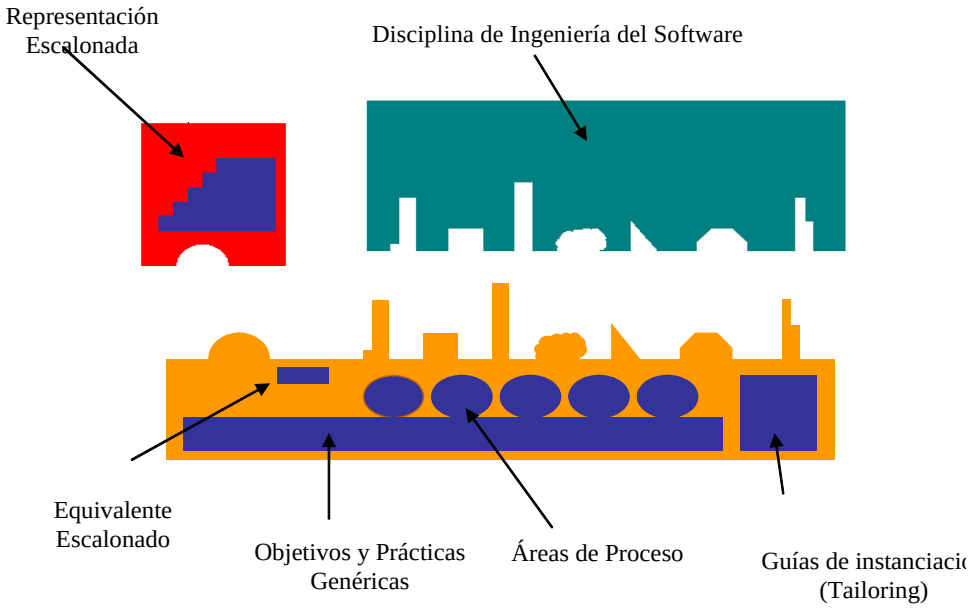



***Figura 18 El Modelo CMMI-SW Representación escalonada (Anterior CMM-SW)***

## 2.5 Las metodologías Ágiles

Del otro lado dónde se encuentran los procesos estandarizados, hay una corriente totalmente opuesta, la forman las llamadas metodologías ágiles[49], las cuales reclaman un mayor protagonismo de las personas sobre los procesos de desarrollo por sí mismos. Ágil, denota la cualidad de ser ágil, facilidad para el movimiento, actividad y destreza en el movimiento. Durante algún tiempo se conocían como las metodologías ligeras, pero el término aceptado ahora es metodologías ágiles [49]. Las metodologías ágiles para el desarrollo de software estuvieron motivadas por una conciencia particularmente aguda de la crisis del software, por la responsabilidad que se la asignado a las grandes metodologías en la gestación de esa crisis y por el propósito de articular soluciones.

El término ágil aplicado la industria del software nació en febrero del 2001, tras una reunión realizada en Utah, Estados Unidos, en la que se creó The Agile Alliance –La Alianza Ágil, una organización dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue un documento que resume la filosofía ágil: el Manifiesto ágil [1]. Este documento describe su filosofía a través de un conjunto de principios y valores, los cuales se describen a continuación.

Los firmantes del manifiesto ágil fueron: Kent Beck (XP), Mike Beedle, Arie van Bennekum (DSDM), Alistair Cockburn (Crystal), Ward Cunnimgham (XP), Martin Fowler (XP), James Grenning (XP), Jim Highsmith (ASD), Andrew Hunt (Pragmatic Programming), Ron Jeffries (XP), Jon Kern (FDD), Brian Marick, Robert C. Martin (XP), Steve Mellor, Ken Schwaber (Scrum), Jeff Sutherland (Scrum) y Dave Thomas (Pragmatic Programming).

### *2.5.1 Valores expuestos en el manifiesto ágil*

Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas: El movimiento ágil enfatiza en las relaciones y la comunicación entre los desarrolladores de software y los roles personales en lugar de procesos institucionalizados y las herramientas de desarrollo. En las prácticas ágiles existentes, estas manifiestan estar muy cercanas a la interrelación de los miembros del equipo, disposición del ambiente de trabajo, y otros elementos que impulsan el espíritu de equipo.

Desarrollar software que funciona en lugar de conseguir una buena documentación: el gran objetivo del equipo de desarrollo es continuamente al final de cada iteración realizar pruebas de funcionamiento de software. Los desarrolladores deben mantener el código simple y técnicamente tan avanzado como sea posible.

La colaboración con el cliente más que la negociación de un contrato: la relación y colaboración entre desarrolladores y clientes es preferible a un contrato estricto. Desde el punto de vista del negocio, el desarrollo ágil esta enfocado en entregar valor al negocio desde el inicio del proyecto, así se reduce inmediatamente el riesgo de no cumplimiento del contrato.

Responder a los cambios más que seguir estrictamente un plan: el grupo de desarrollo, comprende desarrolladores y representantes del cliente. El grupo debe estar bien informado, capacitado y autorizado para considerar posibles ajustes y necesidades que surgen durante el ciclo del proceso de desarrollo. Esto significa que los participantes están preparados para realizar cambios y que los contratos existentes contienen herramientas que permiten y apoyan que estos cambios se realice.

### *2.5.2 Principios expuestos en el manifiesto ágil*

Nuestra prioridad más alta es satisfacer al cliente a través de la entrega temprana y continua de software valioso.

El cambio en los requisitos es bienvenido, incluso cuando llegan tarde en el desarrollo. Los procesos ágiles se acogen al cambio para lograr una ventaja competitiva para el cliente.

Entregar software que funcione con frecuencia, desde un par de semanas hasta un par de meses, con preferencia en períodos de tiempo más cortos.

La gente de negocios y los desarrolladores deben trabajar juntos regularmente a través de todo el proyecto.

Construir proyectos en torno de individuos motivados. Darles la oportunidad y el respaldo que necesitan y darles confianza para que realicen sus tareas.

La forma más eficiente y efectiva de comunicar información de un lado a otro dentro de un equipo de desarrollo es mediante la conversación cara a cara.

El software que funciona es la medida primaria de progreso.

Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante indefinidamente.

La atención continua a la excelencia técnica enaltece la agilidad.

La simplicidad (el arte de maximizar la cantidad de trabajo que no se hace) es esencial.

Las mejores arquitecturas, requerimientos y diseños emergen de equipos que se auto-organizan.

A intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo, y de acuerdo a esto, ajusta su conducta.

### *2.5.3 Principales metodologías y sus prácticas*

#### 2.5.3.1 Programación Extrema XP (Extreme Programming)

La programación extrema XP (Extreme Programming) es método típicamente atribuido a Kent Beck, Ron Jeffries and Ward Cunningham [52]. El objetivo de XP son grupos pequeños y medianos de construcción de software en donde los requisitos aún son muy vagos, cambian rápidamente o son de alto riesgo. Las recomendaciones de XP están encaminadas a producir software de calidad. XP fue diseñado teniendo en cuenta los problemas que existían con las metodologías tradicionales de programación en cuanto a tiempos de entrega y satisfacción del cliente. El objetivo principal para XP es buscar la satisfacción del cliente tratando de mantener durante todo el tiempo su confianza en el producto.

##### 2.5.3.1.1 Valores de la Programación Extrema

XP es una metodología ágil, la cual consiste básicamente en un conjunto de guías generales para el desarrollo de software por equipos de pequeño y mediano tamaño. El ciclo de vida de XP [54] incluye cuatro actividades básicas: codificación, prueba, escucha, y diseño, y su dinámica es demostrada a través de cuatro valores:

**V1. Comunicación continua:** XP involucra comunicación extrema. El cliente debe ser parte del equipo de trabajo, haciendo la definición de los requisitos, permitiendo producir versiones cada 2 a 4 semanas. Esto permite también que, si el programador necesita aclarar algo en los requisitos, se reúne con el equipo de trabajo y se resuelven sus dudas. La programación de pares y la propiedad colectiva del código (prácticas de XP) promueven la comunicación de conocimiento técnico a través de todo el equipo.

**V2. Realimentación:** mientras más rápido se identifica el cambio, más rápido se le puede manejar. En general, el principio es encontrar el error lo más cerca posible al tiempo en que fue introducido. XP se esfuerza para que se pueda recibir la retroalimentación lo más rápido posible en todos los aspectos del proyecto. Se generan pruebas unitarias para la mayor parte del código producido. El código debe pasar las pruebas unitarias sin errores antes de ser actualizado en el repositorio central. Además se elaboran pruebas de aceptación con el cliente para verificar que la aplicación está haciendo lo que el cliente necesita que haga.

**V3. Simplicidad:** el cliente es la el rol conductor en XP, se busca hacer lo que el cliente necesita, tan simple como sea posible y nada más. El propósito es reducir la complejidad desde el principio. Todo el código debería ser refactorizado (una práctica de XP) tan frecuentemente como sea posible.

**V4. Valentía:** es coraje que debe tener el equipo para resolver problemas de manera preactiva. XP permite que el programador modifique el código existente cuando lo considere necesario, de

forma responsable, sin temor a que los cambios que realice afecten la funcionalidad existente o causen errores en otras partes, esto gracias a las pruebas unitarias, que exigen que cada vez que se cambie el código, antes de que se pueda subir al repositorio central debe pasar todas las pruebas establecidas.

#### 2.5.3.1.2 Principios de la Programación Extrema

Adicionalmente a los valores, la programación extrema está fundamentada en cinco principios que son los siguientes:

**Pr1. Trabajo en equipo:** El trabajo en equipo es fundamental para sacar un proyecto adelante. XP sugiere grupos no muy grandes (de 2 a 12 desarrolladores) para manejar mejor los problemas del cambio de los requisitos. Es necesario que todos los miembros del grupo estén comprometidos con todo el software producido. XP acepta que uno de los principales y más importantes factores en un equipo de desarrollo son los integrantes del equipo en sí.

**Pr2. Participación del cliente:** Para que el cliente pueda quedar satisfecho al final de un proyecto de software debe ser parte activa del equipo. Un cliente no puede entregar los requisitos, irse y luego volver a mirar los resultados. Involucrar al cliente no significa que el esté al tanto de los detalles mínimos de implementación, simplemente la idea es que el cliente (o un representante) esté disponible para responder a dudas o preguntas que surjan dentro das iteraciones del desarrollo.

**Pr3. Simplicidad:** mantener el diseño y el código lo más simple posible. Con esto se logra una reducción muy importante en los costos de mantenimiento. Por lo tanto se pide al desarrollador, ante todo, simplicidad a la hora de diseñar o implementar. Es tal la importancia de la simplicidad que si se encuentra una forma más sencilla de implementar algo después de hecho, se debería considerar seriamente volver a implementarlo. Esto puede parecer sin sentido, pero en realidad el tiempo ahorrado por simplificar algo a lo que se le va a hacer mantenimiento debe considerarse como una inversión y no un gasto.

**Pr4. Buenas pruebas:** Con XP se busca eliminar malas prácticas como probar todo el software en los últimos días antes de salir el proyecto. Para esto se propone lograr un buen diseño de pruebas y una ejecución completa y eficiente. XP introduce el concepto de pruebas automatizadas las cuales ayudan a desarrollar mejores aplicaciones. Las pruebas automatizadas proveen al equipo de desarrollo una garantía de que los cambios hechos durante el desarrollo no afectan a otros componentes de la aplicación, o si los afectan, poder ubicar las partes comprometidas fácilmente.

**Pr5. La productividad es secundaria:** La satisfacción del cliente debe ser el primer objetivo. Idealmente XP mejora la productividad de los desarrolladores pero si en la realidad esta baja, no debe importar mientras que se logre la satisfacción del cliente.

2.5.3.1.3 El Proceso XP

El ciclo de vida de XP se basa en el concepto de iteración de desarrollo. Una iteración de desarrollo es un periodo de tiempo en el que se realiza un conjunto de funcionalidades determinadas que en el caso de XP corresponden a un conjunto de historias de usuario. Las iteraciones aquí son particularmente cortas ya que se piensa que entre más rápido se le entreguen desarrollos al cliente, más retroalimentación se va a obtener y esto va a representar una mejor calidad del producto a largo plazo. Existe una fase de análisis inicial encaminada a programar las iteraciones de desarrollo y cada iteración incluye diseño, codificación y pruebas, con la diferencia de que no son etapas separadas sino que están superpuestas de tal manera que no es correcto separar un proyecto XP en estas fases porque el esquema de trabajo no se divide en el tiempo de esta forma.

El ciclo de vida de XP consta de cinco ***fases***: Exploración, planeacion, iteraciones, producción, mantenimiento y muerte, a continuación se describen según [54]:

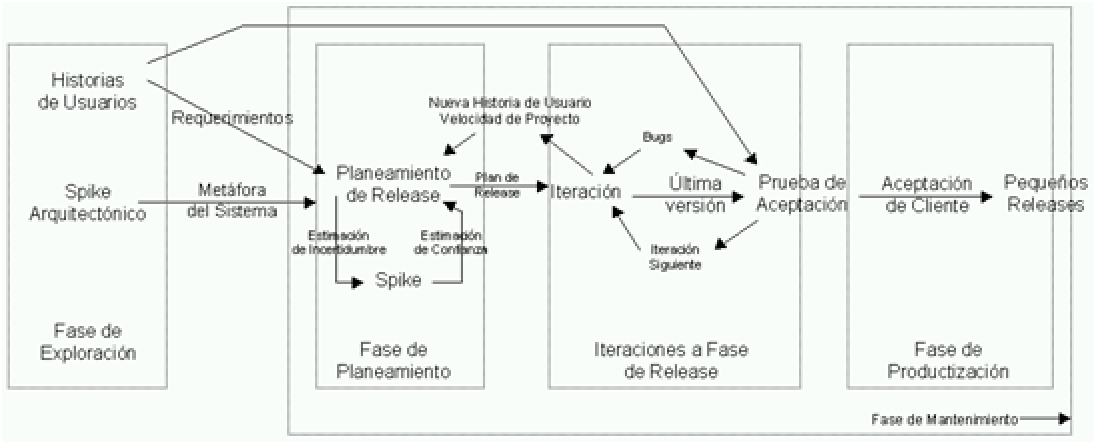



***Figura 19 Ciclo de vida de XP, tomado de [54]***

**F1. Fase de la exploración:** los clientes ponen por escrito en las tarjetas de historia de usuario lo que desean sea incluido en cada release. Cada tarjeta de historia describe una característica que se agregará en el programa. Al mismo tiempo el equipo del proyecto se familiariza con las herramientas, la tecnología y las prácticas que utilizarán en el proyecto. La tecnología que se utilizará será probada y las posibilidades de la arquitectura del sistema son exploradas construyendo un prototipo del sistema. La fase de exploración toma entre algunas semanas a algunos meses, dependiendo en gran parte de cómo están familiarizados con la tecnología los programadores.

**F2. Fase del planeamiento:** se priorizan las historias de usuario y se acuerda el alcance del release. Los programadores estiman cuánto esfuerzo requiere cada historia y a partir de allí se define el cronograma. La duración del cronograma del primer release no excede normalmente dos meses. La fase de planeamiento toma un par de días. Se deben incluir varias iteraciones para lograr un release. El cronograma fijado en la etapa de planeamiento se realiza a un número de

iteraciones, cada una toma de una a cuatro semanas en ejecución. La primera iteración crea un sistema con la arquitectura del sistema completo. Esto es alcanzado seleccionando las historias que harán cumplir la construcción de la estructura para el sistema completo. El cliente decide las historias que se seleccionarán para cada iteración. Las pruebas funcionales creadas por el cliente se ejecutan al final de cada iteración. Al final de la última iteración el sistema esta listo para producción.

**F3. Fase de producción:** requiere prueba y comprobación extra del funcionamiento del sistema antes de que éste se pueda liberar al cliente. En esta fase, los nuevos cambios pueden todavía ser encontrados y debe tomarse la decisión de si se incluyen o no en el release actual. Durante esta fase, las iteraciones pueden ser aceleradas de una a tres semanas. Las ideas y las sugerencias pospuestas se documentan para una puesta en práctica posterior por ejemplo en la fase de mantenimiento. Después de que se realice el primer release productivo para uso del cliente, el proyecto de XP debe mantener el funcionamiento del sistema mientras que realiza nuevas iteraciones.

**F4. Fase de mantenimiento:** requiere de un mayor esfuerzo para satisfacer también las tareas del cliente. Así, la velocidad del desarrollo puede desacelerar después de que el sistema esté en la producción. La fase de mantenimiento puede requerir la incorporación de nueva gente y cambiar la estructura del equipo.

**F5. Fase de muerte:** es cuando el cliente no tiene más historias para poner en ejecución. Esto requiere que el sistema satisfaga las necesidades del cliente respecto al funcionamiento y calidad. En esta etapa se escribe la documentación necesaria y ya no se realizan más cambios a la arquitectura, diseño o código elaborado. La fase de muerte también puede ocurrir si el sistema no satisface los resultados esperados o si llega a ser demasiado costoso para el desarrollo adicional.

Existen diferentes ***roles y responsabilidades*** en XP para diferentes tareas y propósitos durante el proceso y las practicas de éste. A continuación se presentan los roles y practicas de acuerdo a [54]:

**R1. Programador:** los programadores escriben las pruebas y mantienen el código del programa tan simple y definido como sea posible. El primer aspecto para tener éxito con XP es comunicarse y coordinar con los otros miembros del equipo

**R2. Cliente:** es quien escribe las historias y pruebas funcionales, y decide cuando cada requerimiento es satisfecho. El cliente selecciona el orden de prioridad para los requisitos.

**R3. Probador:** el probador o tester ayuda al cliente a escribir las pruebas funcionales. Ellos realizan pruebas funcionales regularmente, difunden resultados de la prueba y mantienen las herramientas de pruebas

**R4. Rastreador (Tracker):** proporciona realimentación en XP, rastrea los estimativos hechos por el equipo, por ejemplo esfuerzo y tiempo; y proporciona realimentación para mejorar estimativos futuros. También rastrea el progreso de cada iteración y evalúa si el objetivo es alcanzable con los recursos dados y en el tiempo límite o si son necesarios cambios en el proceso.

**R5. Entrenador (Coach):** es la persona responsable por el proceso completo. Esta en capacidad de guiar a otros miembros del equipo en el seguimiento del proceso.

**R6. Consultor:** es una persona externa al proyecto que posee el conociendo técnico especifico

necesario. El consultor guía al equipo en la resolución de un problema especifico.

**R7. Director:** es quien toma las decisiones. Para poder hacer esto, él se comunica con el equipo de proyecto para determinar la situación actual, y distinguir cualquier dificultad o deficiencia en el proceso.

#### 2.5.3.1.4 Prácticas de la programación Extrema

La principal suposición que se hace en XP es la posibilidad de disminuir la curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes doce prácticas[54]

**P1. El juego de la planificación:** La planificación en XP permite contestar dos preguntas clave en el desarrollo de software: predecir lo que se hará ahora, y determinar que se hará después. El énfasis se pone en la dirección del proyecto, más que en hacer una predicción exacta de que será necesario y cuanto tardará en hacerlo. Hay una comunicación frecuente del cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y el tiempo de las entregas.

XP promueve la planificación de entregas (release planning) donde se planifica las distintas iteraciones. Para ello existen una serie de reglas que hay que seguir para que las tres partes implicadas en este proceso (equipo de gestión, equipo de desarrollo y cliente) tengan voz y se sientan parte de la decisión tomada, que al final deberá satisfacer a todos. La planificación debe de seguir unas ciertas premisas. La primordial es que las entregas se hagan cuanto antes y que con cada iteración el cliente reciba una nueva versión. Cuanto más tiempo se tarde en introducir una parte esencial, menos tiempo habrá para trabajar en ella posteriormente. Se aconsejan muchas entregas y muy frecuentes. De esta forma, un error en una parte esencial del sistema se encontrará pronto y, por tanto, se podrá arreglar antes.

Sin embargo, los requisitos anteriores en cuanto a la planificación no deben suponer horas extra para el equipo de desarrollo. El argumento que se da es que lo que se trabaja de más un día, se deja de trabajar al siguiente. Diversas prácticas como las pruebas unitarias, la integración continua o el juego de la planificación permiten eliminar los principales motivos por los que suele ser necesario trabajar muchas horas extra. Pero lo mejor de todo es que en el momento de planificar el grupo se puede equivocar. Es más, debido a que el equivocarse puede ser visto como algo común, la metodología ya tiene previsto mecanismos de revisión. Por lo tanto, es normal que cada 3 a 5 iteraciones se tengan que revisar las historias de los usuarios y renegociar nuevamente la planificación.

En cada iteración también hay que realizar la planificación de la misma, lo que ha venido a llamarse planificación iterativa. En la planificación iterativa se especifican las historias de los usuarios cuya implementación se considera primordial y se añaden aquéllas que no han pasado las pruebas de aceptación de anteriores iteraciones. La planificación de una iteración hace uso de tarjetas en las que se escribirán tareas, que durarán entre uno y tres días (la duración la deben decidir los propios desarrolladores). Es por eso, que el diseño que sigue XP se puede calificar de

continuo, añade agilidad al proceso de desarrollo y evita mirar demasiado adelante e implementar tareas que no estén programadas (algo que se ha venido a llamar programación justo a tiempo).

**P2. Entregas pequeñas:** Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Los equipos de XP entregan software a sus usuarios finales tan frecuentemente como sea posible.

**P3. Metáfora:** La metáfora del proyecto es una descripción informal de la arquitectura del sistema; describe el sistema en conceptos simples. Los conceptos pueden ser literales o figurativos, dependiendo de la claridad del sistema actual. La idea es que en base a esta metáfora, todos los miembros del equipo, clientes y desarrolladores, puedan entender cómo trabaja el sistema de manera general. El principal objetivo de la metáfora es mejorar la comunicación entre los todos integrantes del equipo al crear una visión global y común del sistema que se pretende desarrollar. La metáfora debe estar expresada en términos conocidos para los integrantes del grupo, por ejemplo comparando lo que se va a desarrollar con algo que se puede encontrar en la vida real. Aunque también se incluye información sobre las principales clases y patrones que se usarán en el sistema. Un apoyo a la metáfora a lo largo del proyecto es una correcta elección y comunicación de los nombres que se escojan durante el proyecto para los módulos, sistemas, clases, métodos, etc. Nombres bien puestos pueden tener implicaciones en la claridad, reusabilidad y simplicidad, tres conceptos a los que XP otorga una gran importancia. Aunque en general el diseño es realizado por los propios desarrolladores en ocasiones se reúnen aquellos con más experiencia o incluso se involucra al cliente para diseñar las partes más complejas. En estas reuniones se emplean un tipo de tarjetas denominadas CRC (Class Responsabilities and Collaboration - Clases, Responsabilidades y Colaboración) cuyo objetivo es facilitar la comunicación, diseñar el sistema y documentar los resultados. Para cada clase identificada se rellenará una tarjeta de este tipo y se especificará su finalidad así como otras clases con las que interaccione.

**P4. Diseño simple:** se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. Esto se basa en la filosofía de que el mayor valor de negocio es entregado por el programa más sencillo que cumpla los requerimientos.

**P5. Pruebas:** las pruebas son una práctica crucial en un proyecto de XP. Una de los fuertes de XP es que hace que los proyectos sean flexibles; la flexibilidad se basa en la retroalimentación exacta y frecuente, y las pruebas proveen esta retroalimentación. En XP existen dos categorías de pruebas: pruebas unitarias y pruebas de aceptación. Una prueba unitaria es código adicional que forma parte del software final, que ejecuta un aspecto de una pieza de código producido. Las pruebas de aceptación se distinguen de las pruebas unitarias en que las pruebas de aceptación requieren probar el sistema de forma total, además el cliente está involucrado en la creación de las pruebas de aceptación. Las pruebas unitarias se centran en que cada detalle técnico esté funcionando correctamente, las pruebas de aceptación aseguran que cada requerimiento del cliente esté funcionando correctamente.

**P6. Refactorización:** consiste en hacer modificaciones en el código que no alteren la funcionalidad del sistema y que impliquen una mejora de alguna cualidad no funcional: simplicidad, flexibilidad, claridad o desempeño. El código debe ser limpio, legible y la duplicación de código debe ser evitada. Se mejora la estructura interna del código sin alterar el

comportamiento externo.

**P7. Programación en parejas:** todo el software producido con XP es construido por parejas de programadores, sentados uno al lado del otro en la misma máquina. Esta práctica asegura que todo el código producido es revisado por al menos un programador, y resulta en unas mejores pruebas y mejor código. Desarrollar en parejas implica dos personas compartiendo un solo monitor y teclado. Quien codifica estará pensando en el mejor modo de implementar un determinado método, mientras que su compañero lo hará de una manera más estratégica [55], haciéndose preguntas como ¿Es el enfoque apropiado?, ¿Esto podría fallar? , ¿Es la forma más simple?, etc. Los roles son intercambiables, de manera que en cualquier momento quien observaba puede tomar el teclado para ejemplificar alguna idea o, simplemente, para cambiar de turno con su compañero. Igualmente, la composición de las parejas cambiará siempre que uno de los dos sea requerido por algún otro miembro del equipo para que le ayude con su código, de tal manera que podría haber rotaciones en donde el conocimiento de los individuos queda compartido, esto incluye el código y el conocimiento técnico y metodológico.

**P8. Propiedad colectiva del código:** En un proyecto XP, cualquier par de programadores puede mejorar cualquier parte del código en cualquier momento. Esto significa que todo el código recibe el beneficio de la atención de bastante gente, lo que incrementa la calidad de código y reduce los defectos. El uso de estándares de codificación y las pruebas dan la confianza que todo va a seguir funcionando bien después de una modificación.

**P9. Integración continua:** cada pocas horas, o máximo al cabo de un día de programación se integra el sistema completo. Para ello debe existir una máquina llamada "máquina de integración", a la que se acercará una pareja de programadores cada vez que tengan un módulo que haya sido probada unitariamente. Si al añadir el nuevo módulo junto con sus pruebas unitarias, el sistema completo sigue funcionando correctamente (pasa las pruebas definidas), los programadores darán por finalizada esa tarea. Si no, serán los responsables de dejar el sistema de nuevo con las pruebas funcionando al 100%. Si después de un cierto tiempo no son capaces de descubrir qué es lo que falla, tirarán el código a la basura y volverán a comenzar de nuevo [55].

**P10. Cuarenta horas por semana:** XP promueve contar con un equipo de trabajo bien descansado. Los desarrolladores cansados están más propensos a cometer errores y a empezar a desear un nuevo trabajo.

**P11. Cliente en el sitio de trabajo:** otra controvertida práctica de XP: el cliente real (o un representante) debe estar permanentemente junto al equipo de desarrollo, para responder cualquier consulta que los programadores requieran, para establecer prioridades, entre otros asuntos. Si el cliente argumenta que su tiempo es demasiado valioso, debe entenderse que realmente el proyecto que no tiene la importancia suficiente como para merecer su atención, y que no importa que esté construido a partir de suposiciones hechas por un grupo de desarrollo que nada, o muy poco, sabe del negocio real.

**P12. Estándares de codificación:** Es decisiva para poder plantear con éxito la propiedad colectiva del código. Ésta práctica sería impensable sin una codificación basada en estándares que haga que todo el mundo se sienta cómodo con el código escrito por cualquier otro miembro del equipo.

El mayor beneficio de las prácticas se consigue con una aplicación conjunta y equilibrada, puesto que se apoyan unas con otras. La mayoría de las prácticas propuestas por XP no son novedosas

sino que de algún modo ya habían sido propuestos en ingeniería de software e incluso demostrado su valor en la práctica, el mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde las perspectivas del negocio, los valores humanos y el trabajo en equipo.

#### 2.5.3.2 SCRUM

Como metodología ágil específicamente referida a ingeniería de software, Scrum fue aplicado por Jeff Sutherland y elaborado más formalizadamente por Ken Schwaber [56]. Poco después Sutherland y Schwaber se unieron para refinar y extender Scrum, en el que se aplicaron principios de procesos de control industrial, junto con experiencias metodológicas de Microsoft, Borland y Hewlett-Packard. Schwaber se dio cuenta entonces de que un proceso necesita aceptar el cambio, en lugar de esperar predictibilidad [56]. A pesar que CMM se concentraba en hacer que los procesos de desarrollo se tornaran repetibles, definidos y predecibles, muchos de ellos eran formalmente impredecibles e irrepetibles porque cuando se está planificando no hay primeros principios aplicables, los procesos recién comienzan a ser comprendidos y son complejos por naturaleza.

Scrum no está concebido como método independiente, es un complemento de otras metodologías, como XP o el Proceso Unificado [53]. Como método, ***Scrum enfatiza valores y prácticas de gestión***, y no incluye prácticas para los requisitos, implementación y demás cuestiones técnicas; de allí su gran insuficiencia, pero al mismo tiempo su capacidad para complementar otros métodos.

##### 2.5.3.2.1 Valores de Scrum

Scrum es un proceso de gestión y control que implementa técnicas de control de procesos. Los valores de Scrum son:

**V1. Equipos auto-dirigidos y auto-organizados**. No hay un líder que decida, ni otros títulos que "miembros del equipo", "Director" o "Marranos"; la excepción es el Scrum Master que debe ser 50% programador y que resuelve problemas, pero no manda. Los observadores externos se llaman "gallinas"; pueden observar, pero no interferir ni opinar.

**V2. Una vez elegida una tarea, no se agrega trabajo extra.** En caso que se agregue algo, se recomienda quitar alguna otra cosa.

**V3. Encuentros diarios** en las que los miembros responden a tres preguntas: (1) ¿Qué ha logrado completar con respecto al cronograma (backlog en Scrum), desde la última reunión?, (2) ¿Qué obstáculos ha encontrado en la forma de hacer su trabajo? y (3) ¿Qué cosas específicas planea completar, relativo al backlog, entre este momento y la siguiente reunión?. Esta reunión se realiza siempre en el mismo lugar formando un círculo.

**V4. Iteraciones de treinta días**, Scrum admite que sean más frecuentes.

**V5.** Se debe hacer una demostración a participantes externos al fin de cada iteración.

**V6.** Al principio de cada iteración, hay un planeamiento adaptativo guiado por el cliente.

2.5.3.2.2 El proceso de de Scrum

Scrum es una metodología de desarrollo de productos incremental y evolutiva. Los requisitos se identifican y se listan en un lugar definido llamado el backlog del producto. Las iteraciones, llamadas sprints, normalmente duran 30 días. En cada sprint el grupo de desarrollo selecciona del backlog un conjunto de ítems de mayor prioridad, y los desarrolla de tal forma que el backlog se convierte en el artefacto base de la medida de progreso del proyecto. Durante el sprint el equipo trabaja en sus tareas sin modificarlas con nuevos requisitos. Todos los días los miembros del equipo se reúnen (Scrum diario) con el líder del equipo (Master Scrum) para contestar las tres preguntas referidas al progreso del proyecto.

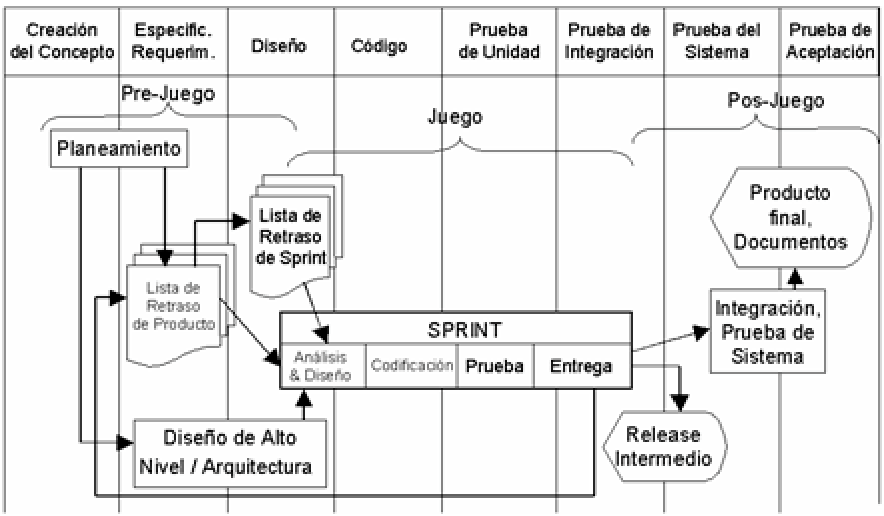



***Figura 20. Ciclo de Scrum, basado en [58]***

Las fases que define Scrum se describen a continuación:

**F1. Fase de planeamiento:** esta fase incluye dos subfases: planeación y diseño de alto nivel/arquitectura.

**F1.1. Planeación:** en esta fase se define el sistema a ser desarrollado y se crea una lista de requisitos del producto conocidos hasta el momento. Los requisitos pueden provenir del cliente, del departamento de mercadeo y ventas o de los desarrolladores de software. Los requisitos son priorizados y se estima el esfuerzo necesario para la implementación de estos. La lista del producto Backlog es constantemente actualizada, en detalles, exactitud y un nuevo orden de prioridades. La planeación también incluye la definición del equipo del proyecto, las herramientas

y otros recursos, valoración y control de riesgos, entrenamiento necesario y una verificación para la aceptación de gerencia. A cada iteración, la actualizacion del producto Backlog es revisada por el equipo Scrum para lograr el objetivo en la iteración siguiente.

**F1.2. Arquitectura:** el diseño de alto nivel del sistema incluye la arquitectura basada en los ítems del producto (requisitos). En caso de un mejoramiento al sistema existente, se identifican los cambios necesarios para la implementación de los ítems del pedido con los problemas que esto pueda generar.

**F2. Fase de desarrollo:** también llamada fase de juego es la parte ágil de la metodología Scrum. Esta fase es tratada como una "caja negra" donde se espera lo imprevisible. Las diversas variables ambientales y técnicas (cronograma, calidad, requisitos, recursos, tecnologías y herramientas de la puesta en práctica, e incluso métodos del desarrollo) identificadas en Scrum, que pueden cambiar durante el proceso, se observan y se controlan con varias prácticas de Scrum durante el sprint de la fase del desarrollo. Más que tener en cuenta estas variables solamente al inicio del proyecto de desarrollo del software, Scrum tiene como objetivo controlarlas constantemente para poder adaptarse flexiblemente a los cambios. En la fase del desarrollo el sistema se desarrolla en sprints. Un sprint es un ciclo iterativo donde la funcionalidad se desarrolla o se realza para producir nuevos incrementos. Cada Sprint incluye las fases tradicionales del desarrollo del software: requisitos, análisis, diseño, evolución y entrega. La arquitectura y el diseño del sistema se desarrollan durante el desarrollo del sprint.

**F3. Fase de posjuego:** se refiere contiene al cierre del lanzamiento. Esta fase es completada con la aceptación de las variables ambientales y con todos los requisitos cumplidos. En este caso, ya no se encuentran más ítems y el sistema se encuentra listo para el lanzamiento. Esta fase incluye integración, prueba del sistema y documentación.

Existen seis roles identificados en Scrum que tienen diferentes funciones y procesos en las practica, estos se describen según [59]:

**R1. Scrum Master:** el Scrum Master es un nuevo rol de la gerencia introducido por Scrum. El Scrum Master es responsable de asegurar que el proyecto esta llevándose a cabo según las practicas, los valores y las reglas de scrum, y de que el proyecto avanza según lo previsto. El Scrum Master trabaja recíprocamente con el equipo de proyecto, con el cliente y la gerencia durante todo el proyecto. Es también responsable de eliminar problemas y de cambiar el proceso para mantener el funcionamiento del equipo tan productivo como sea posible.

**R2. Propietario del producto**: el propietario del producto es oficialmente responsable del proyecto, manejando, controlando y haciendo visible la lista del pedido del producto. Este es seleccionado por el master Scrum, el cliente y el administrador. El propietario del producto toma la decisión final de las tareas relacionadas con el pedido del producto, participa en la estimación del esfuerzo de desarrollo para los ítems del pedido y publica en la lista del pedido los rasgos o características a ser desarrollados.

**R3. Equipo Scrum:** el equipo Scrum es quien tiene la autoridad de decidir sobre las acciones y organizarlas para alcanzar las metas de cada Sprint. El equipo scrum está implicado, por ejemplo, en la valoración del esfuerzo, en la creación del backlog del Sprint, en la revisión del producto pedido y en la sugerencia de eliminación de problemas e inconvenientes del proyecto.

**R4. Cliente:** el cliente participa en las tareas relacionadas con los ítems (requisitos) de la lista del pedido del producto a ser desarrollados.

**R5. Director:** el director es el encargado de tomar las decisiones finales, también decide los estándares y convenciones a seguir en el proyecto. También participa en la determinación de los objetivos y requisitos. Por ejemplo, el director participa en la elección del propietario del producto, en la medición del progreso y en la reducción del pedido junto con el Scrum master.

2.5.3.2.3 Prácticas de Scrum

Scrum no requiere o provee ningún método o práctica de desarrollo de software específico a ser usado. En cambio, requiere ciertas prácticas y herramientas de administración en varias fases del Scrum para evitar el caos causado por los imprevistos y por la complejidad. A continuación, se describe las prácticas de Scrum:

**P1. Pedido del producto:** define todo que se necesita en el producto final basado en el conocimiento inicial. Así, el pedido del producto define el trabajo que se hará en el proyecto. Abarca una lista con los requisitos técnicos y del negocio, nivel de prioridades y está constantemente actualizado. El pedido puede incluir, por ejemplo, las características, funciones, fallas del sistema a corregir, solicitudes de mejoramientos o revisión tecnológica. Múltiples actores pueden participar en la generación del pedido del producto, como el cliente, el equipo del proyecto, el departamento de mercadeo y ventas, gerencia, etc.

**P2. Estimación de esfuerzo:** la estimación del esfuerzo es un proceso iterativo, en que los estimativos de los ítems del pedido esta enfocado en un nivel mas exacto cuando mas información está disponible sobre cierto ítem de un producto dado. El dueño del producto y el equipo Scrum son responsables de realizar la estimación del esfuerzo.

**P3. Sprint:** Sprint es el procedimiento de adaptación al cambio de las variables ambientales (requisitos, tiempo, recursos, conocimiento, tecnología, etc.). El equipo Scrum se organiza para producir un nuevo incremento ejecutable del producto en un Sprint que dura aproximadamente treinta días calendario. Las herramientas de trabajo del equipo son reuniones de planeamiento del Sprint, pedido del Sprint y reuniones diarias de Scrum. El sprint con sus practicas se muestra en la figura 21.

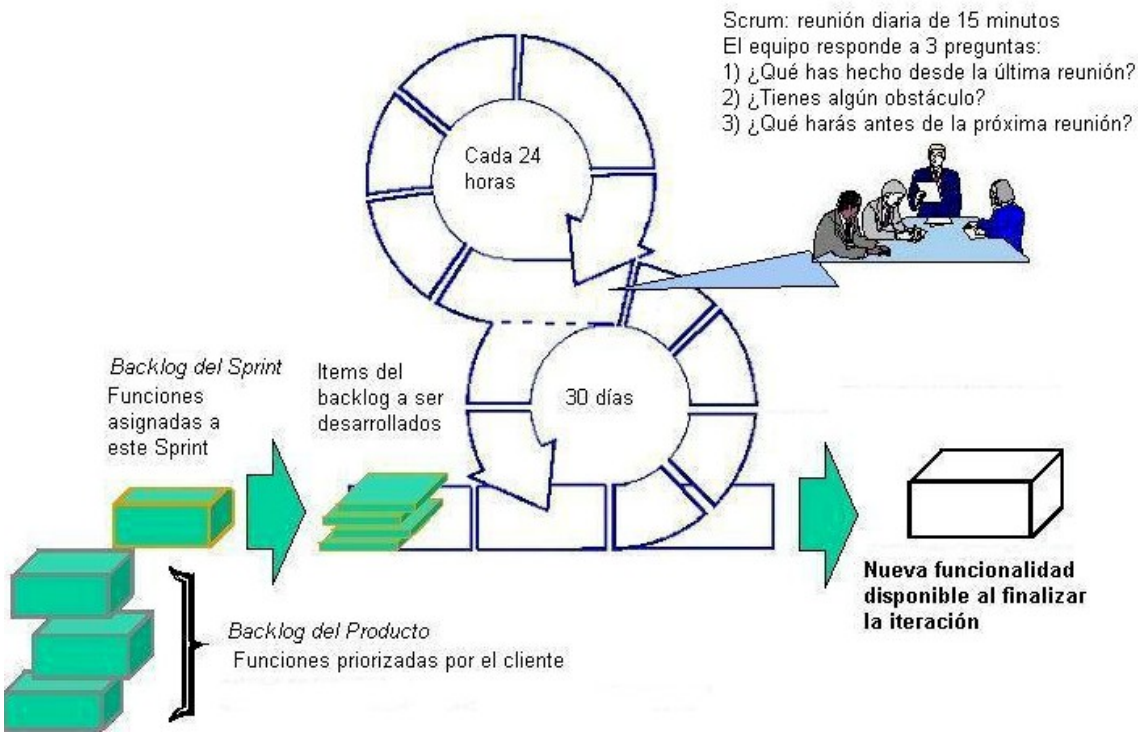



Figura 21. El Sprint de Scrum

**P4. Reunión de planeación:** la reunión de planeación es organizada por el Scrum master y consta de dos fases. Los clientes, usuarios, gerentes, el propietario del producto y equipo Scrum participan en la primera fase de la reunión para decidir sobre los objetivos y funcionalidad del nuevo sprint. La segunda fase de la reunión se realice entre el Scrum master y el equipo scrum, se enfoca en cómo el incremento del producto es implementado durante el proceso.

**P5. Pedido del sprint:** el pedido Sprint es el punto de partida para cada sprint. Esta es una lista de ítems de pedido del producto seleccionados para ser implementados en el próximo sprint. Los ítems son seleccionados por el equipo Scrum en conjunto con el Scrum master y el propietario del producto en la reunión de planeación del sprint, basándose en nivel de prioridad de los ítems y los objetivos para el sprint. El pedido sprint, contrario al pedido del producto, es estable hasta que se termina el sprint (es decir 30 días). Cuando todos los ítems del pedido sprint se completan, se entrega una nueva iteración del sistema.

**P6. Reunión diaria scrum:** las reuniones diarias o Scrum diario son organizadas para mantener una revisión constante del progreso del equipo y también sirve como reunión de planeamiento: que se ha hecho desde la ultima reunión y que debe hacerse antes de la siguiente. También se discuten las dificultades y otras variables. Las reuniones son cortas, de aproximadamente 15 minutos. Se buscan, identifican y se combaten posibles deficiencias o impedimentos en la práctica del proceso de desarrollo con el fin de mejorarlo. El Scrum master dirige las reuniones diarias. Además del equipo Scrum también la gerencia puede participar de la reunión.

**P7. Revisión a reunión de sprint:** en el último día de reunión Sprint, el equipo Scrum y el scrum master presentan los resultados (por ejemplo, incremento del producto trabajado) del sprint a la gerencia. Clientes, usuarios y propietario del producto participan de esta reunión informal. Los

participantes evalúan el incremento del producto y toman decisiones acerca de la siguiente actividad. La reunión de revisión trae nuevos ítems e incluso puede cambiar la dirección del sistema en construcción.

#### 2.5.3.3 Modelado Ágil (Agile Modelling - AM)

Modelado Ágil (AM) es un complemento a las metodologías ágiles para facilitar el modelado y documentación efectiva de sistemas basados en software. AM es una colección de prácticas, guiadas por principios y valores, para ser aplicadas por los ingenieros del software. AM no define detalladamente procedimientos para crear un tipo de modelo dado, en cambio, provee consejos para hacer efectivo tal modelado. Los modelos ágiles son más efectivos que los modelos tradicionales porque ellos son sólo lo que se necesita, no tienen que ser perfectos y completos. AM puede ser aplicado para modelar los requisitos, el análisis, la arquitectura y el diseño [50].

Las técnicas de AM pueden y deben ser aplicadas por equipos de trabajo que deseen tener un acercamiento ágil al desarrollo de software, en particular aquellos que están siguiendo un proceso ágil, como programación extrema u otros; sin embargo, pueden ser usadas para mejorar y simplificar frecuentemente los resultados del modelo en proyectos donde no se usa un proceso ágil. El secreto de AM no son las técnicas de modelado en sí mismas - tales como modelos de casos de uso, modelos de clase, modelos de datos, o modelos de interfaz de usuario – sino cómo debieran aplicarse [50].

AM es una estrategia de modelado (de clases, de datos, de procesos) pensada para contrarrestar la sospecha de que los métodos ágiles no modelan y no documentan. Se lo pude definir como un proceso de software basado en prácticas cuyo objetivo es orientar el modelado de una manera efectiva y ágil.

Los principales objetivos de AM son:

1. Definir y mostrar de qué manera se debe poner en práctica una colección de valores, principios y prácticas que conducen al modelado de peso ligero.

2. Enfrentar el problema de la aplicación de técnicas de modelado en procesos de desarrollo ágiles.

3. Enfrentar el problema de la aplicación de las técnicas de modelado independientemente del proceso de software que se utilice.

##### 2.5.3.3.1 Los Valores de Modelado Ágil (AM)

Los valores de AM incluyen a los de XP: comunicación, simplicidad, feedback y coraje, añadiendo humildad. Una de las mejores caracterizaciones de los principios subyacentes a AM está en la definición de sus alcances [50].

**V1. AM es una actitud, no un proceso prescriptivo**: comprende una colección de valores a los que los modeladores ágiles se adhieren, principios en los que creen y prácticas que aplican. Describe un estilo de modelado; no es una metodología. AM es por lo tanto un suplemento de otros métodos. Como primer foco tiene el modelado y como segundo, la documentación.

V2. AM es una tarea de conjunto de los participantes: no hay “yo” en AM.

**V3. La prioridad es la efectividad**: AM ayuda a crear un modelo o proceso cuando se tiene un propósito claro y se comprenden las necesidades de los interesados; contribuye a aplicar los artefactos correctos para afrontar la situación inmediata y a crear los modelos más simples que sea posible.

**V4. AM es algo que funciona en la práctica, no una teoría académica**: las prácticas han sido discutidas desde la creación del manifiesto ágil.

V5. AM no es una bala de plata.

V6. AM es para el programador promedio, pero no reemplaza a la gente competente.

V7. AM no es un ataque a la documentación. La documentación debe ser mínima y relevante.

V8. AM no es un ataque a las herramientas CASE.

V9. AM no es para cualquiera.

2.5.3.3.2 Principios del Modelado Ágil

Los principios de AM especificados por Ambler [50] se describen a continuación:

**Pr1. Presuponer simplicidad:** la solución más simple es la mejor.

**Pr2. El contenido es más importante que la representación**: pueden ser notas, pizarras o documentos formales. Lo que importa no es el soporte físico o la técnica de representación, sino el contenido.

**Pr3. Aceptar el cambio:** aceptar que los requerimientos cambian.

**Pr4. Habilitar el esfuerzo siguiente:** garantizar que el sistema es suficientemente “robusto” para admitir mejoras posteriores; debe ser un objetivo, pero no el primordial.

**Pr5. Todas las personas pueden aprender del otro:** reconocer que nunca se domina realmente todo el conocimiento.

**Pr6. Cambio incremental**: no esperar hacerlo bien la primera vez.

**Pr7. Conocer tus modelos:** saber cuáles son sus fuerzas y sus debilidades.

**Pr8. Adaptación local**: producir sólo el modelo que resulte suficiente para el propósito.

Pr9. Maximizar la inversión del cliente.

**Pr10. Modelar con un propósito:** si no se puede identificar para qué se está haciendo algo ¿para qué molestarse haciéndolo?

**Pr11. Modelos múltiples:** múltiples paradigmas en convivencia, según se requiera.

Pr12. Comunicación abierta y honesta.

Pr13. Trabajo de calidad.

**Pr14. Realimentación rápida:** no esperar que sea demasiado tarde.

**Pr15. El software es el objetivo primario:** debe ser de alta calidad y coincidir con lo que el

usuario espera.

**Pr16. Viajar ligero de equipaje.** No crear más modelos de los necesarios.

Pr17. Trabajar con los instintos de la gente.

#### 2.5.3.3.3 Prácticas del Modelado Ágil

Lo más concreto de AM es un conjunto de prácticas [50], cada una de las cuales se asocia a lineamientos decididamente narrativos, articulados con minuciosidad:

P1. Colaboración activa de los participantes.

P2. Aplicación de estándares de modelado.

P3. Aplicación adecuada de patrones de modelado.

P4. Aplicación de los artefactos correctos.

P5.Propiedad colectiva de todos los elementos.

P6. Considerar la verificabilidad.

P7. Crear diversos modelos en paralelo.

P8. Crear contenido simple.

P9. Diseñar modelos de manera simple.

P10. Descartar los modelos transitorios.

P11. Exhibir públicamente los modelos.

P12. Formalizar modelos de contrato.

P13. Iterar sobre otro artefacto.

P14. Modelo en incrementos pequeños.

P15. Modelar para comunicar.

P16. Modelar para comprender.

P17. Modelar con otros.

P18. Poner a prueba con código.

P19. Reutilizar los recursos existentes.

P20. Actualizar sólo cuando duele.

P21. Utilizar las herramientas más simples (CASE, o mejor pizarras, tarjetas, etc).

#### 2.5.3.3.4 El proceso de Modelado Ágil (AM)

Como AM se debe usar como complemento de otras metodologías, nada se especifica sobre

métodos de desarrollo, tamaño del equipo, roles, duración de iteraciones, trabajo distribuido, etc., todo lo cual dependerá del método que se utilice. Los principios y prácticas citadas pueden dar impresión de puro sentido común; en realidad a los aspectos pertinentes al uso de herramientas de modelado y documentación, así como la articulación con metodologías específicas, las técnicas de bases de datos y el uso de artefactos están especificados cuidadosamente, como se puede constatar en el sitio de AM y en el libro de Ambler [50]. En este texto se muestra como el Modelado Ágil se puede aplicar a XP y al Proceso Unificado, cumpliendo sus metas de agilidad y documentación liviana.

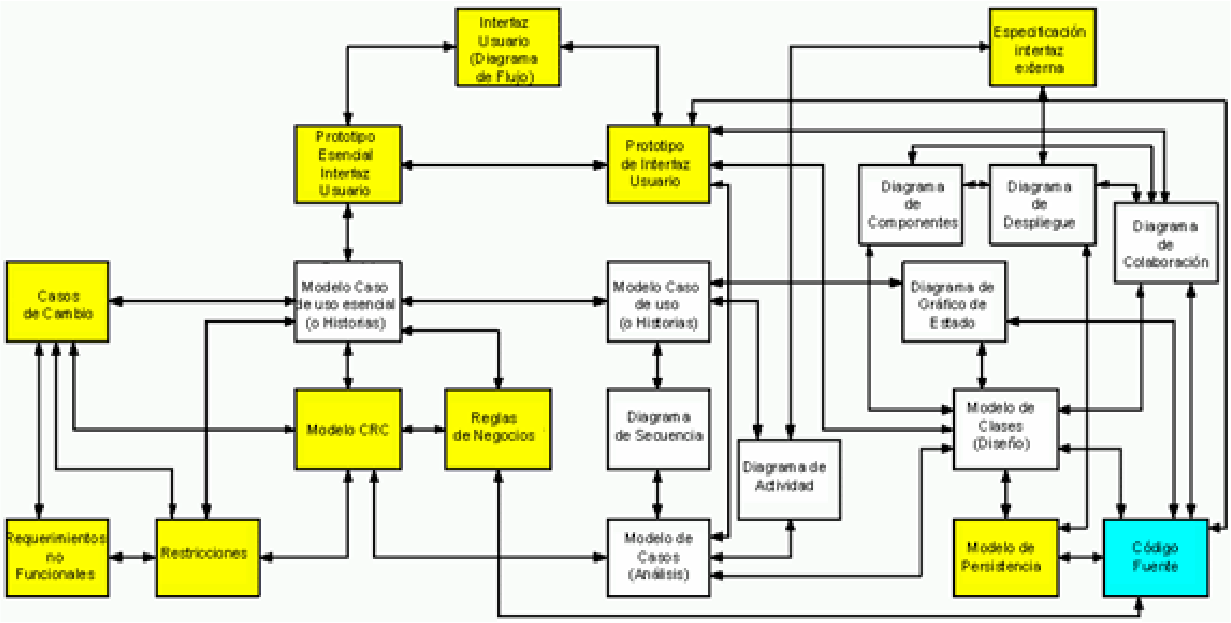



***Figura 22: Diagrama de artefactos de Modelado Ágil en el proceso UP agilizado***

Los diagramas de UML y los artefactos del Proceso Unificado, por ejemplo, han sido explorados en extremo detalle describiendo cómo debería ser su tratamiento en un proceso ágil, tal como se muestra en la figura 22. Las fases, los roles, los artefactos, el ciclo de vida, no son un objetivo para AM, por ello no presenta ninguna información de este tipo, a menos que se aplique sobre un proceso definido, como XP.

#### 2.5.3.4 Evolutionary Project Management (Evo)

Evo, creado por Tom Gilb, es el método iterativo ágil más antiguo [60]. Se lo llama también Evolutionary Delivery, Evolutionary Management, Requirements Driven Project Management y Competitive Engineering. Fue elaborado inicialmente en Europa. En 1976 Gilb trató temas de desarrollo iterativo y gestión evolutiva, luego desarrolló en profundidad esos temas y en 1981 publicó Evolutionary Development [61].

En la década de 1980 Gilb cayó bajo la influencia de los valores de W. Edward Deming y del método Planear-Hacer-Estudiar-Actuar (PDSA) de Walter Shewhart, que se constituyeron en modelos conceptuales subyacentes a Evo. En los 90s Gilb continuó el desarrollo de Evo, que tal vez fue más influyente por las ideas que éste proporcionara a XP, Scrum e incluso UP que por su éxito como método particular.

El modelo de Evo consiste en cinco elementos mayores:

1. **Metas, Valores y Costos –** Cuánto y cuántos recursos. Las Metas y Valores de los Participantes se llaman también, según la cultura, objetivos, metas estratégicas, requerimientos, propósitos, fines, ambiciones, cualidades e intenciones.

2. **Soluciones –** Banco de ideas sobre la forma de alcanzar Metas y Valores dentro del rango de los Costos.

3. **Estimación de Impacto –** Mapear las Soluciones a Metas y Costos para averiguar si se tienen ideas adecuadas para lograr las Metas dentro de los Costos.

4. **Plan Evolutivo –** Inicialmente una idea general de la secuencia a desarrollar y evolucionar hacia las Metas. Los detalles necesarios evolucionan junto con el resto del plan a medida que se desarrolla el producto/servicio.

5. **Funciones –** Describen qué hace el sistema. Son extremadamente secundarias, más de lo que se piensa, y deben mantenerse al mínimo.

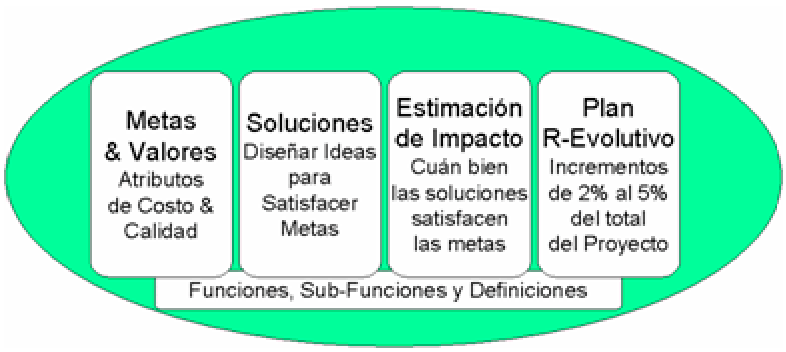



***Figura 23. Elementos de Evo***

En las breves iteraciones de Evo, se efectúa un progreso hacia las máximas prioridades definidas por el cliente, liberando algunas piezas útiles para algunos participantes y solicitando su retroalimentación. Esta es la práctica que se ha llamado Planeamiento Adaptativo Orientado al Cliente y Entrega Evolutiva. Otra idea distintiva de Evo es la clara definición, cuantificación, estimación y medida de los requisitos de ejecución que necesitan mejoras. La ejecución incluye requisitos de calidad tales como robustez y tolerancia a fallas, al lado de estipulaciones cuantitativas de capacidad de carga y de ahorro de recursos. En Evo se espera que cada iteración

constituya una re-evaluación de las soluciones en procura de la más alta relación de valor contra costo, teniendo en cuenta tanto la retroalimentación como un amplio conjunto de estimaciones métricas. Evo requiere, igual que otros MAs, activa participación de los clientes. Todo debe cuantificarse; se desalientan las apreciaciones cualitativas o subjetivas como "usable", "mantenible" o "ergonómico". A diferencia de otros MAs, en Evo hay una especificación semántica y una pragmática rigurosa, completamente alejadas del sentido común, pero con la fundamentación que les presta se ha derivado de prácticas productivas suficientementes probada [62].

#### 2.5.3.4.1 Principios de Evo

Los diez principios fundamentales de Evo son:

**Pr1.** Se entregarán temprano y con frecuencia resultados verdaderos, de valor para los participantes reales.

**Pr2.** El siguiente paso de entrega de Evo será el que proporcione el mayor valor para el participante en ese momento.

**Pr3.** Los pasos de Evo entregan los requerimientos especificados de manera evolutiva.

**Pr4.** No podemos saber cuáles son los requerimientos por anticipado, pero podemos descubrirlos más rápidamente intentando proporcionar valor real a los participantes reales.

**Pr5.** Evo es ingeniería de sistemas holística, todos los aspectos necesarios del sistema deben ser completos y correctos, con entrega a un ambiente de participantes reales (no es sólo sobre programación; es sobre satisfacción del cliente).

**Pr6.** Los proyectos de Evo requieren una arquitectura abierta, porque habremos de cambiar las ideas del proyecto tan a menudo como se necesite hacerlo, para entregar realmente valor a nuestros participantes.

**Pr7.** El equipo del proyecto de Evo concentrará su energía como equipo hacia el éxito del paso actual. En este paso tendrán éxito o fracasarán todos juntos. No gastarán energías en pasos futuros hasta que hayan dominado los pasos actuales satisfactoriamente.

**Pr8.** Evo tiene que ver con aprendizaje a partir de la dura experiencia. Tan rápido como se pueda se debe responder: ¿qué es lo que verdaderamente funciona?, ¿qué es lo que realmente entrega valor?. Evo es una disciplina que hace confrontar los problemas tempranamente, pero que nos permite progresar rápido cuando se hace bien las cosas.

**Pr9.** Evo conduce a una entrega temprana, a tiempo, porque se lo ha priorizado así desde el inicio, y porque se debe aprender desde el principio a hacer las cosas bien.

**Pr10.** Evo permite poner a prueba nuevos procesos de trabajo y deshacerse tempranamente de los que funcionan mal.

2.5.3.4.2 El Proceso

Evo distingue claramente entre los Pasos de la Entrega Evolutiva y las iteraciones propias de los modelos iterativos tradicionales o de algún método ágil. Las iteraciones siguen un modelo de flujo, tienen un diseño preestablecido y son una secuencia de construcciones definidas desde el principio, ver la figura 24; los pasos evolutivos se definen primero de una manera genérica y luego se van refinando en cada ciclo, adoptando un carácter cada vez más formal.

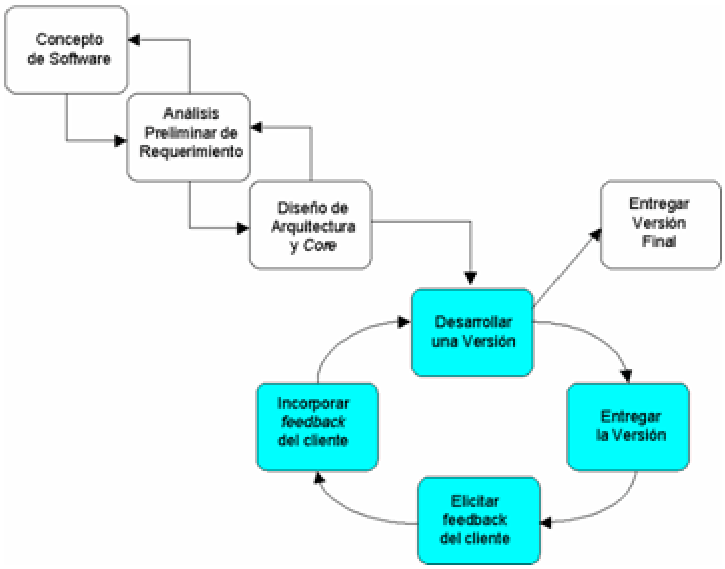



***Figura No. 24 Modelo de entrega evolutiva en Evo***

En proyectos evolutivos, las metas se desarrollan tratando de comprender de quiénes vienen (participantes), qué es lo que son (medios y fines) y cómo expresarlas (cuantificables, medibles y verificables). Se procura pasar el menor tiempo posible en tareas de documentación. En las formas más simples y relajadas de entrega evolutiva, a veces llamada entrega incremental. Liberar parte de una solución es un paso de entrega evolutiva. En la entrega evolutiva más pura y más rica, sólo las mejoras en las metas de los participantes se consideran un paso de entrega evolutiva. Hay que distinguir bien entre los medios y los fines, por un lado, y las metas y las soluciones por el otro. Las metas deben separarse de las soluciones; las metas deben ser sagradas, y se deben alcanzar por cualquier medio; las soluciones son sólo posibles, contingentes: son caballos de trabajo, y deben cambiarse cuando se consigue un caballo mejor.

2.5.3.5 Métodos Crystal

Las metodologías Cristal son una familia de metodologías, fueron creadas por el Alistair Cockburn quien escribe que mucha gente piensa que el desarrollo de software es una actividad de

ingeniería. Esa comparación, piensa, es de hecho más perniciosa que útil, y nos lleva en una dirección equivocada.

Comparar el software con la ingeniería nos conduce a preguntarnos sobre "especificaciones" y "modelos" del software, sobre su completitud, corrección y vigencia. Esas preguntas son inconducentes, porque cuando pasa cierto tiempo no nos interesa que los modelos sean completos, que coincidan con el mundo "real" (sea ello lo que fuere) o que estén al día con la versión actual del lenguaje. Intentar que así sea es una pérdida de tiempo [63].

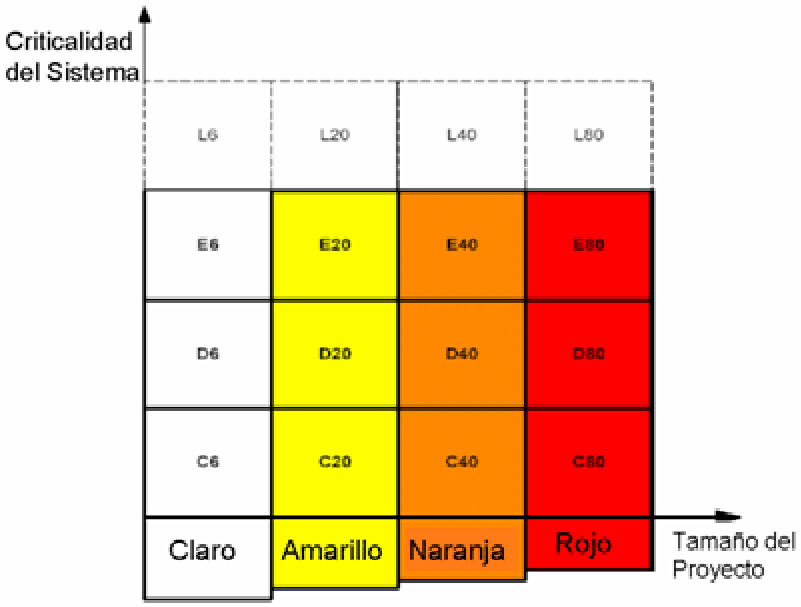



***Figura 25. Familia de Métodos Crystal***

La familia Crystal dispone un código de color para marcar la complejidad de una metodología: cuanto más oscuro un color, más "pesado" es el método. Ver figura 25. Cuanto más crítico es un sistema, más rigor se requiere. El código cromático se aplica a una forma tabular elaborada por Cockburn que se usa en muchos MAs para situar el rango de complejidad al cual se aplica una metodología. En la figura se muestra una evaluación de las pérdidas que puede ocasionar la falla de un sistema y el método requerido según este criterio. Los parámetros son Comodidad (C), Dinero Discrecional (D), Dinero Esencial (E) y Vidas (L). En otras palabras, la caída de un sistema que ocasione incomodidades indica que su nivel crítico es C, mientras que si causa pérdidas de vidas su nivel es L. Los números del cuadro indican el número de personas afectadas a un proyecto.

Los métodos se llaman Crystal evocando las facetas de una gema: cada faceta es otra versión del proceso, y todas se sitúan en torno a un núcleo idéntico. Hay cuatro variantes de metodologías: Crystal Clear ("Claro como el cristal") para equipos de 8 o menos integrantes; Amarillo, para 8 a 20; Naranja, para 20 a 50; Rojo, para 50 a 100. Se promete seguir con Marrón, Azul y Violeta. La más exhaustivamente documentada es Crystal Clear (CC), y es la que se describe a continuación. CC puede ser usado en proyectos pequeños de categoría D6, aunque con alguna extensión se aplica también a niveles E8 a D10. El otro método elaborado en profundidad es el Naranja, apto

para proyectos de duración estimada en 2 años. Como casi todos los otros métodos, CC consiste en valores, técnicas y procesos

#### 2.5.3.5.1 Valores de CC.

**V1. Entrega frecuente:** consiste en entregar software a los clientes con frecuencia, no solamente en compilar el código. La frecuencia dependerá del proyecto, pero puede ser diaria, semanal, mensual o lo que se determine. La entrega puede hacerse sin despliegue, si es que se consigue algún usuario cortés o curioso que suministre retroalimentación.

**V2. Comunicación osmótica:** todos juntos en el mismo cuarto. Una variante especial es disponer en la sala de un diseñador Senior; eso se llama Experto al Alcance de Todos.

**V3. Mejora reflexiva:** tomarse un pequeño tiempo (unas pocas horas cada semana o una vez al mes) para pensar bien qué se está haciendo, ver las notas, reflexionar y discutir.

**V4. Seguridad personal:** hablar cuando algo molesta: decirle amigablemente al manager que la agenda no es realista, o a un colega que su código necesita mejorarse, o incluso de asuntos personales. Esto es importante porque el equipo puede descubrir y reparar sus debilidades. No es provechoso tratar de ocultar los desacuerdos con gentileza y conciliación. Técnicamente, estas cuestiones se han caracterizado como una importante variable de confianza y han sido estudiadas con seriedad en la literatura.

**V5. Foco:** saber lo que se está haciendo y tener la tranquilidad y el tiempo para hacerlo. LA tranquilidad debe venir de la comunicación sobre dirección y prioridades, típicamente con el patrocinador Ejecutivo. El tiempo para hacerlo de un ambiente en que la gente no se le exija hacer otras cosas incompatibles.

**V6. Fácil acceso a usuarios expertos:** el contacto con expertos en el desarrollo de un proyecto debe ser directo. Uno o dos encuentros semanales con llamados telefónicos adicionales puede ser una buena pauta. Otra variante es que los programadores se entrenen para ser usuarios durante un tiempo. El equipo de desarrollo, de todas maneras, incluye un experto en negocios.

**V7. Ambiente técnico con prueba automatizada, gestión de la configuración e integración frecuente.** Se puede hacer construcciones cotidianas, y no es una mala práctica, pero debe haber soporte tecnológico para que muchos equipos ágiles compilan e integran varias veces al día.

#### 2.5.3.5.2 El proceso de CC

El proceso de CC se basa en una exploración refinada de los inconvenientes de los modelos clásicos. Dice Cockburn que la mayoría de los modelos de proceso propuestos entre 1970 y 2000 se describían como secuencias de pasos. Aún cuando se recomendaran iteraciones e incrementos (que no hacían más que agregar confusión a la interpretación) los modelos parecían dictar un proceso en cascada, por más que los autores aseguraran que no era así. El problema con estos procesos es que realmente están describiendo un flujo de trabajo requerido, un grafo de dependencia: el equipo no puede entregar un sistema hasta que esté integrado y ejecute

adecuadamente. No puede integrar y verificar hasta que el código no está escrito y corriendo. Y no puede diseñar y escribir el código hasta que se le dice cuáles son los requerimientos. Un grafo de dependencia se interpreta necesariamente en ese sentido, aunque no haya sido la intención original.

En lugar de esta interpretación lineal, CC enfatiza el proceso como un conjunto de ciclos anidados. En la mayoría de los proyectos se perciben siete ciclos: (1) el proyecto, (2) el ciclo de entrega de una unidad, (3) la iteración, (4) la semana laboral, (5) el período de integración, de 30 minutos a tres días, (6) el día de trabajo, (7) el episodio de desarrollo de una sección de código, de pocos minutos a pocas horas, como se puede observar en la figura 26.

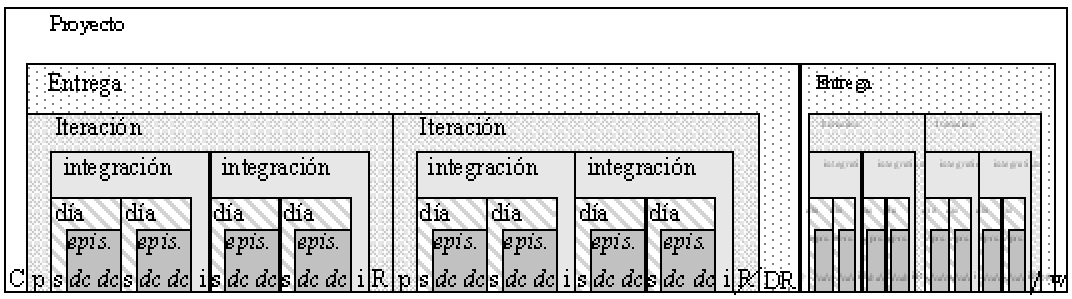



***Figura 26. Ciclos anidados de Crystal Clear***

A pesar que no contempla el desarrollo de software propiamente dicho, CC involucra unos veinte productos de trabajo o artefactos. Mencionamos los más importantes:

**Art1. Declaración de la misión:** documento de un párrafo a una página, describiendo el propósito.

**Art2. Estructura del equipo:** lista de equipos y miembros.

**Art3. Metodología:** comprende roles, estructura, proceso, productos de trabajo que mantienen, métodos de revisión.

**Art4. Secuencia de entrega:** declaración o diagrama de dependencia; muestra el orden de las entregas y lo que hay en cada una.

**Art5. Cronograma de visualización y entrega:** lista, planilla de hoja de cálculo o herramienta de gestión de proyectos.

**Art6. Lista de riesgos:** descripción de riesgos por orden descendente de prioridad.

**Art7. Estatus del proyecto:** lista hitos, fecha prevista, fecha efectiva y comentarios.

**Art8. Lista de actores-objetivos:** lista de dos columnas, planilla de hoja de cálculo, diagrama de caso de uso, historia de usuario o similar.

Art9. Casos de uso o historias de usuario anotados: requerimientos funcionales.

**Art10. Archivo de requisitos:** colección de información indicando qué se debe construir, quiénes han de utilizarlo, de qué manera proporciona valor y qué restricciones afectan al diseño.

Los métodos Crystal no prescriben las prácticas de desarrollo, las herramientas o los productos que pueden usarse, pudiendo combinarse con otros métodos como Scrum, XP y Microsoft Solutions Framework. Hay ocho roles nominados en CC: Patrocinador, Usuario Experto, Diseñador Principal, Diseñador-Programador, Experto en Negocios, Coordinador, Verificador, Escritor. En Crystal Naranja se agregan aun más roles: Diseñador de IU, Diseñador de Base de Datos, Experto en Uso, Facilitador Técnico, Analista/Diseñador de Negocios, Arquitecto, Mentor de Diseño. A continuación se describen los roles y sus responsabilidades.

**R1. Patrocinador:** produce la declaración de misión con prioridades de compromiso, consigue los recursos y define la totalidad del proyecto.

**R2. Usuario Experto**: junto con el experto en negocios produce la lista de actores-objetivos y el archivo de casos de uso o historias de usuario y los requisitos. Debe familiarizarse con el uso del sistema, sugerir atajos de teclado, modos de operación, información a visualizar simultáneamente, navegación, etc.

**R3. Diseñador Principal:** produce la descripción arquitectónica. Debe ser al menos un profesional de alto nivel. El diseñador principal tiene roles de coordinador, arquitecto, mentor y programador más experto.

**R4. Diseñador-Programador:** produce, junto con el diseñador principal, los borradores de interfaces de usuario, el modelo de dominio, las notas y diagramas de diseño, el código fuente, el código de migración, las pruebas y el sistema empaquetado. CC no distingue entre diseñadores y programadores. Un programa en CC es "diseño y programa"; sus programadores son diseñadores-programadores. En CC un diseñador que no programe no tiene cabida.

**R5. Experto en Negocios:** junto con el usuario experto produce la lista de actores-objetivos y el archivo de casos de uso y los requerimientos. Debe conocer las reglas y políticas del negocio.

**R6. Coordinador:** con la ayuda del equipo, produce el mapa de proyecto, el plan de entrega, el estado del proyecto, la lista de Riesgos, el plan y estado de iteración y la agenda de visualización.

**R7. Verificado**r: produce el reporte de defectos. Puede ser un programador en tiempo parcial, o un equipo de varias personas.

**R8. Escritor:** produce el manual de usuario.

**R9. El Equipo como Grupo:** es responsable de producir la estructura y convenciones del Equipo y los resultados del taller de reflexión

2.5.3.5.3 Estrategias y técnicas de CC

Las estrategias documentadas de CC, comunes a otros Métodos Ágiles, son:

**E1. Exploración de 360°**: verificar o tomar una muestra del valor de negocios del proyecto, los requisitos, el modelo de dominio, la tecnología, el plan del proyecto y el proceso. La exploración es preliminar al desarrollo y equivale al período de incepción de UP. Mientras en RUP esto puede demandar algunas semanas o meses, en Crystal Clear debe consumir unos pocos días, máximo dos semanas si se requiere usar una tecnología nueva o inusual. El modelo de negocio se puede hacer verbalmente, con casos de uso u otros mecanismos de listas, pero debe resultar en una lista de los casos de uso esenciales del sistema. Respecto de la tecnología conviene correr unos pocos experimentos basándose en los Spikes de Scrum.

**E2. Victoria temprana:** es mejor buscar pequeños triunfos iniciales que aspirar a una gran victoria tardía. La fundamentación de esta estrategia proviene de algunos estudios sociológicos específicos. Usualmente la primera victoria temprana consiste en la construcción de un Esqueleto Ambulante. Conviene no utilizar la técnica de "lo peor primero" de XP, porque puede bajar la moral. La preferencia de Cockburn es "lo más fácil primero, lo más difícil segundo".

**E3. Esqueleto ambulante:** es una transacción que debe ser simple pero completa. Podría ser una rutina de consulta y actualización en un sistema cliente-servidor, o la ejecución de una transacción en un sistema transaccional de negocios. Un Esqueleto Ambulante no suele ser robusto; sólo camina, y carece de la carne de la funcionalidad de la aplicación real, que se agregará incrementalmente. El Esqueleto debe producirse con buenos hábitos de producción y pruebas de regresión, y está destinado a crecer con el sistema.

**E4. Reconstruir la arquitectura de manera incremental:** se ha demostrado que no es conveniente interrumpir el desarrollo para corregir la arquitectura. Más bien la arquitectura debe evolucionar en etapas, manteniendo el sistema en ejecución mientras ella se modifica.

**E5. Radiadores de información:** consiste de una lámina pegada en algún lugar que el equipo pueda observar mientras trabaja o camina. Tiene que ser comprensible para el observador casual, entendida de un vistazo y renovada periódicamente para que valga la pena visitarla. Puede estar en una página Web, pero es mejor si está en una pared, porque en la pantalla se acumulan tantas cosas que ninguna llama la atención. Podría mostrar los requisitos de la iteración actual, el número de pruebas pasadas o pendientes, el número de casos de uso o historias de usuario entregado, el estado de los servidores, los resultados del último taller de reflexión, etc.

Crystal Clear favorece las siguientes prácticas:

**P1. Entrevistas de proyectos:** se suele entrevistar a más de un responsable para tener visiones más ricas. La idea es averiguar cuales son las prioridades, obtener una lista de rasgos deseados, saber cuáles son los requerimientos más críticos y cuáles los más negociables. Si se trata de una actualización o corrección, se debe saber cuáles son las cosas que se hicieron bien y merecen preservarse y los errores que no se quieren repetir.

**P2. Talleres de reflexión:** el equipo debe detenerse treinta minutos o una hora para reflexionar sobre sus convenciones de trabajo, discutir inconvenientes y mejoras y planear para el período siguiente. De aquí puede salir material para poner en un poster como Radiador de Información.

**P3. Planeamiento Blitz.** Una técnica puede ser el Juego de Planeamiento de XP. En este juego, se ponen tarjetas indexadas en una mesa, con una historia de usuario o función visible en cada una. El grupo finge que no hay dependencias entre tarjetas, y las alinea en secuencias de desarrollo preferidas. Los programadores escriben en cada tarjeta el tiempo estimado para desarrollar cada función. El representante del usuario escribe la secuencia de prioridades, teniendo en cuenta los tiempos referidos y el valor de negocio de cada función. Las tarjetas se agrupan en períodos de tres semanas llamados iteraciones que se agrupan en entregas (releases), usualmente no más largas de tres meses. Pueden usarse tarjetas CRC. Las diferencias entre la versión de Cockburn y el juego de XP están en que, en XP las tarjetas tienen historias, en CC listas de tareas; el juego de XP asume que no hay dependencias, el de CC que sí las hay.

**P4. Estimación Delphi con estimaciones de pericia:** La técnica se llama así por analogía con el oráculo de Delfos. En el proceso Delphi se reúnen los expertos responsables y proceden como en un remate para proponer el tamaño del sistema, su tiempo de ejecución, la fecha de las entregas según dependencias técnicas y de negocios y para equilibrar las entregas en paquetes de igual

tamaño.

**P5. Encuentros diarios de pie:** la palabra clave es "brevedad", cinco a diez minutos como máximo. No se trata de discutir problemas, sino de identificarlos. Los problemas sólo se discuten en otros encuentros posteriores, con la gente que tiene que ver en ellos. La técnica se origina en Scrum. Se deben hacer de pie para que la gente no escriba en sus computadores portátiles, agendas electrónicas, en papeles o se quede dormida.

**P6. Miniatura de procesos:** la "Hora Extrema" fue inventada por Peter Merel para introducir a la gente en XP en 60 minutos y proporciona lineamientos canónicos que pueden usarse para articular esta práctica. Una forma de presentar Crystal Clear puede consumir entre 90 minutos y un día. La idea es que la gente pueda "degustar" la nueva metodología.

**P7. Gráficos de quemado:** su nombre viene de los gráficos de quemado de calorías de los regímenes dietéticos; se usan también en Scrum. Se trata de una técnica de graficación para descubrir demoras y problemas tempranamente en el proceso, evitando que se descubra demasiado tarde que todavía no se sabe cuánto falta. Para ello se hace una estimación del tiempo faltante para programar lo que resta al ritmo actual, lo cual sirve para tener dominio de proyectos en los cuales las prioridades cambian bruscamente y con frecuencia. Esta técnica se asocia con algunos recursos ingeniosos, como la Lista Témpana, llamada así porque se refiere al agregado de ítems con alta prioridad en el tope de las listas de trabajos pendientes, esperando que los demás elementos se hundan bajo la línea de corte; los elementos que están sobre la línea se entregarán en la iteración siguiente, los que están por debajo en las restantes. En otros métodos ágiles la Lista temprana no es otra cosa que un gráfico de retraso.

**P8. Programación lado a lado:** mucha gente siente que la programación en pares de XP involucra una presión excesiva y no la patrocina; la versión de Crystal Clear establece proximidad, pero cada quien se dedica a su trabajo asignado, prestando un ojo a lo que hace su compañero, quien tiene su propia máquina. Esta es una ampliación de la Comunicación Osmótica al contexto de la programación.

**P9. Estacionamiento:** el estacionamiento incluye la planeación del próximo incremento del sistema. El equipo selecciona los requisitos a ser implementados en el incremento y programa cuales de ellos consideran deben entregar.

**P10. Revisión:** cada incremento incluye varias iteraciones. Cada iteración incluye las siguientes actividades: construcción, demostración y revisión de los objetivos del incremento.

**P11. Monitoreo:** el progreso es monitoreado teniendo en cuenta las entregas del equipo durante el proceso de desarrollo con respecto al progreso y estabilidad de estas. El progreso es medido por avances (inicio, i-ésima revisión, pruebas, entrega).

**P12. Paralelismo y flujo:** una vez el monitoreo es estable se da el resultado "suficientemente estable al revisar" y se pueden empezar nuevas tareas. En crystal orange, esto significa que los múltiples equipos pueden proceder con máximo paralelismo satisfactoriamente. Para alcanzar esto, el equipo de monitoreo y arquitectura revisan sus planes de trabajo, estabilidad y sincronización.

**P13. Diversidad de estrategia holística:** Crystal orange incluye un método llamado diversidad de estrategia holística para dividir grandes grupos funcionales en grupos cruzados. La idea central de esto es incluir multiples especialistas en un solo equipo. La diversidad de estrategia holística

también permite formar pequeños grupos con el conocimiento específico necesario, y también considera aspectos como la localización de los equipos, comunicación y documentación y coordinación de equipos multiples.

**P14. Sintonización metodológica:** utiliza entrevistas del proyecto y talleres del equipo para resolver un método cristal especifico para cada proyecto individual (Tailoring). Una de las ideas del desarrollo incremental es permitir la mejora del proceso de desarrollo. En cada incremento, puede utilizar el conocimiento aprendido para desarrollar el proceso para el incremento siguiente.

**P15. Puntos de vista del usuario:** sugeridos por Crystal Clear y Orange En crystal orange, los puntos de vista del usuario deben ser organizados tres veces por cada incremento.

Crystal Clear y Crystal naranja no definen ninguna práctica o técnica a ser usada por los miembros del proyecto en sus tareas de desarrollo de software. La adopción de prácticas de otras metodologías como XP y Scrum es permitida en crystal para reemplazar algunas prácticas propias, como talleres de reflexión, por ejemplo.

#### 2.5.3.6 Desarrollo Manejado por Rasgos – (Feature Driven Develpment - FDD)

FDD es un método ágil, iterativo y adaptativo. A diferencia de otros métodos ágiles, no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción y se considera adecuado para proyectos mayores y de misión crítica [64], tampoco requiere de un proceso específico para ser usado. FDD contiene desarrollo iterativo con las mejores prácticas encontradas para ser efectivo en la industria. Hace énfasis en aspectos de calidad a lo largo de los procesos e incluye entregas tangibles y pequeñas, junto con monitoreo preciso del progreso del proyecto.

FDD no requiere un modelo específico de proceso y se complementa con otras metodologías. Enfatiza cuestiones de calidad y define claramente entregas tangibles y formas de evaluación del progreso. FDD es, además, marca registrada de una empresa, Nebulon Pty. Aunque hay coincidencias entre la programación orientada por características (Features) y el desarrollo guiado por rasgos, FDD no necesariamente implementa FOP – Feature Oriented Progamming.

##### 2.5.3.6.1 El proceso de FDD

FDD consiste en cinco procesos secuenciales durante los cuales se diseña y construye el sistema. La parte iterativa soporta desarrollo ágil con rápidas adaptaciones a cambios en requisitos y necesidades del negocio. Cada fase del proceso tiene un criterio de entrada, tareas, pruebas y un criterio de salida. Típicamente, la iteración de un rasgo insume de una a tres semanas. Ver la figura 27. Las fases se describen a continuación:

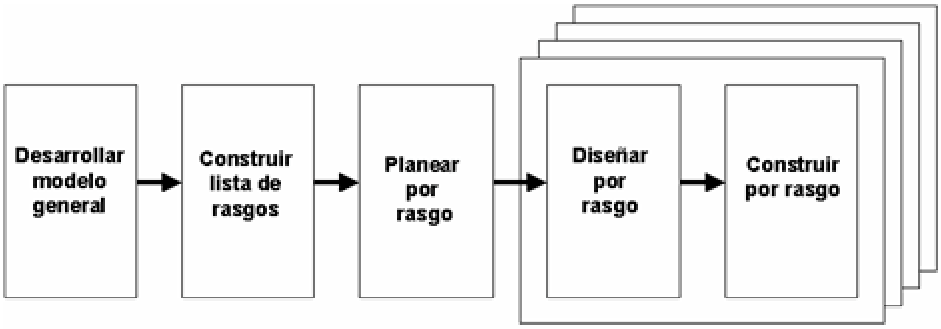



***Figura 27. Proceso de FDD***

**F1. Desarrollo de un modelo general:** cuando comienza este desarrollo, los expertos de dominio ya están al tanto de la visión, el contexto y los requisitos del sistema a construir. En esta fase se espera que existan requisitos tales como casos de uso o especificaciones funcionales. FDD, sin embargo, no cubre este aspecto. Los expertos de dominio presentan un ensayo (walkthrough) en el que los miembros del equipo y el arquitecto principal se informan de la descripción de alto nivel del sistema. El dominio general se subdivide en áreas más específicas y se define un ensayo más detallado para cada uno de los miembros del dominio. Luego de cada ensayo, un equipo de desarrollo trabaja en pequeños grupos para producir modelos de objeto de cada área de dominio. Simultáneamente, se construye un gran modelo general para todo el sistema.

**F2. Construcción de la lista de rasgos:** los ensayos, modelos de objeto y documentación de requisitos proporcionan la base para construir una amplia lista de rasgos. Los rasgos son pequeños ítems útiles a la vista del cliente. Son similares a las tarjetas de historias de XP y se escriben en un lenguaje que todas las partes puedan entender. Las funciones se agrupan conforme a diversas actividades en áreas de dominio específicas. La lista de rasgos es revisada por los usuarios y patrocinadores para asegurar su validez y completitud. Los rasgos que requieran más de diez días se descomponen en otros más pequeños.

**F3. Planeación por rasgo:** incluye la creación de un plan de alto nivel, en el que los conjuntos de rasgos se ponen en secuencia conforme a su prioridad y dependencia, y se asigna a los programadores jefes. Las listas se priorizan en secciones que se llaman paquetes de diseño. Luego se asignan las clases definidas en la selección del modelo general a programadores individuales, o sea propietarios de clases. Se pone fecha a los conjuntos de rasgos.

**F4. Diseño y construcción por rasgo:** se selecciona un pequeño conjunto de rasgos del conjunto y los propietarios de las clases seleccionan los correspondientes equipos dispuestos por rasgos. Se procede luego iterativamente hasta que se producen los rasgos seleccionados. Una iteración puede tomar de unos pocos días a un máximo de dos semanas. Puede haber varios grupos trabajando en paralelo. El proceso iterativo, como se muestra en la figura 28, incluye ***inspección de diseño, codificación, prueba de unidad, integración e inspección de código***. Luego de una iteración exitosa, los rasgos completos se llevan la construcción principal. Este proceso puede demorar una o dos semanas en implementarse.

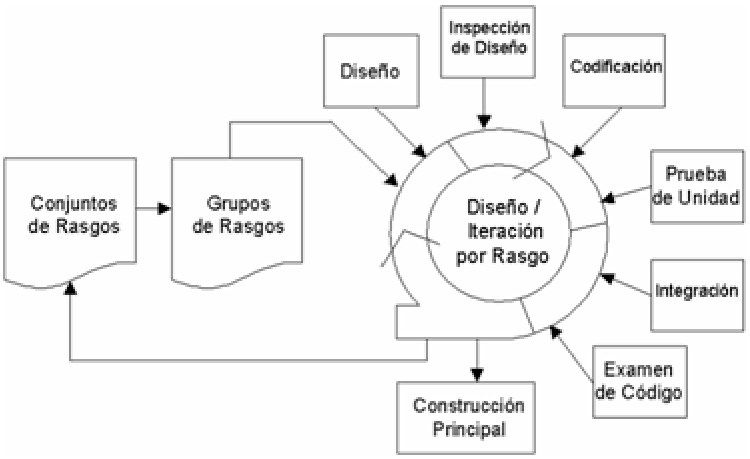



***Figura 28: Iteración de FDD***

Hay tres categorías de rol en FDD: roles claves, roles de soporte y roles adicionales. A continuación se presentan los roles y responsabilidades.

Roles clave:

**R1. Administrador del proyecto:** es el líder administrativo y financiero del proyecto. Una de sus tareas es proteger al equipo del proyecto de distracciones externas y capacitar al equipo para trabajar conjuntamente en condiciones apropiadas. Es quien decide en materia de visión, cronograma y asignación del personal.

**R2. Arquitecto jefe:** el diseñador jefe es responsable por el diseño completo del sistema, es quien toma la decisión final en todos los aspectos de diseño. Si es necesario este rol puede ser dividido en arquitecto de dominio y arquitecto técnico.

**R3. Manager de desarrollo:** es el encargado de las actividades diarias de desarrollo y de solucionar cualquier conflicto que pueda presentarse en el equipo. Además, este rol incluye la responsabilidad de solucionar problemas de recursos. Las responsabilidades de este rol pueden combinarse con el arquitecto jefe o manager de proyecto.

**R4 Programador jefe:** es un desarrollador experimentado, que participa en el análisis de requisitos y el diseño de proyectos. El programador jefe es responsable de conducir a equipos pequeños en el análisis, diseño e implementación de nuevos rasgos. Selecciona rasgos del conjunto de rasgos para ser desarrollados en la iteración siguiente e identifica la clase y los propietarios de clase que se necesitan en el equipo.

**R5 Propietario de clase:** trabaja bajo la guía del programador jefe en diseño, codificación, prueba y documentación, repartido por rasgos. Es el responsable del desarrollo de la clase que le han asignado como propietario. Los propietarios de clase forman equipos de rasgos. Por cada iteración los propietarios de clase están involucrados en las clases que se incluyan en los rasgos seleccionados para el desarrollo de la siguiente iteración.

**R6 experto de dominio:** puede ser un cliente, patrocinador, analista de negocios o una mezcla de todo estos. Su tarea es definir y compartir el conocimiento de cómo los diversos requisitos para el sistema en desarrollo deben realizarse. Los expertos de dominio pasan este conocimiento a los desarrolladores para asegurarse que los desarrolladores entregan un sistema competente.

Roles de soporte

**R7 Administrador de entrega:** controla el progreso del proceso revisando los reportes del programador jefe y manteniendo reuniones breves con él; reporta al manager del proyecto.

**R8 Abogado (gurú) de la tecnología:** es un miembro del equipo que debe tener todo el conocimiento necesario relacionado con un lenguaje o una tecnología especifica. Este rol es particularmente importante cuando el equipo de desarrollo se esta ocupando de una nueva tecnología.

**R9. Ingeniero de construcción:** es el responsable de la creación, funcionamiento y mantenimiento de la estructura del proceso, incluye la tarea de manejar el sistema de control y de publicar la documentación.

**R10. Herramientista:** es un rol para construcción de pequeñas herramientas para desarrollo, pruebas y conversión de datos en el equipo del proyecto. También puede trabajar manteniendo bases de datos y sitios Web para proyectos de propósito específico en el proyecto.

**R11. Administrador del sistema:** debe configurar, manejar y localizar averías en los servidores, sitios de red y de los ambientes de desarrollo y pruebas usados por el equipo del proyecto.

Roles adicionales:

**R12. Verificadores:** verifican que el sistema producido satisface los requisitos del cliente. Puede ser un equipo independiente o una parte del equipo del proyecto.

**R13. Instalador (Deployer):** el trabajo del instalador está relacionado con la conversión de datos existentes al formato requerido por el nuevo sistema y participar en el despliegue de nuevos lanzamientos. Pueden ser un equipo independiente o parte del equipo del proyecto.

**R14. Escritor técnico:** la documentación del usuario es preparada por el escritor técnico, que también puede ser un equipo independiente o parte del equipo del proyecto.

FDD consiste en un conjunto de “mejores prácticas” y aunque las prácticas seleccionadas no son nuevas, la mezcla especifica de estas hace que los cinco procesos de FDD sean únicos. FDD recomienda que todas las prácticas disponibles se utilicen para obtener las máximas ventajas del método, pues ninguna práctica domina el proceso completo.

**P1. Modelado de objetos del dominio**: resultante en un framework cuando se agregan los rasgos. Esta forma de modelado descompone un problema mayor en otros menores; el diseño y la implementación de cada clase u objeto es un problema pequeño a resolver. Cuando se combinan las clases completas, constituyen la solución al problema mayor. Una forma particular de la técnica es el modelado en colores, que agrega una dimensión adicional de visualización. Si bien se puede modelar en blanco y negro, en FDD el modelado basado en objetos es imperativo.

**P2. Desarrollo por rasgo:** hacer simplemente que las clases y objetos funcionen no refleja lo que el cliente pide. El seguimiento del progreso se realiza mediante examen de pequeñas funcionalidades descompuestas y funciones valoradas por el cliente. Un rasgo en FDD es una función pequeña expresada en la forma <acción> <resultado> <por | para | de | a> <objeto> con los operadores adecuados entre los términos. Por ejemplo, calcular el total de una venta; determinar la última operación de un cajero; validar la contraseña de un usuario.

**P3. Propiedad individual de clases (código):** cada clase tiene una sola persona nominada como

responsable por su consistencia, desempeño e integridad conceptual.

**P4. Equipos de Rasgos, pequeños y dinámicamente formados**: la existencia de un equipo garantiza que un conjunto de mentes se apliquen a cada decisión y se tomen en cuenta múltiples alternativas.

**P5. Inspección.** Se refiere al uso de los mejores mecanismos de detección conocidos.

**P6. Construcciones (builds) regulares**: siempre se tiene un sistema disponible. Las construcciones forman la base a partir de la cual se van agregando nuevos rasgos.

**P7. Administración de configuración**. Permite realizar seguimiento histórico de las últimas versiones completas de código fuente.

**P8. Reporte de progreso**: se comunica a todos los niveles organizacionales necesarios el estado actual del proyecto.

FDD es un método de desarrollo de ciclos cortos que se concentra en la fase de diseño y construcción. En la primera fase, el modelo global de dominio es elaborado por expertos del dominio y desarrolladores; el modelo de dominio consiste en diagramas de clases con clases, relaciones, métodos y atributos. Los métodos no reflejan conveniencias de programación sino rasgos funcionales. FDD es demasiado jerárquico para ser un método ágil, porque demanda un programador jefe, quien dirige a los propietarios de clases, quienes dirigen equipos de rasgos. No hay procedimientos detallados de prueba en FDD, pues se supone que las empresas ya tienen implementadas sus herramientas de prueba, pero subsiste el problema de su adecuación a FDD. Un rasgo llamativo de FDD es que no exige la presencia del cliente. .

#### 2.5.3.7 Desarrollo de Software Adaptativo – ASD

ASD se centra principalmente en los problemas de desarrollar sistemas grandes y complejos. El método anima fuertemente al desarrollo incremental, iterativo, con prototipado constante. Este método ágil pretende abrir una tercera vía entre el “desarrollo monumental de software” y el “desarrollo accidental”. Se pretende buscar más bien, “el rigor estrictamente necesario”; para ello hay que situarse en un punto apenas un poco fuera del caos y ejercer menos control que el que se cree necesario [65].

#### 2.5.3.8 Principios de ASD

ASD presenta las siguientes características que pueden ser tomados como principios del método:

**Pr1. Misión Dirigida:** las actividades en cada ciclo de desarrollo se deben justificar versus la misión total del proyecto. La misión puede ser ajustada, con el avance del desarrollo.

**Pr2. Basado en componentes:** el desarrollo de actividades no debe ser orientado a la tarea, en cambio, debe centrare en el funcionamiento del desarrollo de software. Por ejemplo, construir un Componente para acceso a Bases de Datos.

**Pr3. Iterativo:** el método serial en cascada trabaja bien solamente en ambientes bien definidos y bien entendidos. La mayoría de los desarrollos son turbulentos, y el esfuerzo del desarrollo se debe centrar en rehacer en vez de hacer todo correctamente la primera vez.

**Pr4. Caja de tiempo (Time-boxed):** en proyectos de software complejos la ambigüedad puede ser aliviada por la fijación de la fecha límite regularmente. La administración de un proyecto a caja de tiempo definido solicita a los participantes del proyecto tomar decisiones inevitables, duras de entender al inicio de un proyecto.

**Pr5. Tolerante al cambio:** los cambios son frecuentes en el desarrollo de software. Sin embargo, es más importante, estar capacitado para adaptarse a ellos que intentar controlarlos. Para construir un sistema tolerante a cambios los desarrolladores deben evaluar constantemente cómo o cuánto los componentes que están construyendo probablemente cambiaran.

**Pr6. Dirigido por el riesgo:** el desarrollo de los ítems de alto riesgo debe empezar tan temprano como sea posible.

2.5.3.8.1 Proceso

El ciclo de vida, ver Figura 29, de ASD se divide en las fases: especulativa, colaborativa y aprendizaje.

**F1. La fase de especulación** es utilizada en lugar de planeación, como un "plan", que está generalmente mirando la incertidumbre como una debilidad, y donde las desviaciones indican fallas. En la iniciación del proyecto se define la misión del proyecto. La misión básicamente hace un bosquejo de los objetivos del producto, y se dirige todo el desarrollo de modo que la misión sea lograda. Una de las cosas importantes al definir la misión del proyecto es calcular que información es necesaria para realizar el proyecto. Las facetas importantes de la misión son definidas en tres ítems: una carta de visión del proyecto, una tabla de datos del proyecto y un contexto de especificaciones del producto. En la iniciación del proyecto se fija el horario y los objetivos para los ciclos de desarrollo. Los ciclos típicamente duran de cuatro a ocho semanas. De igual forma se realiza un planea para cada ciclo adaptativo, planear los ciclos de vida es parte del proceso de desarrollo iterativo.

**F2. Fase colaborativa:** se realza la importancia del trabajo en equipo como la manera de desarrollar sistemas altamente cambiantes. ASD es explícitamente orientado a los componentes más que a la función. En la práctica esto significa que esta enfocado mas en obtener resultados y su calidad que a las tareas o al proceso utilizado para producir dicho resultado. La manera como ASD direcciona este punto de vista es a través de ciclos de desarrollo adaptativos que contiene la fase colaborativa, donde varios componentes pueden estar en desarrollo concurrente.

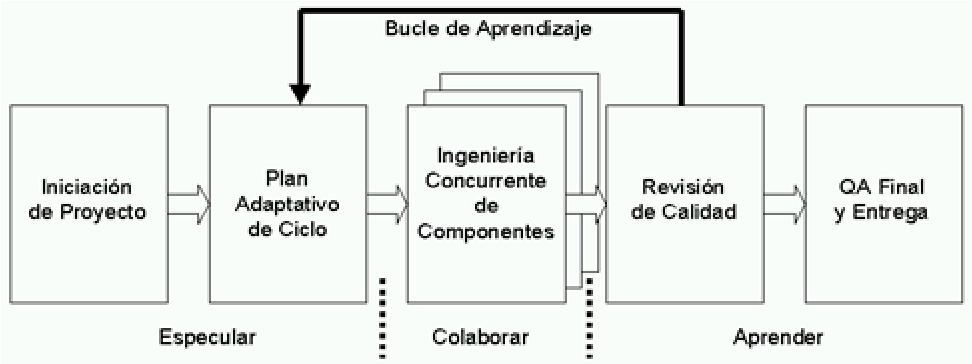



***Figura 28: Fases del ciclo de vida ASD***

**F3. Fase de aprendizaje:** acentúa la necesidad de admitir y corregir errores, y el hecho que los requisitos pueden cambiar durante el desarrollo. La base para otros ciclos (ciclo de aprendizaje) se gana de las revisiones de calidad repetidas, que se centran en demostrar la funcionalidad del software desarrollado durante el ciclo. Un factor importante en la ejecución de las revisiones es la presencia del cliente, como grupo de expertos (llamado grupo enfocado al cliente). Sin embargo, puesto que las revisiones de calidad son algo escasas (ocurren solamente en el extremo de cada ciclo), la presencia del cliente en ASD es sostenida por sesiones comunes sobre el uso del producto. Estas sesiones son esencialmente talleres, en donde los desarrolladores y representantes se reúnen para discutir características de producto, y enriquecer la comunicación. La etapa final de esta fase y del proyecto, es la etapa de Aseguramiento de Calidad (QA) y de lanzamiento. ASD no considera cómo debe ser realizada esta parte, solo la orienta a capturar las lecciones aprendidas.

El proceso ASD se origina en gran parte de la cultura de organizacional, de gerencia y especialmente, de la importancia de equipos de colaboración y trabajo en equipo. El enfoque, sin embargo, no describe las estructuras del equipo detalladamente. Asimismo, muy pocos roles o responsabilidades se enumeran. Se define un **R1. Patrocinador ejecutivo** como la persona con la responsabilidad total del producto que es desarrollado y los participantes en una sesión de desarrollo. Estos participantes pueden ser definen un **R2. facilitador de sesión**, un **R3. Redactor** para levantar actas y documentos, el **R4. Administrador del proyecto**, los **R5. Representantes del cliente** y el **R6. Desarrollador.**

ASD proponen muy pocas prácticas para el trabajo de desarrollo del software. Básicamente, nombra tres: **P1. Desarrollo iterativo, P2. Desarrollo basado en componentes, P3. Planificación y P4. Grupos de revisión enfocados al cliente.**

#### 2.5.3.9 Método de Desarrollo de Sistema Dinámico DSDM

DSDM es un framework de desarrollo rápido de aplicaciones (RAD) popular en Gran Bretaña [66] y se ha llegado a promover como el estándar de facto para desarrollo de soluciones de negocios sujetas a márgenes de tiempo estrechos. Se calcula que uno de cada cinco desarrolladores en Gran Bretaña utiliza DSDM [66].

DSDM puede complementar metodologías como XP, RUP o Microsoft Solutions Framework - MSF, o combinaciones de todas ellas. Ya no se habla de sistemas sino de soluciones, y en lugar de priorizar el desarrollo se prefiere enfatizar la entrega.

#### 2.5.3.9.1 El proceso

DSDM consiste en cinco fases:

**F1. Estudio de viabilidad.**
**F2. Estudio del negocio.**
**F3. Iteración del modelo funcional.**
**F4. Iteración de diseño y versión.**
**F5. Implementación.**

Las últimas tres fases son iterativas e incrementales. De acuerdo con la iniciativa de mantener el tiempo constante, las iteraciones de DSDM son cajas de tiempo. La iteración acaba cuando el tiempo se consume. Se supone que al cabo de la iteración los resultados están garantizados. Una caja de tiempo puede durar de unos pocos días a unas pocas semanas. A diferencia de otros métodos ágiles, DSDM ha desarrollado sistemáticamente el problema de su propia implantación en una empresa. El proceso de examen de salud (Health Check) de DSDM se divide en dos partes que se interrogan, sucesivamente, sobre la capacidad de una organización para adoptar el método y sobre la forma en que éste responde a las necesidades una vez que el proyecto está encaminado. Un examen de salud puede consumir entre tres días y un mes de trabajo de consultoría. A continuación se describen las fases en detalle:

**F1. Estudio de factibilidad:** se evalúa el uso de DSDM o de otra metodología conforme al tipo de proyecto, variables organizacionales y de personal. Si se opta por DSDM, se analizan las posibilidades técnicas y los riesgos. Se preparan como productos un reporte de viabilidad y un plan sumario para el desarrollo. Si la tecnología no se conoce bien, se hace un pequeño prototipo para ver qué pasa. No se espera que el estudio completo consuma más de unas pocas semanas. Es mucho para un método ágil, pero menos de lo que demandan algunos métodos clásicos.

**F2. Estudio del negocio:** se analizan las características del negocio y la tecnología. La estrategia recomendada consiste en el desarrollo de talleres, donde se espera que los expertos del cliente consideren las facetas del sistema y acuerden sus prioridades de desarrollo. Se describen los procesos de negocio y las clases de usuario en una definición del área de negocios. Se espera así reconocer e involucrar a gente clave de la organización en una etapa temprana. La definición utiliza descripciones de alto nivel, como diagramas de entidad-relación o modelos de objetos de negocios. Otros productos son la definición de arquitectura del sistema y el plan del bosquejo del prototipado. La definición arquitectónica es un primer bosquejo y se admite que cambie en el curso del proyecto DSDM. El plan debe establecer la estrategia de prototipado de las siguientes etapas y un plan para la gestión de configuración.

**F3. Iteración del modelo funcional:** en cada iteración se planea el contenido y la estrategia, se realiza la iteración y se analizan los resultados pensando en las siguientes iteraciones. Se lleva a cabo el análisis como el código; se construyen los prototipos y con base en la experiencia se mejoran los modelos de análisis. Los prototipos no han de ser descartados por completo, sino

gradualmente mejorados hacia la calidad que debe tener el producto final. Se produce como resultado un modelo funcional, conteniendo el código del prototipo y los modelos de análisis. También se realizan pruebas constantemente. Hay otros cuatro productos emergentes: (1) funciones priorizadas, la cual es una lista de funciones entregadas al fin de cada iteración; (2) los documentos de revisión del prototipado funcional, que reúnen los comentarios de los usuarios sobre el incremento actual para ser considerados en iteraciones posteriores; (3) los requisitos funcionales, son listas que se construyen para ser tratadas en las fases siguientes; (4) el análisis de riesgo de desarrollo posterior, el cual es un documento importante en la fase de iteración del modelo, porque desde la fase siguiente en adelante los problemas que se encuentren serán más difíciles de tratar.

**F4. Iteración de diseño y construcción:** aquí es donde se construye la mayor parte del sistema. El producto es un sistema probado que cumple por lo menos el conjunto mínimo de requisitos acordados inicialmente. El diseño y la construcción son iterativos y el diseño y los prototipos funcionales son revisados por los usuarios. El desarrollo posterior se atiene a sus comentarios.

**F5. Despliegue:** el sistema se transfiere del ambiente de desarrollo al de producción. Se entrena a los usuarios, que ponen las manos en el sistema. Eventualmente en la fase puede llegar a iterarse. Otros productos son el manual de usuario y el reporte de revisión del Sistema. A partir de aquí hay cuatro caminos de acción posibles: (1) Si el sistema satisface todos los requisitos, el desarrollo ha terminado. (2) Si quedan muchos requisitos por resolver, se puede correr el proceso nuevamente desde el comienzo. (3) Si se ha dejado de lado alguna prestación no crítica, el proceso se puede correr desde la iteración funcional del modelo en adelante. (4) si algunas cuestiones técnicas no pudieron resolverse por falta de tiempo se puede iterar desde la fase de diseño y construcción.

La configuración del ciclo de vida de DSDM se representa con un diagrama característico que se muestra en la figura 29.

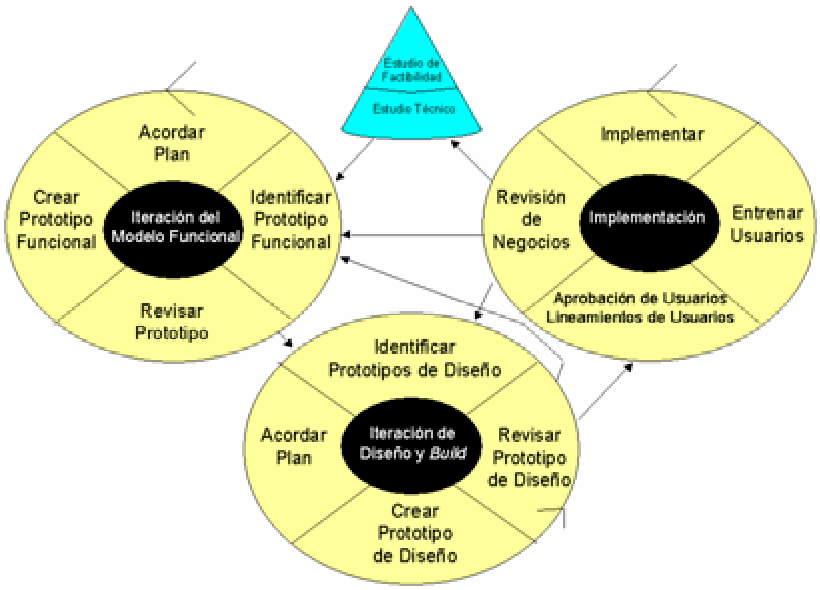



***Figura 29. Proceso de desarrollo DSDM***

DSDM define quince roles. Los más importantes se describen a continuación:

**R1. Programadores y Programadores Senior:** son los únicos roles de desarrollo. El título de senior indica el nivel de liderazgo dentro del equipo. Ambos títulos cubren todos los roles de desarrollo, incluyendo analistas, diseñadores, programadores y verificadores.

**R2. Coordinador técnico:** define la arquitectura del sistema y es responsable por la calidad técnica del proyecto, el control técnico y la configuración del sistema.

**R3. Usuario embajador:** proporciona al proyecto conocimiento de la comunidad de usuarios y brinda la información sobre el progreso del sistema hacia otros usuarios. Se define adicionalmente un rol de Usuario Asesor (Advisor) que representa otros puntos de vista importantes; puede ser alguien del personal de tecnologías de la información o un auditor funcional.

**R4. Visionario:** es un usuario participante que tiene la percepción más exacta de los objetivos de negocio del sistema y del proyecto. Asegura que los requisitos esenciales se cumplan y que el proyecto vaya en la dirección adecuada desde el punto de vista de los requisitos.

**R5. Patrocinador Ejecutivo:** es la persona de la organización que representa la autoridad y responsabilidad financiera, y es quien tiene la última palabra en las decisiones importantes.

**R6. Facilitador:** es responsable de administrar el progreso de los talleres y es el motor de la preparación y la comunicación.

**R7. Escritor:** registra los requisitos, acuerdos y decisiones alcanzadas en las reuniones, talleres y sesiones de prototipado.

En DSDM las prácticas, también llamadas para DSDM principios, se describen a continuación:

**P1. Es imperativo el compromiso activo del usuario:** pocos usuarios bien informados estarán presentes a lo largo del desarrollo del sistema para asegurar una oportuna realimentación.

**P2. Los equipos de DSDM deben tener el poder de tomar decisiones:** el proceso de tomar grandes decisiones no puede ser tolerado en ciclos de desarrollo rápido. Los usuarios involucrados en el desarrollo tienen el conocimiento para decir hacia donde debe ir el sistema.

**P3. El foco radica en la frecuente entrega de productos:** Decisiones erróneas pueden ser corregidas, si el ciclo de vida es corto los usuarios pueden proporcionar una oportuna realimentación.

**P4. El criterio esencial para la aceptación de los entregables es la adecuación a los propósitos de negocios:** la excelencia técnica es menos importante que las necesidades del negocio, estas áreas no se enfrentarán después.

**P5. Requiere desarrollo iterativo e incremental enfocado a conseguir una adecuada solución de negocio:** muy rara vez el sistema requiere quedarse intacto desde el inicio al final. Permitiendo a los sistemas evolucionar a través del ciclo de desarrollo, pueden encontrarse y corregirse errores en

etapas tempranas.

**P6. Todos los cambios durante el desarrollo son reversibles:** en el curso del desarrollo puede tomarse un mal camino, pero utilizando iteraciones cortas y asegurando que la condición previa puede ser revertida, el mal camino podrá seguramente ser corregido.

**P7. La línea de base de los requerimientos es de alto nivel:** Esto permite que los requerimientos de detalle se cambien según se necesite y que los esenciales se capten tempranamente.

**P8. La prueba está integrada a través de todo el ciclo de vida:** se recomienda particularmente la prueba de regresión, de acuerdo con el estilo evolutivo de desarrollo.

**P9. Es esencial una estrategia colaborativa y cooperativa entre todos los participantes**: Las responsabilidades son compartidas y la colaboración entre usuario y desarrolladores no debe tener barreras.

## 2.6 Trabajos relacionados

Han existido con anterioridad otras iniciativas que también pretendían relacionar los métodos ágiles con los modelos de certificación. Uno de los trabajos relacionados más relevantes es el de la migración de métodos ágiles a prácticas estandarizadas [69]. En este trabajo, las perspectivas de ingeniería y del mundo ágil tienen la misma importancia en el desarrollo de software moderno, para ello desarrollaron un framework que se enfoca a implementar los valores ágiles y los principios en el contexto de una organización puntual que está obligada a usar consistentemente el RUP y CMM. En este trabajo, la instanciación del enfoque sugiere que las capacidades del proceso pueden ser certificadas a nivel definido, lo cual explícitamente incluye la instanciación y configuración de los procesos.

En [70] se presenta una experiencia de certificación de una empresa en CMM nivel 2 e ISO9001 a través de XP y Scrum. En esta experiencia se unieron los aportes de cada uno de los métodos ágiles, XP con el objetivo de soportar los procesos técnicos y después de un año Scrum para soportar los aspectos organizacionales y de gestión con un éxito puntual: lograron la certificación de la empresa en estos dos modelos de calidad. A este método se le dio el nombre de Xp@Scrum. Xp@Scrum también fue utilizado en la agilización del proceso productivo de "Software Global", de Motorola Argentina, una empresa que ya estaba certificada en CMM Nivel 5, luego de notar las dificultades de adaptar un proceso tan complejo a los pequeños proyectos o en los cuales los requisitos eran inestables o estaban vagamente definidos. Con este trabajo se mostró que Xp@Scrum no es incompatible con los niveles superiores de CMMI [71].

Mark Paul, hace una exposición de XP desde la perspectiva CMM [72] en donde concluye que los métodos ágiles como XP incluyen muy buenas prácticas de ingeniería, aunque algunas prácticas deben manejarse con prudencia pues pueden ser controversiales y hasta contraproducentes. Al comparar XP con CMM, las expresa como buenas ideas que pueden ir juntas si se combinan adecuada y sinérgicamente con otras ideas y prácticas de gestión. Mientras XP provee una perspectiva de programación del sistema, CMM provee una perspectiva de mejora del proceso organizacional. La idea es que las empresas puedan aprovechar las ventajas de cada uno de ellos haciendo un ejercicio de selección e implementación de sus prácticas.

Otro trabajo en la línea es el expuesto en [73]; en él se propone un modelo de madurez para XP para que una organización lo pueda adoptar por etapas, ya que todas las prácticas sugeridas por XP

son difíciles de implementar de un momento a otro, y estas deben estar bien implementadas para obtener todos los beneficios reales de XP.
Aunque todos estos trabajos son unas valiosas evidencias prácticas de la certificación de procesos, no dejan de ser puntuales, bien sea por su aplicación en una organizacional particular o con respecto a un método definido. El primer paso de nuestro trabajo, antes de implementar programas de mejora de procesos a través de la utilización de métodos ágiles, fue valorar hasta qué punto un grupo de los más conocidos métodos ágiles posibilitarían la certificación en CMMI de un proceso basado en sus prácticas. Además no existen muchos antecedentes de comparación de métodos ágiles con CMMI, sino con CMM en sus versiones anteriores.

## 3. Marco Conceptual de Agile SPsL

El marco conceptual de Agile SPsL es el acercamiento al dominio de interés. Para este caso el dominio de interés viene dado por la integración de varias tecnologías, con los objetivos de lograr mejoras significativas a los procesos de las PyMES que puedan conducir a una certificación CMMI utilizando activos de proceso ágiles.

El esquema conceptual de CMMI de la figura 8, expresa de manera abstracta los requisitos que debe cumplir un proceso, dividido por áreas, para lograr la certificación. Igualmente se ha hecho necesario desarrollar un modelo conceptual referido a las metodologías ágiles, el cual ha sido obtenido de las metodologías expuestas. La figura 30 muestra el modelo conceptual estereotipado con elementos del Perfil de SPEM, con el fin de estereotiparlo con el mismo descriptor de proceso se utilizado en Agile SPI.

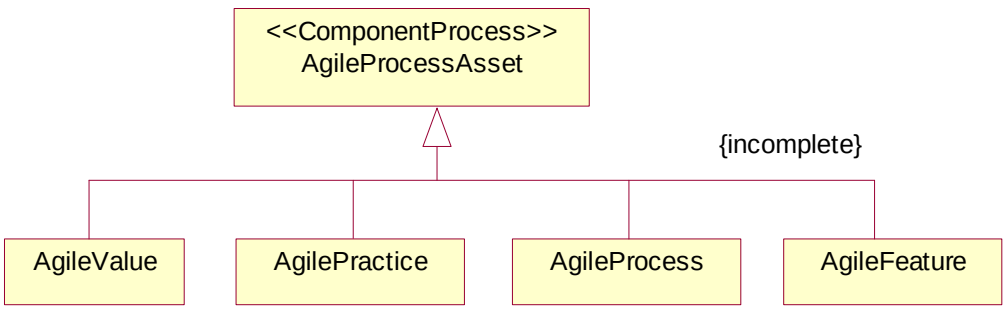



***Figura 30. Modelo conceptual básico de activos de proceso ágiles***

Es importante como primer paso hacer dos acercamientos, uno general y uno específico, de cómo los activos de proceso ágiles pueden lograr cumplir de manera individual o conjunta con algún requisito CMMI. La figura 31, muestra la relación entre estos dos modelos, con el fin de que sirva de mapa general, para hacer estos acercamientos. El acercamiento general se realizará a partir de la descripción general de las áreas de proceso de CMMI, sus objetivos generales y prácticas genéricas, con lo que se espera tener una idea general de cómo los activos ágiles pueden posibilitar la certificación CMMI de una empresa, hasta qué punto y nivel de capacidad y/o madurez. Sin embargo es importante considerar un punto específico en profundidad, para delimitar con mayor detalle cual es el cumplimiento en un área específica y delimitar el trabajo que debería realizarse para cada área con el fin de delimitar un poco el espacio de prácticas necesarias en el mundo ágil par que las PyMES logren sus objetivos de certificación. Este punto específico en el presente

trabajo es el área definida en CMMI como Administración de requisitos.

A continuación se presenta el acercamiento general entre las prácticas ágiles y cada una de las áreas del proceso. El acercamiento se hará partiendo de los objetivos generales con una medición de la capacidad por área que incluye el Nivel de Capacidad a lograr o lograda (La capacidad más cercana a alcanzar) midiendo en un porcentaje el cumplimiento entre el 0 – 100%, igualmente se listarán los aspectos necesarios para lograr alcanzar los niveles de capacidad 2 y 3 en las áreas de proceso correspondientes al nivel de madurez 2. Los niveles capacidad 4 y 5 no son necesarios pues no son requeridos para alcanzar una certificación CMMI en ninguna de las representaciones. En la figura 31 se muestran los Perfiles objetivos, que permitirán agrupar las áreas de trabajo, básicamente son los niveles de madurez del 2 al 5.

El nivel de madurez objetivo (target staging) es una secuencia de perfiles objetivo que describen el camino de mejora del proceso a ser seguido por la organización. Cuando se construyen perfiles objetivos, la organización debe poner atención a las dependencias entre las prácticas genéricas y las áreas de proceso. Cuando una práctica genérica es dependiente de un área de procesos, se debe sacar la práctica genérica o proveer un producto de prerrequisito, la práctica genérica será inefectiva cuando el área de proceso no es implementada. Cuando se escoja en perfil objetivo teniendo en cuenta esas dependencias, el perfil es admisible.

Algunas veces se requiere convertir un perfil logrado por una organización a un nivel de madurez. Esta conversión es hecha a través del nivel equivalente. El nivel equivalente es un estado objetivo equivalente que es equivalente al nivel de madures de la representación escalonada. La figura 31 muestra los perfiles objetivo que deben ser logrados cuando se utiliza la representación continua para ser equivalente a un nivel de madurez para conocer el nivel de madurez de la representación escalonada

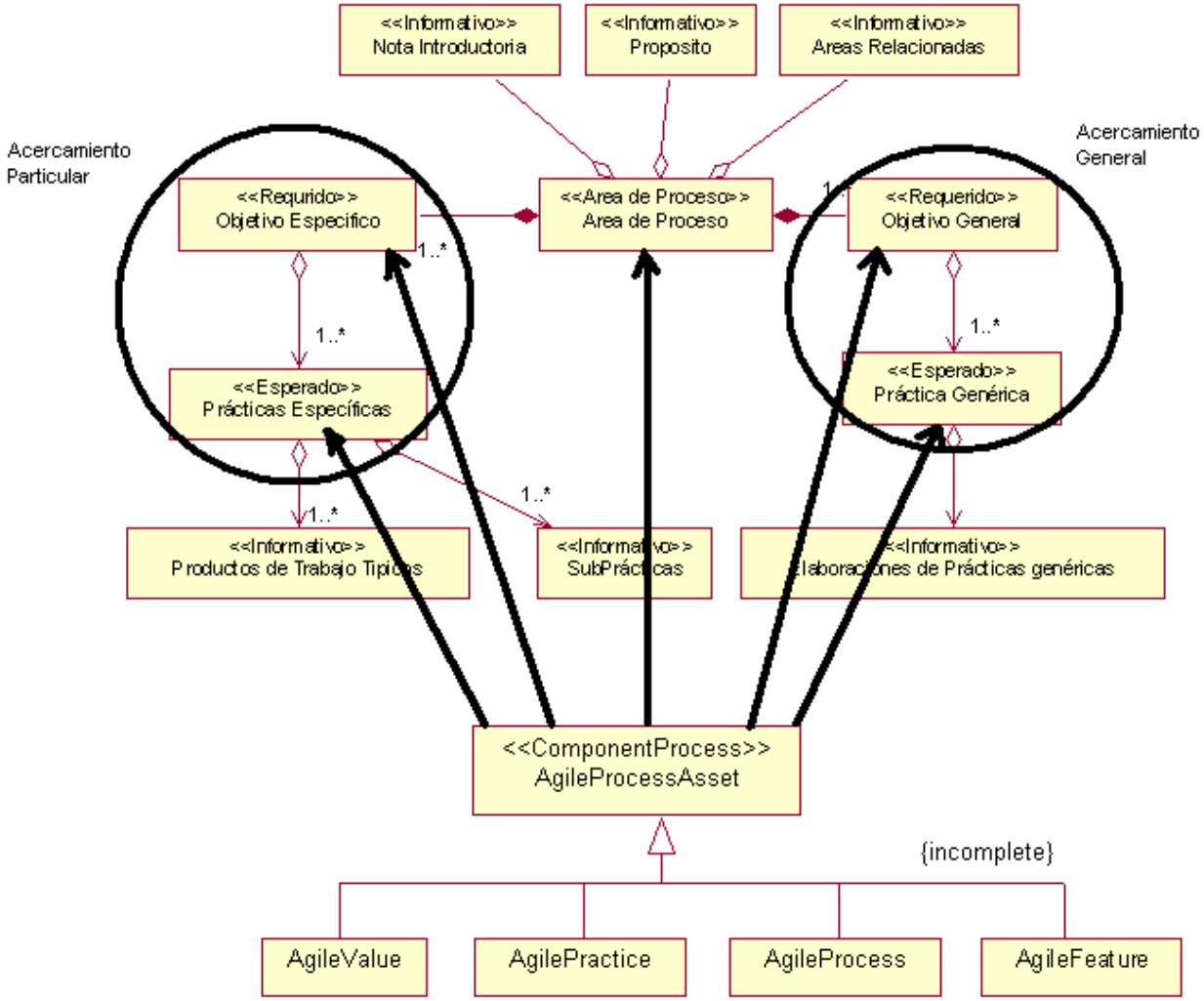



Figura 30. Relación de implementación de los requisitos de CMMI con respecto

Las columnas de la figura 31 significan lo siguiente:

- Name: el nombre del área clave
- Abbr: el acrónimo correspondiente al nombre
- ML: es el nivel de madurez asignado al area de proceso en la representación escalonada.
- CL1,CL2,CL3, CL4, CL5 son los encabezados de las columnas que representan los niveles de capacidad en la representación continua.
- Las areas ensombrecidas en las columnas de niveles de capacidada indicant los perfiles objetivo que son equivalents a los niveles de madurez en la representación escalonada.
- Para lograr el perfil objetivo 2 (y por tanto el nivel de madurez equivalente 2) se deben haber satisfecho los niveles de capacidad 1 y 2 de las áreas de proceso de la izquierda.
- Para lograr el perfil objetivo 3( y por tanto en nivel de madurez equivalente 3) se deben haber satisfecho el perfil objetivo 2 y los niveles de capacidad 1,2 y 3 las áreas del proceso de la izquierda
- Para alcanzar el perfil 4 (y por tanto el nivel de madurez 4), se deben haber satisfecho los perfiles 2 , 3, y los niveles de capacidad 1,2 y 3 de las áreas de proceso de la izquierda.
- Para alcanzar el perfil 5 (y por tanto el nivel de madurez 5), se deben haber satisfecho los perfiles 2 , 3 y 4, y los niveles de capacidad 1,2 y 3 de las áreas de proceso de la izquierda.

| Name | Abbr | ML | CL1 | CL2 | CL3 | CL4 | CL5 |
|---|---|---|---|---|---|---|---|
| **Requirements Management** | **REQM** | **2** | | | | | |
| **Measurement and Analysis** | **MA** | **2** | | | | | |
| **Project Monitoring and Control** | **PMC** | **2** | | | | | |
| **Project Planning** | **PP** | **2** | | | | | |
| **Process and Product Quality Assurance** | **PPQA** | **2** | | | | | |
| **Supplier Agreement Management** | **SAM** | **2** | | | | | |
| **Configuration Management** | **CM** | **2** | | | | | |
| **Decision Analysis and Resolution** | **DAR** | **3** | | | | | |
| **Product Integration** | **PI** | **3** | | | | | |
| **Requirements Development** | **RD** | **3** | | | | | |
| **Technical Solution** | **TS** | **3** | | | | | |
| **Validation** | **VAL** | **3** | | | | | |
| **Verification** | **VER** | **3** | | | | | |
| **Organizational Process Definition** | **OPD** | **3** | | | | | |
| **Organizational Process Focus** | **OPF** | **3** | | | | | |
| **Integrated Project Management (IPPD)** | **IPM** | **3** | | | | | |
| **Risk Management** | **RSKM** | **3** | | | | | |
| **Integrated Supplier Management** | **ISM** | **3** | | | | | |
| **Organizational Training** | **OT** | **3** | | | | | |
| **Integrated Teaming** | **IT** | **3** | | | | | |
| **Organizational Environment for Integration** | **OEI** | **3** | | | | | |
| **Organizational Process Performance** | **OPP** | **4** | | | | | |
| **Quantitative Project Management** | **QPM** | **4** | | | | | |
| **Organizational Innovation and Deployment** | **OID** | **5** | | | | | |
| **Causal Analysis and Resolution** | **CAR** | **5** | | | | | |



***Figura 31. Perfiles Objetivo de CMMI***

### 3.1 El nivel de Madurez 2, Perfil Objetivo 2 con métodos ágiles – Una visión general

Las siguientes tablas resumen las capacidades de las áreas de proceso asociadas al nivel 2 de madurez, muestra el aporte que cada método ágil hace y deduce a priori un nivel de capacidad a alcanzar y el porcentaje de este cumplimiento.

| **Área** | Área de Administración de Requisitos ML-2 | |
|---|---|---|
| **Propósito** | Administrar los requisitos de los productos y componentes de productos del proyecto e identificar consistencias entre los requisitos y los planes del proyecto y los productos de trabajo. | |
| **Valores, principios y prácticas Ágiles que favorecen la implementación del área** | **Método** | **Conclusiones de cumplimiento por Método** |
| V1. Comunicación continua<br>V2. Realimentación<br>Pr2. Participación del cliente<br>Art. Historias de Usuario<br>R2. Cliente<br>R4. Rastreador<br>R5. Entrenador<br>P5. Pruebas<br>P9. Integración continua | XP | Provee los elementos base para administrar los requisitos, pero no provee elementos para la gestión en sí mismo. El cliente en el proyecto es un actor clave para la administración. Considera un entrenador del proceso. |
| V3. Encuentros diarios<br>Art. Backlog<br>R2. Propietario del producto<br>P1. Pedido del producto<br>P5. Pedido del sprint<br>P6. Reunión diaria scrum | Scrum | Provee dos elementos esenciales para lograr la administración de requistos el Backlog, que permite hacerle seguimiento al pedido y un rol responsable de su seguimiento el propietario del producto. La reunión diaria promueve la mejora contínua. |
| Pr3. Los pasos de Evo entregan los requerimientos especificados de manera evolutiva. | Evo | Considera la evolución de los requisitos. |
| Art4. Secuencia de entrega<br>Art10. Archivo de requisitos<br>R2. Usuario Experto<br>E5. Radiadores de información<br>P6. Miniatura de procesos<br>P10. Revisión<br>P15. Puntos de vista del usuario | Crystal | Provee dos artefactos muy importantes para la gestión el Archivo de requisistos y los radiadores de información. Los puntos de vista de usuario son pertinentes. Provee entrenamiento del proceso en general. |
| F2. Construcción de la lista de rasgos<br>R6 Experto de Dominio<br>R7 Administrador de entrega<br>P8. Reporte de progreso | FDD | Provee una lista de rasgos y un reporte del progreso del proyecto relacionado con los requisitos. Hay dos roles que pueden jugar un papel importante en la administración de los requisitos, el experto del dominio y el administrador de entrega. |
| Pr5. Tolerante al cambio | ASD | Cuenta con un representante del cliente, quién puede |

| | | |
|---|---|---|
| R5. Representantes del cliente | | cumplir el rol de administrador de requistos y promueve la tolerancia al cambio. |
| F2. Estudio del negocio.<br>R3. Usuario embajador<br>R4. Visionario<br>P1. Es imperativo el compromiso activo del usuario<br>P6. Todos los cambios durante el desarrollo son reversibles<br>P7. La línea de base de los requerimientos es de alto nivel<br>P4. El criterio esencial para la aceptación de los entregables es la adecuación a los propósitos de negocios | DSDM | Lo favorece el usuario como parte del equipo, un compromiso alto de este. Promueve la reversibilidad en los cambios y para la administración recomienda tener un línea base de los requisitos de alto nivel para facilitar su evolución. |

| **Conclusión** |
|---|
| El Área de la administración de requisitos es viable, aunque ninguno de los métodos define un proceso de administración de requisitos, es posible armarlo a partir de varios de los elementos presentes en los diferentes métodos. Con base en un artefacto base (Lista de Requisitos, Pedido, Historia de Usuario, Rasgo) es posible que la PyME pueda definir un modelo de gestión de requisitos.<br><br>El nivel de capacidad alcanzable es CL-2 al 75%<br>CL-3 al 50%<br><br>Delta de cumplimiento:<br><br>GP2.8. Monitoreo y control del proyecto. No hay elementos explícitos de monitoreo y control de la gestión de requisitos.<br>GP3.1. Establecer un proceso definido. Sólo es posible definirlo a nivel de la organización que lo aplique.<br>GP 3.2. Recolectar información de mejoramiento. Aunque muchas promueven estrategias de comunicación para mejorar el trabajo del proyecto, ninguna promueve elementos para mejorar la administración de requisitos como tal. |

***Tabla 11. Valoración a priori de los métodos ágiles en concierto del cumplimiento del área de administración de requisitos***

| **Área** | Área de Medida y Análisis ML-2 | |
|---|---|---|
| **Propósito** | Desarrollar y sostener una capacidad de medición que se utiliza para soportar las necesidades de información de la gestión. | |
| **Valores, principios y prácticas Ágiles que favorecen la implementación del área** | **Método** | **Conclusiones de cumplimiento por Método** |
| La medida principal del progreso es el código | Todos | Hay una política clara de medición, aunque no deshabilita otro tipo de medición de avance, de gestión y de calidad. |

| | | |
|---|---|---|
| funcionando. | | |
| Programación Extrema<br><br>Pr4. Buenas pruebas<br>R4. Rastreador<br>P1. El juego de la planificación<br>P5. Pruebas<br>Art. Historias de Usuario | XP | En XP se rastrea los estimativos hechos por el equipo, por ejemplo esfuerzo y tiempo, esto implica la medición de estos elementos de gestión. Se proporciona realimentación para mejorar estimativos futuros. También rastrea el progreso de cada iteración y evalúa si el objetivo es alcanzable con los recursos dados y en el tiempo límite o si son necesarios cambios en el proceso. Este progreso implica medir la velocidad del proyecto. La planificación en XP permite contestar dos preguntas clave en el desarrollo de software: predecir lo que se hará ahora, y determinar que se hará después. La historia de usuario es utilizada como unidad de medida tanto para la gestión, como para la calidad, dado el énfasis de pruebas de software que tiene y que podría ayudar a determinar la eficacia de las otras prácticas y arrojar información para la toma de decisiones tanto a nivel de proyecto como de proceso. |
| V3. Encuentros diarios<br>Art. Backlog<br><br>R5. Director<br>P2. Estimación de esfuerzo<br>P3. Sprint<br>P4. Reunión de planeación<br>P6. Reunión diaria Scrum<br>P7. Revisión a reunión de sprint. | Scrum | Scrum tiene sus fortalezas en la gestión, así que su aporte a esta área es de los más significativos. Un Sprint es el procedimiento de adaptación al cambio de las variables ambientales (requisitos, tiempo, recursos, conocimiento, tecnología, etc.), este implica hacer mediciones de gestión que puede salir de estimativos y del seguimiento de cada sprint. El Director es el rol que se encarga de tomar decisiones a partir del estado del proyecto, y de la información recolectada con anterioridad. |
| Pr16. Viajar ligero de equipaje. | AM | Permite reducir de manera significativa el número de artefactos, lo cual disminuye el esfuerzo de medición. Las mediciones se deberán extender a los artefactos que se definan (tiempo, esfuerzo, calidad). |
| Pr2. El siguiente paso de entrega de Evo será el que proporcione el mayor valor para el participante en ese momento.<br>Pr8. Evo tiene que ver con aprendizaje a partir de la dura experiencia<br>Pr10. Evo permite poner a prueba nuevos procesos de trabajo y deshacerse tempranamente de los que funcionan mal. | Evo | En Evo se espera que cada iteración constituya una re-evaluación de las soluciones en procura de la más alta relación de valor contra costo, teniendo en cuenta tanto la retroalimentación como un amplio conjunto de estimaciones métricas. |

| | | |
|---|---|---|
| V3. Mejora reflexiva<br>Art7. Estatus del proyecto.<br>Art8. Lista de actores-objetivos.<br>R5. Experto en Negocios.<br>R6. Coordinador<br>E1. Exploración de 360°.<br>P1. Entrevistas de proyectos<br>P2. Talleres de reflexión<br>P11. Monitoreo | Crystal | El énfasis al monitoreo puede ayudar a la recolección de datos, la cual tiene un alcance hasta el mismo negocio (variables que le generen valor). Esta información recolectada puede ser analizada en las reuniones y talleres de reflexión a diferentes niveles, técnico y de gestión, para la toma de decisiones. |
| R1. Administrador del proyecto<br>R2. Arquitecto jefe<br>R3. Manager de desarrollo<br>R4 Programador jefe<br>R5 Propietario de clase<br>R7 Administrador de entrega<br>R10. Herramientista<br>P8. Reporte de progreso | FDD | Hace énfasis en el monitoreo preciso del progreso del proyecto. Es un método de desarrollo de ciclos cortos que se concentra en la fase de diseño y construcción por lo que es aquí donde ofrece fortalezas. Define roles muy específicos en la parte técnica y de gestión, llamados en general administradores que pueden facilitar la recolección y análisis de datos a diferentes niveles. El reporte del progreso se puede alimentar de mediciones de cada uno de los roles y esta información servir de entrada para el análisis y la toma de decisiones. Considera el rol del herramientista que puede extender su trabajo a generar la infraestructura necesaria para soportar los procesos de medición y de pruebas. |
| Pr2. Desarrollo basado en componentes<br>Pr4. Caja de tiempo (Time-boxed)<br>R4. Administrador del proyecto | ASD | El desarrollo de software basado en componentes, y el enfoque de reuso detrás de este puede ayudar a definir unidades de medida más grandes y más manejable a nivel de producto (mejor que líneas de código por ejemplo). La caja de tiempo es un elemento clave, pues manteniendo el tiempo constante es posible dedicar más esfuerzo a medir el cumplimiento de los objetivos y con esto derivar la información necesaria para el análisis y la toma de decisiones. |
| F3. Iteración del modelo funcional.<br>P2. Los equipos de DSDM deben tener el poder de tomar decisiones. | DSDM | En cada iteración se planea el contenido y la estrategia, se realiza la iteración y se analizan los resultados pensando en las siguientes iteraciones. Se lleva a cabo el análisis como el código; se construyen los prototipos y con base en la experiencia se mejoran los modelos de análisis. Esto requiere una estrategia de Medición. |

| **Conclusión** |
|---|
| El Área de medición y análisis no es definida por ninguno de los procesos ágiles, sin embargo es viable implementarla con la combinación de los elementos de los métodos ágiles. Las variables, aunque pocas pueden prestarse para mediciones básicas que soporten la toma de decisiones. Dos estrategias interesantes son medir con caja de tiempo o medir la velocidad del proyecto y a partir de estos soportar las mediciones que pueden ser repartidas entre los diferentes roles asignados en el proyecto.<br><br>El nivel de capacidad alcanzable es CL-2 al 75 % ( CL-3 al 50%)<br><br>Delta de cumplimiento: |

| |
|---|
| GP2.8. Monitoreo y control del proyecto. No hay elementos explícitos de monitoreo y control de la medida y análisis.<br>GP3.1. Establecer un proceso definido. Sólo es posible definirlo a nivel de la organización que lo aplique.<br>GP3.2. Recolección de información de mejoramiento. No se recolecta información explícitamente para mejorar la recolección y análisis de información. |

***Tabla 12. Valoración a priori de los métodos ágiles en concierto del cumplimiento del área de Medición y Análisis***

| **Área** | Planificación de Proyectos ML-2 | |
|---|---|---|
| **Propósito** | Establecer y mantener planes que definan las actividades del proyecto. | |
| **Valores, principios y prácticas Ágiles que favorecen la implementación del área** | **Método** | **Conclusiones de cumplimiento por Método** |
| V2. Realimentación<br>F2. Fase del planeamiento<br>R7. Director<br>P1. El juego de la planificación | XP | En XP se rastrea los estimativos hechos por el equipo, por ejemplo esfuerzo y tiempo, esto implica la medición de estos elementos de gestión. Se proporciona realimentación para mejorar estimativos futuros. También rastrea el progreso de cada iteración y evalúa si el objetivo es alcanzable con los recursos dados y en el tiempo límite o si son necesarios cambios en el proceso. Este progreso implica medir la velocidad del proyecto. La planificación en XP permite contestar dos preguntas clave en el desarrollo de software: predecir lo que se hará ahora, y determinar que se hará después. El cliente es un actor activo en la planeación. El énfasis se pone en la dirección del proyecto, más que en hacer una predicción exacta de que será necesario y cuanto tardará en hacerlo. Hay una comunicación frecuente del cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y el tiempo de las entregas. XP promueve la planificación de entregas (release planning) donde se planifica las distintas iteraciones. |
| V3. Encuentros diarios<br>Art. Backlog<br>R5. Director | Scrum | Scrum tiene sus fortalezas en la gestión, así que su aporte a esta área es de los más significativos. Un Sprint es el procedimiento de adaptación al cambio de las variables |

| | | |
|---|---|---|
| P2. Estimación de esfuerzo<br>P3. Sprint<br>P4. Reunión de planeación<br>P6. Reunión diaria Scrum<br>P7. Revisión a reunión de sprint.<br>F1.1. Planeación<br>R1. Scrum Master | | ambientales (requisitos, tiempo, recursos, conocimiento, tecnología, etc.), este implica hacer mediciones de gestión que puede salir de estimativos y del seguimiento de cada sprint. El Director es el rol que se encarga de tomar decisiones a partir del estado del proyecto, y de la información recolectada con anterioridad. |
| Plan Evolutivo | Evo | Inicialmente sirve como plan una idea general de la secuencia a desarrollar y evolucionar hacia las metas. Los detalles necesarios evolucionan junto con el resto del plan a medida que se desarrolla el producto/servicio. |
| Art4. Secuencia de entrega<br>Art5. Cronograma de visualización y entrega<br>Art7. Estatus del proyecto<br>Art8. Lista de actores-objetivos.<br>R5. Experto en Negocios.<br>R6. Coordinador<br>E1. Exploración de 360°.<br>E5. Radiadores de información<br>P1. Entrevistas de proyectos<br>P2. Talleres de reflexión<br>P11. Monitoreo | Crystal | Provee los roles y prácticas necesarios para soportar el área de planificación de proyectos. |
| R1. Administrador del proyecto<br>F3. Planeación por rasgo<br>P8. Reporte de progreso | FDD | Define una fase para la planeación para la cual se realiza a partir de los rasgos. Define un rol encargado de esta fase, el del administrador del proyecto. |
| Pr4. Caja de tiempo (Time-boxed)<br>R4. Administrador del proyecto<br>Pr6. Dirigido por el riesgo | ASD | Es un método de desarrollo de ciclos cortos que se concentra en la fase de diseño y construcción por lo que es aquí donde ofrece fortalezas. Define una fase para la planeación, un rol líder. Sigue una política clara el tiempo es cerrado, así que en la planeación el tiempo es una constante a la que hay que ajustar el esfuerzo, de igual forma los requisitos más críticos se deben planear de primeros. El plan es usado para seguir el proyecto. |
| F2. Estudio del negocio<br>F3. Iteración del modelo funcional.<br>R6. Facilitador | DSDM | La estrategia recomendada consiste en el desarrollo de talleres, donde se espera que los expertos del cliente consideren las facetas del sistema y acuerden sus prioridades de desarrollo. En cada iteración se planea el contenido y la estrategia, se realiza la iteración y se analizan los resultados pensando en las siguientes iteraciones. El facilitador es el rol adecuado para realizar el trabajo de planificación en conjunto con los demás involucrados. |

| Conclusión |
|---|
| El Área de Planificación de Proyectos es una constante en casi todos los métodos ágiles, la diferencia es que la planificación que se propone se aplica a iteraciones cortas y no son documentos obligatorios, son la base para el trabajo, pero los planes pueden cambiar, la meta es desarrollar el producto con la satisfacción del cliente, los planes son adecuados a lo largo del proyecto. El riesgo es tenido en cuenta para priorizar las actividades planeadas. En los planes se involucran los diferentes participantes.<br><br>El nivel de capacidad alcanzable es CL-3 al 100% |

***Tabla 13. Valoración a priori de los métodos ágiles en concierto del cumplimiento del área de Planificación de Proyectos***

| **Área** | Seguimiento y Control de Proyectos ML-2 | |
|---|---|---|
| **Propósito** | Proveer la información acerca del progreso del proyecto para poder tomar las acciones correctivas apropiadas cuando la ejecución del proyecto se desvía significativamente del plan. | |
| **Valores, principios y prácticas Ágiles que favorecen la implementación del área** | **Método** | **Conclusiones de cumplimiento por Método** |
| R4. Rastreador (Tracker)<br>P2. Entregas pequeñas<br>P10. Cuarenta horas por semana. | XP | En XP se rastrea el progreso de cada iteración de acuerdo a los planes establecidos y evalúa si el objetivo es alcanzable con los recursos dados y en el tiempo límite o si son necesarios cambios en el proceso. Este progreso implica medir la velocidad del proyecto. El énfasis se pone en la dirección del proyecto, más que en hacer una predicción exacta de que será necesario y cuanto tardará en hacerlo. Hay una comunicación frecuente del cliente y los programadores. La medida real de avance del proyecto es el software ejecutable. Promueve el trabajo de 40 horas a la semana para que el trabajo de los integrantes sea lo más productivo posible. |
| V3. Encuentros diarios<br>Art. Backlog<br>R5. Director<br>P3. Sprint<br>P4. Reunión de planeación<br>P6. Reunión diaria Scrum<br>P7. Revisión a reunión de sprint.<br>F1.1. Planeación<br>R1. Scrum Master | Scrum | Scrum tiene sus fortalezas en la gestión, así que su aporte a esta área es de los más significativos. Un Sprint es el procedimiento de adaptación al cambio de las variables ambientales (requisitos, tiempo, recursos, conocimiento, tecnología, etc.), este implica hacer mediciones de gestión que puede salir de estimativos y del seguimiento de cada sprint. El Director es el rol que se encarga de tomar decisiones a partir del estado del proyecto, y de la información recolectada con anterioridad. |
| Plan Evolutivo | Evo | Los detalles necesarios evolucionan junto con el resto del plan a medida que se desarrolla el producto/servicio. Las metas deben separarse de las soluciones; las metas deben ser |

| | | |
|---|---|---|
| | | sagradas, y se deben alcanzar por cualquier medio; las soluciones son sólo posibles, contingentes: son caballos de trabajo, y deben cambiarse cuando se consigue un caballo mejor. No define elementos específicos de planificación de proyectos. |
| V3. Mejora reflexiva<br>Art4. Secuencia de entrega<br>Art5. Cronograma de visualización y entrega<br>Art6. Lista de riesgos<br>Art8. Lista de actores-objetivos.<br>R5. Experto en Negocios.<br>R6. Coordinador<br>E1. Exploración de 360°.<br>P1. Entrevistas de proyectos<br>P2. Talleres de reflexión<br>P11. Monitoreo | Crystal | Provee los roles y prácticas necesarios para soportar el área de seguimiento y control de proyectos. Los riesgos son definidos y gestionados. |
| R1. Administrador del proyecto<br>F3. Planeación por rasgo<br>P8. Reporte de progreso | FDD | Promueve la práctica de reporte de los progresos y define un rol encargado de esto, el administrador del proyecto. |
| Pr4. Caja de tiempo (Time-boxed)<br>R4. Administrador del proyecto<br>Pr6. Dirigido por el riesgo | ASD | Se realza la importancia del trabajo en equipo como la manera de desarrollar sistemas altamente cambiantes. Está enfocado más en obtener resultados y su calidad que a las tareas o al proceso utilizado para producir dicho resultado. La manera como ASD direcciona este punto de vista es a través de ciclos de desarrollo adaptativos que contiene la fase colaborativa, donde varios componentes pueden estar en desarrollo concurrente. |
| **Conclusión** | | |
| El área de seguimiento y control de proyectos es llevada a cabo en los métodos ágiles, pero no es un área disciplinada, excepto en algunas de ellas como en XP, donde hay una aproximación y en Scrum donde el área de gestión es la fortaleza más grande. Es posible cumplir con el área uniendo los esfuerzos de varios métodos.<br><br>El nivel de capacidad alcanzable es CL-3 al 100% | | |

***Tabla 14. Valoración a priori de los métodos ágiles en concierto del cumplimiento del área de Seguimiento y control***

| **Área** | Aseguramiento de Calidad del Proceso y del Producto ML-2 | |
|---|---|---|
| **Propósito** | Proveer la gestión y el poder a un grupo de trabajo para revisar internamente procesos. | |
| **Valores, principios y** | **Métod** | **Conclusiones de cumplimiento por Método** |

| prácticas Ágiles que favorecen la implementación del área | o | |
|---|---|---|
| Pr4. Buenas pruebas<br>F3. Fase de producción<br>R3. Probador<br>P5. Pruebas<br>P6. Refactorización<br>P7. Programación en parejas<br>P10. Cuarenta horas por semana<br>P11. Cliente en el sitio de trabajo:<br>P12. Estándares de codificación | XP | En XP el código es el principal artefacto y por tanto el trabajo de aseguramiento de la calidad se centra en el código con las prácticas de programación por pares, la refactorización y la prueba. El aseguramiento de la calidad de las historias de usuario se logra con la participación activa del cliente. Sin embargo no existe un rol o equipo para asegurar la calidad de los productos y de los procesos de manera integrada. |
| V3. Encuentros diarios<br>P6. Reunión diaria Scrum<br>P7. Revisión a reunión de sprint. | Scrum | Se hacen reuniones periódicas para enfrentar los problemas de desarrollo, algunas de ellas incluyen al cliente. Aunque pueden verse como elementos de aseguramiento de calidad no definen un proceso que pueda servir de base para satisfacer los requisitos de ésta área del proceso. |
| P2. Aplicación de estándares de modelado<br>P4. Aplicación de los artefactos correctos<br>P6. Considerar la verificabilidad.<br>P10. Descartar los modelos transitorios<br>P17. Modelar con otros<br>P18. Poner a prueba con código | AM | Fortalece el uso de artefactos intermedios útiles que pueden ser considerados para el aseguramiento de calidad. Siguiendo su filosofía, sería incluir artefactos que permiten hacerle un seguimiento a la calidad del producto y del proceso, así como los artefactos mismos de este seguimiento. Obviamente AM no define ningún proceso de aseguramiento de calidad. |
| Pr5. Evo es ingeniería de sistemas holística.<br>Pr6. Los proyectos de Evo requieren una arquitectura abierta.<br>Pr10. Evo permite poner a prueba nuevos procesos de trabajo y deshacerse tempranamente de los que funcionan mal. | Evo | La ejecución incluye requisitos de calidad tales como robustez y tolerancia a fallas, al lado de estipulaciones cuantitativas de capacidad de carga y de ahorro de recursos. En Evo se espera que cada iteración constituya una re-evaluación de las soluciones en procura de la más alta relación de valor contra costo, teniendo en cuenta tanto la retroalimentación como un amplio conjunto de estimaciones métricas. Tiene como principio probar y mejorar los procesos(la forma de hacer su trabajo), pues su filosofía es el de proyectos evolutivos donde el producto y los procesos evolucionan en este proceso de desarrollo. Sin embargo, no define un proceso base de entrada para el aseguramiento de la calidad. |
| V1. Entrega frecuente<br>V3. Mejora reflexiva<br>V6. Fácil acceso a usuarios expertos<br>V7. Ambiente técnico con prueba automatizada, gestión de la configuración e integración frecuente. | Crystal | Provee estrategias documentadas para gestionar información del proyecto, tanto del producto, como de los problemas en su ejecución. Incluye más artefactos, valores y prácticas para reflexionar sobre el producto y la forma en que se está haciendo. Tiene una práctica enfocada a la sintonización de la metodología, como forma de asegurar la calidad del proceso. Esta metodología aporta significativamente a esta área, sin embargo igual que las demás no define un proceso de aseguramiento de calidad |

| | | |
|---|---|---|
| Art6. Lista de riesgos.<br>E1. Exploración de 360°.<br>E3. Esqueleto ambulante.<br>E5. Radiadores de información<br>P2. Talleres de reflexión<br>P7. Gráficos de quemado<br>P8. Programación lado a lado.<br>P10. Revisión.<br>P14. Sintonización metodológica. | | específico. |
| F4. Diseño y construcción por rasgo.<br>R12. Verificadores<br>P5. Inspección | FDD | El proceso iterativo incluye inspección de diseño, codificación, prueba de unidad, integración e inspección de código; con lo cual se definen actividades básicas de aseguramiento de calidad del producto. El proceso de aseguramiento va mezclado con el desarrollo, no hay un equipo separado para realizar esto. No hay elementos de aseguramiento de calidad del proceso. |
| **Conclusión** | | |
| El área aseguramiento y control de proyectos debe es un área débil en los métodos ágiles, pues no se presenta muy claramente. FDD incluye una estructura base que se sale de los límites de la prueba en el resto de metodologías, Crystal ofrece unas alternativas para sintonizar el proceso y Evo un enfoque donde el producto y el proceso se mejoran como objetivo inherente al proyecto de desarrollo. Dado que Scrum es fuerte en la gestión, las prácticas de gestión son aplicables a un posible proceso de aseguramiento de calidad.<br><br>El nivel de capacidad alcanzable es CL-2 al 75%<br>El nivel de capacidad alcanzable es CL-3 al 50%<br><br>GP2.6. Manejar las configuraciones: no se definen estrategias para gestionar las configuraciones de los artefactos de aseguramiento de calidad (incluso estos no están definidos).<br>GP2.8. Monitoreo y control del proceso: mientras no haga parte del proceso de la organización, no es posible monitorear y controlar el proceso de aseguramiento de calidad.<br>GP3.2. Recolectar información de mejoramiento. No hay elementos conducentes a la mejora de las tareas mismas de aseguramiento de calidad. | | |

***Tabla 15. Valoración a priori de los métodos ágiles en concierto del cumplimiento del área de Aseguramiento de Calidad***

| Área | Gestión de la Configuración ML-2 |
|---|---|
| **Propósito** | Establecer y mantener la integralidad de los productos de trabajo usando identificación de la configuración, control de la configuración, estado de cuenta de la configuración, y las auditorías de configuración. |

| Valores, principios y prácticas Ágiles que favorecen la implementación del área | Método | Conclusiones de cumplimiento por Método |
|---|---|---|
| P8. Propiedad colectiva del código<br>P9. Integración continua | XP | La gestión de configuración está asociada al código a través de las prácticas asociadas a este. No provee un proceso de gestión de la configuración, sino que recomienda manejar una herramienta adecuada para el manejo de versiones. |
| | Scrum | No hay ningún elemento asociado, sin embargo su fortaleza en la gestión puede ser aplicada a la organización del trabajo de gestión de la configuración. |
| P4. Aplicación de los artefactos correctos<br>P10. Descartar los modelos transitorios<br>P14. Modelo en incrementos pequeños | AM | Fortalece el uso de artefactos intermedios útiles que pueden ser considerados para la gestión de configuración. Al descartar modelos transitorios favorece las metodologías ágiles en el cumplimiento de esta área de proceso, pues se pueden reducir considerablemente los ítems de trabajo tanto para el proceso técnico, como para los asociados. |
| V7. Ambiente técnico con prueba automatizada, gestión de la configuración e integración frecuente | Crystal | A pesar que define más artefactos que el código, no da estrategias para la gestión de la configuración de estos ítems. |

**Conclusión**

El área de gestión de la configuración es débil en el mundo de los métodos ágiles pues estas máximo alcanzan elementos de gestión de la configuración del código. Es posible llevar las estrategias de gestión de configuración del código hacia otros ítems y apoyarse significativamente de herramientas que automaticen muchas de las tareas rutinarias de esta área.

El nivel de capacidad alcanzable CL-1 al 10%

Delta de cumplimiento:

SG 1.1 Establecer la línea base de la gestión de configuración
SG 1.2 Seguimiento y control del cambio
SG 1.3 Establecer integridad
SG 2.1. Establecer un política organizacional
SG 2.2. Planear el proceso
SG 2.3. Proveer recursos
SG 2.4. Asignar responsabilidades
SG 2.5. Entrenar personas
SG 2.6. Gestionar la configuración

| |
|---|
| SG 2.7. Identificar e involucrar los participantes relevantes<br>SG 2.8. Monitorear y controlar el proceso<br>SG 2.9. Evaluar objetivamente la adherencia<br>SG 2.10. Revisar el estado con la alta gerencia.<br>SG3.1. Establecer un proceso definido<br>SG3.2. Recolectar información de mejoramiento. |

***Tabla 16. Valoración a priori de los métodos ágiles en concierto del cumplimiento del área de Gestión de la Configuración***

La tabla 17 resume las valoraciones de las capacidades de las áreas de proceso asociadas al nivel 2 de madurez por parte de los métodos ágiles en concierto. Esta tabla cual muestra el nivel de capacidad más cercano a alcanzar y el porcentaje que el área ha avanzado en este nivel. Si los elementos de los métodos juntos dan todas las pautas para implementar el área en una organización se supone que se establece un proceso gestionado en la organización para soportar dicha área.

| Área | Métodos de mayor soporte | Porcentaje de cumplimiento potencial (Nivel de capacidad + Porcentaje) |
|---|---|---|
| Gestión de requisitos | XP | CL-2 al 75%<br>CL-3 al 50% |
| Planificación de Proyectos | XP, Scrum | CL-3 al 100% |
| Seguimiento y control | XP, Scrum | CL-3 al 100% |
| Gestión de la subcontratación | No aplica | |
| Aseguramiento de la calidad del producto y del proceso | | CL-2 75%<br>CL-3 50% |
| Gestión de la configuración | Ninguno | CL-1 (10%) |
| **Porcentaje de cumplimiento para el nivel de madurez 2** | | 72 % |
| **Porcentaje de cumplimiento para el nivel de madurez 3 (Relativo a las áreas de nivel de madurez 2)** | | 60 % |

***Tabla 17. Valoración a priori de los métodos ágiles en concierto del cumplimiento de las áreas de proceso de nivel de madurez 2 - Resumen***

Es importante observar que el nivel de madurez 2 es posible alcanzarlo si se superan los siguientes problemas:

1. Se definen procesos en las organizaciones asociados a cada una de las áreas, lo cual es posible, con algunas excepciones, a partir de los elementos de los métodos ágiles.
2. Se complementan los procesos de gestión de requisitos, de aseguramiento de la calidad y de gestión de la configuración con elementos de los métodos o de otros orígenes para eliminar los deltas de cumplimiento.

Una de las suposiciones hechas es que es posible satisfacer cada una de las áreas con un proceso, pero para todos los casos de éstas áreas no existen procesos definidos para planificar y aplicar fácilmente en un proyecto.

De allí la importancia de poder contar con unos activos de proceso que permitan definirlos e instanciarlos, que permitan alcanzar los requisitos del área a través de elementos de origen ágil. Para ello se hace necesario contar con un catálogo de valores, técnicas, prácticas, elementos de proceso, etc., que permitan implementar estos requisitos de una forma práctica.

Todos los procesos tendrán los siguientes elementos comunes (**commonalities**):

1. Una política clara, obtenida de valores y principios de las metodologías ágiles.
2. Un ciclo de vida que pueda ser gestionado y definido: obtenido de los ciclos de vida de los procesos ágiles y de otras posibles fuentes.
3. Unos flujos de trabajo que implementen en detalle cada parte del ciclo de vida: artefactos, actividades y roles: obtenido de los artefactos, prácticas y roles definidos en los métodos ágiles.

Cada proceso será particularmente único, puesto que depende de las metas y de la estructura de la organización (**variabilities**).

### 3.2 Un acercamiento específico: Cumplimiento del área de administración de requisitos con un método ágil – Programación Extrema

De acuerdo a la tabla 18 se obtienen los siguientes ítems:

***Conclusión***

El área de Administración de requisitos en una empresa que trabaja con XP puro, cumple con el nivel de capacidad 1, como esta área es importante para alcanzar una certificación CMMI, se debe mínimo alcanzar el nivel de capacidad 2.

**Problema**

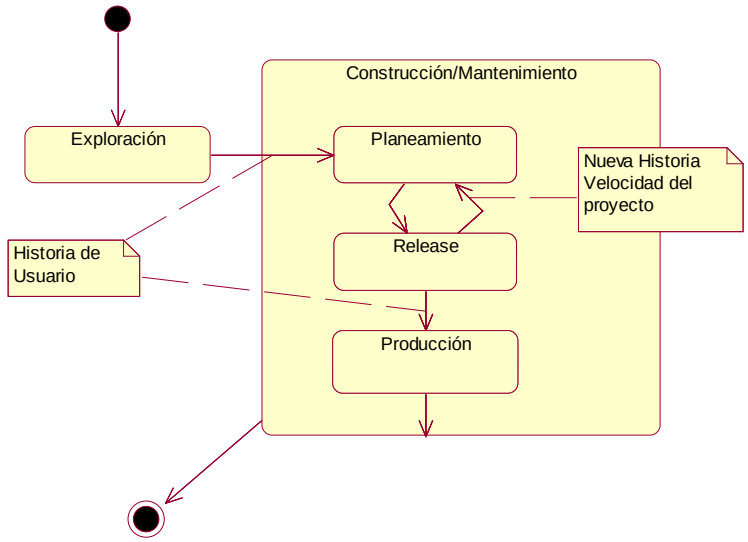
No define un proceso de administración de requisitos para ser institucionalizado como un proceso gestionado, y por tanto: no se planea, ejecuta, gestiona y mejora.

**Soluciones:**

Implementar una estrategia ágil de gestión de los requisitos a través de prácticas que cumplan con los principios y valores del manifiesto ágil. Si no es posible hacerlo recurrir a una práctica no ágil.

**Ejemplo de implementación:**

1. Solución Ágil




Artefactos: Repositorio de Historias de Usuario y requisitos como radiador de información (Crystal): Estado de los requisitos. Seguimiento del proyecto XP + Requisitos

Nuevas disciplinas: Ninguna (seguimiento y control integrado de producto y proyecto)
Nuevas actividades:
Actualizar requisitos en puntos de requisistos {Exploración, Planeamiento, Release}
Planificar la administración de requisitos

**Artefactos involucrados:** Repositorio de Historias de Usuario y requisitos como radiador de información (Crystal): Estado de los requisitos.
Seguimiento del proyecto XP + Requisitos

**Nuevas disciplinas:** Ninguna
**Nuevas actividades:**
Actualizar requisitos en puntos de requisitos {Exploración, Planeamiento, Release}
Planificar la administración de requisitos – Responsable Tracker
Seguimiento y Control del proyecto - Responsable Tracker

**Solución 2.**

Implementación no ágil – Definir una disciplina completa para la administración de requisitos que vaya paralela al proceso productivo.

### 3.3 Un acercamiento específico: Cumplimiento del área de administración de requisitos con el

**concierto de los principales métodos ágiles.**

La tabla 18, presenta el acercamiento específico entre los métodos ágiles presentes en este trabajo y el área de administración de requisitos. Este acercamiento se ha hecho utilizando los objetivos específicos del área de administración de requisitos definidos en CMMI, los cuales posibilitan el alcance de los objetivos generales. Este acercamiento específico en esta área ha permitido corroborar el acercamiento general realizado anteriormente, en gran parte porque el acercamiento general se realizó teniendo en cuenta las particularidades de cada área.

| Área de Administración de Requisitos (ML-2) para el concierto de algunos métodos ágiles | | | | | |
|---|---|---|---|---|---|
| GG | SG Objetivos específicos | SP/GP<br>Practicas específicas/Practicas genéricas | Valores y Prácticas XP | Componentes del proceso | Cumplimiento |
| GG1 **El proceso soporta y permite lograr los objetivos específicos del área transformando productos de trabajo identificables de entrada y produciendo productos de trabajo identificables de salida.** | SG1<br>Administrar los requisitos<br>**Los requisitos son administrados y se identifican las inconsistencias con los planes del proyecto y los productos de trabajo.** | SP 1.1-1 Comprender los requisitos<br>**Comprender los requisitos con los proveedores de requisitos** | XP.V1. Comunicación<br>XP.P3. Metáfora<br>XP.P11. Cliente como parte del equipo<br>Scrum.V3. Encuentros diarios<br>Scrum.P1. Pedido del producto<br>Scrum.P5. Pedido del sprint<br>ASD. R5. Representantes del cliente | XP.A1. Historia de Usuario<br>XP.A2. Spike Arquitectónico<br>XP.A3. Prueba de aceptación<br>XP.D1. Requisitos<br>XP.D4. Prueba<br>Scrum.Art. Backlog<br>Scrum.R2. Propietario del producto<br>Crystal.R2. Usuario experto<br>FDD.F2. Construcción de la lista de rasgos<br>FDD.R6 Experto de Dominio<br>DSDM.F2. Estudio del negocio.<br>DSDM.R3. Usuario embajador | **Alto** |
| | | SP 1.1-2 Acordar los requisitos<br>**Los participantes del proyecto deberán acordar los requisitos** | XP.V1. Comunicación<br>XP.P3. Metáfora<br>XP.P11. Cliente como parte del equipo<br>Scrum.V3. Encuentros diarios<br>Scrum.P1. Pedido del producto<br>Scrum.P5. Pedido del sprint | XP.A1. Historia de Usuario<br>XP.A2. Spike Arquitectónico<br>XP.A3. Prueba de aceptación<br>XP.D1. Requisitos<br>XP.D4. Prueba<br>XP.F1. Fase de Exploración<br>XP.F3. Fase de Release<br>FDD.F2. Construcción de la lista de rasgos<br>DSDM.F2. Estudio del negocio.<br>DSDM.R3. Usuario embajador<br>Crystal.R2. Usuario Experto | **Alto** |

| | | | | | |
|---|---|---|---|---|---|
| | | SP 1.1-3 Administrar el cambio de los requisitos<br>**Administrar el cambio de los requisitos a través de todo el proyecto** | XP.V2. Realimentación contínua<br>XP.P11. Cliente in situ<br>XP.V4. Coraje<br>DSDM. P7. La línea de base de los requerimientos es de alto nivel | XP.A1. Historia de Usuario<br>XP.A3. Prueba de aceptación<br>Scrum.R2. Propietario del producto | **Medio** |
| | | SP 1.1-4 Mantener la trazabilidad bidireccional de los requisitos<br>**Mantener la trazabilidad bidireccional entre los requisitos y los planes del proyecto y los productos de trabajo.** | XP.V1. Comunicación<br>XP.V2. Realimentación contínua<br>XP.P1. Juego de la planificación<br>XP.P2. Iteraciones pequeñas<br>XP.P9. Integración contínua<br>XP.P11. Cliente in situ<br>Scrum.V3. Encuentros diarios<br>Scrum.P1. Pedido del producto<br>Scrum.P5. Pedido del sprint<br>ASD. Pr5. Tolerante al cambio | XP.A1. Historias de Usuario<br>XP.A3. Prueba de aceptación<br>Scrum.R2. Propietario del producto<br>Crystal.E5. Radiadores de información | **Medio** |
| | | SP 1.1-5 Identificar inconsistencia entre el trabajo del proyecto y los requisitos<br>**Identificar inconsistencias entre los planes del proyecto, los productos de trabajo y los requisitos.** | XP.V1. Comunicación<br>XP.V2. Realimentación contínua<br>XP.P1. Juego de la planificación<br>XP.P2. Iteraciones pequeñas<br>XP.P9. Integración contínua<br>XP.P11. Cliente in situ<br>Scrum.V3. Encuentros diarios<br>Scrum.P1. Pedido del producto<br>Scrum.P5. Pedido del sprint | XP.A1. Historias de Usuario<br>XP.A3. Prueba de aceptación<br>XP.A4. Bugs<br>XP.D4. Prueba<br>Scrum.R2. Propietario del producto<br>Crystal.E5. Radiadores de información | **Alto** |
| | GP 1.1. Realizar las prácticas base<br>**Realizar las prácticas base del área del proceso para desarrollar los productos de trabajo y proveer los servicios para alcanzar los objetivos específicos del área de proceso** | | DSDM. P7. La línea de base de los requerimientos es de alto nivel<br>Cristal. P10. Revisión<br>FDD. P8. Reporte de progreso | FDD.R7 Administrador de entrega.<br>**Falta:**<br>**D. Administración de requisitos** | **Alto** |
| GG2 Institucionalizar un proce | **Compromisos a lograr** | GP 2.1. Establecer una política organizacional<br>**Establecer y mantener una política organizacional para planear y ejecutar el proceso de administración de requisitos.** | **Depende de la política de la organización**<br>XP.V1. Comunicación<br>XP.V2. Realimentación<br>XP.P1. Juego de la planificación | | **Alto** |

| | | | | |
|---|---|---|---|---|
| **Habilidades a lograr** | GP 2.2. Planear el proceso<br>**Establecer y mantener el plan de ejecución del proceso de administración de requisitos.** | **No hay proceso de administración de requisitos en ninguno de los métodos.** | Scrum.Art5. Cronograma de visualización y entrega | **Medio** |
| | GP 2.3. Proveer recursos<br>**Proveer los recursos adecuados para ejecutar el proceso de gestión de requisitos, desarrollar los productos de trabajo y proveer los servicios del proceso.** | Scrum.V3. Encuentros diarios<br>Scrum.P5. Pedido del sprint<br>Scrum.P6. Reunión diaria. | XP.Tracker<br>Scrum. Encuentros diarios<br>Scrum.Art. Backlog<br>Scrum.R2. Propietario del producto | **Bajo** |
| | GP 2.4. Asignar responsabilidad<br>**Asignar la responsabilidad y la autoridad para realizar los procesos, desarrollar los productos de trabajo y proveer los servicios del proceso de gestión de requistos.** | **No hay un rol definido como administrador de requistos** | Scrum.R2. Propietario del producto | **Bajo** |
| **Implementación** | GP 2.6. Gestionar configuraciones<br>**Darle a los productos de trabajo designados en el proceso de admisnitración de requisitos los niveles de apropiados de gestión de configuración.** | **No hay administración de configuración para los requisitos** | | **Bajo** |
| | GP 2.7. Identificar e involucrar los participantes relevantes<br>**Identificar e involucrar los participantes relevantes del proceso de gestión de requisitos de acuerdo a lo planeado.** | **Todos están involucrados**<br>XP.V1. Comunicación<br>Crystal.P15. Puntos de vista del usuario | Scrum.R2. Propietario del producto<br>XP.Tracker<br>Crystal.R2. Usuario experto | **Medio** |
| | GP 2.8. Monitorear y controlar el proceso<br>**Monitorear y controlar el proceso de gestión de requisitos a partir del plan para realizar el proceso y tomar los acciones correctivas apropiadas.** | **No hay seguimiento del proceso explícito**<br>XP.V1. Comunicación<br>XP.P3. Metáfora<br>XP.P11. Cliente como parte del equipo<br>Scrum.V3. Encuentros diarios<br>Scrum.P1. Pedido del producto<br>Scrum.P5. Pedido del sprint | FDD. R7 Administrador de entrega<br>ASD. R5. Representantes del cliente<br>Scrum.R2. Propietario del producto<br>XP.Tracker<br>Crystal.R2. Usuario experto | **Medio** |
| **V e ri** | GP 2.9. Evaluar objetivamente la adherencia | Pr3. Los pasos de Evo entregan los requerimientos especificados de manera evolutiva. | Evo.A.Plan Evolutivo | **Bajo** |

| | | | | | |
|---|---|---|---|---|---|
| | | **Evaluar objetivamente la adherencia del proceso de gestión de requisitos de acuerdo a su descripción de proceso, estándares, procedimientos y orientar las no conformidades.** | | | |
| | | GP 2.10. Revisar el estado con la más alta dirección<br>**Revisar las actividades, el estado y los resultados del proceso de gestión de requistos con el más alto nivel de gestión y resolver problemas.** | V1. Comunicación<br>Scrum.P6. Reunión diaria Scrum<br>Scrum. P7. Revisión a reunión de sprint. | Crystal. R6. Coordinador | **Medio** |
| GG3 Institucionalizar un proceso definido<br>**El proceso es institucionalizado como un proceso definido** | **Habilidad** | GP 3.1. Establecer un proceso definido<br>**Establecer y mantener una descripción del proceso de gestión de requisitos definido.** | **Depende de la organización**<br>Es posible armarlo con las prácticas anteriores. | | **Medio** |
| | **Implementación** | GP 3.2. Recolectar información de mejora<br>**Recolectar información de productos de trabajo, mediciones, resultados de medición y de mejora derivados de la planeación y ejecución del proceso de gestión de requisitos para soportar el uso y mejora de los activos de proceso y el proceso de la organización.** | Ax. Velocidad del proyecto<br>Scrum.P4. Reunión de planeación<br>Scrum.P6. Reunión diaria Scrum<br>Scrum.P7. Revisión a reunión de sprint.<br>Pr4. Caja de tiempo (Time-boxed)<br>Crystal.E5. Radiadores de información<br>Crystal.P1. Entrevistas de proyectos<br>Crystal.P2. Talleres de reflexión<br>Crystal.P11. Monitoreo | Scrum.Art. Backlog | **Medio** |

***Tabla 18. Valoración del área de administración de requisitos a través de los métodos ágiles en concierto***

## 4. Hacia una línea de procesos ágiles: la Arquitectura de SPrL

Todo proceso ágil, en este trabajo, comparte los siguientes elementos comunes (**commonalities**):

1. Una política clara, obtenida de valores y principios de las metodologías ágiles.
2. Un ciclo de vida que pueda ser gestionado y definido: obtenido de los ciclos de vida de los procesos ágiles y de otras posibles fuentes.
3. Unos flujos de trabajo que implementen en detalle cada parte del ciclo de vida: artefactos, actividades y roles: obtenido de los artefactos, prácticas y roles definidos en los métodos ágiles.

Cada proceso será particularmente único, puesto que depende de las metas y de la estructura de la organización (**variabilities**). Por ello la forma de especificar el proceso ágil deberá permitir la configuración a diferentes contextos de *activos de proceso*. Para poder especificar todo el conjunto de procesos hemos optado por especificarla como una *línea de procesos*. Una línea de procesos ágiles, es una línea de productos donde el producto es un proceso ágil. Las características comunes que comparten los procesos son:

- Agilidad: una política clara, el manifiesto ágil.
- Cumplimiento de estándares internacionales: será la información que se asociará a cada uno de los activos del proceso para facilitar el aseguramiento de algún ítem de cumplimiento por parte de un modelo.
- Arquitectura de proceso: describe y estructura un catálogo de activos de proceso, para la definición e instanciación de procesos ágiles.

La agilidad y el cumplimiento de los estándares internacionales, imponen ciertas restricciones a lo que serán los activos del proceso:

- Agilidad: cada proceso y cada componente del proceso deberá contar con una *medida de la agilidad.* La medida de la agilidad es el grado de cumplimiento de un proceso o un componente con el manifiesto ágil. Dado que se va a dar cierta formalización al proceso la agilidad podría verse afectada en algún grado. La agilidad puede ser utilizada por la organización, bien sea para ponerle un límite a la estandarización, o para medir su evolución hacia un proceso estandarizado.
- Cumplimiento de los estándares internacionales: los elementos para modelar el proceso serán definidos a través de SPEM, los estándares internacionales ISO: 9001:2000 y CMMI. Para ello cada componente del proceso deberá ser especificado en términos SPEM y del cumplimiento de algún estándar y de la parte de este que ayuda a cumplir.

Atributos controladores de la Arquitectura:

- Flexibilidad: la arquitectura debe permitir el cambio de una práctica o de una parte significativa del proceso sin restarle agilidad a la organización.
- Desempeño: el proceso resultante debe cumplir con características de alto desempeño en: tiempo y recursos. Un alto desempeño es un proceso que permite desarrollar proyectos de software con éxito en situaciones de bajo presupuesto y tiempos estrechos.

- Calidad: la calidad del proceso debe ser garantizada hasta el cumplimiento del nivel de capacidad o de madurez que la empresa desee acoger.

Las tácticas definidas para que la arquitectura alcance los atributos controladores de la arquitectura son:

- Ocultamiento de la información: se debe especificar los componentes del proceso como elementos que ofrecen un servicio y su calidad (agilidad y cumplimiento de un estándar), más no como lo implementan. Sólo a quién le concierne instanciar el componente deberá preocuparse por su implementación. De hecho su implementación podrá variar de acuerdo a las necesidades sin importar como se implementen otros componentes.
- Evolución en tiempo de ejecución el proceso, muchos de los elementos de un proceso, pueden tardar tiempo implementarse ¿Cómo un proyecto pude beneficiarse sin perder agilidad?, en la ejecución del proyecto, puede paulatinamente incorporarse un componente, haciendo que el proceso evolucione con el proyecto. Componentes del proceso pueden ser reemplazados y completados en el tiempo, a medida que los requisitos, los estimativos y los recursos se van haciendo más claros.

Agile SPrL es un catálogo de procesos ágiles, donde cada activo de este proceso es un AgileProcessAsset que permite implementar con un elemento de una estructura de modelo de calidad (por ejemplo un área de proceso de CMMI, un proceso de ISO). De acuerdo a CMMI los activos de proceso ágil: "son artefactos relacionados a la descripción, implementación y mejoramiento del proceso (por ejemplo políticas, mediciones, descripciones de procesos, y herramientas de soporte a la implementación del proceso). Incluyen: conjunto de procesos estándares (incluso la arquitectura del proceso y los elementos del proceso), descripciones de modelos de ciclo de vida apropiados a la organización, líneas guías y criterios para utilizar los procesos, el repositorio de métricas de la organización y la librería de activos de los procesos. Las líneas base del proceso y los modelos de desempeño del proceso también son activos del proceso". Para ello hemos definido el modelo de la figura 33, el cual presenta la descripción lógica de un activo de proceso ágil, extendido del modelo conceptual de la figura 30, para soportar la propuesta de contar con una línea de procesos ágiles. Los elementos de este modelo se exponen a continuación:

**Activo de Proceso**: es un artefacto relacionado con la definición, implementación y mejora del proceso de software.

**Agile Value**: hace referencia a valores, principios y políticas organizacionales que definen la orientación práctica del proceso o de parte del proceso (e.g. comunicación contínua en XP).

**Agile Process**: es un proceso ágil puro (p.e. XP) o híbrido (Obtenido de la instanciación de Agile SPrL).

**Agile Practice**: es una práctica ágil presente en cualquier método ágil (e.g. programación por pares en XP).

**AgileFeature:** es cualquier otra característica que no pueda ser expresada en términos de las anteriores (p.e. el elemento mayor Plan Evolutivo de Evo).

**Agile Process Pattern:** es un patrón de proceso ágil. Describe una solución general a un problema dado. El problema está asociado a la implementación de un requisito del modelo de calidad o de negocio, a través de la implementación de un proceso.

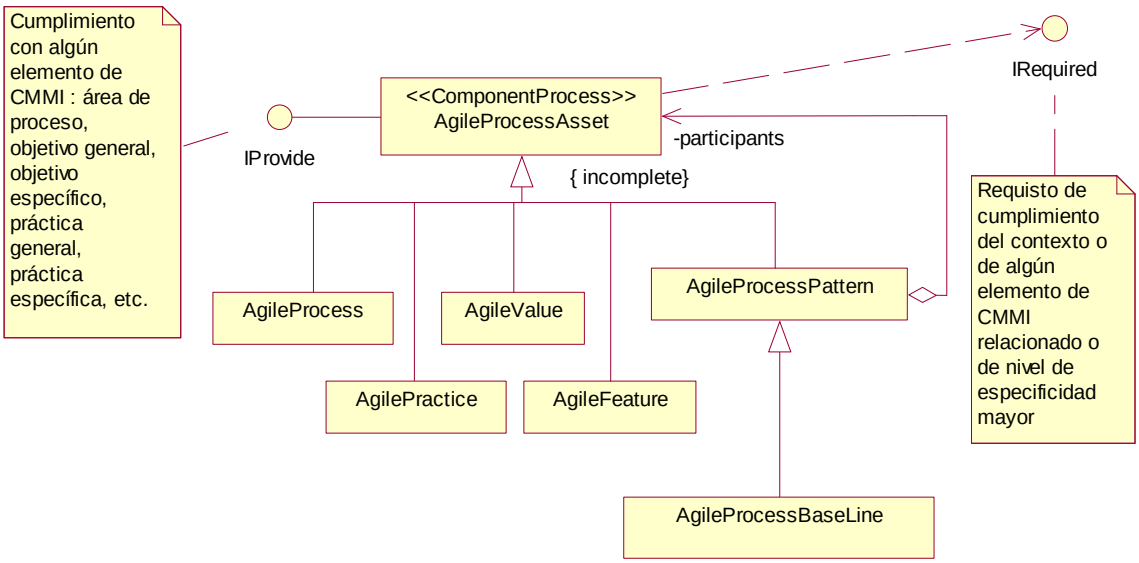



***Figura 33. Modelo conceptual de un activo de proceso ágil para***

Este trabajo inicia con la definición a nivel medio de toda esta estructura, dado que los elementos simples pueden ser obtenidos de los métodos ágiles (faltaría empaquetarlos), los elementos más compuestos como Agile Process Patterns y Agile Process Base Line deberán ser definidos. Dado que Agile Process Base Line es un patrón de proceso de grano grueso, depende en gran medida de la construcción de su padre en la jerarquía, así que el punto de partida de esta especificación son los Agile Process Patterns.

Dado que el interés es lograr la mejora y la certificación de las PyMES, es importante encontrarse con la posibilidad de combinar diferentes activos de proceso, provenientes de múltiples métodos ágiles. La pregunta ahora es como agruparlos para: organizar los procesos en la organización y para ofrecerles una presentación frente a una entidad certificadora. Dado que hemos venido haciendo un análisis con respecto a CMMI, como modelo integrador de modelos y estándares, es importante seguir con su estructura agrupándolas en áreas de proceso. En terminología ISO un área de proceso se define como un proceso y en SPEM un área es definida como una disciplina. De ahora en adelante, se agruparán los elementos bajo el nombre de disciplina, la cual será descrita a través de un tipo de patrón de proceso.

Un patrón es una regla de tres partes, la cual expresa una relación entre cierto contexto, un problema y una solución **¡Error!No se encuentra el origen de la referencia.**. Un patrón de proceso, es un patrón que describe una serie de acciones exitosamente probadas para desarrollar software [68]. Para este trabajo el patrón de proceso ágil es una regla de tres partes, la cual expresa la relación en el contexto de las metodologías ágiles, el problema de cumplir con un requisito X de CMMI y una solución modelada con SPEM que lo garantiza. A continuación se expone brevemente los elementos básicos de SPEM para facilitar la comprensión de los patrones de proceso ágiles más adelante descritos.

## 4.1 Terminología SPEM base para la especificación de los patrones

A continuación se presenta el Metamodelo de Procesos de Ingeniería del Software (Software Process Enginnering Metamodel – SPEM). Este metamodelo es usado para describir un proceso de desarrollo de software o una familia de procesos de desarrollo de software relacionados.

La figura 34 muestra las cuatro capas arquitectónicas de abstracción definidas por la OMG. Un proceso en ejecución que es un proceso de producción en un entorno real representado en el diagrama como el nivel M0. La definición del proceso está representada por el nivel M1, cualquier proceso genéricos como el RUP o XP, u otras especificaciones adaptadas a un proyecto dado están en el nivel M1. SPEM se ubica en el metamodelo que está en el nivel 2 -M2 y sirve como plantilla para el nivel 1 –M1.

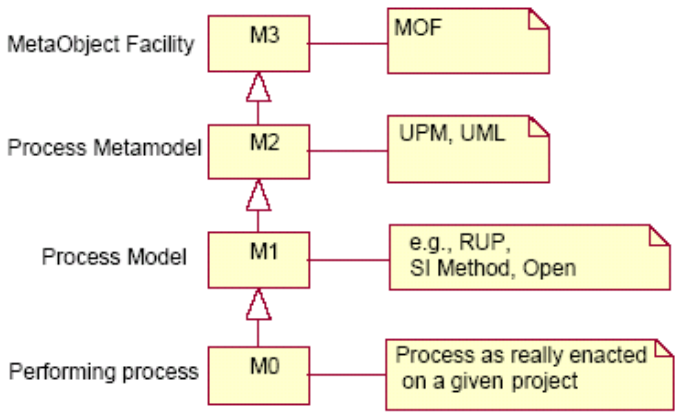



***Figura 34. Niveles de abstracción***

La especificación SPEM está estructurada como un perfil UML y como un metamodelo completo basado en MOF (Meta Object Facility). Este aprovecha las facilidades de intercambio entre las herramientas UML y las herramientas y repositorios basados en MOF. El MOF es una tecnología adoptada por la OMG para definir y representar metadatos como objetos CORBA. La especificación MOF 1.3 fue finalizada en septiembre de 1999 (documento OMG ad/99-09-05). Un metamodelo MOF define la sintaxis abstracta de la meta dato en la representación MOF del modelo. El modelo MOF describe la sintaxis abstracta para representar meta modelos MOF. Los meta modelos MOF pueden ser representados usando un subconjunto de la sintaxis UML.

XMI (XML Metadata Interchange) es la tecnología adoptada por la OMG para intercambiar modelos en forma serializada (Documento OMG ad/98-10-05). La versión 1.1 de XMI fue formalmente adoptada por la OMG en febrero de 2000 (Documento OMG ad/99-10-04). XMI se enfoca sobre el intercambio de meta datos de MOF; que es, la conformación de meta datos para un metamodelo MOF.

XMI está basado en el Lenguaje Extensible de Marcado de la W3C (eXtensible Markup Language, XML) y tiene dos componentes principales. Reglas de producción DTD XML para generar Definición de Tipos de Documento XML (DTDs) para meta dato codificado XMI. Las DTDs XMI sirven como especificaciones de sintaxis para documentos XML, y permiten generar herramientas XML usadas para componer y validar documentos XMI. Las reglas de producción de documentos sirven para codificar meta data dentro de un formato compatible XML. Las reglas de producción pueden ser aplicadas en reversa para decodificar documentos XMI y reconstruir la meta data.

XMI puede ser usado para manipular el metamodelo SPEM de las siguentes formas:

- Para crear definición de tipos de documentos SPEM.
- Para transferir modelos de procesos basados en SPEM como documentos XML, o describiendo el modelo como una instancia directa de SPEM (Uso del DTD SPEM) o describiendo este como un modelo de UML conforme al perfil UML para SPEM (Uso del DTD UML).
- Transformar el metamodelo SPEM en un documento XML basado en el DTD MOF, para intercambio entre los repositorios de conformidades MOF.

### *4.1.1 Modelo Conceptual*

En la base del Metamodelo de Procesos de Ingeniería del Software (SPEM) está la idea de que un proceso de desarrollo de software es una colaboración entre las entidades activas abstractas llamadas Roles del proceso que realizan operaciones llamadas actividades en entidades concretas, tangibles llamadas productos de trabajo **¡Error!No se encuentra el origen de la referencia.**. La Figura 35 describe un ejemplo de modelo conceptual fundamental usando la notación UML para una clase.

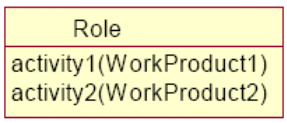



***Figura 35. Modelo Conceptual***

Múltiples roles interactúan o colaboran para intercambiar productos de trabajo y provocar la ejecución, o representación de ciertas actividades. El objetivo principal de un proceso es traer unos productos de trabajo a un estado bien definido. Para este modelo, un primer paso consistirá en reestructurar el rol, la actividad y el producto de trabajo. Esto nos conduce al modelo simple mostrado en la Figura 36.

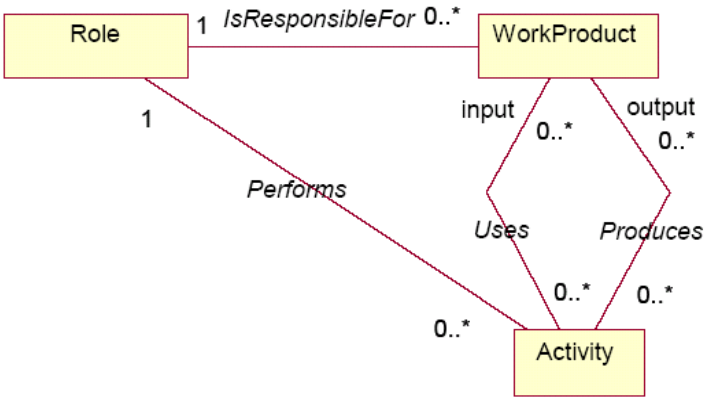



***Figura 36. Modelo Conceptual Reificado***

*4.1.2 Elementos básicos de SPEM y su notación gráfica:*

| | |
|---|---|
| Activity |  |
| Document |  |
| Guide |  |
| Process |  |
| Phase |  |
| Process Package |  |
| Rol |  |
| WorkProduct |  |
| WorkDefinition |  |
| Model |  |

***Figura 37. Estereotipos gráficos de los elementos SPEM***

**WorkProduct:** un WorkProduct o Producto de Trabajo es cualquier elemento producido, consumido, o modificado por un proceso. Esto puede ser una pieza de información, un documento, un modelo, código fuente y demás. Un WorkProduct describe una clase de producto

de trabajo producido en un proceso **¡Error!No se encuentra el origen de la referencia.**.

**WorkDefinition** (Definición de trabajo): es un tipo de operación que describe el trabajo que se ejecuta en un proceso. Este trabajo puede ser Activity, Phase, Iteration y LifeCycle.

**Activity (Actividad):** es la principal subclase de WorkDefinition. Esta describe una parte del trabajo desarrollado por un ProcessRole: las tareas, operaciones y acciones que son desempeñadas por un rol o las que el rol puede asistir. Una Actividad se puede componer de elementos atómicos llamados pasos.

**ProcessPerformer (Ejecutor de Proceso):** define la ejecución para un conjunto de definiciones de trabajo en un proceso. ProcessPerformer es una representación abstracta del representante del "proceso completo" o uno de sus componentes.

**ProcessRol(Rol de Proceso)**: define responsabilidades sobre productos de trabajo específicos y define los roles que ejecutan y asisten en actividades específicas.

**ProcessPackage** (Paquete de Procesos): al igual que en UML, puede contener procesos propios e importar elementos de definición de procesos. Las actividades y definiciones de trabajo son contenidos respectivamente, por Roles de proceso y Ejecutores de proceso; Las máquinas de estado son contenidas por Productos de trabajo y sus propios estados y transiciones. Los grafos de actividad pueden ser contenidos por Paquetes, Clasificadores o características de comportamiento; Otros Elementos de Modelo SPEM pueden ser contenidos por Paquetes.

**Guidance(Guía):** pueden ser asociados con los Elementos del Modelo, para proporcionar información más detallada a los ejecutores acerca de los elementos asociados. Los tipos de guía dependen de la familia de procesos y pueden ser por ejemplo: Guías, Técnicas, Métricas, Ejemplos, Perfiles UML, Tutoriales, Listas de comprobación, Plantillas, etc.

**ProcessComponent (Componente de Proceso)**: es una porción de descripción de proceso que es consistente internamente y puede ser reutilizado por otros ProcessComponents para ensamblar un proceso completo.

**Process (Proceso):** es un ProcessComponent que es capaz de mantenerse aislado de principio a fin. El proceso se distingue de un componente de proceso normal por que no está pensado para ser accedido por otros componentes.

**Discipline (Disciplina):** es una especialización particular de Package que divide las actividades dentro de un proceso de acuerdo a un tema común. El particionamiento de actividades de esta forma implica que la guía asociada (Guidance) y la salida de WorkProducts se categorizan similarmente en un tema. La inclusión de una actividad en una Disciplina es representada por la dependencia de Categorías, con una restricción adicional de que toda Actividad es categorizada por exactamente una Disciplina.

Un proceso se puede ver como una colaboración entre roles para alcanzar cierta meta u objetivo. Para dirigir su representación, podemos restringir la orden en la cual las actividades deben ser, o pueden ser ejecutadas. También es necesario definir la "forma" del proceso en un determinado tiempo, y la estructura del ciclo de vida en términos de fases y de iteraciones.

**Phase (Fase):** es una especialización de WorkDefinition tal que su condición previa define los criterios de entrada de la fase y su meta (a menudo llamada "milestone") define los criterios de la salida de la fase. Las fases son definidas con una restricción adicional de secuencialidad, que es,

sus representaciones son ejecutadas con una serie de fechas de tiempos de programación (milestone) que normalmente toman un mínimo (o ningún) traslapo de sus actividades en el tiempo.

**Ciclo de vida.** El ciclo de vida de un proceso es definido como una secuencia de fases que alcanzan una meta específica. Este define el comportamiento de un proceso completo que será representado en un proyecto o un programa dado.

**Iteración.** Una Iteración es un componente WorkDefinition que se caracteriza por tener el menor "milestone".

### 4.2 Patrones Ágiles de procesos

Un patrón de proceso ágil es una regla de tres partes, la cual expresa la relación en el contexto de las metodologías ágiles, el problema a resolver y una solución al problema. El catálogo de patrones que a continuación se expresa presenta las siguientes restricciones de alta prioridad:

1. Está realizado para permitir a las organizaciones en el contexto de los métodos ágiles su mejoramiento y certificación. El mejoramiento consiste en introducir disciplina de una manera suave y ágil. Para la certificación se busca presentar los métodos ágiles de una forma tal, que se visualice el cumplimiento de un estándar ante una entidad certificadora, partiendo del hecho, de que los métodos ágiles en conjunto pueden lograr el cumplimiento de las metas propuestas por un modelo de calidad. La razón de todo esto, es que este trabajo está enmarcado en Agile SPI, un Framework de mejora para la pequeña y la mediana empresa, basado en los principios del manifiesto ágil.
2. El problema ha sido expresado en términos de metas de certificación CMMI. Debido a que CMMI, es especial en el área de ingeniería del software y ofrece dos representaciones moldeables fácilmente a modelos heredados o de otros estándares.

Estos patrones combinan estrategias y demás elementos de diferentes metodologías ágiles, sin embargo se hace necesario considerar:

1. Que diferentes métodos ágiles, e.g. XP, sugieren que todas sus prácticas sean utilizadas en conjunto para alcanzar los beneficios del método.
2. Que algunos métodos, e.g. Scrum, sugieren ser utilizados en el contexto de otros métodos, pues no traen consigo cómo llevar a cabo el proceso desde la perspectiva técnica.
3. Al adicionar elementos de otros métodos ágiles a un método ágil ¿Cómo asegurar que este método mixto también sea un método ágil?

Para solucionar el item 1 y 2 hemos limitado la estructura de las soluciones a los métodos XP, Scrum, manteniendo las variantes suficientes para soportar elementos de otras metodologías. Para solucionar el punto 3 se han considerado dos caminos: uno de ellos es la valoración del proceso resultante con Agile SPI - Light Quality Evaluation Model, el cual además de medir la capacidad de un proceso, es capaz de medir la agilidad del mismo. El otro, es medir la apreciación por parte de las pequeñas y medianas empresas sobre la relación beneficio/costo de tener una solución ágil con más disciplina que la que inicialmente trae el método.

A continuación se especifica el template que se seguirá para la descripción de los patrones de procesos y luego se mostrará su utilización en la descripción de 6 patrones asociados a las seis

áreas de proceso del nivel 2. Estos patrones han sido diseñados con el fin de brindar un camino hacia la certificación CMMI, así que el desarrollo de estos seguirá hasta dónde las necesidades de certificación de la pequeña y mediana industria lo requieran. Obviamente, estos patrones deberán ser experimentados, evolucionados, hasta convertirlos en patrones que provean un camino de implementación probado de CMMI.

### *4.2.1 Template de los patrones de procesos*

A continuación, en la tabla 19 se describe la estructura a seguir por parte de un patrón de proceso ágil para la certificación CMMI basado en el *Perfil SPEM*. Se ha decidido trabajar con el perfil, pues la orientación de este trabajo es para las PyMES, a las cuales les resultará más fácil asimilar unos estereotipos UML que asimilar un metamodelo. Adicionalmente el uso del metamodelo, está más orientado a las herramientas de soporte de Agile SPI.

| **Template de Patrón de Proceso Ágil** | |
|---|---|
| Nombre del Patrón | Aquí se especifica el nombre del patrón, se recomienda darle un nombre fuerte que represente la esencia de la solución. |
| Tipo | Se especifica el tipo de patrón, los tipos de patrón pueden ser de Ciclo de Vida, Iteración, Fase o Componente de Proceso de acuerdo a la definición SPEM. |
| Problema | Especifica mediante una pregunta el problema que resuelve. |
| Contexto | Describe el contexto en el cual el patrón presenta la solución al problema. |
| Solución | Describe mediante la notación SPEM la solución general al problema, planteado por el patrón: esta solución puede incluir los participantes, productos de trabajo y la representación estática o dinámica de la disciplina. |
| Ejemplo | Describe mediante un ejemplo una solución específica de implementación de la solución general, mediante elementos SPEM más concretos. |
| Consecuencias | Al aplicar el patrón a una organización, va a implicar cambios culturales, de entrenamiento, de disciplina, de agilidad, etc. En las consecuencias se deben describir las implicaciones que deberán ser tenidas en cuenta al aplicar el patrón. |
| Uso | Especifica el uso que se le ha dado al patrón, este espacio podrá quedar en blanco mientras el patrón no haya sido probado (también conocido como pseudopatrón) |
| Patrones relacionados | Especifica patrones que guardan relación con este patrón, y donde la implementación de este patrón tendrá consecuencias sobre ellos. |

***Tabla 19. Plantilla de un patrón ágil de proceso***

### *4.2.2 Un ejemplo: el patrón Administrador Ágil de Requisitos*

| **Patrón Administrador Ágil de Requisitos** | |
|---|---|
| **Nombre del Patrón** | Administrador Ágil de Requisitos |
| **Tipo** | Componente de proceso – Disciplina |
| **Problema** | ¿Cómo identificar las inconsistencias entre los requisitos de los productos y los componentes de productos con respecto a los planes del proyecto y los productos de |

| | |
|---|---|
| | trabajo? |
| **Contexto** | La empresa cuenta con un proceso de administración de requisitos insuficiente o no cumple aparentemente con el estándar CMMI, además no tiene personal adicional para el soporte de nuevos procesos. |

**Solución**

Implementación de la Disciplina de Administración de requistos:

**Participantes**

- **Representante del cliente:** es en el mundo ágil un rol muy importante, puesto que la participación de este es crucial tanto para el proceso de ingeniería, como el de administración de requisitos. De hecho muchas de las metodologías lo ven como un miembro más del equipo de desarrollo. Este representante del cliente, puede ser el cliente o una persona que lo represente en el grupo de desarrollo y que tenga la suficiente disponibilidad de tiempo para participar en las áreas asociadas a los requisitos. Actividades:
  - **Adicionar/Modificar/Eliminar requisitos:** en los métodos ágiles el cliente es involucrado en el trabajo de captura y administración de los requisitos. La idea es que el sea capaz de expresarlos de una manera sencilla y administrar sus contenidos a través del modelo de requisitos definido por la organización. Algo bien interesante podría ser manejar un modelo de requisitos con diferentes vistas y ofrecerle a este participante una vista manejable a su capacidad, pero lo suficientemente adecuada para soportar la administración del contenido de los requisitos. De todas formas el proceso deberá incluir el entrenamiento de este participante.
- **Administrador de Requisitos:** es el encargado de gestionar las prácticas de ingeniería y administración de requisitos (planes y ejecución), de administrar el cambio de estos y de evaluar contínuamente el estado de estos con respecto a los planes y los recursos. La idea es que los demás participantes actualicen los requisitos, pero haya un responsable de su administración. Actividades:
  - **Gestionar Prácticas:** hace referencia a lograr la efectividad de la planificación, la ejecución y el monitoreo y control de las prácticas asociadas. (p.e. el juego de la planificación en XP, Pedido del sprint en Scrum).
  - **Gestionar Cambios:** los cambios realizados al modelo de requisitos, de acuerdo a las políticas de gestión (valores y principios del método ágil) deberán ser evaluados, aprobados o reprobados, y mantener un historial de estos cambios.
  - **Evaluar estado:** el estado de los requisitos debe ser evaluado periódicamente con el fin de detectar inconsistencias entre estos y el estado del proyecto y el desarrollo del producto, encontrar problemas y resolverlos lo más oportunamente.
- **Participante:** es cualquier otro participante del proyecto, que a medida que avanza en su trabajo actualiza la información de avance de acuerdo a las tareas terminadas, asociadas a estas están los requisitos o directamente a los requisitos (e.g. el Probador). Actividades:
  - **Actualizar el estado de los requisitos:** los requisitos a medida que son analizados, diseñados, implementados, probados, etc., deberán ser actualizados, esto le compete hacerlo a cada participante dentro del proceso productivo, cada

vez que termina una tarea que cambia el estado de un producto de trabajo y a través del cual un requisito puede quedar parcial o totalmente cubierto.

**Productos de Trabajo**

- **Modelo de requisitos:** un modelo liviano de requisitos es la forma en que la organización representa y administra sus requisitos (e.g. una lista, un pedido, un conjunto de historias de usuario, un conjunto de casos de uso, etc.)

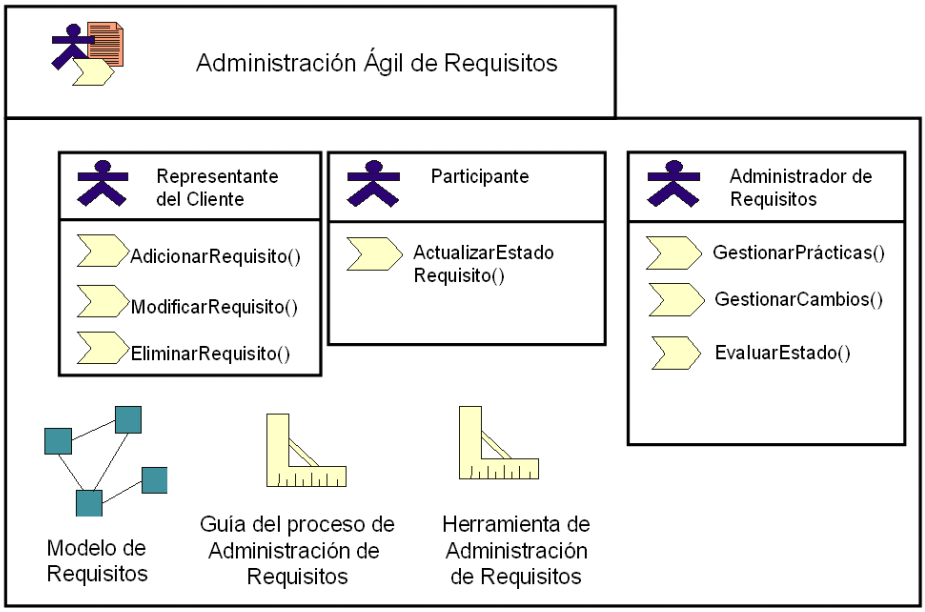
**Guías:**

- **Guía de administración de requisitos:** es una guía liviana para la planificación y ejecución del proceso de administración de requisitos. Esta guía deberá incluir:
  - Una política clara de administración de requisitos
  - Un manual de procedimiento de la disciplina de administración de requisitos que incluye: procedimiento, planificación, asignación de recursos y responsabilidades, monitoreo y control y recolección de información para la mejora.
  - Gestión de la configuración de los requisitos de acuerdo a la disciplina gestión de la configuración.
  - Un manual de entrenamiento y capacitación para cada tipo de participante.

**Herramientas**

- **Herramienta de administración de requisitos**: es una herramienta de soporte a la disciplina de administración de requisito basada en el modelo de requisitos utilizado por la organización y en el modelo de gestión de la configuración. La herramienta debe soportar: múltiples vistas, la interacción con las herramientas de gestión del proyecto y de gestión de la configuración. Igualmente debe permitir la recolección de información, para medir el desempeño de la disciplina y ofrecer pautas para su mejoramiento. Son ejemplos de herramientas ***RMTrak*** [http://www.rmtrak.com/] ***Analyst Pro*** de Goda [http://www.analysttool.com/ ], entre otras.

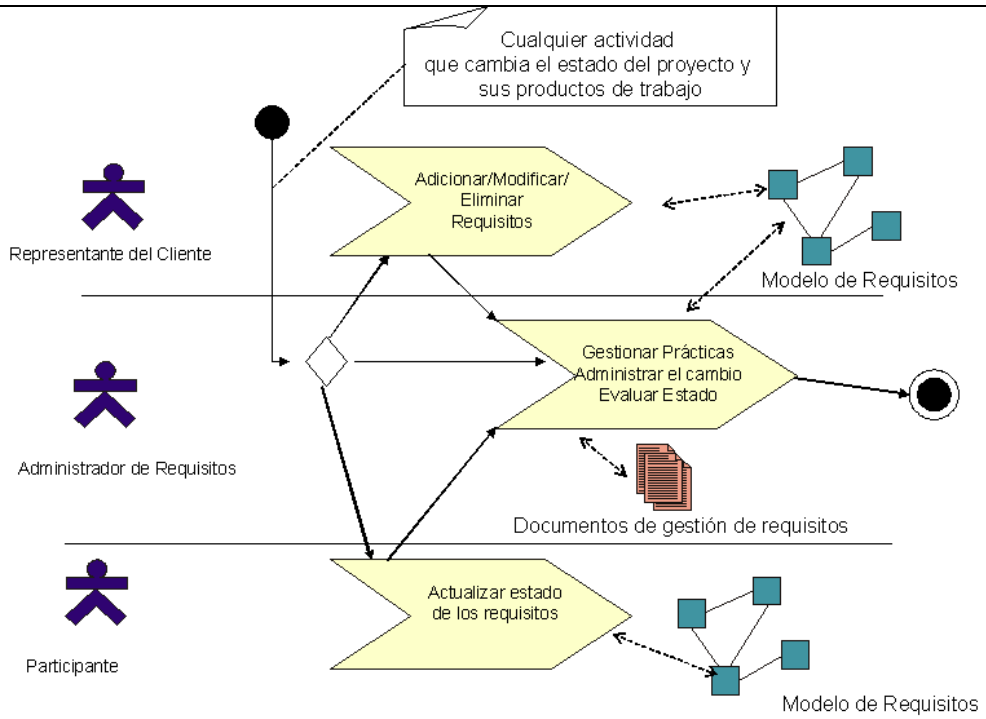
**Estructura Estática**




La disciplina de administración de requisitos es una disciplina que corta a todo el proceso ágil, en cualquier momento del proceso un participante podrá actualizar el estado de un requisito, por ejemplo "una historia de usuario fue completamente implementada, esta pasa de su estado *capturada* a *implementada*". A pesar de que diferentes participantes puedan actualizar el modelo puedan actualizar el modelo de requisitos, sólo el administrador de requisitos administra los cambios en estos.

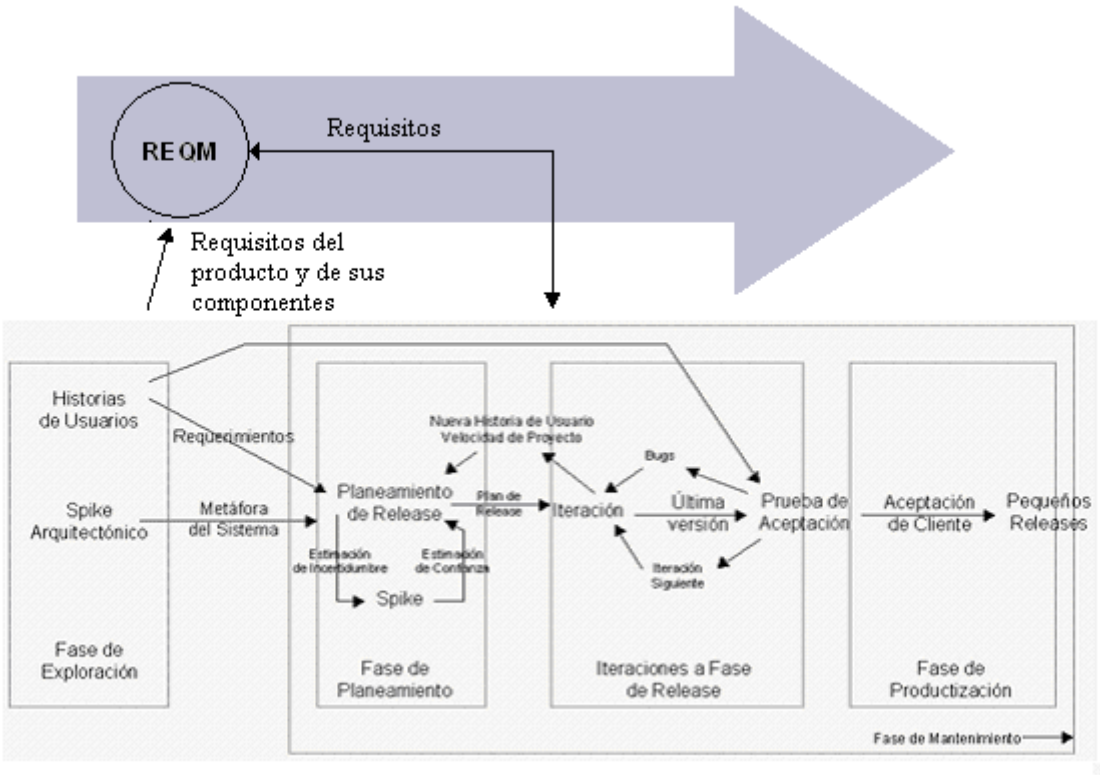
**Estructura dinámica**

El diagrama de actividad visualizado a continuación, esboza la dinámica básica de la disciplina de administración de requisitos. Básicamente, todos los participantes tienen acceso a los requisitos, bien sea para actualizar su estado o hacer modificaciones en ellos. Esto puede realizarse en cualquier momento dentro del proyecto de desarrollo. Siempre que se haga un cambio en los requisitos o en su estado, estos deben reposar en la base de requisitos del sistema (modelo de requisitos), y es el administrador de requisitos quien administra este cambio y verifica el estado de los requisitos con respecto a la evolución del proyecto de una manera planificada. El rol de administrador de requisitos, es un rol de gestión, quien ve los requisitos como la base sobre la cual se evalúa el cumplimiento del producto y del proyecto de acuerdo a las expectativas del cliente y del proyecto respectivamente. Para planificar y realizar las actividades de gestión, se deberá seguir la guías y normalmente se generan documentos del proyecto asociados a la administración de los requistos.




Ejemplo

La siguiente implementación a nivel de ejemplo se ha definido la administración de requisitos en el marco del proceso XP, con el apoyo de Scrum, el método fuerte en la parte de gestión. El proceso seguido presenta la siguiente estructura:



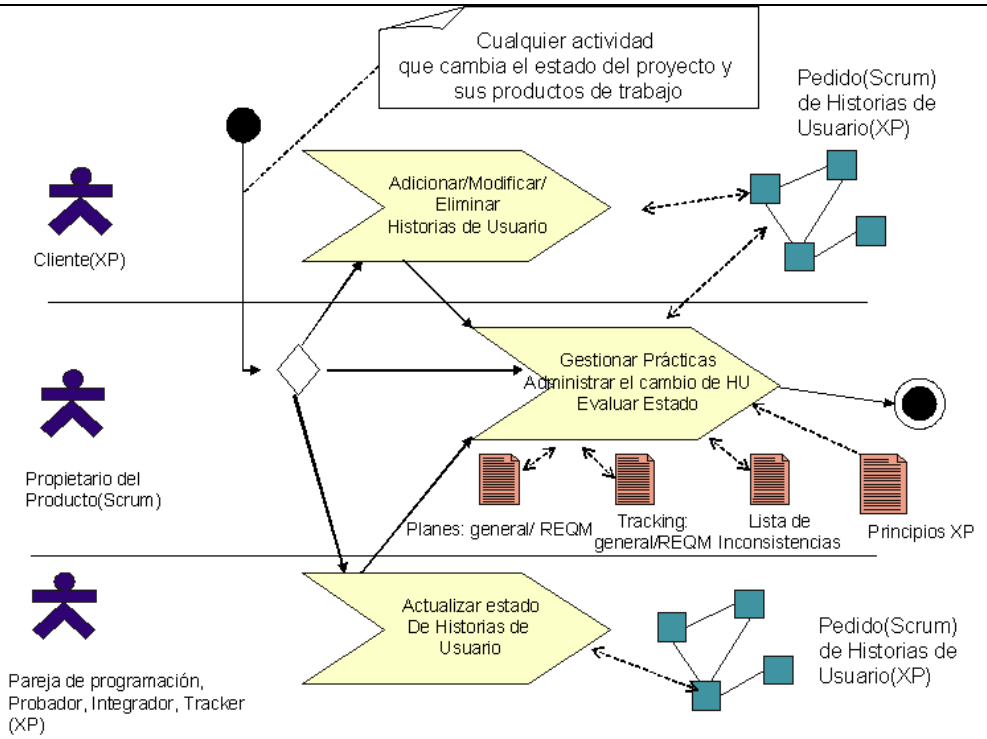

La disciplina de administración de requisitos acompaña todo el proceso de desarrollo, así que es importante esbozarlo de manera resumida. Las historias de usuarios son la forma en XP de capturar las necesidades del cliente, guiar la planificación, el diseño, la prueba, la integración y en general el desarrollo del proyecto. Un conjunto de historias de usuarios se realiza en una iteración de XP, y se pueden ver afectadas por el trabajo en otras iteraciones. La historia de usuario es la base para la planificación y el diseño de las pruebas, así que finalmente, facilita la validación y verificación del sistema. Gestionar esta información es definitiva en XP, la forma de lograrlo es involucrando al cliente en el proceso de desarrollo como parte del equipo, sin embargo es importante controlar un poco este proceso de administración, asignarle un responsable (que incluso podría ser el mismo cliente, si cuenta con las habilidades) y unos recursos. Normalmente es posible ayudarse de una herramienta software que facilite estas tareas de una manera menos monótona.

Al aplicarle el patrón se tiene las siguientes asignaciones:

**Los participantes:** el cliente (representante del cliente), el propietario del producto (Administrador de requisitos) y el conjunto de participantes {seguidor, pareja programadora, pareja integradora, probador}.

**Los productos de trabajo:** el pedido de historias de usuario(modelo de requistos), el conjunto de documentos: planes, seguimiento y control, lista de inconsistencias, principios y valores de XP, entre otros.

**Herramienta:** RMTrak




| | |
|---|---|
| Consecuencias | Aplicar este patrón a la organización puede implicar lo siguiente:<br>• Hay que crear el rol de Administrador de requisitos o en su defecto, asignarle tales responsabilidades bien sea al Cliente si cuenta con la disposición o a un rol de tipo administrativo con conocimiento en el área (director, integrador, tracker, etc.). Se debe medir la relación beneficio/costo de tener este nuevo rol.<br>• Cada vez que se dispara un trabajo hacia la administración de requisitos siempre se anexará una tarea al administrador de requisitos.<br>• El área de administración de requisitos definida, documentada e implementada de manera repetida en la organización permite que el área de administración de requisitos tenga un nivel de capacidad 3 en CMMI (CL-3), el aporte justo de parte del área REQM para la certificación CMMI 5 en cualquiera de las representaciones.<br>• Una herramienta software de administración de requisitos adecuada es altamente necesaria para garantizar este proceso, hacerlo certificable y lo más importante conservar la agilidad del proceso general. |
| Uso | Pendiente |
| Patrones relacionados | Patrón Tracker<br>Patrón Gestor de la Configuración<br>Patrón Planificador de Proyectos<br>En general es un patrón que corta el proceso de al organización en puntos definidos en |

| | |
|---|---|
| | tiempo de ejecución. Todos los participantes al hacer sus labores actualizan el estado del proyecto y normalmente esto implica cambiar el estado de los requisitos. Periódicamente el administrador de requisitos estará haciendo las labores de gestión para controlar y evaluar este cambio de acuerdo a las metas del proyecto. |

***Tabla 20. Patrón de proceso ágil. Administrador de Requisitos***

### 4.3 Patrones de línea base del proceso

Presentan la estructura base de un proceso, es un patrón especial de proceso de tipo "Proceso", y este permite describir un proceso completo a partir de otros patrones de proceso (de disciplina, de iteración, de ciclo de vida). Un proceso completo puede obtenerse aplicando un patrón de línea base inicialmente, y cada uno de los componentes del proceso podrá realizarse aplicando los patrones de proceso y estos a su vez aplicando activos de proceso más simples. La línea base del proceso es un patrón descrito a través de un patrón de tipo Ciclo de Vida, integrador de patrones de proceso de tipo Iteración, Fase y Disciplina. Al ser esta la primera aproximación a una línea de procesos ágil, el alcance de las definiciones de todos los activos queda pendiente para una primera iteración de ingeniería de dominio que se realice sobre su construcción. Sin embargo, hemos tratado de dejar trazado un camino a seguir.

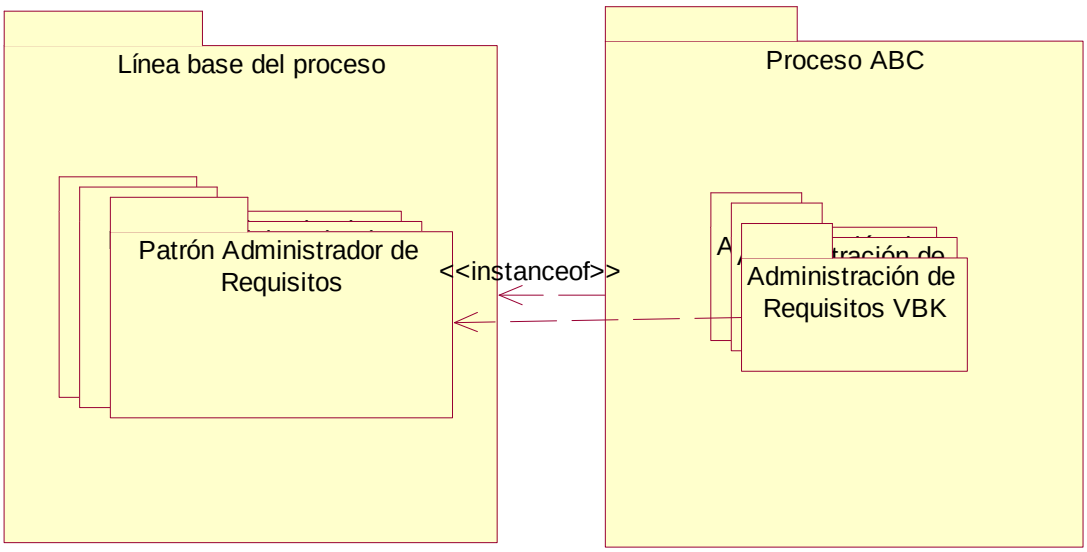
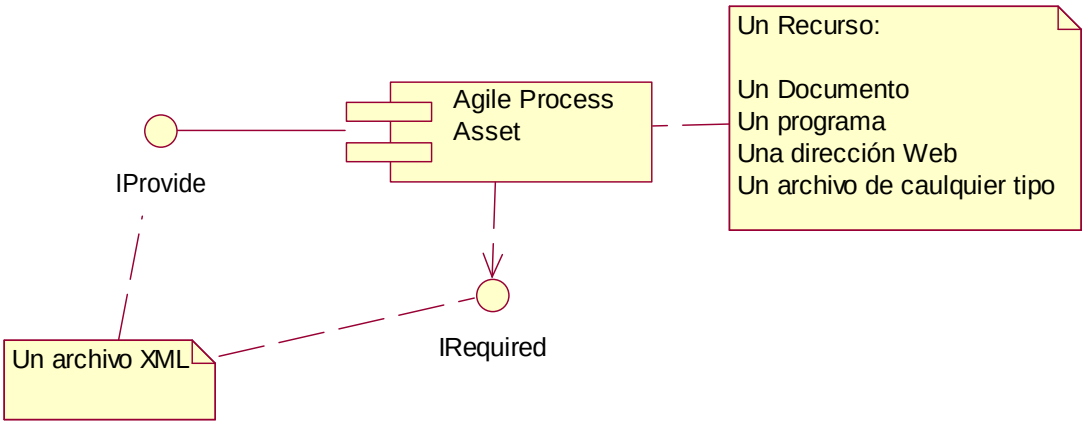



***Figura 38. La línea base del proceso y una instancia***

### 4.4 Representación de los activos de proceso

Los activos de proceso son finalmente componentes físicos que pueden ser administrados y configurado para armar soluciones concretas de procesos. Un proceso definido a partir de estos activos podrá usarlos tal como se han descrito, sin embargo, la idea es que puedan ser extendidos para soportar su adecuación a las necesidades de la organización.




Para especificar las interfaces de los componentes, la que provee (IProvide) y la que requiere (IRequired) se ha optado por usar XML Extensible Markup Language, por ser el lenguaje estándar para la interoperabilidad de la información, con el fin de garantizar la integración de estos activos de proceso a una herramienta de composición o definición de procesos, o a un sistema de workflow. Dado que la Arquitectura de la línea de procesos y de los activos de proceso ha sido someramente definida, no podemos tener un descriptor de los tipos de datos (DTD) completo, pero si hemos hecho una aproximación, para dejar trazado el camino, de cómo luciría este DTD y como luciría el archivo XML (descriptor) del patrón de proceso administrador de requisitos.

Una descripción como prueba de concepto de lo que sería la DTD se describe a continuación:

```
< ?xml version="1.0"?>
< !DOCTYPE agileprocessasset
[

< !ELEMENT record (idProcessAsset nombre tipo assetsInclude?
                   localizacion iRequired? iProvide? >
< !--  cada activo de proceso 6 campos, los dos ultimos opcionales -->

< !ELEMENT idProcessAsset(#PCDATA)>
< !-- Numero serial -->

< !ELEMENT nombre (#PCDATA)>
< !—- nombre del activo  -->

< !ELEMENT tipo(#PCDATA)>
< !!—- tipo de activo  -->

< ¡ELEMENT assetsInclude(agileprocessasset+)>
< !!—- activos que incluye -->

< !ELEMENT localizacion(#PCDATA)>
< !!—- ubicación del activo  -->

< !ELEMENT iRequired(itemRequirement+)>
< !!-- Lista completa de requisitos requeridos -->
```

```
< !ELEMENT iProvide(itemRequirement+)>
< !!—- Lista completa Una lista de citas al documento -->

< !ELEMENT itemRequirement EMPTY>
< !ATTLIST itemRequirement
        idRequirement CDATA #REQUIRED;
        tipoSPEM CDATA #REQUIRED;
        nombreReq CDATA #REQUIRED;
        tipoCMMIComponent CDATA #OPTIONAL
        descripcion CDATA #OPTIONAL;>
< !—Un item de requisito -->
]>
```

Basado en este dtd, el patrón de proceso iría acompañado de su respectivo archivo de configuración. Este archivo de configuración puede ser utilizado para su búsqueda y para su adecuación a un proceso determinado. Veamos como podría quedar el archivo de configuración del activo de proceso patrón administrador de requisitos.

```
<!DOCTYPE SYSTEM
"http://www.unicauca.edu.co/~ahurtado/DTD/agileprocessasset.dtd">
        <agileprocessasset>
                <idProcessAsset>24</idProcessAsset>
                <nombre>Administrador de Requisitos</nombre>
                <tipo> AgileProcessPattern</tipo>
                <localizacion>  http://www.unicauca.edu.co/~ahurtado/patronREQM
                </localizacion>
                <iProvided>
                        <itemRequirement>
                                <idRequirement> 1 </idRequirement>
                                <tipoSPEM> discipline </tipoSPEM>
                                <nombreReq> REQM </nombreReq>
                                <tipoCMMI>GG3 Institucionalizar un proceso
                                definido</tipoCMMI>
                        </itemRequirement>
                </iProvided>
                <iRequired>
                        <itemRequirement>
                                <idRequirement> 2 </idRequirement>
                                <tipoSPEM> process </tipoSPEM>
                                <nombreReq> proceso </nombreReq>
                        </itemRequirement>
                        <itemRequirement>
                                <idRequirement> 3 </idRequirement>
                                <tipoSPEM> Rol </tipoSPEM>
                                <nombreReq> Administrador de requisitos </nombreReq>
                        </itemRequirement>
                </iRequired>
        </agileprocessasset>
```

## 5. Conclusiones y Trabajo futuro

A través de este trabajo hemos evidenciado la posibilidad de que las empresas desarrolladoras de software puedan certificar sus procesos con CMMI nivel 2 a través de prácticas ágiles. Las áreas sobre las que deberán complementarse un poco los métodos ágiles son la gestión de requisitos, medición y análisis, el aseguramiento integrado de producto y proceso. Con respecto a la gestión de la configuración deberá hacerse un esfuerzo más grande puesto que los métodos ágiles alcanzan a cubrir poco más que la gestión de la configuración del producto entregable.

De este trabajo se evidencia la necesidad de poder contar con unos activos de proceso que permitan definir e instanciar procesos para alcanzar los requisitos de cada área a través de elementos de origen ágil. Para ello se hace necesario contar con un catálogo de activos de proceso que incluyan valores, técnicas, prácticas, elementos de proceso (roles, disciplinas, fases, ciclos de vida, etc.), que permitan implementar estos requisitos de una forma práctica. Todos los procesos deberán tener los siguientes elementos básicos:

- Una política clara, obtenida de valores y principios de las metodologías ágiles;
- Un ciclo de vida que pueda ser gestionado y definido obtenido de los ciclos de vida de los procesos ágiles y de otras posibles fuentes,
- Unos flujos de trabajo que implementen en detalle cada parte del ciclo de vida: artefactos, actividades y roles: obtenidos de los artefactos, prácticas y roles definidos en los métodos ágiles.

Cada proceso será particularmente único, puesto que depende de las metas y de la estructura de la organización. Para ello es importante definir un catálogo de activos de proceso que agrupe todos estos elementos del mundo ágil, los estructure de una forma más estandarizada y brinde las guías de aplicación a una empresa para alcanzar algún nivel de disciplina en la organización a través de métodos ágiles.

Hemos esbozado la estructura preliminar de un catálogo de procesos ágiles, a lo que nos hemos atrevido a ponerle Agile SPsL – Agile Software Process Lines (Línea de Procesos Ágiles) y ejemplificado a través de la definición de un activo y de un caso de aplicación que los visualice en conjunto para cumplir un área de proceso CMMI a un nivel de madurez específico. Ese activo fue visualizado como un patrón de proceso, con el que se busca visualizar un conjunto de prácticas ágiles como una implementación de un requisito CMMI, ya que la idea es que las PyMEs puedan visualizar sus procesos (ágiles) como procesos certificables. De igual forma somos conscientes de que al darle un poco de disciplina y organización a las prácticas las organizaciones podría alcanzar un equilibrio entre agilidad y disciplina que les sirva de base de mejoramiento de sus procesos de desarrollo: satisfacción del cliente y mayor productividad.

A mediano plazo se espera realizar una ingeniería de dominio completa para especificar e implementar la Arquitectura de Agile SPsL, poblarla con los activos de proceso y experimentar su uso en los ámbitos académico e industrial para poder medir su eficacia en el marco de un programa de mejora del proceso de desarrollo de software consistente con los objetivos de SIMEP-SW.

## 6. Referencias


[1] Manifiesto for Agile Software Development. http://agilemanifesto.org/

[2] Fuggetta, A. Software Process: A Roadmap. International Conference on Software Engineering. Proceedings of the Conference on The Future of Software Engineering. ACM Limerick, Ireland. 2000. pp 25-34

[3] Object Management Group – OMG. SPEM, Software Process Engineering Metamodel Specification. Version 1.0 formal/02-11-14. 2002.

[4] Software Engineering Institute. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1) Staged Representation. CMU/SEI-2002-TR-012 ESC-TR-2002-012. 2002.

[5] Planning Extreme Programming, Kent Beck, Martin Fowler, Addison-Wesley, 2001

[6] Clements, P., Northrop, L. Software Product Lines: Practices and Patterns. Boston, MA: Addison-Wesley, 2002.

[7] Asociación Española de Normalización y Certificación. Tecnología de la Información Proceso de Ciclo de Vida del Software. ISO/IEC. (1995). ISO/IEC 12207 – UNE 71044, 1999.

[8] Kival C., Weber et Al. , Modelo de Referência para Melhoria de Processo de Software: uma abordagem brasileira.

[9] OKTAVA. H. "Modelo de Procesos para la Industria de Software MoProSoft" Versión 1.1. México 2003.

[10] Lycett, M. et. Al. Migrating Agile Methods to Standarized Development Practice. IEEE Computer Society. June 2003. p.79-85.

[11] Linda M. Northrop. SEI's Software Product Line Tenets. IEEE Software. July/August 2002. p. 32-40.

[12] García F. et Al. Líneas de productos, componentes, frameworks y mecanos. Informe Técnico – Technical Report DPTOIA-IT-2002-004. Marzo, 2002.

[13] Griss, M. L. Implementing Product-Line Features by Composing Component Aspects. Proceedings of First International Software Product Line Conference. Denver, CO (USA). August, 2000.

[14] Bosch, J. (2000). "Design & Use of Software Architectures. Adopting and Evolving a Product-Line Approach". Addison-Wesley. 2000

[15] Frank van der Linden. Software Product Familias in Europe: The Esaps & Café Projects. IEEE Software. P. 41-49.

[16] Bass, L., et. Al. Software Architecture in Practice, Second Edition. Addison Wesley. 2003.

[17] Shaw M, Garlan D. Software Architecture: Perspectives on an Emerging Discipline. Prentice Hall. 1996.

[18] McGregor., J. Testing a Software Product LineTechnical Report CMU/SEI-2001-TR-022

[19] The Product Line Technical Probe (PLTP). Disponible en la dirección http://www.sei.cmu.edu/productlines/pltp.html

[20] Matinlassi, M. Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, KobrA and QADA. Proceeding of the International Conference on Software Engineering. 2204

[21] Deming, W. E. Out of the Crisis. MIT Center for Advanced Engineering Study, Cambridge, MA, 1982.

[22] Juran, J. and Gryna, F. Juran's Quality Control Handbook, Fourth Edition. New York: McGrawHill Book Company, 1988.

[23] Fagan, M. "Advances in Software Inspections." IEEE Transactions on Software Engineering, SE-12, 7, (July 1986).

[24] Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., and Paulk, M. "Software Quality and the Capability Maturity Model," Communications of the ACM, 40, 6 (June 1997): 30-40.

[25] PSP

[26] Humphrey, W. S. Introduction to the Team Software Process. Read-ing, MA: Addison-Wesley, 2000.

[27] Software Engineering Institute. Standard CMMISM Appraisal Method for Process Improvement (SCAMPI$^{SM}$), Version1.1: Method Definition Document. CMU/SEI-2001-HB-001. 2001.

[28] Gremba, J., Meyers, C. The IDEALSM Model: A Practical Guide for Improvement. 1997.

[29] ISO/IEC. (1995). ISO/IEC 12207 – UNE 71044 (1999) Tecnología de la Información Proceso de Ciclo de Vida del Software. AENOR – Asociación Española de Normalización y Certificación.

[30] ISO/IEC. (1998a). ISO/IEC 15504 TR2:1998, Software Process Assessment - Part 2: A reference model for processes and process capability. International Organization for Standardization.

[31] ISO/IEC. (2002). ISO/IEC 12207 AMENDMENT 1: Information Technology - Software Life Cycle Processes Amendment 1. International Organization for Standardization.

[32] ISO/IEC. (1998c). ISO IEC 15504 TR2:1998, Software Process Assessment - Part 4: Guide to conducting assessment. International Organization for Standardization.

[33] ISO/IEC. (1998b). ISO IEC 15504 TR2:1998, Software Process Assessment - Part 7 : Guide for Use in Process Improvement. International Organization for Standardization.

[34] Batista J. Figueiredo A. SPI in very small team: a case with CMM. Software Process Improvement and Practice. IEEE. 2000.

[35] Calvo-Manzano, J. (1999). Métodos de mejora del proceso de desarrollo de sistemas de información en la pequeña y mediana empresa. Tesis Doctoral. Universidad de Castilla-La Mancha. Ciudad Real.

[36] Moen, R., Nolan T. W., Provost Ll. Improving Quality Through Planned Experimentation. Ed. McGraw-Hill. 1991.

[37] TANTARA INC. Software process improvement & related standards/models. 2.001. Disponible en http://www.tantara.ab.ca/a_stds.htm

[38] L. Scott, R. Jeffery, L. Carvalho, J. D'Ambra and P. Rutherford. Practical Software Process Improvement – The IMPACT Approach in Proceedings 2001 Australian Software Engineering Conference, pp. 182-189, IEEE Computer Society Press, 2001.

[39] Moen, R., Nolan T. W. Process Improvement, Quality Progress, Vol. 23, no. 5, pp. 62-68, September. 1987.

[40] Scott, L. et Al. Practical Software Process Improvement - The IMPACT. Project Proceedings of the 13th Australian Software Engineering Conference. IEEE 2001.

[41] FUGGETTA, Alfonso, CONRADI Reidar, Improving Software Process Improvement. 2.002. IEEE Software July/August 2002 (Vol. 19, No. 4)

[42] Paulk, Mark C.; Curtis, Bill; Chrissis, Mary Beth Chrissis, and Weber, Charles. Capability Maturity Model. Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403. 1993.

[43] SEI. Systems Engineering Capability Maturity Model SE-CMM. CMU/SEI-95-MM-003. 1995.

[44] Curtis, B, Hefley, B and Miller, S. The People Capability Maturity Model CMU/SEI-2001-MM-001. 2001.

[45] SEI, IPD-CMM - Integrated Product Development. CMU/SEI-MM-97-001. 1997.

[46] Jack Cooper (editor) Matt Fisher (editor) Software Acquisition Capability Maturity Model. Technical Report. CMU/SEI-2002-TR-010. 2002.

[47] Electronic Industries Association. Systems Engineering Capability Model (EIA/IS-731). Washington, D.C.: Electronic Industries Association, 1998.

[48] Chrissis, M., Konrad, M., Shrum, Sandy. CMMI Guidelines for Process Integration and Product Improvement. SEI Series in Software Engineering. Addison-Wesley. 2003.

[49] Fowler, Martin. The New Methodology. http://www.martinfowler.com/articles/newMethodology.html, April 2003.

[50] Ambler, Scott. "Agile modeling: effective practices for extreme programming and the unified process". Wiley Computer Publlishing. 2002.

[51] Canós, J., Letelier, P., Penadés, M. "Metodologías ágiles en el desarrollo de software". Actas de las VIII Jornadas de Ingeniería del software y Bases de Datos, JISBD 2003. Alicante, Noviembre de 2003.

[52] Beck, K. Extreme Programming Explained: Embrace Change. Addison-Wesley. 1999.

[53] Jacobson, I. et. Al. The Unified Software Development Process. Addison-Wesley. 1999.

[54] Paulk, M. Extreme Programming from CMM Perspective. IEEE Software. November/December. 2001.

[55] Acebal C., Cueva J. Extreme Programming (XP): un nuevo método de desarrollo de software. NOVATICA/UPGRADE mar./abr., nº 156. 2002.

[56] Schwaber, K. "The Scrum development process". OOPSLA '95 Workshop on Business Object Design and Implementation, Austin, 1995.

[57] Rising, L., Janoff, N. The Scrum Software Development Process for Small Teams. IEEE Software July/August. Pp. 25-32. 2000.

[58] Abrahamsson, P. Agile Software development methods: A minitutorial. VTT Technical Research Centre of Finland, http://www.vtt.fi/virtual/agile/seminar2002/Abrahamsson_agile_methods_minitutorial.pdf, 2002.

[59] Shwaber K., Beedle, M. Agile software development process with Scrum. Prentice Hall, 2002.

[60] Gilb, T. Evolutionary Development. ACM SIGSOFT Software Engineering Notes. Volume 6, Issue 2, April 1981, p. 17-17, ACM Press.

[61] en ACM Software Engineering Notes y "Evolutionary Devivery versus the 'Waterfall Model'" en ACM Sigsoft Software Requirements Engineering Notes.

[62] GILB. Kai. “Evolutionary Project Management & Product Development (or The Whirlwind Manuscript)”. 2003. http://www.iti.upv.es/~squac/JTS/JTS2004/docs/Wmodel.pdf

[63] COCKBURN. Alistair. “Balancing Lightness with Sufficiency)”. Septiembre de 2000. http://alistair.cockburn.us/crystal/articles/blws/balancinglightnesswithsufficiency.html.

[64] Palmer S., Felsing M. A Practical Guide to Feature-Driven Development. Prentice Hall. 2002.

[65] Jim Highsmith. Adaptive software development: A collaborative approach to managing complex systems. New York, Dorset House, 2000

[66] Jennifer Stapleton. Dynamic Systems Development Method – The method in practice. Addison Wesley, 1997.

[67] Gamma, E. et. Al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley. 1994.

[68] Scott, A. et. Al. Process Patterns. Cambridge University Press/SIGS Books. 1998

[69] Lycett, M. et al. Migrating Agile Methods to Standardized Development Practice. Computer. Vol 36, No. 6 (June 2003), pp.79- 85.

[70] Vriens C. Certifying for CMM Level 2 and ISO9001 with XP@Scrum. Proceedings of the Agile Development Conference (ADC’03).Salt Lake City, Utah, 2003, pp. 120-124.

[71] Maller P, et al. Agilizando el Proceso de Producción de Software en un Entorno CMM de nivel 5. Revista IEEE América Latina. Special Edition - JISBD'2004. IX Jornadas de Ingeniería del Software y Bases de Datos. Vol. 3 Issue 1.

[72] Paulk, M. Extreme Programming from a CMM Perspective. IEEE Software. Vol. 18, No.6, (November/December 2001) pp. 19-26.

[73] Nawrocki J. et al. Toward Maturity Model for eXtreme Programming. Proceedings of the 27th EUROMICRO Conference 2001.Warsaw, Polland (September 2001), pp. 233-239.

[Fin de Documento]