Metodología AUP (Agile Unified Process – Proceso Unificado Agil) para Implementación del ERP ADempiere

Todos los esfuerzos están vinculados por un universo elegante que llamamos realidad en donde las dos dimensiones de tiempo y espacio establecen el panorama para la danza entre las dos fuerzas naturales del cambio y la complejidad. Es en este ámbito en el que convergen los ingredientes claves que son las personas, las mejores prácticas de contextuales y automatización como puente para el abismo entre la visión y la realidad. Sin embargo, a lo largo de nuestros esfuerzos a través de dominios (y con independencia de cualquier dominio particular), cambian más y la complejidad que uno intenta abordar es mas cambiante y la complejidad en si misma se torna como un círculo vicioso, plagado de riesgos y oportunidades.

Dentro de los negocios, la industria de sistemas y tecnología de la información, hay diversos medios para hacer frente a estas fuerzas y, en última instancia, aprovechar oportunidades, incluyendo el proceso unificado (UP) y enfoques ágiles. Este documento se centra en la definición de agilidad en el análisis de procesos, explorando la definición relativa a la UP y enfoques ágiles y describir el Agile Unified Process (AUP).

1.- El Proceso Unificado (UP) y Proceso Unificado Relacional (RUP):

El Proceso Unificado (UP) es un producto de software marco de ingeniería de procesos (un caso de uso impulsada por la arquitectura centrada en el riesgo, iterativo e incremental, en paralelo, la lucha, orientado a objetos, y el componente de enfoque). También es conocido como el Proceso Unificado de Desarrollo de Software (USDP). Ha sido desarrollado por Grady Booch, James Rumbaugh, e Ivar Jacobson (los Tres Amigos). La UP ofrece una infraestructura para la ejecución de proyectos de software de ingeniería de productos, un marco integrado de los hitos principales, secundarios y de disciplinas.

El Rational Unified Process (RUP) es un producto de proceso desarrollado y comercializado por IBM Rational Software. El RUP proporciona los detalles necesarios para la ejecución de proyectos de uso de la UP, incluyendo directrices, plantillas y asistencia de la herramienta.

La UP surgió como la unificación de Rational Rational Software Corporation enfoque y el proceso de Objectory AB, cuando Rational Software Corporation adquirió Objectory AB en 1995. El Rational Software Corporation, desarrolló el enfoque racional como resultado de diversas experiencias de los clientes. Ivar

Jacobson creó el proceso de Objectory principalmente como resultado de su experiencia con Ericsson en Suecia. Rational Software Corporation fue adquirida por IBM a finales de 2002.

2.- El Proceso Unificado Ágil (AUP):

La UP es un software producto ingeniería marco de proceso que puede abordarse mediante tres perspectivas, incluyendo colaboraciones, contexto y las interacciones que se centran en un ciclo de vida de compuesto por fases, disciplinas y iteraciones.

En la siguiente figura se muestra una vista conceptual de los elementos esenciales que constituyen la UP y AUP.

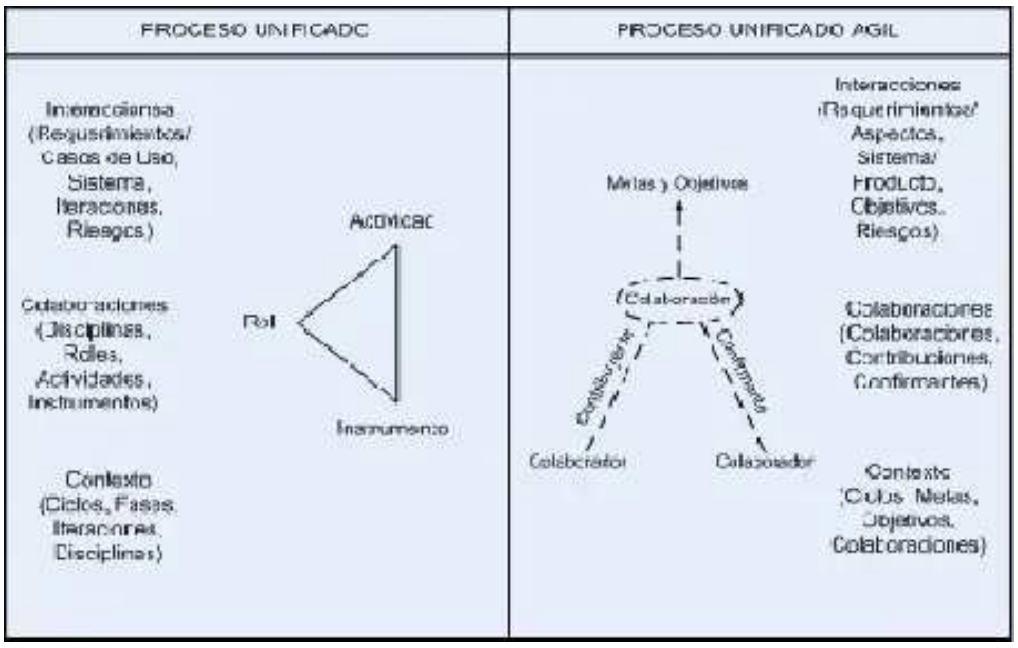

Figura: AUP vs RUP

Una colaboración implica una interacción dentro de un contexto. En la UP, una colaboración captura quién hace qué actividades (cómo) sobre lo que funcionan los productos. Por lo tanto, establece los elementos de un proyecto. Colaboraciones implican a trabajadores (funciones), actividades y productos (artefactos) de trabajo. En el AUP, colaboraciones se centrarán en colaboradores (colaboradores y confirmantes), metas y objetivos y resultados. Un contexto se hace hincapié en el aspecto estructural o estático de una colaboración, los elementos que colaboran y su conglomerado o las relaciones espaciales. En la UP, un contexto captura cuando y donde estas actividades deben ser hechas, trabajo, productos (artefactos) producidos y consumidos. Por lo tanto, establece el marco para un proyecto. Contextos implican ciclos de desarrollo y fases, iteraciones y disciplinas. Las fases de la UP incluyen principio, diseño, construcción y transición. Disciplinas de la UP incluyen Business Modeling, requisitos, análisis y diseño, prueba, implementación, configuración y administración de cambios,

gestión de proyectos y medio ambiente. En el AUP, contextos centrarán en objetivos.

Una interacción enfatiza el aspecto de comportamiento o dinámica de una colaboración, los elementos que colaboran y en su cooperación o la comunicación temporal. En la UP, una interacción captura cuándo y por qué esas actividades se debe hacer y los productos de trabajo (artefactos) producidos y consumidos. Así, establece la ejecución de un proyecto. Interacciones implican requisitos o casos de uso, un sistema y su arquitectura, iteraciones, y el riesgo. En el AUP, las interacciones se centran en los objetivos.

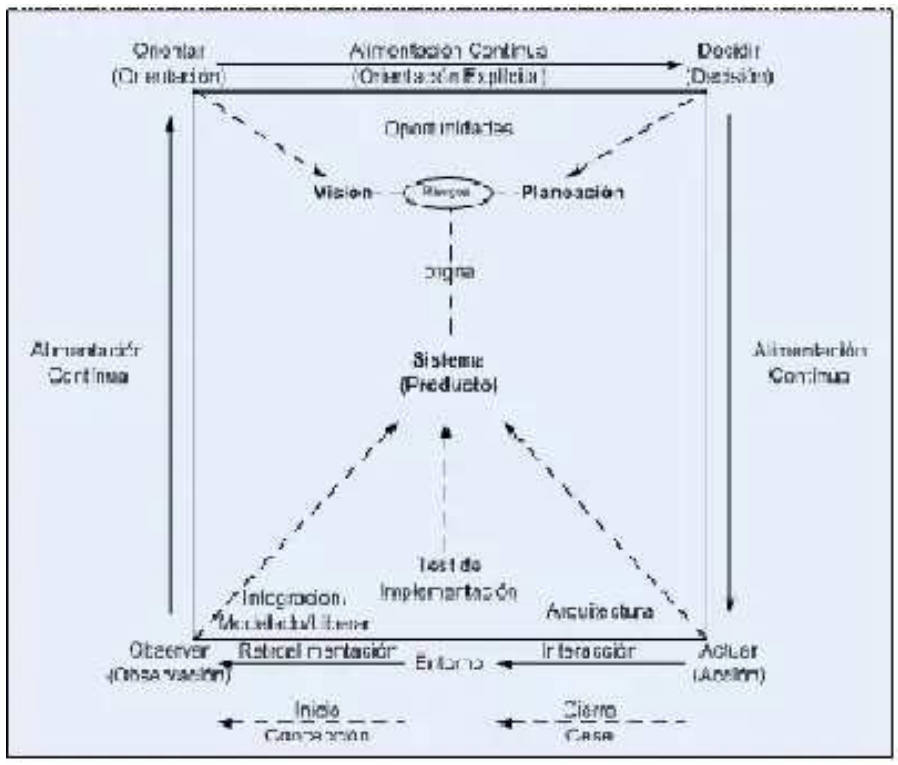

Figura: AUP

El AUP define metas y objetivos donde las colaboraciones se utilizan para lograr resultados. Una colaboración puede ser el trabajo realizado por una persona que puede interactuar con otros usuarios. Una colaboración puede ser un taller en el que participaron varios individuos que interactúan entre sí y pueden interactuar con otros fuera del taller. Objetivos son similares a UP fases en las que se utilizan como puertas que implican las partes interesadas y los usuarios. Objetivos son similares a UP iteraciones en caja el tiempo que implican a los usuarios y dar como resultado un producto de la versión.

El objetivo de iniciar (Concepción) es establecer el producto visión, plan de proyecto, negocios y justificación tecnológica e identificar los riesgos y las oportunidades. Puede haber uno o más objetivos que deben lograrse en alcanzar este objetivo.

- Las partes interesadas se centran en el problema, solución y restricciones (equipo de trabajo visionario).
- Los usuarios se centrarán en sus necesidades y las características de los productos de alto nivel (análisis de requerimientos por parte del equipo de trabajo).
- Las partes interesadas y los usuarios se centrarán en el establecimiento de la justificación del negocio para el producto y el proyecto.
- Los usuarios y el equipo de desarrollo de software se centrarán en el establecimiento de la justificación de tecnología para el producto y el proyecto.
- El equipo (equipo de desarrollo de software, las partes interesadas y los usuarios) se centran en dar prioridad a los riesgos y las oportunidades (labor de planificación por parte del equipo de trabajo).
- El equipo se centra sobre las características de los productos Execute (innovación) metas y objetivos en el plan de trabajo del proyecto (labor de planificación por parte del equipo de trabajo).
- El equipo establece la infraestructura de desarrollo.

  El objetivo de ejecutar (innovación) es evolucionar el visión de producto, plan de proyecto y justificación de negocio y tecnología; identificar y hacer frente a riesgos; identificar y aprovechar oportunidades mientras diseñar, desarrollar (aplicación y pruebas) y desplegar (integración, construcción y libertad) el producto. Normalmente hay muchos objetivos y metas, cada una resultante de la implementación del producto.

- Estrategias (Definir).
- El equipo considera posibles cambios en el producto y el proyecto y sus ramificaciones, decide aceptar o rechazar los posibles cambios y incorpora los cambios aceptados en el producto y el proyecto (equipo de trabajo y los cambios - consideraciones).
- Las partes interesadas y los usuarios se centran en evolucionar la visión (equipo de trabajo visión y requisitos).
- El equipo se centra en la evolución de los negocios y la justificación de la tecnología para el producto y el proyecto.
- El equipo se centra en la evolución de los riesgos y oportunidades y plan de proyecto (labor de planificación del equipo de trabajo).
- El equipo se centra en la adopción de un pulso del producto y el proyecto en metas y objetivos (talleres de pulso).

- Ejecutar (arquitectos, desarrollar (implementación y pruebas), implementar).
- El equipo se centra en aprovechar oportunidades mientras que enfrentan riesgos en diseñar, desarrollar (aplicación y pruebas) y desplegar (integración, creación y liberación) el producto (talleres de desarrollo de productos).
- El equipo aborda cuestiones (talleres de resolución de problemas) El equipo aborda cuestiones (talleres de resolución de problemas).
- El equipo identifica posibles cambios en el producto y el proyecto.

El objetivo de cerrar (dejar) es retirar el producto y cerrar el proyecto (talleres de cierre). Puede haber uno o más objetivos que debe lograrse en alcanzar este objetivo.

La visión provee dirección y un enfoque general, esta guía provee dirección específica a través de las metas y las misiones a través de los objetivos, y las colaboraciones establecen intuición, confianza, unidad, y cohesión.

3.- Justificación del Uso del Proceso Unificado Ágil como Modelo de Desarrollo:

Un modelo de ciclo de vida de software es una vista de las actividades que se llevan a cabo durante el desarrollo de éste, e intenta determinar el orden de las etapas involucradas y proporcionar unos criterios para avanzar de unas a otras. Por tanto, definir un ciclo de vida permite llevar un mayor control sobre las tareas, evitando que estas se vayan eligiendo y realizando de manera desordenada, según parezca que van surgiendo necesidades, que podrían ser puntuales y fácilmente evitables.

A.- Uso del Proceso Unificado de Desarrollo

Debido al carácter relativamente investigador de este proyecto, y a la necesidad de modificar los requisitos que surgirían según se fueran evaluando y probando las distintas posibilidades con las que se cuenta para desarrollarlo, un modelo pesado no se ajusta de manera adecuada.

Sin embargo, un modelo puramente ágil necesita de un equipo de desarrollo con experiencia para ser llevado a cabo de manera satisfactoria, por lo que éste tampoco es el caso más adecuado para su aplicación. Es por ello que se ha optado por un modelo que combina características de ambas orientaciones, proporcionando un enfoque iterativo e incremental: el Proceso

Unificado de Desarrollo propuesto por Rumbaugh, Booch y Jacobson.

B.- Características del Proceso Unificado de Desarrollo

Al igual que con cualquier otro modelo de desarrollo, del Proceso Unificado también se pueden destacar ciertas características.

B.1.- Iterativo e incremental:

El Proceso Unificado es un marco de desarrollo compuesto de cuatro fases:

- Inicio
- Elaboración
- Construcción
- Transición

Cada una de ellas es, a su vez, dividida en una serie de iteraciones que ofrecen como resultado un incremento del producto desarrollado, que añade o mejora las funcionalidades del sistema en desarrollo. Es decir, un "incremento" no implica necesariamente una ampliación de dicho sistema.

Durante cada una de estas iteraciones se realizarán a su vez las actividades definidas en el ciclo de vida clásico: requisitos, análisis, diseño, implementación, prueba e implantación. Aunque todas las iteraciones suelen incluir trabajo en casi todas estas actividades, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto. Por ejemplo, en la fase de inicio se centrarán más en la definición de requisitos y en el análisis, y durante la de construcción quedarán relegadas en favor de la implementación y las pruebas.

Si una iteración cumple sus metas, publicando una nueva versión del producto que implemente ciertos casos de uso, el desarrollo continúa con la siguiente. Cuando no las cumple, los desarrolladores deben revisar sus decisiones previas y probar un nuevo enfoque.

B.2.- Dirigido por los casos de uso:

Un sistema software se crea para servir a sus usuarios por lo que, para construir un sistema exitoso, se debe conocer qué es lo que quieren y necesitan. El término "usuario" no se refiere solamente a los usuarios humanos sino también a otros sistemas,

es decir, representa a algo o alguien que interactúa con el sistema a desarrollar.

En el Proceso Unificado, los casos de uso se utilizan para capturar los requisitos funcionales y para definir los objetivos de las iteraciones. En cada una, los desarrolladores identifican y especifican los casos de uso relevantes, crean el diseño usando la arquitectura como guía, implementan el diseño en componentes y verifican que los componentes satisfacen los casos de uso.

B.3.- Centrado en la arquitectura:

El concepto de arquitectura del software involucra los aspectos estáticos y dinámicos más significativos del sistema, y actúa como vista del diseño, dando una perspectiva completa y describiendo los elementos más importantes. La arquitectura surge de los propios casos de uso, sin embargo, también está influenciada por muchos otros factores, como la plataforma en la que se ejecutará, el uso de estándares, la existencia de sistemas heredados (aunque éste no sea el caso que nos ocupa) o los requisitos no funcionales.

Puesto que la arquitectura y los casos de uso están relacionados, por una parte, los casos de uso deben, cuando son realizados, acomodarse en la arquitectura, y ésta debe ser lo bastante flexible para realizar todos los casos de uso, hoy y en el futuro. De palabras de los propios creados del Proceso Unificado, es un problema semejante al del “huevo y la gallina”. En la realidad, arquitectura y casos de uso deben evolucionar en paralelo.

B.4.- Enfocado en los riesgos:

Para disminuir la posibilidad de fallo en las iteraciones o incluso la de cancelación del proyecto, se deben llevar a cabo sucesivos análisis de riesgos durante todo el desarrollo. Por supuesto, los riesgos principales deben ser identificados en una etapa temprana del ciclo de vida, y además, los resultados de cada iteración deben seleccionarse en un orden que asegure que estos son considerados primero.

B.5.- Ciclo de Vida del Proceso Unificado de Desarrollo:

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Al final de cada uno de ellos se obtiene una versión final del producto, que no sólo satisface ciertos casos de uso, sino que está lista para ser entregada y puesta en producción. En caso de que fuese necesario publicar otra versión, deberían repetirse los mismos pasos a lo largo de otro ciclo.

Como se es sabido cada ciclo se compone de varias fases, y dentro de cada una de ellas, los directores o los desarrolladores pueden descomponer adicionalmente el trabajo en iteraciones, con sus incrementos resultantes. Cada fase termina con un hito, determinado por la disponibilidad de un conjunto de artefactos, modelos o documentos. Las iteraciones de cada fase se desarrollan a través de las actividades de identificación de requisitos, análisis, diseño, implementación, pruebas e integración.

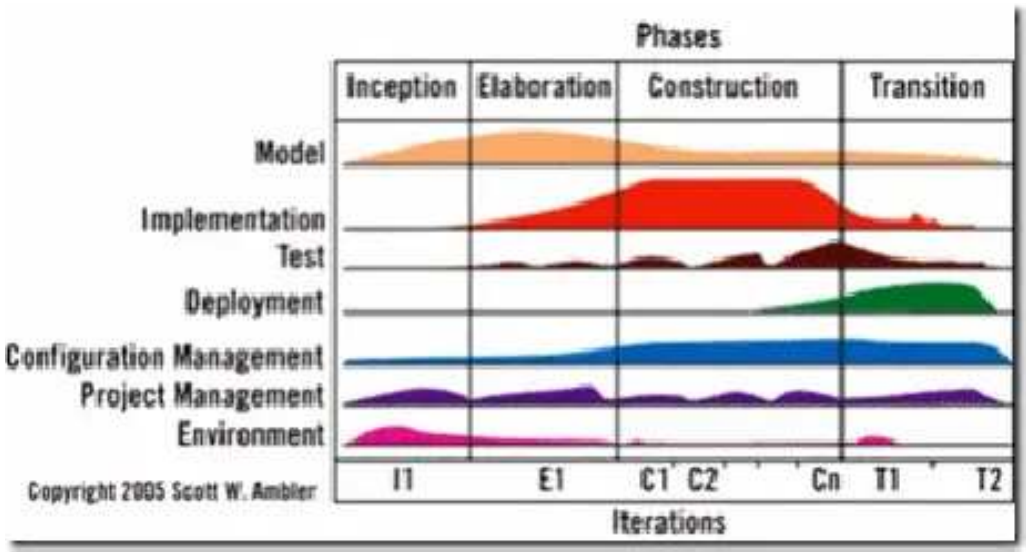

Figura: Fases AUP

4.- Fases e Hitos:

Fase: Inception (Concepción):

Suele ser la fase más corta del desarrollo, y no debería alargarse demasiado en el tiempo. En caso contrario, podríamos encontrarnos en una situación de excesiva especificación inicial, yendo en contra del enfoque relativamente ágil del Proceso Unificado. Los clientes deben participar activamente del modelado en alto nivel de los requerimientos ya que este determina el alcance inicial del proyecto. El objetivo es identificar una arquitectura viable.

- Explorar la utilización del producto escribiendo casos de uso.
- Identificar los procesos de negocio mediante la realización de diagramas de flujo de datos.
- Identificar principales entidades del negocio y sus relaciones. Esto se puede hacer trabajando en un dominio reducido del negocio.
- Identificar las principales reglas de negocio y los principales requerimientos técnicos.
- Comenzar la realización de un glosario que contenga los principales términos técnicos y del negocio.

Fase: Elaboración:

Durante esta fase deberían capturarse la mayoría de requisitos del sistema, aunque los objetivos principales son tratar los riesgos ya identificados y establecer y validar la base de la arquitectura del sistema. Esta base se llevará a cabo a través de varias iteraciones, y servirá de punto de partida para la fase de construcción. La fase de elaboración termina, por tanto, al alcanzar el hito de la arquitectura del sistema.

- **Identificar riesgos técnicos.** Los artefactos de requerimientos, en particular los casos de uso o los requerimientos técnicos pueden revelar potenciales riesgos del proyecto.
- **Modelado de la arquitectura.** Al construir el modelado de la arquitectura puede resultar útil hacer model storming para resolver alguna de las partes de la arquitectura.
- **Realizar un prototipo de la interfaz de usuario.** En paralelo con el modelado de la arquitectura debiera realizarse un prototipo de la interfaz que contenga las pantallas principales. No se necesita hacer mucho prototipado en esta fase porque los requerimientos seguramente van a cambiar y entonces el trabajo será descartado. El objetivo debería ser entender las principales pantallas de la interfaz e identificar el "look and feel" básico de la aplicación.

Fase: Construcción:

Es la fase más larga del proyecto, y completa la implementación del sistema tomando como base la arquitectura obtenida durante la fase de elaboración. A partir de ella, las distintas funcionalidades son incluidas en distintas iteraciones, al final de cada una de las cuales se obtendrá una nueva versión ejecutable del producto.

Por tanto, esta fase concluye con el hito de obtención de una funcionalidad completa, que capacite al producto para funcionar en un entorno de producción. También en esta fase es necesario trabajar al lado del cliente para identificar sus necesidades paso a paso.

- **Participación activa del cliente y modelado inclusivo.** La idea es que los clientes participen del modelado. Para esto se utilizan herramientas y técnicas sencillas, cuanto más sencillas mejor.
- **Mostrar los detalles de casos de uso.** Para ello se sugiere usar diagramas de actividad UML en lugar de descripciones textuales.

- **Explorar reglas de negocios y requerimientos técnicos con la misma profundidad.**
- **Aplicar model storming para el diseño.** En las iteraciones de construcción se debe realizar el modelado suficiente para resolver un único requerimiento antes de implementarlo.
- **Puede resultar útil realizar diagramas de secuencia UML, modelo de deployment, diagramas de clase UML, modelo de seguridad frente a amenazas, modelo de datos físicos.** No es necesario utilizar todas las técnicas, la utilidad o no de ellas depende del proyecto.
- **Documentar las decisiones de diseño críticas.** Se recomienda documentar aquellas decisiones que no resulten obvias y a aquellas que puedan resultar interesantes para alguien en el futuro. Estas decisiones documentadas pueden considerarse como un comienzo para el documento general del sistema que se finaliza en la fase de transición.

Fase: Transición:

En la fase final del proyecto se lleva a cabo el despliegue del producto en el entorno de los usuarios, lo que incluye la formación de éstos.

- Aplicar model stroming para intentar comprender la causa de defectos detectados.
- Finalizar la documentación general del sistema. El mejor momento para terminar la documentación general del sistema es en esta fase donde el alcance del sistema está verdaderamente estabilizado.
- Comprende tests del sistema, de los usuarios e instalación del sistema.
- Entre sus objetivos busca la aceptación de los stakeholders del negocio, de operaciones, soporte y de costos estimados.

5.- Disciplinas:

Las disciplinas se llevan a cabo de manera sistemática, a la definición de las actividades que realizan los miembros del equipo de implementación a fin de desarrollar, validar, y entregar el software de trabajo que responda a las necesidades de sus interlocutores. Las disciplinas son:

a.- Modelo: El objetivo de esta disciplina es entender el negocio de la organización, el problema de dominio que se abordan en el proyecto, y determinar una solución viable para resolver el problema de dominio.

b.- Aplicación: El objetivo de esta disciplina es transformar su modelo (s) en código ejecutable y realizar un nivel básico de las pruebas, en particular, la unidad de pruebas.

c.- Prueba: El objetivo de esta disciplina consiste en realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificando que se cumplan los requisitos.

d.- Despliegue: El objetivo de esta disciplina es la prestación y ejecución del sistema y que el mismo este a disposición de los usuarios finales.

e.- Gestión de Configuración: El objetivo de esta disciplina es la gestión de acceso a herramientas de su proyecto. Esto incluye no sólo el seguimiento de las versiones con el tiempo, sino también el control y gestión del cambio para ellos.

f.- Gestión de Proyectos: El objetivo de esta disciplina es dirigir las actividades que se lleva a cabo en el proyecto. Esto incluye la gestión de riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos, etc), coordinación con el personal y los sistemas fuera del alcance del proyecto para asegurarse de que es entregado a tiempo y dentro del presupuesto.

g.- Entorno: El objetivo de esta disciplina es apoyar el resto de los esfuerzos por garantizar que el proceso sea el adecuado, la orientación (normas y directrices), y herramientas (hardware, software, etc.) estén disponibles para el equipo según sea necesario.

Recomendaciones:

- No es necesario invertir mucho tiempo en detallar el modelado y la documentación en las fases de inicio y elaboración. Sólo deberían llevar unos días el Modelado Inicial de Requerimientos y el Modelado Inicial de la Arquitectura.
- Model storming se realiza cuando se precisa y sólo para obtener los detalles estrictamente necesarios. Típicamente esto se hace en la fase de construcción y no debería llevar más de unos minutos.
- El objetivo es crear modelos con la profundidad necesaria para lo que se está haciendo y no tratar de ir más allá. Hay que tener presente que en cualquier momento se puede volver atrás y explicitar más detalles si se necesita.

- La mayor parte de los modelos se descarta (sólo una pequeña parte de ellos se mantiene) Los modelos deberían hacerse empleando técnicas simples, tomando en cuenta que una vez que los requerimientos estén bien especificados serán descartados.
- Siempre hay que tener en cuenta oportunidades de re-uso. Esto no refiere sólo al código.
- La participación activa del cliente se considera fundamental para el éxito del proyecto.
- Se recomienda emplear arquitectura en capas.

6.- Entregables:

Entre los diagramas UML recomendados por el sitio web **ambysoft** (Ambysoft, 2010) aplicables para AUP tenemos:

- Diagramas de Casos de uso.
- Especificación de Casos de Uso.
- Diagrama de Clases.
- Diagramas de Secuencia
- Diagramas de Estado.
- Interfaces Graficas de Usuario.
- Diagrama de Componentes
- Diagrama de Despliegue

7.- Consejos

- Mantener los entregables tan simples y concisos como se pueda.
- Se necesita mucha menos documentación de la que se piensa.
- Trabajar conjuntamente con la gente que crea los entregables, para producir sólo lo necesario.
- Documentos ágiles son justo lo suficientemente buenos.
- Producir documentos es la peor manera de comunicar la información.
- Utilizar pizarrones, papel y Wikis para modelar y capturar la documentación.

8.- Conclusión para la Metodología AUP:

Si deseamos un método ágil entre XP y RUP tradicionales, que incluya explícitamente las actividades y las herramientas que están acostumbrados, entonces la más aconsejable es la AUP. XP no muestra explícitamente cómo crear algunos de las herramientas que la administración quiere ver. En el otro extremo del espectro está RUP, que es el gestor más utilizado de los desarrolladores, pero presenta una gran cantidad de herramientas. La AUP en comparación entre los dos, es la

adopción de muchas de las técnicas ágiles de XP y otros procesos ágiles que mantiene de las RUP.

En relación al XP, el AUP resulta ser un proceso muy pesado y en relación al RUP resulta ser un proceso muy simplificado, entonces, los desarrolladores o implementadores deberán decidir en: si desea buscar una forma de trabajo ligero esta XP y si desea trabajar con un proceso más detallado esta RUP.

Sin lugar a dudas, las personas son y seguirán siendo el "ingrediente original" necesarias para tener éxito. Sin embargo, con una mejor comprensión de agilidad, individuos, equipos y organizaciones están más facultados no sólo para hacer simplemente al cambio y complejidad, sino aprovechar el cambio y la complejidad para lograr una ventaja competitiva. Además, es experiencia, experimentación y aplicación de agilidad que permitirá obtener sus beneficios.