BUENAS PRÁCTICAS A USAR EN LAS IMPLANTACIONES SAP R/3 Y SAP NETWEAVER EN LAS PERSONALIZACIONES CON LENGUAJE ABAP

SERGIO MARTÍNEZ MARÍN

UNIVERSIDAD EAFIT
DEPARTAMENTO DE INFORMATICA Y SISTEMAS
MEDELLÍN
2008

ASESOR DEL PROYECTO:

Hernán Darío Toro Escobar

ESTE PROYECTO FUE ELABORADO POR:

Sergio Martínez Marín.

DEDICATORIA

A mis padres por el apoyo incondicional en toda mi vida.

AGRADECIMIENTOS:

Agradezco a Ángela María González Álvarez por sus aportes en el diplomado de construcción de software de la universidad EAFIT, al Asesor Hernán Darío Toro por su ayuda en los momentos de crisis de este proyecto, por último a Sandra Hernández por ser mi editora ad hoc en este trabajo.

No se puede dejar de lado a toda la comunidad informática, sin la cual el proyecto no hubiera sido posible.

Firma de jurados

_____

Saul Peralta

_____

Edder Zea

BUENAS PRÁCTICAS A USAR EN LAS IMPLANTACIONES SAP R/3 Y SAP NETWEAVER EN LAS PERSONALIZACIONES CON LENGUAJE ABAP

SERGIO MARTÍNEZ MARÍN

UNIVERSIDAD EAFIT
DEPARTAMENTO DE INFORMATICA Y SISTEMAS
MEDELLÍN
2008

Contenidos

Tabla de contenido

INDICE DE GRÁFICAS

INDICE DE TABLAS

INDICE ANEXOS

A QUIEN VA DIRIGIDO ESTE PROYECTO.

La tesis ha sido pensada para 3 tipos de roles en un proyecto de implantación de la plataforma SAP.

1. **Gerente general del proyecto**: Persona que por lo general es puesta por la empresa a la que se va a realizar la implantación de la plataforma SAP, posee conocimientos de gerencia de proyectos, pero no tiene conocimiento alguno de la plataforma previo al proyecto de implantación y mucho menos de cómo debe ser la gestión de las personalizaciones de esta.

2. **Líder de desarrolladores ABAP**: Persona que posee conocimientos técnicos de la plataforma, conoce a fondo su funcionamiento, tiene dotes de liderazgo y conciliador de problemas, pero por lo general carece de conocimientos básicos de gerencia de proyectos. Esto es de vital importancia para cualquier líder en cualquier proyecto.

3. **Desarrollador**: Persona con altos conocimientos técnicos en la plataforma, es muy importante que esta persona comprenda la importancia de los reportes y actividades “extras” que se les solicita en los proyectos.

Objetivos

Objetivo general:

Brindar una guía básica de los conceptos necesarios para llevar exitosamente las personalizaciones del sistema en un proyecto bajo plataformas SAP R/3 NetWeaver con Web AS en ABAP.

Objetivos específicos

Tener criterios y nociones de las metodologías más usadas en el medio con sus principales componentes

Dar a conocer los pasos básicos en la metodología propuesta por SAP

Explicar los conceptos más esenciales en la ingeniería del software para gestionar adecuadamente un proyecto.

INTRODUCCIÓN

En la actualidad, al enfrentar un proyecto de gran tamaño como lo son las implantaciones con la plataforma SAP R/3 y SAP NETWAVER, debemos tener en cuenta muchos aspectos para que el proyecto salga exitoso.

Con respecto a las personalizaciones en “algunas” organizaciones, esto se toma con poca importancia y después ocurren retrasos y problemas en el proyecto.

El siguiente trabajo explica los aspectos más relevantes a tener en cuenta en las personalizaciones para la plataforma SAP R/3 y SAP NETWAVER. Prácticamente se debe gestionar como un subproyecto dentro del proyecto de implantación en la organización.

1¿QUÉ ES SAP?

"SAP es un sistema de racionalización de operaciones para empresas. Es una herramienta estándar, pero a la vez parametrizable, y una solución modular, que aporta un alto nivel de flexibilidad y posibilita su personalización y adecuación a las necesidades de cada empresa."[1]

1. ¿Qués es SAP R/3?

"El Sistema SAP R/3 consta, en la vista modular, de áreas empresariales homogéneas, que soportan las operaciones empresariales y trabajan integradas en tiempo real. Las siglas SAP (System, Applications and Products) identifican a una compañía de sistemas informáticos con sede en Alemania, que se introdujo en el mercado de los sistemas de información con un producto denominado SAP R/2, antecesor al SAP R/3.

La integración en SAP se logra a través de la puesta en común de la información de cada uno de los módulos y por la alimentación de una base de datos común. El sistema SAP está compuesto de una serie de módulos funcionales que responden de forma completa a los procesos operativos de las compañías. "[2]

[1] Utilización del sistema SAP R/3 [sitio en Internet], disponible en http://www.tirant.com/libreria/detalle?articulo=8484681831 Último acceso: Noviembre 19 de 2007.

[2] Módulos De Sap R3 [sitio en Internet], disponible en http://www.mundosap.com/foro/showthread.php?t=281 Último acceso: Noviembre 19 de 2007.

Ilustración 1 Algunos módulos de SAP

“Desde un punto de vista funcional y de su arquitectura técnica, SAP R/3® puede definirse como un software abierto, basado en la tecnología cliente/servidor, diseñado para manejar las necesidades de información de una empresa. SAP R/3® es el software de estas características de mayor divulgación en todo el mundo, contando con más de 18.000 instalaciones en más de 100 países. Es la versión mejorada de un producto anterior (sistema R/2®) que ha permitido a SAP AG convertirse en la empresa líder de software empresarial, que es en lo que consiste básicamente SAP R/3®.

El sistema ECC es un sistema "On-line" y en tiempo real diseñado para cubrir de forma global las necesidades de gestión o información de corporaciones de tipo medio/grande. Consta de un conjunto de módulos totalmente integrados que cubren una amplia variedad de funciones de negocio entre las que se incluyen: Gestión Económico Financiera (Contabilidad General, Contabilidad Analítica, Activos Fijos, Módulo Financiero, etc.), Logística, Comercial y Distribución, Producción (Planificación, Control, Sistemas de Producción en serie, lotes, JIT, etc.), Control de Calidad, Mantenimiento, Gestión integrada de Proyectos, Recursos Humanos, Workflow, etc.

En definitiva, puede afirmarse que cubre todas las áreas funcionales de la empresa. Además, se están desarrollando y en su caso mejorando, las llamadas Soluciones Industriales, lo que significa una mayor adecuación del sistema SAP a las particularidades de cada negocio sectorial: Petróleo, Automoción, Publishing, Laboratorios Farmacéuticos, Retail, Alimentación, Sector Público, Telecomunicaciones, etc.

1.1.1 Características del sistema R/3:

SAP R/3® ha tenido tanto éxito debido a que combina unas características únicas que son ideales a la hora de trabajar en gestión empresarial.

SAP es muy flexible. Permite agilizar las tareas diarias de cualquier empresa independientemente del sector y del país en que trabaje, de su tamaño (si bien es cierto que parece estar dirigido más bien a grandes empresas) y de otros factores que pueden suponer un problema con otro software.

Otro aspecto importante es que es altamente integrado: supera las limitaciones jerárquicas y funcionales típicas de la empresa. Todo está integrado en un mismo software que coordina las distintas estructuras, procesos y eventos de todos los departamentos y áreas funcionales, permitiendo a cada empleado disponer de toda la información necesaria en todo momento. Así, no solo actualiza la información en tiempo real (importantísima característica de SAP que constituye una enorme ventaja), sino que además basta con introducir los datos una sola vez, puesto que es el sistema se encarga de pasar y actualizar los datos en el resto de los módulos o programas.

Así la interconexión entre centrales, oficinas, centros de producción, etc. queda asegurada. Antes de los sistemas SAP, todas las operaciones podían hacerse, en cada departamento, oficina, fábrica.... con sus programas específicos para cada una (software para la gestión de materiales, software para controlar salarios, ventas, compras, etc. y cada uno de ellos trabajando con sus propios protocolos, con su propia información, adaptados para hardware distinto, sin conectar ni compartir información) con lo que se trabajaba el doble: los datos que se repiten en diversas áreas se manejan varias veces (por ejemplo, en el almacén y en la administración) y, al no estar interconectados, (aunque exista una red interna, los diversos programas podrían trabajar con formatos, datos, máquinas incompatibles) es necesario que alguien se dedique a pasar la información de unos a otros, perdiendo un tiempo que se podría dedicar a mejorar la estrategia.

Imagine por ejemplo una fábrica de sillas metálicas, con su almacén de piezas, maquinaria, oficinas, muelle de carga para los camiones, oficinas, salas de reuniones etc. La empresa tendrá personal de almacén, conductores, operarios de maquinaria, encargados, jefe de personal, administrativos, comerciales, jefe de ventas, jefe de administración, gerente, y puede ser una pequeña empresa (tendrá mucho menos personal e instalaciones) o una gran empresa (más camiones, más personal etc.), o no ser un fábrica, sino un empresa de servicios, (en lugar de operarios podría tener dependientes), para el caso es lo mismo. En la fábrica se genera y se necesita información constantemente: entrada de mercancías y materias primas, necesidades de barras metálicas y tornillos, presupuestos que piden los clientes, albaranes de venta, notas de consumo de combustible, vencimientos de pedidos, pagarés que nos entregan los clientes, letras que firmamos a proveedores, facturas de acreedores, balances que hay que presentar en el Registro Mercantil, etc. etc. SAP R/3® evita que se repita innecesariamente la información (los datos que introduzca uno no hace falta que los introduzca otro, aunque sea de otra sección) y asegura que si en nuestra fábrica de Berlín se fabrica una silla, en ese mismo momento, nuestra sucursal en Albacete sabe que tiene una silla más que podría vender.

SAP es también abierto. Fue diseñado como un producto integrado, pero existe la posibilidad de instalar sólo parte del software (los módulos pueden utilizarse individualmente) para luego ir ampliando paso a paso según sus necesidades. Permite

además la comunicación con terceros (clientes o proveedores de su empresa que no sean SAP AG y sus partners).

SAP está directamente conectado a Internet y preparado para el comercio electrónico. Así la World Wide Web (www) puede servir como una interfase de usuario alternativa para las aplicaciones de empresa SAP R/3 , abriendo nuevas vías de negocio para los clientes.

La universalidad de SAP ECC no consiste sólo en la adaptabilidad a monedas, lenguajes o leyes, sino que es capaz de satisfacer las necesidades tanto de empresas pequeñas y medianas (más del 50% de las instalaciones, por ejemplo, HUMO COLPISA S.A.) como de grandes multinacionales (Mercedes Benz o Microsoft) y empresas de cualquier sector (aerospacial y defensa, automoción, banca y seguros, bienes de consumo, gestión sanitaria, ingeniería y construcción, petróleo y gas...).

SAP ECC® tiene además otras ventajas. Ofrece algo más que soluciones informáticas. Las aplicaciones también enlazan sus procesos empresariales con los de sus clientes y proveedores, permitiendo integrar a los bancos y otras empresas colaboradoras dentro de las comunicaciones internas (a nivel nacional e internacional).

Ofrece también la posibilidad de escoger entre más de 100 escenarios y 1000 procesos empresariales ya confeccionados en numerosos sectores industriales, permitiendo beneficiarse de los modelos empresariales de las empresas líderes. SAP ha desarrollado una amplia librería de procesos de negocio predefinidos que abarcan cada requerimiento de software funcional. Nuevos procesos de negocio y tecnologías se ponen a disposición de los clientes regularmente, facilitándoles soluciones empresariales al último nivel tecnológico, lo que les permite satisfacer la siempre cambiante demanda del mercado.

Dispone asimismo de sistemas EIS y de alerta temprana como son EarlyWatch y GoingLive Cheks que ayudan a detectar y corregir (incluye procesos de toma de decisiones) los problemas (como valores umbrales críticos) antes de que afecten a las operaciones.

SAP es infinitamente ampliable: es posible diseñar software específico en varios lenguajes de programación.

Necesita breves periodos de implantación. El tiempo que dure ésta, la empresa no podrá rendir al 100%, pero actualmente el 90% de las instalaciones se hacen en menos de un año, permitiendo un rápido retorno de la inversión. La duración no debe parecernos excesiva si tenemos en cuenta que supone la adaptación de maquinaria, procesos, personal e incluso el diseño específico de funcionalidades no cubiertas con el software estándar.

Tiene un sistema de arquitectura abierta que facilita a las compañías el elegir los equipos informáticos y los sistemas operativos de tal manera que se pueda aprovechar al máximo los avances en la tecnología. Emplea estándares internacionales reconocidos, lo que le permite a la empresa hacer distintas combinaciones (según sus

preferencias) entre proveedores de hardware, bases de datos, sistemas operativos y lenguajes de programación.

La compatibilidad con HTML y JAVA permite diseñar programas específicos en estos lenguajes, haciéndose más visual. Así JAVA permite programación de objetos para trabajar con ventanas, ratón, botones, etc.., y, por tanto, más fácil de manejar (más cómodo para el personal); mientras que el HTML está orientado a Internet (es el lenguaje de las páginas web).

La combinación de estructuras organizativas y funciones informáticas "on-line" (ya sea a través de Internet o de intranets -redes internas-) conduce, si se desea, a un proceso de datos descentralizado bajo una arquitectura cliente/servidor de tres niveles distintos:

- **Servidor de base de Datos:**
  El ordenador central (Host, miniordenadores, etc.) gestiona todas las funciones de la base de datos tales como actualización, consulta y otros.
- **Servidor de aplicaciones:**
  Está conectado al servidor de Base de Datos, y para cada departamento de la empresa, carga y ejecuta los programas y aplicaciones (ordenadores departamentales).
- **Servidor de presentación:**
  Estaciones de trabajo y ordenadores personales se conectan al nivel anterior de aplicación y presentan y hacen accesible la información y los procesos al usuario.

La flexibilidad de la arquitectura cliente/servidor le permite a las organizaciones obtener un óptimo uso de los recursos informáticos, pudiendo disfrutar de las ventajas que las nuevas y más avanzadas tecnologías de hardware van ofreciendo, así como la posibilidad de adaptar los procedimientos de negocio a los nuevos requerimientos que el mercado establece".[3]

[3] SAP R/3 [sitio en Internet], disponible en http://usuarios.lycos.es/cblsap/sisr3.html Último acceso: Noviembre 19 de 2007.

2. ¿Qué es SAP NETWEAVER?

Como reemplazo de SAP R/3 y poder responder a las exigencias del mercado actual SAP AG ha creado Sap Netweaver, la cual es una plataforma en la que se pueden alinear la infraestructura de tecnología con los requerimientos del negocio. SAP tiene una combinación de tecnologías y aplicaciones que reduce la complejidad e incrementa la flexibilidad del negocio. Con SAP NetWeaver, se pueden crear aplicaciones usando servicios empresariales, orquestándolo con procesos del negocio y eventos, administrando información empresarial que entregue aplicaciones y contenido a los usuarios de una forma más rápida y efectiva en costos.

Sap Netweaver es basado en web, posee una integración abierta y es una plataforma que provee una base de arquitectura orientada al servicio (SOA Service-Oriented Architecture) y permite la integración y alineamiento de las personas, información y los procesos de negocio. Sap Netwaver estándares abiertos que permiten la integración con la información y aplicaciones.

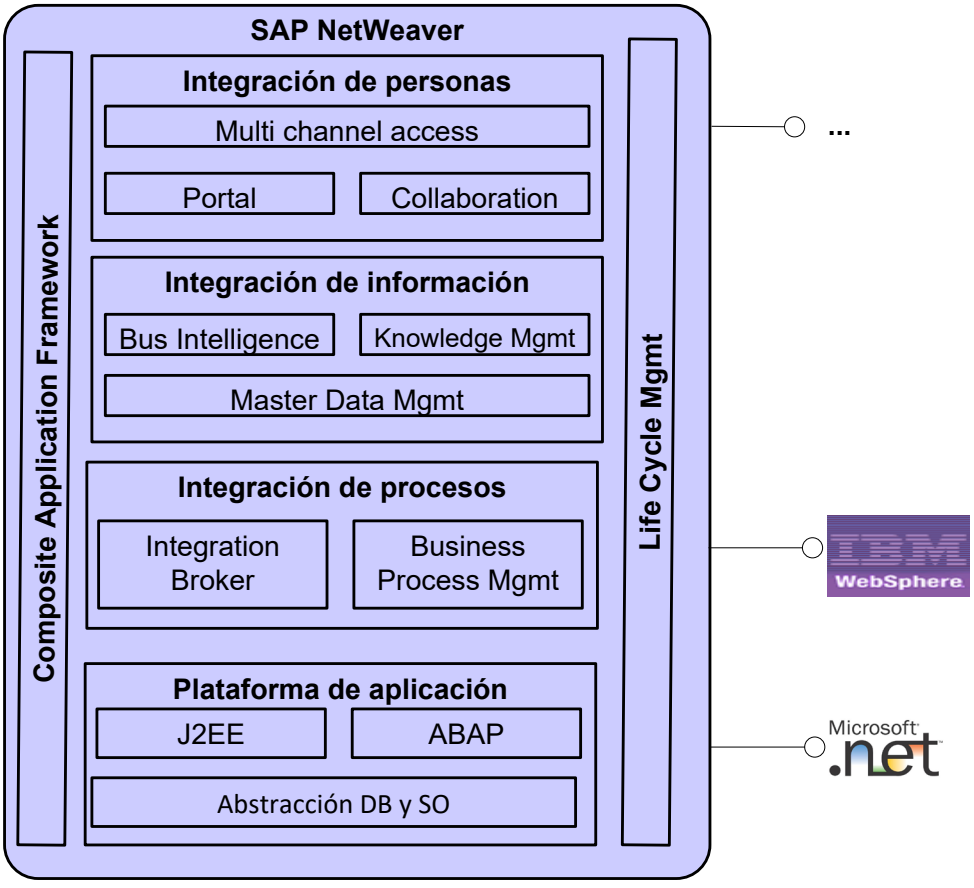


Ilustración 2 Arquitectura SapNetwaver

1.2.1 Arquitectura de SapNetweaver

Multi channel Access (Sap Mobile infrastructure):

Con una máquina virtual especial en los dispositivos móviles, permite desplegar e integrarse con aplicaciones SAP.

Portals y collaboration:

Son un conjunto de herramientas web, que brindan un punto de acceso común, maneja perfiles y paquetes de aplicaciones.

Business Intelligent:

Componente de SAP relacionado con la minería de datos.

Knowledge Management:

Posibilita a los usuarios para ordenar la información externa e interna usando funciones inteligentes de búsqueda.

Los usuarios pueden buscar , clasificar o manejar el contenido.

El Knowledge Management, está compuesto de:

- **Gestión de contenido:** Soporta el ciclo completo de la vida de los documentos, desde su creación, hasta el archivamiento de un documento.

- **Búsqueda y clasificación:** Búsqueda a nivel de textos completos a través de toda clase de documentos y su clasificación automática.

Master Data Management:

Cuando se guarda la información en diferentes localizaciones y sistemas al mismo tiempo usando la base tecnológica de SAP Exchange infraestructure se puede obtener la información de forma adecuada y garantizar la no duplicidad de estos.

Integración de procesos(Integration Broker y Business Process Mgmt):

Básicamente está cubierto por el SAP XI, el cual se encarga de la conexión de sistemas no SAP que están en diferentes plataformas y versiones con sistemas SAP.

XI está basado en una arquitectura abierta, usa principalmente estándares y ofrece servicios que son esenciales en un sistema complejo y heterogéneo.

Plataforma de aplicación:

El web application server actualmente viene en 2 versiones, con soporte para ABAP o Java, está el interpretador de la base de datos el cual nos permite tener un solo lenguaje SQL sin importar su motor ni plataforma en la que esté.

3. ¿Qué es una personalización?

La plataforma SAP ofrece un número de programas altamente parametrizables, pero por exigencias del negocio, se requiere un comportamiento de estos programas que no cubre el estándar, existen reportes que varían de organización en organización interfaces muy particulares que varían de empresa en empresa o programas que no existen en la plataforma. SAP ofrece las herramientas suficientes para suplir todas estas necesidades a la cuales se les llaman personalizaciones.

2 JUSTIFICACIÓN DEL PROYECTO

Hay muchos, aspectos por los cuales un proyecto no sale como se desea, ya sea por excederse en el tiempo inicialmente proyectado, errores de diseño, errores de codificación, cambios del negocio en el transcurso del proyecto, una inadecuada gestión de proyectos y en general son muchos los factores a controlar que hacen un gran reto sacar adelante un proyecto de índole tecnológico.

Según la envergadura del proyecto a enfrentar es directamente su complejidad y por ende el número de entregables a manejar en cada fase del ciclo de vida del proyecto, para poder tener un adecuado control sobre él. Muchas veces los entregables que se definen en un proyecto surgen por la misma necesidad natural del proyecto, pero debemos tener cuidado en no caer en los extremos de tener un exceso de documentación para todo o en tener una documentación escasa.

Por experiencia propia, en muchos proyectos de implantación del sistema SAP ECC, con respecto a las personalizaciones se incurre en varios detalles que se deben mejorar como son:

Inadecuada estimación: La complejidad y duración de una personalización, se llevan a cabo por la experiencia y no se acompaña esto de alguna métrica más precisa como lo puede ser los puntos de función. Cuando se lleva una mala estimación de los desarrollos a realizar en su totalidad, queda muy difícil hacer un cronograma de actividades ajustado a un contexto más real y a la vez se hace más difícil estimar el número adecuado de desarrolladores a utilizar en el proyecto.

Inadecuado manejo de riesgos: Todo proyecto a emprender tiene unos riesgos, inherentes, si no se tiene un manejo adecuado de éstos. Con sus debidos planes de contingencia y mitigación el proyecto se puede complicar.

Métricas inadecuadas: Las métricas son el indicador de cómo va el proyecto y si no son llevadas de forma adecuada, no se pueden tomar las medidas correctivas a tiempo o saber cómo está el proyecto realmente.

Falta de estándares: Es vital para el mantenimiento de las personalizaciones a futuro que tengan un estándar en su codificación, si no es así y cada desarrollador tiene su propio estándar de programación, las personas que a futuro le hagan mantenimiento a estos programas se les dificultara mucho en tiempo el entendimiento de cada desarrollo.

Falta de buenas prácticas de programación: Muchos programas son ineficientes y hacen lo que se menciona en su especificación pero no de la manera más adecuada, lo cual hace que se desperdicie tiempo y recursos de cómputo.

Desconocimiento de la arquitectura: Teniendo en cuenta se deben crear programas que hagan un uso adecuado del acceso a la base de datos, recursos del sistema,

consumo de red y tiempos de respuesta. Usualmente los desarrollos solo tienen en cuenta la especificación funcional, pero no la arquitectura de la plataforma.

Mala documentación: Al tener una mala documentación en las especificaciones de las personalizaciones se pierde mucho tiempo en entender lo que el funcional y usuario líder desean, así mismo se dificultaría enviar estas especificaciones a terceros que estén por fuera del proyecto dado el caso que se necesitaran. Por parte de la documentación técnica, si esta no es llevada de forma adecuada a futuro se complicaría en forma muy notable el mantenimiento de las aplicaciones por parte de una persona que no haya elaborado el programa y la gestión de cambios se dificultaría en forma extrema al no tener una trazabilidad del programa y poder medir el impacto real que tiene un cambio sobre la aplicación, además que un cambio en un programa muchas veces puede alterar un conjunto de programas y en estas ocasiones el impacto es mayor sobre el proyecto.

Gestión inadecuada de transportes: En los proyectos, una forma de garantizar que las versiones adecuadas de los programas van a pasar del ambiente de pruebas a productivo, es transportar las ordenes en un orden específico y las que son.

En algunos proyectos por decisiones gerenciales, se deben usar pocas ordenes de transporte (se deben rehusar) lo cual inhabilita el manejo de versiones de las aplicaciones.

En los grupos de funciones y en casos que deben intervenir varios desarrolladores se deben poner de acuerdo las órdenes de transporte a usar para luego no dañar el trabajo de los demás.

Las órdenes de transporte deben estar debidamente documentadas, para poder medir los impactos a la hora de hacer actualizaciones del sistema.

Si no se tienen en cuenta como mínimo estos aspectos en el manejo de las personalizaciones, se tendrán muchos problemas en el proceso de desarrollo y un cliente no muy contento con la situación. Todo proyecto tiene atrasos y problemas, pero estos deben ser debidamente manejados y llevados de la mejor manera y no que ellos nos lleven y manejen.

3 MARCO TEÓRICO

3.1 Metodología de desarrollo de software

"Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de este tipo.
Es como un libro de recetas de cocina, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Actualmente es imprescindible considerar los riesgos, aunque habitualmente las empresas, no han sido concienciadas de los riesgos inherentes al procesamiento de la información mediante ordenadores, a lo que han contribuido, a veces, los propios responsables de informática, que no han sabido explicar con la suficiente claridad las consecuencias de una política de seguridad insuficiente o incluso inexistente. Por otro lado, debido a una cierta deformación profesional en la aplicación de los criterios de coste/beneficio, el directivo desconocedor de la informática no acostumbra a autorizar inversiones que no lleven implícito un beneficio demostrable, tangible y mensurable.

Las técnicas indican cómo debe ser realizada una actividad técnica determinada identificada en la metodología. Combina el empleo de unos modelos o representaciones gráficas junto con el empleo de unos procedimientos detallados. Se debe tener en consideración que una técnica determinada puede ser utilizada en una o más actividades de la metodología de desarrollo de software. Además se debe tener mucho cuidado cuando se quiere cambiar una técnica por otra."[4]

En la industria se están aplicando actualmente tres metodologías en los proyectos informáticos, dos de las metodologías pertenecen al mundo de las metodologías ágiles son el SCRUM y el XP; por contraparte está una metodología mas tradicional y probada exitosamente en la industria y es el RUP.

La propuesta por parte de SAP AG en las implantaciones es una metodología llamada ASAP, la cual es muy propia de las implantaciones, esta se mostrará brevemente y se explicará lo que hay que tener en cuenta para poderla llevar de una manera exitosa con respecto a las personalizaciones.

[4] Metodología de desarrollo de software [sitio en Internet], disponible en http://www.um.es/docencia/barzana/IAGP/Iagp2.html Último acceso: Noviembre 19 de 2007.

3.1.1 ¿Qué es el desarrollo ágil de software?

"En los años 90, varias metodologías han llamado la atención del público. Cada una de ellas tiene una diferente combinación de viejas ideas y nuevas ideas.
Todas estas metodologías se han enfatizado en una muy buena colaboración entre el equipo de programación y los expertos del negocio; una comunicación cara a cara (mucho más eficiente que la comunicación escrita); haciendo frecuentemente entregas de valor para el negocio, teniendo equipos que se auto organizan y un código que se puede adecuar a los cambios en los requerimientos sin traumatismos."[5]

"En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.
Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.
Tras esta reunión se creó The Agile Alliance, una organización, sin ánimo de lucro, 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 es fue el Manifiesto Ágil, un documento que resume la filosofía ágil.

Según el Manifiesto se valora:

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

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

---

5 What Is Agile Software Development? [sitio en Internet], disponible en http://www.agilealliance.org/show/2 Último acceso: Noviembre 19 de 2007.

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

Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:"[6]

- "Nuestra mayor prioridad es satisfacer al cliente a través de la entrega temprana y continua de software con valor.
- Aceptamos requisitos cambiantes, incluso en etapas avanzadas. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software frecuentemente, con una periodicidad desde un par de semanas a un par de meses, con preferencia por los periodos más cortos posibles.
- Los responsables de negocio y los desarrolladores deben trabajar juntos diariamente a lo largo del proyecto.
- Construimos proyectos con profesionales motivados. Dándoles el entorno y soporte que necesitan, y confiando en ellos para que realicen el trabajo.
- El método más eficiente y efectivo de comunicar la información a un equipo de desarrollo y entre los miembros del mismo es la conversación cara a cara.
- Software que funciona es la principal medida de progreso.
- Los procesos ágiles promueven el desarrollo sostenible. Patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y los buenos diseños mejoran la agilidad.
- Simplicidad, el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de equipos que se auto-organizan.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo, entonces mejora y ajusta su comportamiento de acuerdo a sus conclusiones."[7]

---

[6] Metodologías Ágiles en el Desarrollo de Software [sitio en Internet], disponible en www.willydev.net/descargas/prev/TodoAgil.Pdf Último acceso: Noviembre 19 de 2007.

[7] Principles behind the Agile Manifesto [sitio en Internet], disponible en

Algunas metodologías ágiles son:

- Programación Extrema (XP).
- Scrum.
- Crystal.
- Evolutionary Project Management (Evo).
- Feature Driven Development (FDD).
- Adaptive Software Developmen(ASD).
- Lean Development (LD) y Lean Software Development (LSD).

A continuación se dará una breve introducción a las metodologías de SCRUM y XP.

---

http://www.agilemanifesto.org/principles.html Último acceso: Noviembre 19 de 2007.

3.1.1.1 SCRUM[8]

3.1.1.1.1 "Historia del SCRUM

El término "Scrum" viene de un estudio de 1986 de los japoneses Takeuchi y Nonaka. En dicho estudio se documentaban una serie de proyectos muy exitosos los cuales tenían en común el uso equipos chicos y multidisciplinarios.
El estudio comparaba a esos equipos hiper-productivos con la formación Scrum de Rugby.

Jeff Sutherland creó el proceso Scrum para desarrollo de software en 1993 usando este estudio como base y adoptó la analogía con los equipos de Rugby. Posteriormente, Ken Schwaber formalizó el proceso y lo abrió a toda la industria del software en 1995."[9]

3.1.1.1.2 ¿Qué es SCRUM?

Scrum es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de software como RUP o Extreme Programming.

3.1.1.1.3 ¿Porque SCRUM?

Scrum se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión.

Está diseñado especialmente para adaptarse a los cambios en los requerimientos, por ejemplo en un mercado de alta competitividad. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. De esta forma se puede adaptar en tiempo real el producto que se está construyendo a las necesidades del cliente. Se busca entregar software que realmente resuelva las necesidades, aumentando la satisfacción del cliente.

---

[8] Fuente principal: Schwaber Ken Beedle Mike. Agile Software development with Srcum USA; Prentice Hall Hill, 2002.

[9] ¿Qué es Scrum? [sitio en Internet], disponible en http://www.baufest.com/spanish/scrum/scrumconference2006/Que_es_scrum.pdf Último acceso: Noviembre 20 de 2007.

3.1.1.1.4 Roles [10]

SCRUM Team

Son las personas conformantes del equipo a realizar el proyecto como tal. Este equipo no debe ser muy numeroso, aproximadamente de 5 a 9 personas para que pueda funcionar adecuadamente.

El equipo tiene mucha autonomía en el cumplimiento de sus labores y entre los propios miembros del equipo se hace la asignación de las tareas a realizar, dando ésto como resultado la posibilidad de estar en diferentes papeles durante el transcurso del proyecto (analista, diseñador, codificador, tester, etc.), según las capacidades y destrezas de cada miembro del equipo y las exigencias de las tareas a realizar.

Dueño del producto

Es la persona que representa los intereses de la empresa en el desarrollo del producto, está encargada de velar porque el equipo SCRUM posea y cumpla la perspectiva del negocio en lo desarrollado.

El dueño del producto debe estar altamente capacitado en el proceso de negocio que se va desarrollar.

SCRUM Master

Persona encargada de facilitar todos los medios posibles para que el SCRUM team pueda realizar sus labores efectiva y correctamente

También debe hacer el rol de de “Project manager” al procurar que se tomen todas las medidas necesarias para el cumplimiento del proyecto y maximizar el retorno de la inversión por parte de la organización dueña del desarrollo.

[10] Scrum in five minutes [sitio en Internet], disponible en http://www.softhouse.se/Uploades/Scrum_eng_webb.pdf Último acceso: Noviembre 20 de 2007.

3.1.1.1.5 Artefactos

Product backlog:

Es una lista en donde van las funcionalidades que requiere el sistema, esta lista va en orden de prioridades y está a cargo del dueño del producto su elaboración

Durante el transcurso del proyecto la lista de funcionalidades y sus prioridades pueden ser cambiadas por el dueño del producto.

	Product Backlog	Estimación inicial	Complejidad	Estim. ajustada	Trabajo pendiente Sprint 1	2	3	4
ID	Elemento							
1	Nuevo formulario para peticiones de clientes	2	0.2	2,4	2,4	0	0	0
2	Configuración de respuestas automáticas	3	0.2	3,6	3,6	0	0	0
3	Envío automático de respuestas	1	0.2	1,2	1,2	0	0	0
4	Consulta para los clientes de peticiones enviadas	1	0.2	1,2	1,2	0	0	0
5	Modificación del cliente de sus peticiones enviadas	2	0.2	2,4	2,4	0	0	0
6	Acceso a peticiones sólo para clientes del portal jurídico	5	0.2	6	6	0	6	0
7	Consulta de peticiones por parte del staff	1	0.2	1,2	1,2	0	0	0
	SPRINT 1	15		18	18	0	0	0
8	Inserción de comentarios y reasignación a peticiones (staff)	2	0.2	1,2	1,2	1,2	0	0
9	Consultas por clientes, fechas y temas	3	0,2	3,6	3,6	3,6	0	0
10	[Continúa]....							

Tabla 1 Product Backlog

Haciendo una comparación con las competiciones de atletismo, en SCRUM un SPRINT es una entrega completamente funcional (Codificada, verificada y documentada) que se hace cada dos o cuatro semanas.

La cantidad de funcionalidades en cada SPRINT es una negociación entre el dueño del producto, el SCRUM master y el SCRUM team.

Sprint backlog

Una vez se ha concertado un acuerdo entre el dueño del producto, el SCRUM master y el SCRUM Team, se hace una lista detallada de las tareas específicas a realizar para poder desarrollar las funcionalidades de determinado SPRINT, en la lista de tareas se

especifica el miembro del equipo que la realiza, el estado en que esta, y por cada día del SPRINT se pone las horas faltantes

Sprint 1 backlog														
				Horas pendientes										
Descripción	**Originador**	**Responsable**	**Estatus**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Tarea 1		Sergio M	Completado	30	15	9	0	0	0	0	0	0	0	0	0
Tarea 2	Carlos M	Ricardo P	Pendiente	20	20	20	20	20	20	20	20	20	20	20	20
Tarea 3		Ricardo L	En Progreso	15	9	7	7	7	7	7	7	7	7	7	7
Tarea n		Pepito Mendez	Completado	50	40	30	20	10	0	0	0	0	0	0	0

Tabla 2 Sprint backlog

Burdown Chart

Gráfica en forma descendente que muestra los resultados que se van presentando en el proyecto.

La gráfica como tal no es una sola y se pueden hacer varias según las necesidades en el proyecto.

Pueden representar varias cosas, entre ellas:

1. El trabajo pendiente de todo un Sprint Vs los días que quedan de un Sprint.
2. Horas pendientes de actividades Vs días pendientes de un Sprint (por miembro del equipo o el equipo)
3. El trabajo pendiente de todo del proyecto Vs cantidad de Sprints planeados para todo el proyecto

Estos Burdown Charts son de gran utilidad para poder hacer el seguimiento al proyecto a muchos niveles, desde el estado general del proyecto, hasta comparativas del rendimiento real Vs El rendimiento esperado de un miembro del equipo

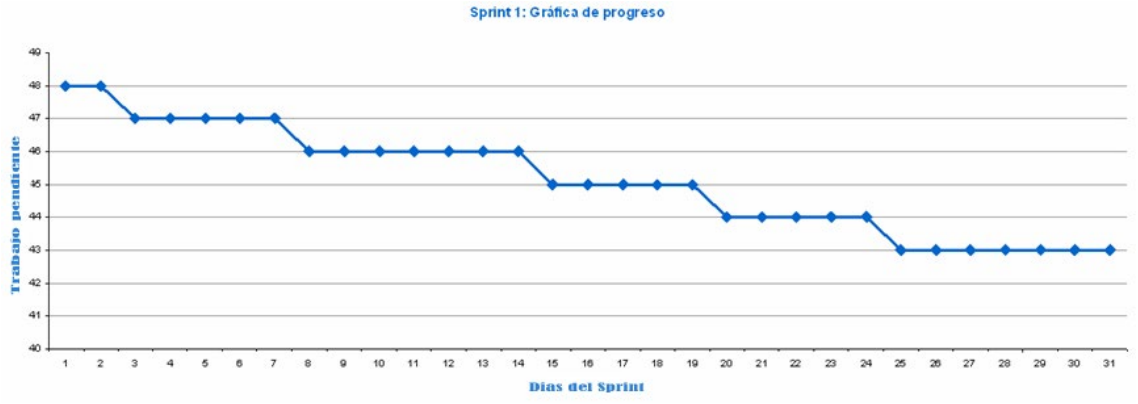


Ilustración 3 Burn down chart

3.1.1.1.6 Metodología de trabajo

Sprint Planning Meeting

Pre-requisitos:

El dueño del producto debe tener el product backlog ya elaborado.

Participantes:

El dueño del producto (son las personas que velan por los intereses de la empresa en el proyecto), el Scrum Master , Scrum Team y personas requeridas con determinados conocimientos en el negocio o en la tecnología a emplear.

Dinámica:

El Sprint planning está constituida en 2 sesiones aproximadamente de 4 horas cada una, la primera sesión es para seleccionar varias funcionalidades del Product Backlog y la segunda es para preparar el Sprint Backlog.

En esta reunión deben participar el Scrum Master, el dueño del producto y el Scrum team. Adicionalmente puede invitarse a gente que provea más información acerca del negocio o tecnologías necesarias, pero una vez que las personas adicionales hayan hecho su aporte ya no es necesaria su participación en la reunión.

El equipo sugiere los ítems del product backlog que considera que puede dejar completamente desarrollados, los cuales se mostrarán en el Sprint review, después de una concertación con los miembros del equipo y el dueño del producto, la

responsabilidad de la elección de que ítems irán exactamente en ese Sprint es del dueño del producto.

En la segunda sesión el equipo se centra en todas las actividades necesarias para llevar a cabo lo que se pacto con el dueño del producto para ese Sprint de esta reunión sale el documento Sprint Backlog. Para esta segunda parte debe estar el dueño del producto para resolver todas las inquietudes que posea el equipo.

Daily Scrum Meeting:

Es una reunión diaria donde se obtiene en general un balance de los avances de cada miembro del equipo.

Participantes:

El Scrum master, el scrum team y personas que deseen ir aunque no deben ser muchas.

Dinámica:

El scrum master y el scrum team se sientan en la mesa y las personas que vienen a ver la reunión, se sientan atrás, esto con el fin de hacerlas caer en cuenta que van en calidad de visitantes, más no a participar en ella.

El Scrum Master le formula 3 preguntas a cada miembro del equipo.
¿Cómo ha avanzado desde la última reunión?
¿Qué vas hacer desde ahora hasta la próxima reunión?
¿Qué problemas ha tenido para trabajar exitosamente?

Estas tres preguntas sirven para que el Scrum Master pueda saber exactamente como van las tareas a cumplir en el Sprint y solucionar los problemas que se presenten, así mismo da la oportunidad de que todo el equipo tenga una noción general del como va el Sprint.

Después de la reunión, si es necesario el Scrum Master toma las medidas correspondientes para poder solucionar los problemas de los miembros del equipo, o si está al alcance de los miembros del equipo entre ellos mismos se colaboran para poder solucionarlos.

Es vital la reunión diaria, ya que de esta forma cada integrante del equipo asume una responsabilidad ante él mismo y su equipo, dando como resultado que cada persona del equipo procure dar lo mejor de si.

Sprint Review Meeting

Pre-requisitos:

Los desarrollos que se trazaron como meta en el sprint backlog, deben estar completamente funcionales (codificados, verificados y documentados) en ambiente de calidad. Lo que no cumpla esta condición no se puede mostrar a menos que sea para explicar algo que ya este completo.

Participantes:

El Scrum master, el scrum team, el dueño del producto y personas que tienen que ver con el proyecto.

Dinámica:

Algún miembro del equipo presenta los objetivos que se pretendían alcanzar en el Sprint que acabo de pasar y los ítems que se cubrieron en el product backlog. Varios miembros del equipo exponen en que les fue bien y en que puntos no, y el porqué de ello.

En la mayoría de estas reuniones los miembros del equipo presentan las funcionalidades hechas y responden las preguntas de las personas que tienen que ver en el proyecto a medida que se van presentado las funcionalidades.

La reunión es un espacio para el dueño del producto y los interesados en el proyecto expresen sus ideas e impresiones de cómo ven el proyecto, que esta bien y que debe corregirse, identificación de nuevas o correcciones en el product backlog así como la reorganización en prioridades del product backlog.

Al final de esta reunión el Scrum master da la fecha del próximo Sprint review meeting.

Sprint Retrospective Meeting

Participantes:

El dueño del producto (opcional), Scrum master y el Scrum Team.

Dinámica:

Cada miembro del equipo responde estas 2 preguntas:
¿En qué le fue bien en el pasado Sprint?
¿En qué se puede mejorar para el próximo Sprint?

A medida que cada miembro del equipo va respondiendo estas preguntas, el scrum Master va tomando notas de ellas.

El equipo debe observar las anotaciones hechas por el Scrum master sobre los aspectos a mejorar y hallar entre todos las mejores formas de solucionarlos, además de dar cabida a cambios y mejoras potenciales al próximo Sprint backlog.

El Scrum master debe hacer todo lo posible en esta reunión de retroalimentación para que el equipo en cada Sprint valla mejorando en sus labores.

3.1.1.1.6 Valores en el SCRUM.

Responsabilidad.

Uno de los pilares del Scrum, es la confianza que se deposita en cada uno de los participantes al realizar sus labores. Es por ello que se les da la mayor autonomía y libertad en el cumplimiento de sus actividades y como requisito para lograr esto, cada integrante debe tener un sentido absoluto de la responsabilidad.

Concentración.

En un proyecto, salen muchas actividades por hacer, pero para poder hacer las cosas de la mejor forma posible, los miembros del equipo se deben centrar en hacer sólo una actividad al tiempo y de esta manera dar lo mejor de si en cada actividad que se esté haciendo.

Respeto.

Al trabajar en un grupo de personas, la gente se debe procurar de tratar a las demás de la mejor forma posible. En esta clase de proyectos es de saberse que las presiones y el tiempo es algo que juega en contra.

Coraje.

En esta metodología, se trabajan tecnologías de alta complejidad, cambiantes o simplemente el proyecto es muy cambiante en sus funcionalidades. Las personas tienen mucha autonomía para realizar sus labores, pero a la vez asume mucha responsabilidad y presión para llevarlas a cabo. Este tipo de proyectos no es para toda clase de personas.

3.1.1.2 XP[11]

3.1.1.2.1Definiendo XP

XP es una Metodología Ágil para equipos de desarrollo pequeños o medianos que encaran un cambio rápido de requerimientos.

XP consiste en 4 partes: Valores, principios, actividades y prácticas, en la figura se muestran como se componen unas de otras a través de actividades que son ejecutadas a través del ciclo de vida del proyecto.

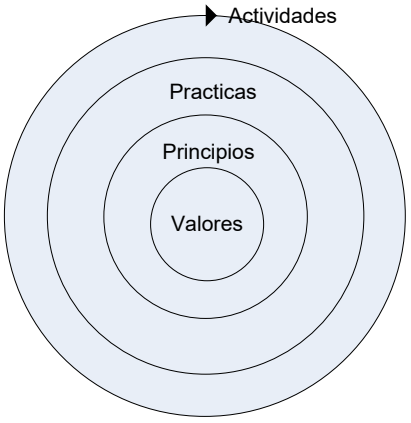


Ilustración 4 XP es basado en unos valores, que son soportados mediante actividades, prácticas y principios.

3.1.1.2.2 Valores:

XP maneja un conjunto de valores que es la esencia de la metodología.

Simplicidad:

Simplicidad en XP es definido como "has las cosas lo más simples posibles y que funcionen". Nosotros resolvemos problemas presentes simplemente y confiamos en que los problemas a futuro también los podamos resolver. En la mayoría de las metodologías tradicionales se crean expectativas en la planeación, donde el cliente hace un esfuerzo por estimar la totalidad del sistema. Hay un entendimiento implícito que este modelo cambiará en el futuro. Se debe implementar únicamente lo que el usuario necesita ahora, ¿Para qué poner características que probablemente no vas a

[11] Fuente principal: Baird Stewart. **Teach Yourself Extreme Programming in 24 Hours** USA; Sams Publishing, 2002

necesitar?, esto ocurre mucho en las metodologías tradicionales porque el valor de un cambio sería muy costoso, pero manteniendo el costo de los cambios constante se remueve la necesidad de este tipo de prácticas.

El análisis de un sistema de forma simple, lo hace más fácil de comunicar, tiene menos puntos de integración y un escalamiento mayor.

Comunicación:

Todas las metodologías tiene un proceso de comunicación, pero en XP tiene un valor muy importante poner hincapié en la comunicación oral, no en documentos, reportes y planes.

Sin una constante comunicación entre todos los miembros del equipo para colaborarse, el proyecto fracasaría.

Retroalimentación:

El valor de la retroalimentación es vital para el éxito de cualquier proyecto. Con XP las constantes preguntas acerca del estado del sistema son respondidas constantemente.

Como se tienen varios periodos de desarrollo cortos, se debe analizar cada x tiempo cómo se están haciendo las cosas para mejorar o tomar métodos correctivos a tiempo, asegurando así un producto de alta calidad.

Coraje:

Es la tenacidad de trabajar rápido y volver a desarrollar si es necesario. Coraje en XP debe ser pensado en el contexto de los otros 2 valores anteriores, sin el coraje se iría al caos el proyecto. El valor del coraje en XP está basado en ser fuerte, y tener pruebas automatizadas y no ser imprudentes en lo desconocido. Las personas que desarrollan en XP cambian lo que son, tentativas de cambio de parecer en codificación incierta, con algo más veraz.

***3.1.1.2.3 Principios*:**

Basados en los valores del XP, que guían el desarrollo de software.

Rápida retroalimentación:

Significa que los desarrolladores usan los cortos loops de desarrollo para verificar que están haciendo las cosas de forma adecuada para satisfacer las necesidades del cliente.

Asuma simplicidad

Asuma la mayor simplicidad posible en cada problema que vaya a resolver, significa que únicamente se hacen soluciones para la presente iteración. No hay una bola de cristal para saber a futuro lo que “de pronto” se necesitará.

Cambios incrementales:

Solucione los problemas con una serie de pequeños cambios. Esto aplica a la planeación, desarrollo y al diseño. En vez de hacer una sola entrega de todo, se van haciendo pequeñas entregas según la necesidad del negocio.

Acoja el cambio:

Adopte la estrategia de preservar opciones cuando este resolviendo problemas difíciles.

Trabajo con calidad:

La calidad del trabajo nunca puede estar comprometida. XP eleva la importancia del código y su chequeo con el criterio de hacer primero los test y luego el programa.

***3.1.1.2.4 Actividades*:**

Teniendo en cuenta los valores y principios, lo que se hace a diario son las siguientes actividades:

Escuchar:

XP está basado en la comunicación y tiene prácticas que requieren escuchar activamente. Al tener una comunicación menos escrita, hay una necesidad sobre la calidad de la comunicación verbal. Más allá de decir que los desarrolladores deben escuchar a los clientes, XP tiene prácticas que propenden a una mejor comunicación, sin necesidad de tanta documentación.

Las personas de los proyectos XP deben aprender y madurar su forma de comunicación.

Haciendo chequeos (testing):

No es algo que se debe hacer justamente antes de entregar algo al cliente. Es algo integral en el proceso, por lo que se requiere que los desarrolladores los hagan antes de escribir el código, no únicamente limitados la búsqueda de errores, si no de funcionalidad y desempeño. Trayendo como ventaja un código de alta calidad y no tener que esperar a capturar errores posteriormente cuando los costos usualmente son mucho más altos.

Codificando:

El código se debe ir refinando a través de prácticas como la refactorización, programación en pares y la revisión de código, con un pequeño diseño, los desarrolladores de XP deben escribir un código lo suficientemente claro para que con sólo leer el código otros miembros del equipo puedan entender la lógica, vale la pena clarificar que no sólo estamos hablando de simples comentarios en el código, el código como tal es una forma de comunicación.

Recuerde que el código es lo más cercano que tenemos al producto final, después de esto el código es compilado.

Diseñando:

Una de las ideas radicales de XP es que el diseño debe ir creciendo y evolucionando a través del proyecto.

El diseño no es estático o asignado a sólo rol y debe estar en la dinámica del equipo y en constante diseño en vez de limitarse a sólo las actividades de diseño, XP acepta la natural evolución del sistema.

***3.1.1.2.5 Practicas*:**

Haciendo más especificas las actividades, en XP se expresan las actividades en 12 prácticas esenciales.

Estas actividades son las que el equipo XP usan cada día para desarrollar los proyectos.

Planeamiento del juego:

Se debe generar un plan de alto nivel para la próxima iteración

Pequeñas entregas:

Los ciclos en XP consisten en constantes entregas que tengan valor al negocio

Metáforas:

Es la visión común, le da el sentido más sencillo de lo que debe hacer el código en XP.

Consiste en poner las descripciones con palabras entendibles tanto por parte del equipo técnico como para el cliente.

Diseño simple:

El diseño debe ser simple, en XP significa que el código debe ser lo más sencillo posible para que funcione correctamente.

Chequeo:

El corazón de XP es el chequeo automático, los test son escritos antes que el código para su posterior utilización**.**

Refactorización:

Es mejorar el diseño del código existente sin cambiar su funcionalidad**.**

Programación en parejas:

Una pareja de programadores se sientan en la misma estación de trabajo y comparten tareas de desarrollo.

Dueños colectivos:

Como se es dueño colectivo del código se está en la posibilidad de mejorar cualquier parte del código en cualquier momento.

Continúa integración:

Los componentes del sistema son hechos e integrados varias veces cada día

40 horas de trabajo a la semana:

Para mantener la calidad y el rendimiento no se necesita tener horas extras al equipo. Con un trabajo sostenido y regular se puede sostener la calidad.

El usuario en el mismo sitio de desarrollo:

El cliente como tal es parte del equipo.

Estándares de código:

Son convenciones de codificación que todo el equipo debe acatar.

3.1.1.2.6 Ciclo de vida de XP

En XP una de las premisas fundamentales es que el cliente y el desarrollador van a trabajar en conjunto para producir un software que tenga un valor real, el cliente direcciona al equipo de desarrolladores en cómo quiere hacer las entregas del producto para que aporten valor al negocio a través del ciclo de vida del proyecto XP.

Los proyectos XP consisten en dividir la visión que tiene el usuario acerca del producto y segmentarlo en varias entregas e iteraciones.

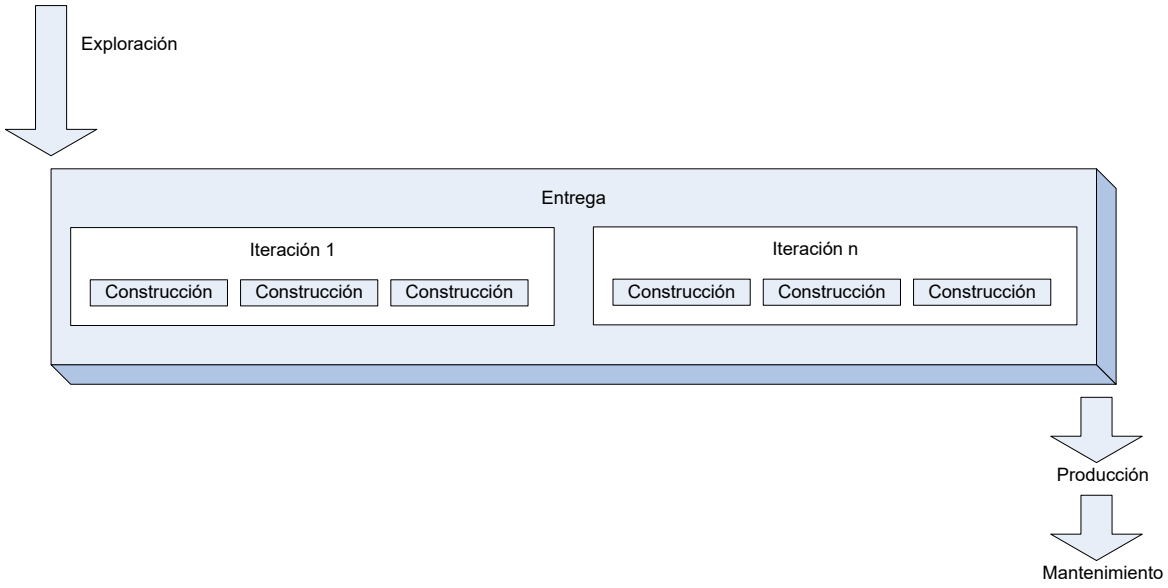


Ilustración 5 Ciclo de vida de una aplicación en XP

Durante cada iteración, hay múltiples construcciones, las cuales ocurren regularmente durante el día cuando el equipo integra nuevo código. El número de construcciones es dependiente del tipo de tecnología usada por el equipo de desarrollo.

Las entregas son hitos que se planifican con el cliente, mientras que las iteraciones son de tipo interno del equipo de desarrollo, que son usadas para lograr la elaboración de una entrega.

3.1.1.2.6.1 Etapas del ciclo de vida

Explorando los requerimientos del usuario:

Es donde se debe descubrir la visión y misión del sistema. Esta visión y misión se deben ser sencillas y aproximadamente entre 20 a 30 palabras de longitud en forma de metáfora. El cliente y los desarrolladores trabajarán en esta fase explorando opciones técnicas, definiendo requerimientos y completando una lista de historias de usuario.

Estimando y descubriendo

La fase de exploración incluye descubrimiento. Los clientes descubren lo que ellos quieren y los desarrolladores descubren acercamientos de soluciones y sus estimaciones.

Muchas veces el equipo de desarrollo debe indagar sobre nuevas tecnologías para brindar las mejores soluciones. En la búsqueda de estas alternativas se deben estimar cuanto tiempo se van a tardar y hacer su seguimiento, luego de esto contrastar el tiempo real que se han demorado en hacerlo, esto se hace con el fin de que el equipo vaya mejorando sus habilidades en las estimaciones ya que es vital cuando equipo empiece a realizar sus planeaciones para cada iteración.

El tamaño de la exploración depende de la naturaleza del proyecto y las tecnologías a usar. El tiempo necesario para buscar alternativas es breve si el equipo de desarrollo está familiarizado con las herramientas y la tecnología. La longitud total de esta fase puede ir entre unas pocas semanas o meses, dependiendo del tamaño y alcance de los requerimientos.

Una vez ha finalizado esta etapa, se documenta en forma de listas de usuario en forma de metáforas.

Creando el plan del proyecto

Es una fase corta en la cual el usuario y desarrolladores se pondrán de acuerdo para la primera entrega. Las historias de usuario son priorizadas y puestas de una forma tal que sean coherentes para el negocio y sean viables técnicamente.

Reponsabilidad del cliente:

- Definir las historias de usuario.
- Definir qué valor para el negocio tienen las historias de usuario.

- Definir que historias van a estar incluidas en una entrega.

Resposabilidad del desarrollador:

- Estimar cuanto se va a demorar una historia de usuario en hacerse.
- Avisar al usuario sobre los riesgos tecnológicos en los cuales se está incurriendo.
- Medir el avance del equipo.

En la etapa de planeamiento deben estar involucrados el equipo de desarrollo y el cliente para crear un perfecto balance de los recursos tecnológicos a usar y dar un real aporte de valor para el negocio. Los desarrolladores deben crear unas historias de usuario con "cero valor al negocio" para la parte de la infraestructura que va a soportar el sistema. Un ejemplo de esto podría ser el establecimiento de los ambientes de prueba y desarrollo. Haciendo esto se asegura que todo el trabajo del equipo va ha ser estimado y va a tener su respectivo seguimiento.

Esta etapa en XP se llama "el planeamiento del juego" la cual es la principal herramienta usada para la planeación. Esta etapa involucra al cliente y programadores a través de historias de usuario, estimaciones y priorizando.

Una vez se tienen el total de las actividades estas se dividen en varias iteraciones.

Dividiendo las entregas en iteraciones.

Una entrega es dividida en varias iteraciones de longitud de 1 a 4 semanas. La iteración inicial está centrada en la arquitectura base del sistema. Esta iteración es llamada de "cero valor para el negocio" ya que estas historias de usuario cubren aspectos técnicos y no traen un beneficio directo al negocio del usuario.

Las otras iteraciones deben aportar valor al negocio. El usuario selecciona las historias que desea y los desarrolladores procederán a desarrollarlas. En estas iteraciones el usuario puede cambiar de parecer en las historias de usuario, pero una vez se comienza una iteración, se realizara la construcción de lo pactado en esas historias de usuario. Lo que garantiza un costo bajo del cambio son los cortos tiempos de cada iteración.

El proceso de iteración

Al principio de cada iteración se hace una reunión de planeación en donde se hace una división de tareas de programación. El usuario selecciona las historias de usuario que se van a implementar. Los programas que hayan fallado en las pruebas de aceptación de la anterior iteración se incluirán en la presente iteración.

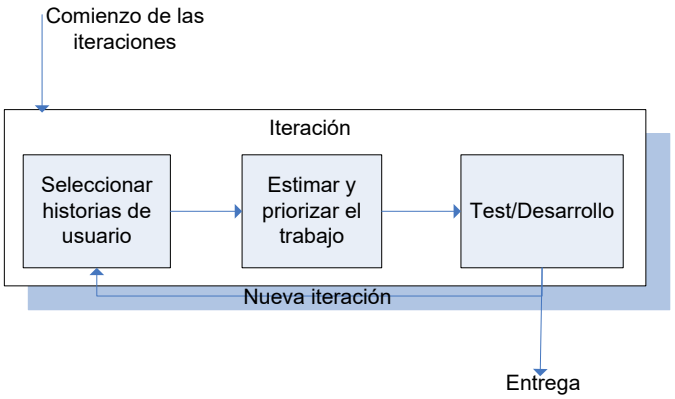


Ilustración 6 Iteraciones y entrega en XP

Al tener las historias de usuario a implementar, éstas se van desagregando en tareas específicas a realizar, estas tareas se escriben a anexo a las historias de usuario respectivas e idealmente deben ser de 3 días de extensión. Si excede esta cantidad de días se recomienda subdividirlo en tareas. Como ejercicio y mejora de las estimaciones las parejas de programadores deben dar a priori cuanto se demorarán en realizar las tareas que se les haya asignado.

Medición del rendimiento

En XP se mide el rendimiento (cantidad de trabajo que el equipo completa en el tiempo). Para empezar a refinar las estimaciones se empieza a preparar el equipo desde la iteración cero para que se vaya familiarizando con este tipo de estimaciones y se puedan hacer cálculos más refinados a futuro.

A medida que el proyecto va transcurriendo los resultados de los pares de programadores se va mostrando públicamente tanto los estimados de las tareas asignadas como su duración real. Esto con el fin de saber cada instante el estado real del proyecto y poder afinar las futuras estimaciones.

Desarrollo y pruebas

Primero se desarrolla las pruebas de la nueva tarea y se añade al framework de pruebas.

El par de programadores hace el desarrollo.

A medida que se va construyendo componentes, estos se van añadiendo al ambiente integrado Este tipo de integraciones se procura hacer con frameworks construidos para este propósito.

Al final de cada iteración, los usuarios corren sus pruebas de aceptación y las tareas que no pasen estas pruebas serán arregladas para la próxima iteración.

Poniendo el software en producción

Al final de la entrega, el producto es verificado y aprobado para ponerlo en ambiente productivo. En este punto el experto de usabilidad crea pruebas de aceptación en esta última construcción de sistema, se pone a prueba con usuarios reales que aseguren que el flujo del programa garantice el cumplimiento de sus labores. Si ocurre algún percance o detalle, se procede a corregirlos.

Muchas veces estos correctivos involucran el sistema en que se ejecutan los programas o en mejoramiento de los aplicativos, esto usualmente se prueba en ambientes espejo al productivo, con el fin de garantizar una evolución segura y estable.

En teoría se puede poner parte del sistema desarrollado desde la primera iteración, esto depende de lo que contribuya valor al negocio y factores externos como migración de datos o sistemas integradores tipo legacy.

Dependiendo del tipo de estrategia utilizada para desplegar el sistema, se deben asumir estrategias para reducir el riesgo que implica realizarlo.

Las formas típicas de desplegar un sistema son:

Bing bang:

Se apaga el sistema antiguo y se inicia con el sistema nuevo. Esto se hace cuando tener un manejo de transición gradual entre sistemas es muy complejo o costoso. Desplegar el nuevo sistema es sencillo, pero el riesgo es alto y las fallas en el nuevo sistema pueden ser críticas.

Por fases:

El Nuevo sistema es desplegado en piezas discretas de funcionalidad, esto se puede hacer cuando el sistema puede ser dividido en partes lógicas, se puede hacer y desplegar el sistema con bajos riesgos, pero puede prolongar mucho el posicionamiento del nuevo sistema en su totalidad.

En paralelo:

Esto requiere que tanto el nuevo sistema como el viejo estén corriendo durante un periodo de tiempo. Este enfoque es costoso por el múltiple manejo de datos, pero el riesgo manejado propende a disminuir ostensiblemente.

Bandera verde:

El sistema a desplegar en un ambiente nuevo, no tiene un sistema existente. Esto es común en un negocio que apenas está empezando. Es un ambiente ideal para desplegar el nuevo sistema ya que los riesgos son bajos y los posibles problemas para integrarse con otros sistemas en mínima.

Manteniendo el sistema después de la entrega:

Después de desplegar el sistema, este pasa al estatus de mantenimiento, en XP el sistema constantemente está evolucionando, en refactorización y refinamiento del sistema. La habilidad para parchar y actualizar el sistema es establecida y verificada por las pruebas automáticas del sistema.

Cuando estamos hablando del sistema productivo se debe tener mucho cuidado y se deben crear estrategias para ello.

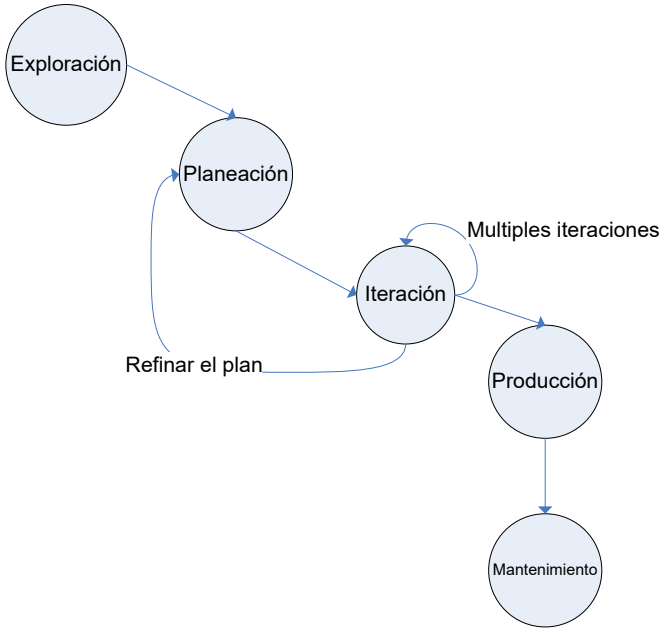


Ilustración 7 Ciclo de vida en XP

3.1.1.2.7 ROLES

Como trabajan juntos los roles en XP

Usualmente un negocio cubre demasiados aspectos que una sola persona no es capaz de abarcar totalmente, o el negocio tendría que ser lo suficientemente pequeño para que haya una persona experta en todo el proceso de negocio.

Ilustración 8 Un solo cliente con el equipo técnico

Las ventajas obvias de usar XP son el proceso de toma de decisiones simplificadas.

El problema al trabajar con 2 equipos es que son más 2 roles (cliente y desarrollador), existen más roles como el diseñador, pruebas, modeladores, arquitectos, etc.

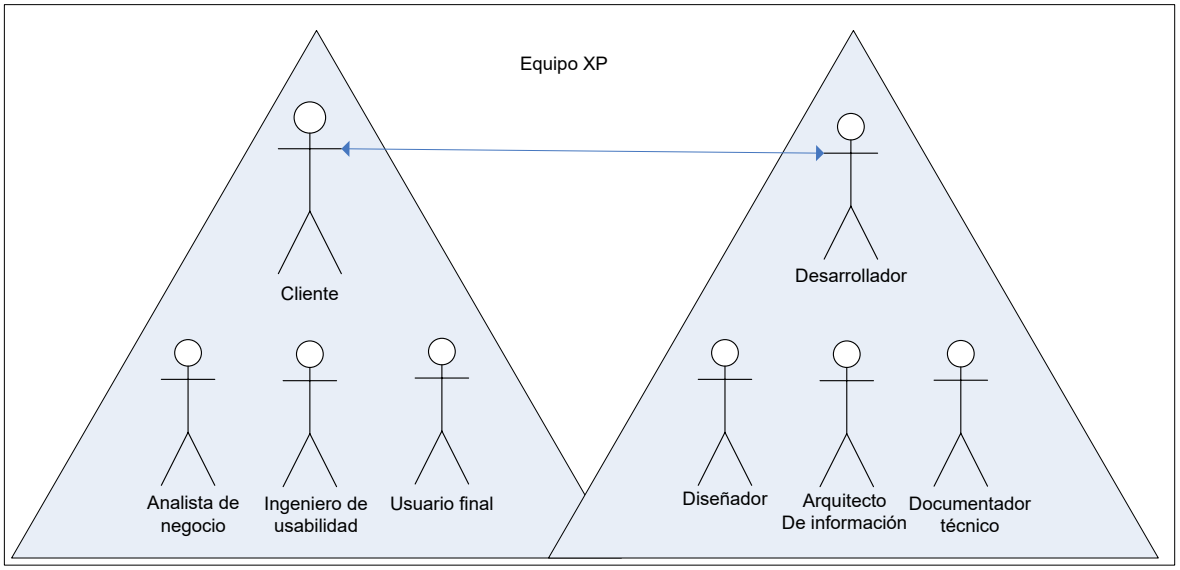


Ilustración 9 Equipo XP

Roles por parte del negocio:

Cliente:

El cliente define lo que el proyecto es, el valor del negocio y la dirección en que se debe mover el proyecto. Nos referimos como cliente el conjunto de personas que:

- Define el valor del negocio.
- Escribe historias de usuario.
- Escribe o especifica pruebas de aceptación.
- Debe ser el usuario final del sistema.

El cliente debe tener conocimientos muy sólidos del proceso del negocio y debe tener claridad de lo que el nuevo sistema está resolviendo. El nivel de sus conocimientos técnicos es menos crítico que el conocimiento que debe tener del negocio.

Administrador

Es la persona por parte del negocio, que lidera la comunidad de clientes, hacia el éxito del proyecto, esclarece y clarifica conceptos entre los usuarios y el grupo técnico. En

general facilita y hace todo lo posible por parte de la organización para liberarla de obstáculos y facilitar los medios para el proyecto.

Entrenador:

El entrenador es el principal facilitador de comunicación en el equipo.

El entrenador sabe que está haciendo el equipo, no 100%, pero si sabe en qué dirección van.
El entrenador necesita tener excelentes habilidades técnicas, porque dirige la refactorización tanto en la parte de diseño como en su implementación, en ocasiones debe guiar y ayudar a los pares de programadores.
Algunas de las características del entrenador deben ser:

- Alto entendimiento de XP
- Un amplio conocimiento de los procesos de desarrollo
- Confidente
- Liderazgo
- Sentido del humor
- Habilidad para balancear las perspectivas de la visón a largo y los problemas que se presentan presentes en el proyecto
- Resolución de problemas.
- Saber enseñar
- Integral
- Experiencia en técnicas orientadas a objetos
- Experiencia en lenguajes (java, c++, etc.)

Desarrollador:

- Hace una estimación de las historias de usuario
- Hace pruebas cuando es necesario
- Escribe pruebas
- Escribe código
- Participa en la reuniones de planificación
- Participa en las reuniones diarias.

Actividades diarias del desarrollador:

Los desarrolladores trabajan en parejas en una estación de trabajo. Si se está trabajando actualmente en una tarea ya asignada, se debe estar con una pareja, si se va a comenzar una nueva tarea se debe conseguir un compañero de trabajo.

Diseñar a este nivel significa ¿Cómo iremos a resolver esta tarea?

Se comienza la tarea de escribir las pruebas que chequean lo que podría fallar, antes de escribir el código. Expanda la prueba a todo lo que sea susceptible de fallos. En esta fase si se requiere más claridad se busca al usuario si es necesario.

Se codifica la tarea. El objetivo es completar únicamente lo que asegura que el componente pase exitosamente las pruebas.

Revisando el código, se miran las áreas que son susceptibles de mejorarlas y simplificarlas. Se para la refactorización una vez han sido satisfechas estas 2 premisas.

Se añade los componentes nuevos al escenario de integración y se verifica que la integración pase las pruebas en el ambiente integrado.

Ejecutor de pruebas

Es el rol para completar las pruebas funcionales. Verifica que el sistema trabaja como es esperado. Más que trabajar desde un punto de aseguramiento de calidad externo, es parte del equipo, probablemente escribirá pruebas con el cliente.

Ejecutor de métricas

Es la persona que junta las métricas como las historias de usuario o tareas completas y las disemina en el equipo, este rol debe decidir cual métrica es más significativa para el equipo, cual métrica es la que trae más aportes significativos al estado actual del proyecto.

En estas métricas debe estar como mínimo como es el estado de los últimos “builds”, número de historias de usuario, velocidad de desarrollo y todo lo que se considere necesario para hacer seguimiento.

Los resultados de las métricas siempre deben ser públicos y se pueden llevar en forma electrónica como documentos ofimáticas, páginas Web de intranet, etc.

3.1.1.2.7 Historias de usuario

Es una de las principales herramientas usadas en esta metodología, puede estar en medios impresos o digitales para su mejor manejo.

Un ejemplo de una historia de usuario en limpio podría ser:

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número: 1</td><td colspan="2">Nombre: Enviar artículo</td></tr>
<tr><td colspan="3">Usuario: Autor</td></tr>
<tr><td colspan="2">Modificación de Historia Número:</td><td>Iteración Asignada:</td></tr>
<tr><td colspan="2">Prioridad en Negocio: Alta
(Alta / Media / Baja)</td><td>Puntos Estimados:</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo:
(Alto / Medio / Bajo)</td><td>Puntos Reales:</td></tr>
<tr><td colspan="3">Descripción:

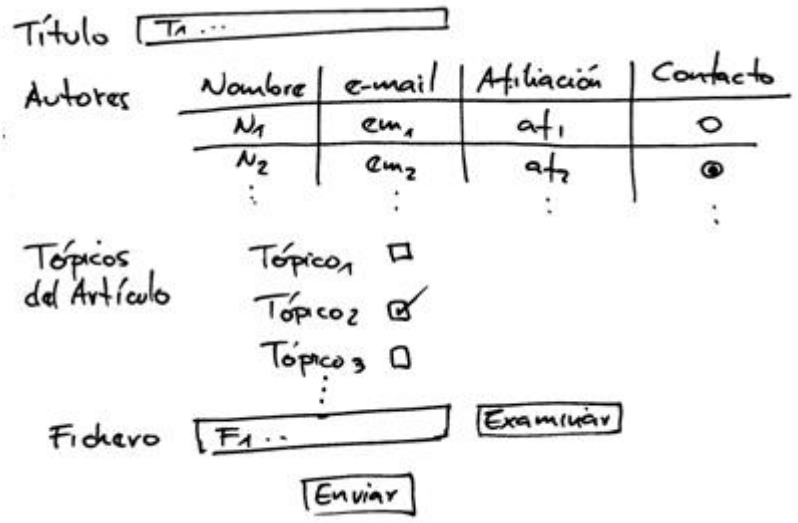
Se introducen los datos del artículo (título, fichero adjunto, resumen, tópicos) y de los autores (nombre, e-mail, afiliación). Uno de los autores debe indicarse como autor de contacto. El sistema confirma la correcta recepción del artículo enviando un e-mail al autor de contacto con un userid y password para que el autor pueda posteriormente acceder al artículo.</td></tr>
<tr><td colspan="3">Observaciones:</td></tr>
<tr><td colspan="3">Interfaz:
 
</td></tr>
</table>

Tabla 3 Historia de usuario

3.1.2 RUP

3.1.2.1 Consideraciones

¿Cómo podemos comparar procesos?

Los procesos se caracterizan por dos dimensiones, nivel de ceremonia y son en cascada o iterativos

- **Baja ceremonia/Alta ceremonia:** En el eje horizontal, la baja ceremonia produce un mínimo de documentación de soporte y posee pequeños formalismos en los procedimientos de trabajo; La alta ceremonia tiene una buena cantidad de documentación y una trasabilidad a través de varios artefactos, control de cambios y todo lo necesarios para proyectos que requieran ser muy rigurosos y predecibles.

- **Cascada/Iterativos:** Cascada es un ciclo de vida lineal con integración y pruebas tardías; El iterativo maneja riegos con una temprana implementación de la arquitectura, integración y pruebas.

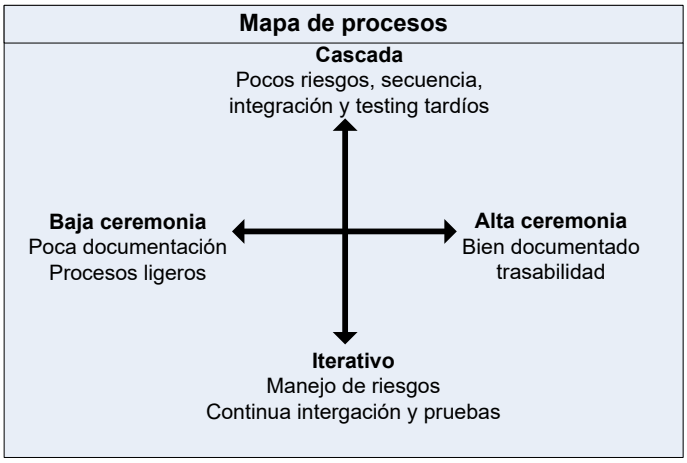


Ilustración 10 Mapa de procesos

Programación ágil: Baja ceremonia e iterativo

Los desarrollos ágiles como son XP y SCRUM, hacen un sacrificio en torno al nivel de ceremonia y rigor en pro de tener una alta flexibilidad y adaptabilidad en el negocio. Se le da una mayor prioridad en producir software y no en crear una extensa y detallada documentación. No se posee un plan rígido de desarrollo, se van haciendo cambios durante el proceso, aunque esto no quiere decir que no exista y que no sea importante

la documentación, esta es requerida para asegurar que el software se pueda hacer, adaptar los planes a la realidad y necesidades que hayan de cambio.

CMM y CMMI: Alta ceremonia, se procura tener todo predecible.

Cuando las compañías tienen una inversión considerable en proyectos de software y no pueden darse el lujo de resultados no predecibles que afecten la calidad, se debe tener una cantidad considerable de documentación que facilite y garantice un adecuado funcionamiento de un proyecto de software.

Las características de un proyecto de este tipo son:

- Los planes son muy detallados y hechos con cuidado.
- Hay muchos artefactos relacionados con la administración, requerimientos y pruebas.
- Se maneja control de versiones de artefactos de administración, requerimientos y pruebas.
- Hay una creación y mantenimiento de la trasabilidad entre los requerimientos, elementos de diseño y artefactos de prueba.
- Existe un mecanismo para manejar adecuadamente los controles de cambio.
- Los resultados de las inspecciones poseen seguimiento.

El enfoque de alta ceremonia es apropiado para sistemas de desarrollo complejo, equipos extensos y distribuidos. Este enfoque produce sistemas de calidad que son fáciles de mantener, pero el costo de producirlo es costoso y el tiempo para su ejecución es mayor que el enfoque de baja ceremonia.

CMM

En la búsqueda de tener mas predecibilidad y una alta calidad, las organizaciones necesitaron primero entender sus procesos que tan buenos eran y mirar hacia el futuro para tener unos procesos de alto nivel de madures. El modelo de capacidad y madurez diseñado por el instituto de ingeniería de software (SEI) , está diseñado para hacer eso, el CMM no es un proceso, es un framework de valoración usado para saber en la organización su nivel de los procesos en una escala de 1 a 5, CMM no dice como desarrollar software como tal, para esto está RUP, XP, etc.

CMM provee una guía detallada de lo que puede ser parte de un desarrollo de software o una adquisición para una organización madura. Se debe tener mucho cuidado de no tener un exceso de artefactos, lo cual haría los procesos pesados e ineficientes. Teniendo un enfoque fuerte en las revisiones en pares, inspecciones, actividades tradicionales para asegurar la calidad, CMM tiene un enfoque en cascada ya que no se obliga ha realizar una temprana integración y chequeo.

CMMI

El CMMI está más adecuado a las nuevas y mejores prácticas como es el manejo de riesgos y un desarrollo iterativo, más que dedicarse a crear más ceremonia, CMMI se

centra en enfocarse en áreas individuales de mejoramiento que ayuden a los objetivos de la organización en sus negocios y disminuya los riesgos.

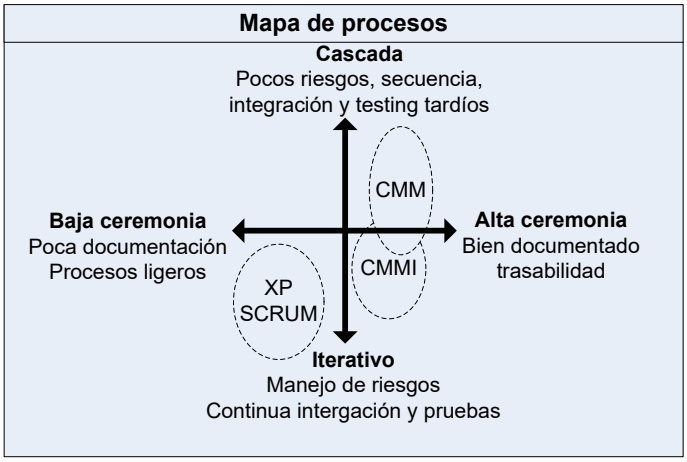


Ilustración 11 Mapa de rocesos CMM CMMI XP SCRUM

RUP: Un acercamiento iterativo con un nivel de ceremonia adaptable

Es un framework configurable que permite a los usuarios producir una gran variedad de procesos, las configuraciones de RUP permite adaptarse si ningún problema desde escalas de poca ceremonia hasta las que requieren alta ceremonia dependiendo de las necesidades del proyecto. RUP promueve ser fuertemente iterativo y maneja riesgos en el desarrollo haciendo frecuentemente integración y pruebas. Hay también alguna flexibilidad entre el eje de Cascada/iterativo y algunas compañías lo usan de la forma de cascada

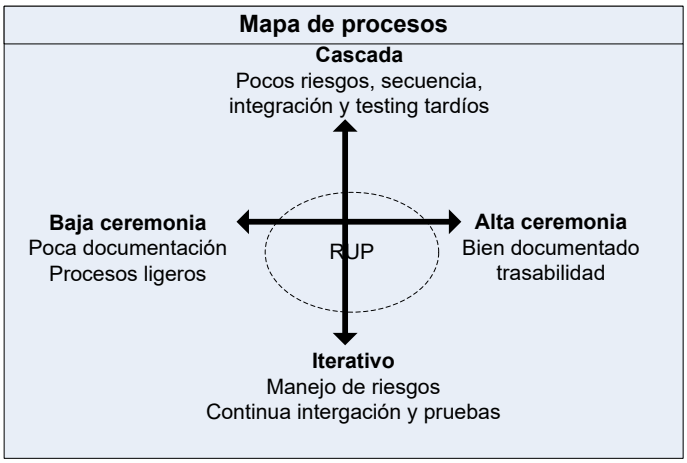


Ilustración 12 Mapa de procesos RUP

El framewok de RUP posee una vasta cantidad de guías, artefactos y roles, como no todos los proyectos van a necesitar todos estos artefactos, se necesita especificar un subjconjunto para usar en un proyecto. Esto es hecho configurando el RUP, el cual constituye unos procesos completos desde una perspectiva particular de los

requerimientos. También se pueden coger configuraciones pre-establecidas como punto de inicio o arrancar de ceros.

Un componente de proceso en RUP: Es un módulo del proceso que puede tener un nombre, ser empacado, intercambiado y ensamblado con otros componente de de proceso.

Una librería de RUP, es una colección de componentes, los cuales sus configuraciones son tratadas con el RUP builder y se pueden poner más procesos en la librería usando el mecanismo de plug-ins.

El proceso de configuración es realizado usando el RUP Builder. Este es comprado y viene con un número predefinido de configuraciones. También ofrece la posibilidad de crear configuraciones que se necesiten. Se puede elegir el nivel de formalidad con la que se puede trabajar, también se pueden usar plantillas que pueden ser muy completas o ligeras dependiendo de las necesidades.

Una vez se halla definido que plug-ins, que componentes de proceso y base de RUP se empleará el Rup Builder valida que todo este correcto y se hace una publicación web de la configuración.

Al tener una configuración del proceso RUP, partes de esta configuración no son del todo aplicables a administrador del proyecto, analista, arquitecto, desarrollador o tester, por esta razón dependiendo del rol y las responsabilidades se crean vistas de procesos las cuales serán usadas según el rol y responsabilidades de la persona en el proyecto.

3.1.2.2 “Definiciones de RUP

- El RUP es un acercamiento al desarrollo de software que es iterativo, centrado en la arquitectura y manejado con casos de uso. Rup describe una variedad de documentación y libros. La mayor información puede ser encontrada en el propio producto el cual tiene guías detalladas, ejemplos y plantillas que cubren completamente el ciclo de vida del software.

- El RUP es un proceso de ingeniería de software bien definido y bien estructurado, está claramente definido quien es responsable de que, como las cosas son hechas y cuando deben hacerse. El RUP provee una buena definición de la estructura del ciclo de vida en un proyecto con RUP, se identifican hitos importantes y puntos de decisión.

El proceso puede ser descrito en dos dimensiones o ejes:

Eje horizontal: Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en

términos de fases, iteraciones e hitos. Se puede observar en la Figura 8 que RUP consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Como se mencionó anteriormente cada fase se subdivide a la vez en iteraciones.

Eje vertical: Representa los aspectos estáticos del proceso. Describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

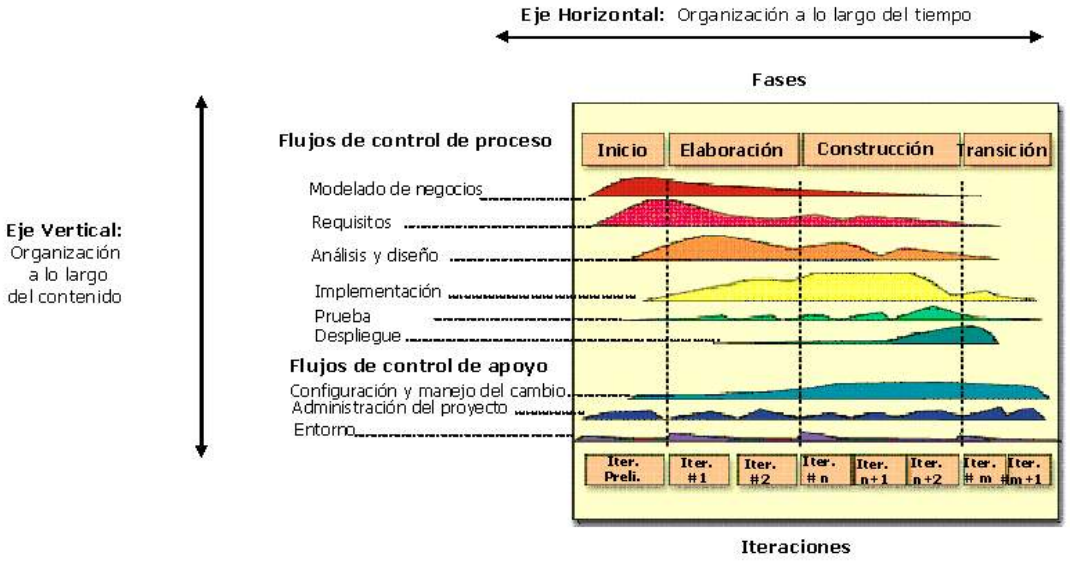


Ilustración 13 Estructura de RUP

3.1.2.3 Estructura Dinámica del proceso. Fases e iteraciones

RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un producto. Cada ciclo concluye con una generación del producto para los clientes. Cada ciclo consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase se subdivide a la vez en iteraciones, el número de iteraciones en cada fase es variable.

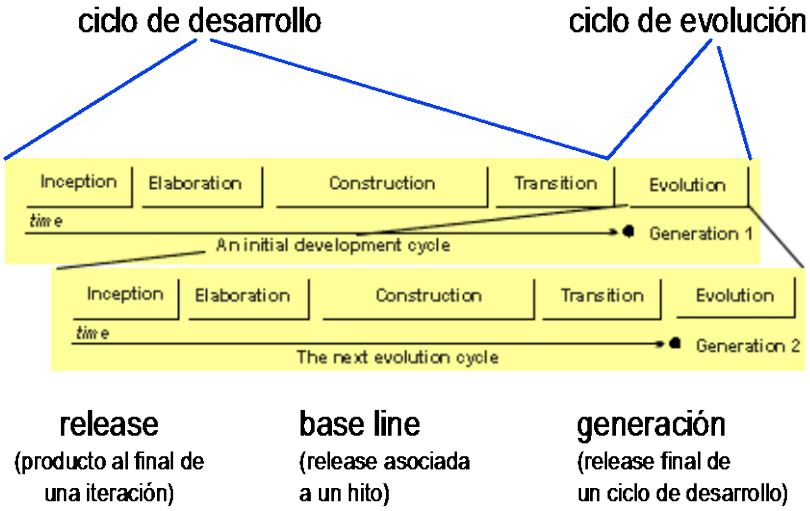


Ilustración 14 Ciclos, releases, baseline

Cada fase se concluye con un hito bien definido, un punto en el tiempo en el cual se deben tomar ciertas decisiones críticas y alcanzar las metas clave antes de pasar a la siguiente fase, ese hito principal de cada fase se compone de hitos menores que podrían ser los criterios aplicables a cada iteración. Los hitos para cada una de las fases son: Inicio - *Lifecycle Objectives,* Elaboración - *Lifecycle Architecture,* Construcción - *Initial Operational Capability,* Transición - Product Release. Las fases y sus respectivos hitos se ilustran en la Figura siguiente.

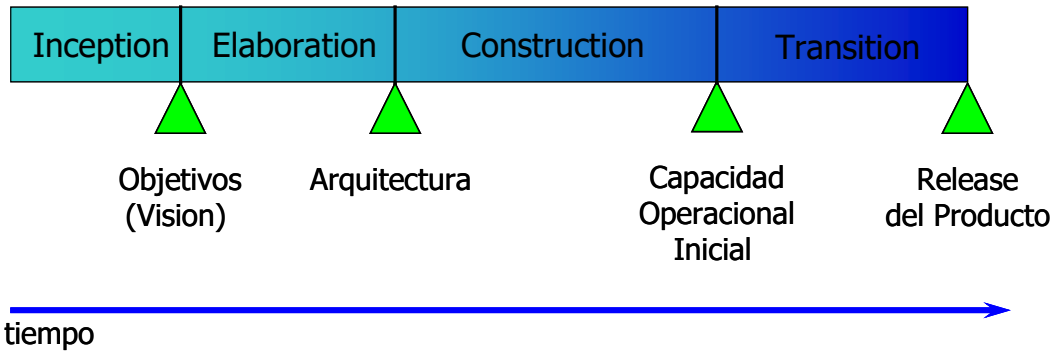


Ilustración 15 Fases e hitos en RUP

La duración y esfuerzo dedicado en cada fase es variable dependiendo de las características del proyecto. Sin embargo, la tabla 4 siguiente ilustra porcentajes frecuentes al respecto. Consecuente con el esfuerzo señalado, la ilustración 16 ilustra una distribución típica de recursos humanos necesarios a lo largo del proyecto.

	Inicio	Elaboración	Construcción	Transición
Esfuerzo	**5 %**	**20 %**	**65 %**	**10%**
Tiempo Dedicado	**10 %**	**30 %**	**50 %**	**10%**

Tabla 4 Distribución típica de esfuerzo y tiempo

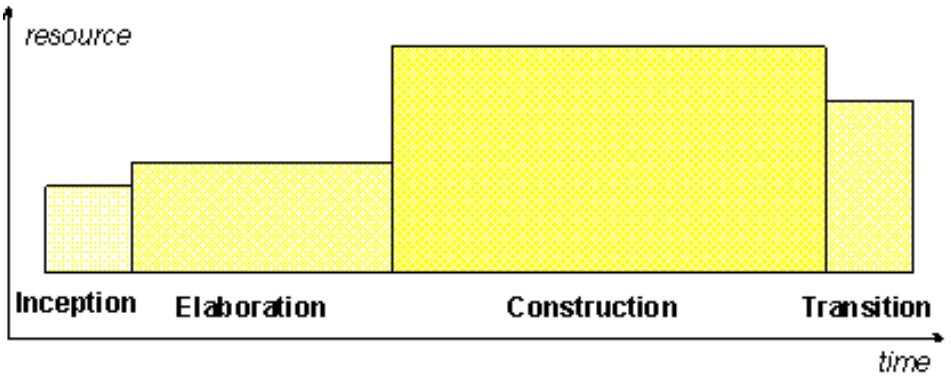


Ilustración 16Distribución típica de recursos humanos

Inicio

Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y Casos de Uso, y se diseñan los Casos de Uso más esenciales (aproximadamente el 20% del modelo completo). Se desarrolla, un plan de negocio para determinar que recursos deben ser asignados al proyecto.

Los objetivos de esta fase son:

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los Casos de Uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre.

Los resultados de la fase de inicio deben ser :

- Un documento de visión: Una visión general de los requerimientos del proyecto, características clave y restricciones principales.
- Modelo inicial de Casos de Uso (10-20% completado).
- Un glosario inicial: Terminología clave del dominio.
- El caso de negocio.
- Lista de riesgos y plan de contingencia.
- Plan del proyecto, mostrando fases e iteraciones.
- Modelo de negocio, si es necesario
- Prototipos exploratorios para probar conceptos o la arquitectura candidata.

Al terminar la fase de inicio se deben comprobar los criterios de evaluación para continuar:

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda.
- Entendimiento de los requisitos, como evidencia de la fidelidad de los Casos de Uso principales.
- Las estimaciones de tiempo, coste y riesgo son creíbles.
- Comprensión total de cualquier prototipo de la arquitectura desarrollado.
- Los gastos hasta el momento se asemejan a los planeados.

Si el proyecto no pasa estos criterios hay que plantearse abandonarlo o repensarlo profundamente.

Elaboración

El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos.

En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

Los objetivos de esta fase son:

- Definir, validar y cimentar la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede.
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Al terminar deben obtenerse los siguientes resultados :

- Un modelo de Casos de Uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados.
- Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un Caso de Uso específico.
- Descripción de la arquitectura software.
- Un prototipo ejecutable de la arquitectura.
- Lista de riesgos y caso de negocio revisados.
- Plan de desarrollo para el proyecto.
- Un caso de desarrollo actualizado que especifica el proceso a seguir.
- Un manual de usuario preliminar (opcional).

En esta fase se debe tratar de abarcar todo el proyecto con la profundidad mínima. Sólo se profundiza en los puntos críticos de la arquitectura o riesgos importantes.

En la fase de elaboración se actualizan todos los productos de la fase de inicio.

Los criterios de evaluación de esta fase son los siguientes:

- La visión del producto es estable.
- La arquitectura es estable.
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles.
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual.

- Los gastos hasta ahora son aceptables, comparados con los previstos.

Si no se superan los criterios de evaluación quizá sea necesario abandonar el proyecto o replanteárselo considerablemente.

Construcción

La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto.

Los objetivos concretos según incluyen:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea práctico.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea práctico.

Los resultados de la fase de construcción deben ser :□□

- Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación)
- Arquitectura íntegra (mantenida y mínimamente actualizada)
- Riesgos Presentados Mitigados
- Plan del Proyecto para la fase de Transición.
- Manual Inicial de Usuario (con suficiente detalle)
- Prototipo Operacional – beta
- Caso del Negocio Actualizado

Los criterios de evaluación de esta fase son los siguientes:

- El producto es estable y maduro como para ser entregado a la comunidad de usuario para ser probado.
- Todos los usuarios expertos están listos para la transición en la comunidad de usuarios.
- Son aceptables los gastos actuales versus los gastos planeados.

Transición

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

Algunas de las cosas que puede incluir esta fase:

- Prueba de la versión Beta para validar el nuevo sistema frente a las expectativas de los usuarios
- Funcionamiento paralelo con los sistemas legados que están siendo sustituidos

por nuestro proyecto.
- Conversión de las bases de datos operacionales.
- Entrenamiento de los usuarios y técnicos de mantenimiento.
- Traspaso del producto a los equipos de marketing, distribución y venta.

Los principales objetivos de esta fase son:
- Conseguir que el usuario se valga por si mismo.
- Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Los resultados de la fase de transición son :
- Prototipo Operacional
- Documentos Legales
- Caso del Negocio Completo
- Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la Arquitectura completa y corregida
- Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

Los criterios de evaluación de esta fase son los siguientes:
- El usuario se encuentra satisfecho.
- Son aceptables los gastos actuales versus los gastos planificados.

Estructura Estática del proceso. Roles, actividades, artefactos y flujos de trabajo

Un proceso de desarrollo de software define quién hace qué, cómo y cuándo. RUP define cuatro elementos los roles, que responden a la pregunta ¿Quién?, las actividades que responden a la pregunta ¿Cómo?, los productos, que responden a la pregunta ¿Qué? y los flujos de trabajo de las disciplinas que responde a la pregunta ¿Cuándo? (ver Ilustración 17 y 18) .

Ilustración 17 Relación entre roles, actividades, artefactos

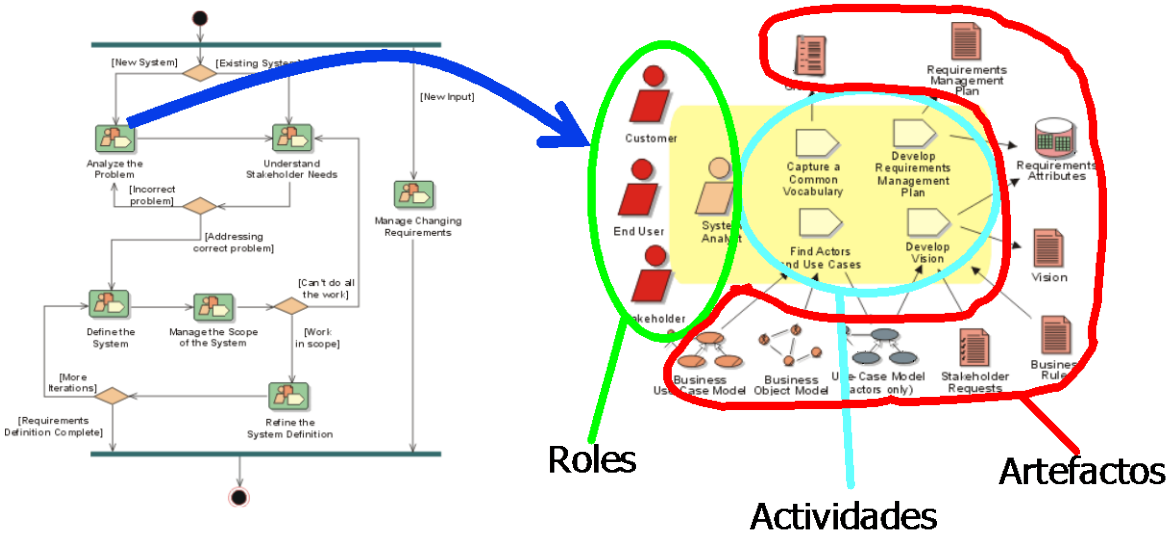


Ilustración 18 Detalle de un workflow mediante roles, actividades y artefactos

3.1.2.4 Roles

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Una persona puede desempeñar diversos roles, así como un mismo rol puede ser representado por varias personas.

Las responsabilidades de un rol son tanto el llevar a cabo un conjunto de actividades como el ser el dueño de un conjunto de artefactos.

RUP define grupos de roles, agrupados por participación en actividades relacionadas. Estos grupos son :

Analistas:

- Analista de procesos de negocio.
- Diseñador del negocio.
- Analista de sistema.
- Especificador de requisitos.

Desarrolladores:

- Arquitecto de software.
- Diseñador
- Diseñador de interfaz de usuario
- Diseñador de cápsulas.
- Diseñador de base de datos.
- Implementador.
- Integrador.

Gestores:

- Jefe de proyecto
- Jefe de control de cambios.
- Jefe de configuración.
- Jefe de pruebas
- Jefe de despliegue
- Ingeniero de procesos
- Revisor de gestión del proyecto
- Gestor de pruebas.

Apoyo:
- Documentador técnico
- Administrador de sistema
- Especialista en herramientas
- Desarrollador de cursos
- Artista gráfico

Especialista en pruebas:
- Especialista en Pruebas (*tester*)
- Analista de pruebas
- Diseñador de pruebas

Otros roles:
- *Stakeholders*.
- Revisor
- Coordinación de revisiones
- Revisor técnico
- Cualquier rol

3.1.2.5 Actividades

Una actividad en concreto es una unidad de trabajo que una persona que desempeñe un rol puede ser solicitado a que realice. Las actividades tienen un objetivo concreto, normalmente expresado en términos de crear o actualizar algún producto.

3.1.2.6 Artefactos

Un producto o artefacto es un trozo de información que es producido, modificado o usado durante el proceso de desarrollo de software. Los productos son los resultados tangibles del proyecto, las cosas que va creando y usando hasta obtener el producto final.

Un artefacto puede ser cualquiera de los siguientes:

- Un documento, como el documento de la arquitectura del software.
- Un modelo, como el modelo de Casos de Uso o el modelo de diseño.

- Un elemento del modelo, un elemento que pertenece a un modelo como una clase, un Caso de Uso o un subsistema.

3.1.2.7 Flujos de trabajo

Con la enumeración de roles, actividades y artefactos no se define un proceso. Necesitamos contar con una secuencia de actividades realizadas por los diferentes roles, así como la relación entre los mismos. Un flujo de trabajo es una relación de actividades que nos producen unos resultados observables. A continuación se dará una explicación de cada flujo de trabajo.

3.1.2.8 Modelado del negocio

Con este flujo de trabajo pretendemos llegar a un mejor entendimiento de la organización donde se va a implantar el producto.

Los objetivos del modelado de negocio son:

- Entender la estructura y la dinámica de la organización para la cual el sistema va ser desarrollado (organización objetivo).
- Entender el problema actual en la organización objetivo e identificar potenciales mejoras.
- Asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de la organización objetivo.
- Derivar los requisitos del sistema necesarios para apoyar a la organización objetivo.

Para lograr estos objetivos, el modelo de negocio describe cómo desarrollar una visión de la nueva organización. Basado en esta visión se definen procesos, roles y responsabilidades de la organización por medio de un modelo de Casos de Uso del negocio y un Modelo de Objetos del Negocio. Complementario a estos modelos, se desarrollan otras especificaciones tales como un Glosario.

3.1.2.9 Requisitos

Este es uno de los flujos de trabajo más importantes, porque en él se establece qué tiene que hacer exactamente el sistema que construyamos. En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifiquemos.

Los objetivos del flujo de datos Requisitos son:

- Establecer y mantener un acuerdo entre clientes y otros *stakeholders* sobre lo que el sistema podría hacer.
- Proveer a los desarrolladores un mejor entendimiento de los requisitos del sistema.
- Definir el ámbito del sistema.

- Proveer una base para la planeación de los contenidos técnicos de las iteraciones.
- Proveer una base para estimar costos y tiempo de desarrollo del sistema.
- Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.

Los requisitos se dividen en dos grupos. Los requisitos funcionales representan la funcionalidad del sistema. Se modelan mediante diagramas de Casos de Uso. Los requisitos no funcionales representan aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica. Por ejemplo requisitos de facilidad de uso, fiabilidad, eficiencia, portabilidad, etc.

Para capturar los requisitos es preciso entrevistar a todos los interesados en el proyecto, no sólo a los usuarios finales, y anotar todas sus peticiones. A partir de ellas hay que descubrir lo que necesitan y expresarlo en forma de requisitos.

En este flujo de trabajo, y como parte de los requisitos de facilidad de uso, se diseña la interfaz gráfica de usuario. Para ello habitualmente se construyen prototipos de la interfaz gráfica de usuario que se contrastan con el usuario final.

3.1.2.10 Análisis y Diseño

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema.

Los objetivos del análisis y diseño son :
- Transformar los requisitos al diseño del futuro sistema.
- Desarrollar una arquitectura para el sistema.
- Adaptar el diseño para que sea consistente con el entorno de implementación, diseñando para el rendimiento.

El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos.

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los Casos de Uso con las interacciones de las clases de análisis. Durante la fase de elaboración se va refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes. Además si el sistema usará una base de datos, habrá que diseñarla también, obteniendo un modelo de datos.

El resultado final más importante de este flujo de trabajo será el modelo de diseño. Consiste en colaboraciones de clases, que pueden ser agregadas en paquetes y subsistemas.

Otro producto importante de este flujo es la documentación de la arquitectura de software, que captura varias vistas arquitectónicas del sistema.

3.1.2.11 Implementación

En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. Además se deben hacer las pruebas de unidad: cada implementador es responsable de probar las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable.

En cada iteración habrá que hacer lo siguiente:

- Planificar qué subsistemas deben ser implementados y en que orden deben ser integrados, formando el Plan de Integración.
- Cada implementador decide en que orden implementa los elementos del subsistema.
- Si encuentra errores de diseño, los notifica.
- Se prueban los subsistemas individualmente.
- Se integra el sistema siguiendo el plan.

La estructura de todos los elementos implementados forma el modelo de implementación. La integración debe ser incremental, es decir, en cada momento sólo se añade un elemento. De este modo es más fácil localizar fallos y los componentes se prueban más a fondo. En fases tempranas del proceso se pueden implementar prototipos para reducir el riesgo. Su utilidad puede ir desde ver si el sistema es viable desde el principio, probar tecnologías o diseñar la interfaz de usuario. Los prototipos pueden ser exploratorios (desechables) o evolutivos. Estos últimos llegan a transformarse en el sistema final.

3.1.2.12 Pruebas

Este flujo de trabajo es el encargado de evaluar la calidad del producto que estamos desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

Esta disciplina brinda soporte a las otras disciplinas. Sus objetivos son:

- Encontrar y documentar defectos en la calidad del software.
- Generalmente asesora sobre la calidad del software percibida.
- Provee la validación de los supuestos realizados en el diseño y especificación de requisitos por medio de demostraciones concretas.
- Verificar las funciones del producto de software según lo diseñado.
- Verificar que los requisitos tengan su apropiada implementación.

Las actividades de este flujo comienzan pronto en el proyecto con el plan de prueba (el cual contiene información sobre los objetivos generales y específicos de las prueba en el proyecto, así como las estrategias y recursos con que se dotará a esta tarea), o incluso antes con alguna evaluación durante la fase de inicio, y continuará durante todo el proyecto.

El desarrollo del flujo de trabajo consistirá en planificar que es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida nos sirva para ir refinando el producto a desarrollar.

3.1.2.13 Despliegue

El objetivo de este flujo de trabajo es producir con éxito distribuciones del producto y distribuirlo a los usuarios. Las actividades implicadas incluyen:

- Probar el producto en su entorno de ejecución final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el software.
- Proveer asistencia y ayuda a los usuarios.
- Formar a los usuarios y al cuerpo de ventas.
- Migrar el software existente o convertir bases de datos.

Este flujo de trabajo se desarrolla con mayor intensidad en la fase de transición, ya que el propósito del flujo es asegurar una aceptación y adaptación sin complicaciones del software por parte de los usuarios. Su ejecución inicia en fases anteriores, para preparar el camino, sobre todo con actividades de planificación, en la elaboración del manual de usuario y tutoriales.

3.1.2.14 Gestión del proyecto

La Gestión del proyecto es el arte de lograr un balance al gestionar objetivos, riesgos y restricciones para desarrollar un producto que sea acorde a los requisitos de los clientes y los usuarios.

Los objetivos de este flujo de trabajo son:

- Proveer un marco de trabajo para la gestión de proyectos de software intensivos.
- Proveer guías prácticas realizar planeación, contratar personal, ejecutar y monitorear el proyecto.
- Proveer un marco de trabajo para gestionar riesgos.

La planeación de un proyecto posee dos niveles de abstracción: un plan para las fases y un plan para cada iteración.

3.1.2.15 Configuración y control de cambios

La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido.

3.1.2.16 Entorno

La finalidad de este flujo de trabajo es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento, así como definir la instancia concreta del proceso que se va a seguir.

En concreto las responsabilidades de este flujo de trabajo incluyen:

- Selección y adquisición de herramientas
- Establecer y configurar las herramientas para que se ajusten a la organización.
- Configuración del proceso.
- Mejora del proceso.
- Servicios técnicos.

El principal artefacto que se usa en este flujo de trabajo es el *caso de desarrollo* que especifica para el proyecto actual en concreto, como se aplicará el proceso, que productos se van a utilizar y como van a ser utilizados. Además se tendrán que definir las guías para los distintos aspectos del proceso, como pueden ser el modelado del negocio y los Casos de Uso, para la interfaz de usuario, el diseño, la programación, el manual de usuario."[12]

[12] Introducción a RUP [sitio en Internet], disponible en https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc Último acceso: Noviembre 20 de 2007.

3.1.3 Consideraciones metodología ASAP

La metodología ASAP, al verla detenidamente nos da la impresión que sus procesos van dirigidos estrictamente a un ciclo tradicional de cascada o lineal.

Para la mayoría de gente que haya participado en proyectos tradicionales de software, en donde se tiene que construir todo el sistema desde cero, claramente propondrían un esquema iterativo como lo proponen en la actualidad los paradigmas de la programación ágil o RUP.

La plataforma SAP ECC, es un sistema altamente modular y configurable, en el cual sólo se programa la parte respectiva a las personalizaciones de la plataforma. Para estas personalizaciones ya se tiene una plataforma tanto de hardware y software. SAP ECC provee al programador una arquitectura y un modelo de datos suficientemente completa como para llevar a cabo las tareas de personalización en el sistema y es por esto que la metodología ASAP se centra en la captura de los procesos del negocio y trasladarlos a la plataforma SAP ECC con sus módulos básicamente.

3.1.3.1 Ciclo de vida cascada

En Ingeniería de software el **desarrollo en cascada**, también llamado **modelo en cascada**, es el enfoque metodológico que ordena rigurosamente las etapas del **ciclo de vida** del software, de forma tal que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

Un ejemplo de una metodología de desarrollo en cascada es:

1. Análisis de requisitos
2. Diseño del Sistema
3. Diseño del Programa
4. Codificación
5. Pruebas
6. Implantación
7. Mantenimiento

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costes del desarrollo. La palabra *cascada* sugiere, mediante la metáfora de la fuerza de la gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto.

Si bien ha sido ampliamente criticado desde el ámbito académico y la industria, sigue siendo el paradigma más seguido al día de hoy.

Fases del modelo.

Análisis de requisitos

Se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos.

Es importante señalar que en esta etapa se deben consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

Diseño del Sistema

Se descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

Diseño del Programa

Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario así como también los análisis necesarios para saber que herramientas usar en la etapa de Codificación.

Codificación

Es la fase de programación propiamente dicha. Aquí se desarrolla el código fuente, haciendo uso de prototipos así como pruebas y ensayos para corregir errores.
Dependiendo del lenguaje de programación y su versión se crean las librerías y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

Pruebas

Los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente antes de ser puesto en explotación.

Implantación

El software obtenido se pone en producción. Se implementan los niveles software y hardware que componen el proyecto. La implantación es la fase con más duración y con más cambios en el ciclo de elaboración de un proyecto. Es una de las fases finales del proyecto.
Durante la explotación del sistema software pueden surgir cambios, bien para corregir errores o bien para introducir mejoras. Todo ello se recoge en los Documentos de Cambios.

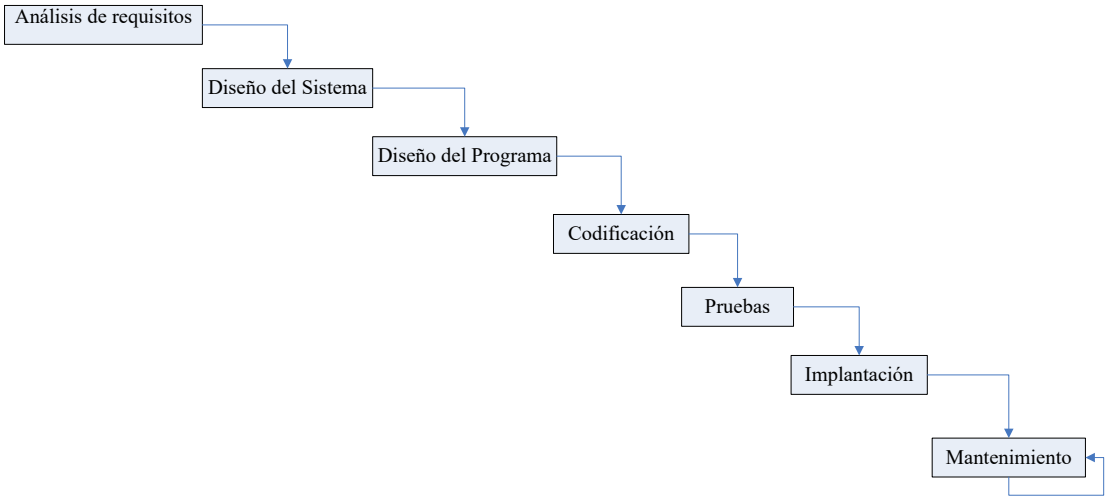


Ilustración 19 Ciclo de vida en cascada

3.1.3.2 La metodología de implementación Accelerated SAP (ASAP)[13]

La metodología de implementación ASAP fue diseñada por SAP para ayudar a sus Clientes y Partners a realizar una implementación rápida y de bajo costo. La metodología ASAP permite:

- Minimizar el tiempo requerido entre la instalación y el arranque productivo.
- Maximizar la utilización de los recursos de los clientes y de los partners.
- Incorporar un esquema de entrenamiento orientado a los procesos.
- Obtener resultados en un "modelo de procedimientos" que puede ser usado en otras implantaciones en el cliente.
- Involucrar y lograr una pronta aceptación del sistema por parte de la comunidad de usuarios.

La metodología ASAP contempla las siguientes fases y entregables:

---

13 La metodología de implementación Accelerated SAP (ASAP) [sitio en Internet], disponible en http://www.sap.com/argentina/partners/howtochoose/index.epx Último acceso: Noviembre 20 de 2007.
Fuente principal: Brand Hartwing. Implementación técnica mediamte ASAP España; Gestión 2000, 2001.

- **Preparación del Proyecto**
  Diseño del proyecto (equipo de trabajo, tareas, recursos, tiempo, metodología)
- **Planos del Negocio**
  Definición de Estructuras Organizativas
  Definición de Procesos Funcionales
  Definición de Datos Maestros
  Migración de datos
  Recopilación de información faltante
  Documento Plano de Negocios
- **Realización**
  Parametrización del Sistema
  Sistema configurado y probado
- **Preparación Final**
  Usuarios entrenados y sistema productivo preparado
- **Arranque Productivo y Soporte**
  Ingreso en Productivo
  Temas pendientes cerrados

Ilustración 20 Fases del ASAP

FASE I: Preparación del Proyecto

Durante esta fase, el equipo del proyecto se entrenará en los fundamentos de mySAP ECC y en el mapa de procedimientos de ASAP, se completará el plan del proyecto de alto nivel y se revisará el esquema del hardware necesario. El proyecto se iniciará oficialmente con una reunión de lanzamiento. Esta reunión no es exclusiva del equipo del proyecto y los consultores del implantador, sino que también para otros miembros clave de la empresa. Este arranque preparará un escenario propicio para el proyecto destacando la importancia de éste con los objetivos futuros de la compañía.

Debido a los grandes costos de estos proyectos en la reunión de lanzamiento debe participar una empresa interventora que garantice los aspectos legales entre las partes y garantice que el proyecto posee unos alcances coherentes y claros.

FASE II: Planos del Negocio (Business Blueprint)

El propósito de la fase de Planos del Negocio (Business Blueprint) es entender las metas del cliente y determinar los procesos de negocio necesarios para cumplir las mismas. En una reunión de Revisión Ejecutiva, se discutirán las metas del grupo, la estructura organizacional y los procesos de negocio de alto nivel. Otros requerimientos más detallados serán discutidos durante reuniones de trabajo de revisión de cada uno de los procesos de negocio. Las reuniones de Revisión Ejecutiva y de los procesos de negocio serán conducidas por los consultores del implantador.

Las reuniones del levantamiento de procesos de la empresa son ejecutadas por los funcionales respectivos del área de proceso y los usuarios líderes que son los expertos del negocio y conocen las singularidades del proceso, allí también es de vital importancia definir las interfaces con sistemas no SAP.

Para verificar que se entendieron apropiadamente los requerimientos del grupo y que se incluyó a todos los involucrados en el proyecto, se preparará un "Plano" del estado futuro y será presentado a sus ejecutivos para la aprobación del mismo. Este Plano consistirá en un diagrama de la estructura de la empresa, además del primer borrador de la definición de los procesos de negocio que se utilizará en su compañía; los dos se presentarán en forma escrita y de diagrama. Con la elaboración de los Planos se finalizará el alcance detallado del proyecto.

FASE III: Realización

Durante esta fase, el equipo del proyecto de su empresa (usuarios líderes) y los consultores del implantador se separarán para terminar las actividades asignadas. El equipo del proyecto asistirá al entrenamiento de SAP de nivel 2 y 3[14] El entrenamiento de SAP se ha organizado alrededor de procesos de negocios. Durante el entrenamiento de nivel 2, el equipo del proyecto se hará más competente, y comenzará a modelar al sistema mySAP ERP con base en los requerimientos de una compañía caso. El entrenamiento, también, proporcionará un entendimiento de las herramientas y ayudas de referencia del sistema; de igual manera, se realizará la integración de sus componentes. El entrenamiento de nivel 3 ayudará al equipo del proyecto a adquirir conocimientos en tópicos detallados dentro de los procesos de negocio.

Mientras que el equipo del proyecto está en entrenamiento, los consultores del implantador configurarán los procesos de negocio definidos en los "planos aprobados". El sistema configurado reflejará la organización del cliente y los catálogos maestros; y deberá soportar un flujo totalmente integrado de los procesos del sistema. Una revisión de los procesos de negocio de su empresa con el equipo del proyecto y con otros usuarios clave de cada uno de los procesos de negocio permitirá la retroalimentación y confirmación de los "planos aprobados".

---

[14] Nivel 1: Cursos de uno a dos días para introducirse en la tecnología de SAP ECC.
Nivel 2: Cursos de tres a cinco días como primera especialización de un área concreta.
Nivel 3: Cursos de tres a cinco días para profundizar en un área de especialidad, a modo de ampliación de un curso de nivel 2.

Un sistema que refleje los catálogos maestros y la organización de su empresa proporcionará un beneficio adicional al equipo del proyecto en el refuerzo del entrenamiento tomado.

La configuración de cada proceso de negocio modular será dividida en interacciones o ciclos de flujos de procesos de negocios relacionados. Los flujos de procesos de negocios son configurados conjuntamente con el desarrollo de reportes, formularios (documentos legales de la empresa como son facturas, recibos de caja, retención del IVA, retención en la fuente, ICA, etc), procedimientos de usuarios, escenarios de prueba (pruebas unitarias e integrales), migración de datos, interfaces (comunicación sistemas SAP no SAP) y perfiles de seguridad, basado en las necesidades globales de la empresa y los requerimientos de los usuarios líderes. Los ciclos no sólo proporcionan indicadores para el equipo del proyecto, sino que también proveen puntos claves para probar y simular partes específicas del proceso global de negocios. Este enfoque proporciona retroalimentación inmediata, así como el involucramiento de toda la organización a lo largo del ciclo de vida del proyecto.

Durante los ciclos, los usuarios líderes del cliente estará trabajando estrechamente con los consultores del implantador para definir los escenarios específicos de negocios y las condiciones de excepción. Este enfoque cuenta con la máxima transferencia de conocimientos permitiendo al equipo de trabajo repetir la configuración de los procesos medulares del negocio mientras pone a punto el sistema para tomar en cuenta procesos comunes de negocios. El equipo del proyecto del cliente completará el entrenamiento detallado de Nivel 3 durante los ciclos.

Como una actividad paralela, son desarrollados y probados los programas de interface y conversión, así como los reportes especiales.

FASE IV: Preparación Final

El propósito básico de la fase de Preparación Final es terminar las pruebas finales del sistema, entrenar a los usuarios finales y llevar los datos y el sistema a un ambiente productivo. Las pruebas finales al sistema consisten en probar los procedimientos y programas de conversión y reportes especiales para fines legales y fiscales, probar los programas de interface a los sistemas actuales, llevar a cabo las pruebas de volumen y estrés, así como las pruebas de aceptación del usuario final.

Para entrenar a los usuarios finales, el equipo de proyecto entrenará usuarios clave utilizando un método de “entrenar al entrenador”. Este método ayudará a ganar la aceptación de los usuarios finales, así como a la construcción de una base de conocimiento para soporte propio de los reportes en línea y futuras mejoras al sistema.

Otro propósito de esta fase es crear una estrategia para la Puesta en Marcha. Este plan específicamente identifica la estrategia de conversión de datos, procedimientos iniciales de auditoría y una estructura de soporte al equipo del proyecto.

El último paso en esta fase es aprobar el sistema y asegurar que el cliente esté listo para la puesta en marcha del Sistema mySAP ERP.

FASE V: Arranque Productivo y Soporte.

Inmediatamente después de la puesta en marcha, el sistema deberá ser revisado y afinado para asegurar que el entorno del negocio está completamente soportado. Este proceso involucra no solamente el verificar la precisión de las transacciones del negocio, sino también, entrevistar informalmente a los usuarios para verificar que sus necesidades hayan sido satisfechas.

El último paso en el proceso involucra la medición de los beneficios que brinda el nuevo sistema al negocio.

4 CONSIDERACIONES PARA REALIZAR EXITOSAMENTE LAS PERSONALIZACIONES

4.1 Roles

Desarrollador líder

Conocimientos y destrezas:

- Alto nivel y destreza en ABAP (estructurado y orientado a objetos).
- Conocimiento medio de los principales módulos de SAP.
- Conocimiento medio de la arquitectura SAP.
- Nivel alto con metodologías y técnicas de investigación de alto nivel, para planificar, ejecutar, administrar y evaluar proyectos informáticos.
- Habilidades para promulgar la comunicación y resolución de conflictos.
- Ser capaz de trabajar en equipo.

Actividades:

- En la fase 2 (Planos del negocio), trabajando en equipo, se debe planear con el líder del proyecto de la empresa que esta montando la implantación y con el gerente del proyecto (persona por el lado de la empresa en la que están montando la implantación), la estrategia de salida en vivo, según tengan previstas las necesidades de la organización y el riego que estén dispuesto a correr.

- En la fase 2 (Planos del negocio), cuando se hacen las especificaciones de las personalizaciones, para poder hacer una aproximación del tiempo de los desarrollos, debe usar los punto de función.

- En la fase 2 (Planos del negocio), Según la estrategia de salida en vivo y duración de las actividades, se debe organizar un cronograma de los desarrollos, para que estén acordes al proyecto en general y poder saber de una forma coherente cuantos recursos ABAP se necesitan.

- En todas las fases del proyecto, debe manejar junto con todas las personas involucradas en el proyecto la parte de riesgos, tanto en su parte de identificación, probabilidad de ocurrencia, impacto, estrategia de mitigación y seguimiento.

- En la fase 3 (realización), debe supervisar el adecuado desarrollo de las personalizaciones, que vayan de acuerdo a lo planeado y en caso contrario establecer las medidas adecuadas para que sea así.

- En la fase 3, 4 y 5 debe supervisar el adecuado control de cambios a las personalizaciones, si un desarrollador determina un impacto muy grande en una personalización ya hecha, se deben tomar las medidas correspondientes para no

atrasar el cronograma en general, si la personalización de todos modos todavía no ha sido realizada se debe medir el impacto y como afecta el cronograma de actividades.

- En la fase 4(Preparación Final), se supervisa que las entregas de los desarrollos sean satisfactorias, habiendo aprobado las pruebas unitarias, integrales y están listas para pasar a productivo.

- En la fase 5(Salida en vivo), se supervisa que los desarrollos estén ejecutando correctamente.

Desarrollador

Conocimientos y destrezas:

- Alto nivel y destreza en ABAP (estructurado y orientado a objetos).
- Conocimiento medio de los principales módulos de SAP.
- Actitud de promulga la comunicación.
- Ser capaz de trabajar en equipo.

Actividades:

- En la fase 2 (Planos del negocio), como la cantidad de personalizaciones usualmente en un proyecto es considerable, 1 o 2 desarrolladores deben colaborar en la tarea de estimación por puntos de función al desarrollador líder.

- En la fase 3 (Realización), deben desarrollar las personalizaciones, llenando los informes respectivos a los avances de estos.

- En la fase 3, 4 y 5, deben analizar el impacto que se reporte de los cambios hechos a las personalizaciones y notificar esto al desarrollador líder, que es el encargado de dar el aval o tomar las medidas indicadas con respecto a la petición de cambio.

Funcional:

Conocimientos y destrezas:

- Alto conocimiento del área de negocio a tratar.
- Alto nivel de conocimiento del módulo a configurar.
- Actitud de promulga la comunicación.
- Ser capaz de trabajar en equipo

Actividades:

- En la fase 2 junto con el usuario líder deben hacer un levantamiento de los procesos de la organización respectivos al módulo a tratar y sus relaciones con los otros procesos de la empresa.

- En la fase 2 debe hacer un mapeo de los proceso de la organización a programas de SAP ECC que cubran estas necesidades y ver las interrelaciones con los otros módulos de SAP ECC. Estos mapeos deben supervisados y abalados por el usuario líder.

- En la fase 2 una vez mapeados los procesos de la empresa a diferentes programas de SAP ECC, quedan detectados exactamente que funcionalidades de estos programas son requeridos por reglas del negocio y la plataforma estándar no los cubre, justamente estas funcionalidades deben ser especificadas para que los desarrolladores las realicen.

- En la fase 3 ya deben empezar a realizar las configuraciones, y pruebas unitarias e integrales de las personalizaciones.

- En la fase 4, se terminan las pruebas. se hacen pruebas de estrés para evaluar el comportamiento y se entrenan a los usuarios que capacitarán al resto del personal que vaya a usar el sistema.

- En la fase 3, 4 y 5 pueden existir solicitudes de cambios, estas deben ser manifestadas y comunicadas al desarrollador que haya realizado el desarrollo, o se le notifican al desarrollador líder.

- En la fase 5, se hace un seguimiento del sistema en producción y se afina el sistema.

Usuario líder:

Conocimientos y destrezas:

- Alto conocimiento del área de negocio a tratar.
- Actitud de promulga la comunicación.
- Ser capaz de trabajar en equipo.
- Ser un facilitador para promulgar las nuevas funcionalidades de las herramientas desarrolladas.

Actividades:

- En la fase 2 junto con el funcional respectivo deben hacer un levantamiento de los procesos de la organización respectivos al módulo a tratar y sus relaciones con los otros procesos de la empresa.

- En la fase 2 debe supervisar que el mapeo de los procesos de la organización a programas de SAP ECC llevado por el funcional, sea adecuado y cubra también las interrelaciones de estos procesos con otros de la organización.

- En la fase 2 una vez mapeados los procesos de la empresa a diferentes programas de SAP ECC, quedan detectados exactamente que funcionalidades de estos programas son requeridos por reglas del negocio y la plataforma estándar no los cubre, justamente estas funcionalidades entre el funcional y el usuario líder deben ser especificadas para que los desarrolladores las realicen.

- En la fase 3 ya se debe empezar a realizar las configuraciones efectuadas por el funcional y supervisadas por el usuario líder.

- En la fase 3 cuando el desarrollador entrega las personalizaciones, el usuario líder debe realizar las pruebas unitarias e integrales para garantizar y constatar que se realizo lo que se estaba solicitando.

- En la fase 4, se terminan las pruebas, y determina con que datos se hacen pruebas de estrés para evaluar el comportamiento.

- En la fase 3, 4 y 5 pueden existir solicitudes de cambios, estas deben ser manifestadas y comunicadas al desarrollador que haya realizado el desarrollo, o se le notifican al desarrollador líder.

- En la fase 5, se hace un seguimiento del sistema en producción y supervisa el afinamiento del sistema.

4.2 Manejo de riesgos[15]

El SEI (*Software Engineering Institute*) define al riesgo como "*la posibilidad de sufrir una pérdida*" [SEI, 2004] y a la **Administración de Riesgos** como "*el proceso formal en el que los factores de riesgos son sistemáticamente identificados, evaluados y mitigados*"

El análisis y gestión del riesgo son una serie de pasos que ayudan a un equipo de software a comprender y manejar la incertidumbre. El primer paso es reconocer que en el proyecto algo puede salir mal. A esto se le denomina identificación del riesgo, segundo paso consiste en determinar la probabilidad de que ocurrirá y el daño que causará si en efecto ocurre, el tercer paso es desarrollar un plan para gestionar aquellos riesgos con gran probabilidad y alto impacto. El cuarto paso consiste en revisar la evolución constante del riego y finalmente tomar las medidas de control necesarias para evitarlo o mitigarlo. Este trabajo lo realizan todos los involucrados en el proceso de software

Ilustración 21 Modelo de Administración de Riesgos[16]

Identificación de riesgos

Va encaminado a especificar las amenazas al plan del proyecto, este es un proceso iterativo ya que se pueden ir descubriendo nuevos riesgos a medida que el proyecto va avanzando.

---

[15] Fuente Pincipal: Pressman Roger S. Ingeniería de software Un enfoque práctico Sexta edición . Cap gestión del riego México;Mc Graw Hill, 2006

[16] Tomada del anteproyecto: Identificación de riesgos de proyectos de software en base a taxonomias Proyecto de maestria de Lic. Sebastián D. Maniasi *ITBA – UPM*

Es un criterio proactivo que busca identificar posibles factores de riesgo y tomar medidas de aseguramiento o planes de contingencia para contrarrestarlos a ellos y a sus efectos.

Existen métodos para identificar riesgos uno de ellos es el de crear una lista de verificación de riesgos.

Con esta lista podemos identificar riesgos y enfocarnos en algún subconjunto de riesgos sean conocidos y predecibles.

Ejemplo:

- Tamaño del producto
- Impacto del negocio
- Características del cliente
- Definición del proceso
- Entorno de desarrollo
- Tecnología que construir
- Tamaño y experiencia de la plantilla de personal

El gestor de proyectos da un primer paso para evitarlos cuando es posible y a controlarlos cuando es necesario.

Se ha identificado dos tipos de riesgos:

Riesgos genéricos: Amenaza potencial para el proyecto de software.

Riesgos específicos: Se los puede identificar teniendo un claro conocimiento de la tecnología, el personal.

Componentes y controladores del riesgo

Es otro criterio, en la que se identifica cuatro componentes a tener en cuenta para la estimación del riesgo en desarrollo de software y son:

Riesgo de desempeño: es el que me permite medir el grado de incertidumbre de que el producto satisfaga los requisitos.

Riesgo de costo: me permite medir el grado de incertidumbre del presupuesto del proyecto.

Riesgo de soporte: mide el grado de incertidumbre de que el SW resulte fácil de corregir, adaptar y mejorar.

Riesgo de calendarización: me permite tener un grado de incertidumbre de que el producto se entregue a tiempo.

Análisis del riesgo

Lo que se intenta con la proyección del riesgo es hacer una medición de los riesgos en función de la probabilidad o posibilidad de que el riesgo sea real o llegara a ocurrir y así como también las consecuencias que causaría al momento de que ocurriese el imprevisto

Para la proyección del riesgo se realizan cuatro pasos:

- Establecer una escala que refleje la posibilidad percibida de un riesgo.
- Definir las consecuencias del riesgo
- Estimar el impacto del riesgo en el proyecto y el producto
- Tener una exactitud general en la proyección del riesgo para así evitar confusiones

Estos pasos nos van a permitir establecer prioridades.

Desarrollo de una tabla de riesgos

Ofrece al gestor de un proyecto una técnica simple para la proyección de riesgos.

Con el fin de facilitar un plan RSGR que es un **Plan de Reducción, Supervisión y gestión de riesgo** se explicará como ingresar la información en la plantilla **"**Evaluación de riesgos.xls". ***Ver anexo 1***.

Riesgo: La construcción de esta tabla se comienza listando todos los riesgos sin importar cuán remotos sean.

Probabilidad: Es la probabilidad de ocurrencia de este riesgo y se clasifica en: Muy baja, Baja, Medio, Alto y Muy alta

Impacto: Es la clasificación que se le da al riesgo si llegará a pasar, indicando el nivel de daño o traumatizmo que puede causar en el proyecto y se clasifica en: Muy bajo, Bajo, Medio, Alto y Muy alto.

Valor probabilidad: Como su nombre lo indica, es el peso que se la da al campo probabilidad.

Probabilidad	Valor probabilidad
1. Muy baja	0,1
2. Baja	0,3
3. Medio	0,5
4. Alto	0,7
5. Muy alta	0,9

Tabla 5 probabilidad valor probabilidad

Nota: Como son probabilidades no puede dársele el valor de 1, ya que no sería una probabilidad, si no un hecho.

Valor impacto: Es el peso que se le da al campo impacto.

Impacto	Valor impacto
1. Muy bajo	1,5
2. Bajo	3,5
3. Medio	5,5
4. Alto	7,5
5. Muy alto	9,5

Tabla 6 Impacto valor impacto

Impacto por probabilidad: Operación matemática, que consiste en multiplicar el valor de la probabilidad por el valor del impacto.

Calificación del riesgo: Indicador visual del riesgo, se clasifica en 3 colores:

Verde: Bajo riesgo

Amarillo: Mediano riesgo

Rojo: Alto riesgo.

Requiere mitigación: Indicador (si/no), si la calificación del riesgo está en verde o amarillo el indicador estará en “No”, si el indicador del riesgo está en rojo el indicador estará en “Si”.

Estrategia de reducción: Descripción, de la estrategia para mitigar el riesgo.

Actividades: Indicar paso a paso de las actividades a realizar para evitar o mitigar el riesgo.

Frecuencia: Cada cuanto tiempo, se debe revisar el estado de este riesgo para saber su estado actual.

Responsable: Encargado de revisar y controlar este riesgo.

Revisiones: Notas y fechas de las revisiones que se la ha dado al riesgo.

Estado: Estado del riesgo (abierto/cerrado).

Enfoque de los riesgos en la primera etapa ASAP (Preparación del Proyecto)

La primera fase o etapa de ASAP es en la que se deben tomar decisiones con respecto al diseño del proyecto:

- Equipo de trabajo.
- Tareas.
- Recursos.
- tiempo.
- metodología.

Cada una de estas actividades conlleva riesgos que deben ser analizados en esta etapa, con el fin de garantizar las otras etapas.

Enfoque de los riesgos en la segunda etapa ASAP (Planos del Negocio)

La segunda fase o etapa de ASAP es en la que se deben tomar decisiones con respecto a:

- Definición de Estructuras Organizativas.
- Definición de Procesos Funcionales.
- Definición de Datos Maestros.
- Documento Plano de Negocios.

Esta etapa es de vital importancia, ya que se plasman los procesos de negocio de la empresa y el mapeo de estos procesos hacia la plataforma SAP.

Justamente en esta etapa es en donde se descubren cuales son las personalizaciones. Los riesgos de esta etapa deben ir enfocadas en estos aspectos.

Enfoque de los riesgos en la tercera etapa ASAP (Realización)

La tercera fase o etapa de ASAP es en la que se deben tomar decisiones con respecto a:

- Configuración de los módulos.
- Interfaces con otros sistemas.
- Conversiones.
- Desarrollos (personalizaciones).
    - Especificaciones.
    - Estimaciones.
    - Seguimientos.
    - Cambios.
    - Nivel del personal

Es muy importante en esta etapa del desarrollo, se tengan los procesos muy bien definidos y revisados de la fase anterior, para poder asegurar una correcta claridad en las especificaciones de las personalizaciones.

Algunos riesgos en los desarrollos son:

¿Las especificaciones son lo suficientemente claras?

¿Las especificaciones tienen su estimación correcta?

¿Qué pasa si hay cambio en las aplicaciones?

¿El nivel del personal es el adecuado para el proyecto?

¿Qué pasa ante un cambio en el negocio con respecto a los desarrollos?

¿Las métricas para los desarrollos es la adecuada, si está dando los resultados esperados?

4.3 Métricas[17]

Las métricas de procesos de software son algo más que una simple medida de algún "atributo" del proceso de software; una métrica es información que sirve para planificar, predecir y evaluar el estado de un proyecto.

"Cuando puedes medir lo que estas diciendo y expresarlo en números, sabes algo sobre ello; pero cuando no puedes medirlo, cuando no puedes expresarlo con números, tu conocimiento es escaso e insatisfactorio: puede ser el comienzo del conocimiento pero en tus pensamientos, apenas estás avanzando hacia el escenario de la ciencia." Lord Kelvin

La medición se aplica al proceso de software con la finalidad de mejorarlo de manera continua. La medición se utiliza a lo largo de un proyecto como apoyo en la estimación, el control de la calidad, la valoración de la productividad y el control del proyecto.

4.3.1 Necesidades de medida

Las necesidades de medida pueden ser diversas, desde medir el rendimiento de los proyectos de una empresa, evaluar las inspecciones de código hasta evaluar las actividades de mejora del proceso software

La medida es una parte esencial para comprender que afecta a la calidad, oportunidad, utilidad y funcionalidad y en la mejora de los procesos y productos software.

Para comprender como aplicar las métricas del software, debemos comprender primero qué significan las medidas y por qué las necesitamos.

Las medidas nos permiten cuantificar conceptos o atributos en orden a manipularlos y aprender más sobre ellos

4.3.1.1 Cuáles son los costes de no medir

Incapacidad para:

- estimar/planificar de forma realista
- determinar el progreso
- evaluar la calidad
- reconocer las oportunidades de mejora
- reconocer mejoras

[17] Fuente Pincipal: Pressman Roger S. Ingeniería de software Un enfoque práctico Sexta edición . Cap gestión del riego México;Mc Graw Hill, 2006

4.3.1.2 Precauciones y Limitaciones

- Medición implica varios usuarios en diversos niveles por toda la organización.
- Medición no deberá ser utilizada para evaluar prestaciones individuales; más bien deberá servir como base para resolución interactiva de aspectos del proceso.
- Dos programas no son iguales: consecuentemente comparación directa de datos de programas deberá evitarse.
- Evaluaciones basadas en medición son solo tan buenas como la oportunidad, consistencia y precisión de los datos de entrada.

4.3.2 Beneficios de las métricas

- Medición por si misma no mejora el proceso, pero la visibilidad permite profundizar en la planificación, control, gestión y mejora.
    - Datos históricos ayudan a predecir y planear
    - Datos actuales frente a los planificados ayudan a comunicar el progreso y soportan la toma de decisiones.
- Análisis de tendencias ayuda a enfocar sobre áreas problema.
- La supervisión de las actividades de mejora del proceso ayuda a identificar lo que funciona y lo que no funciona
- Calidad del producto mejorada.
- Productividad del equipo de desarrollo incrementada.
- Estimación y planificación del proyecto mejor.
- Mejor gestión del proyecto.
- Cultura de calidad de la compañía mejorada.
- Satisfacción del cliente mejorada.
- Visibilidad del proceso software incrementada.

4.3.3 Pasos para realizar usar una métrica

- Establecer los objetivos de la medición
- Especificar medidas
- Especificar procedimientos de recogida y almacenamiento
- Especificar procedimientos de análisis
- Comunicar los resultados

Una de las métricas más utilizadas para estimar la duración de un proyecto según su complejidad y extensión son los puntos de función.

4.3.4 Punto de función

“El Punto Función es una medida del tamaño de un sistema de software y del proyecto que lo construye. Es una unidad de medida así como la hora lo es para medir el tiempo o el kilómetro para medir la distancia. El Análisis de Punto Función se basa en la teoría de que las funciones de una aplicación son la mejor medida del tamaño de un sistema. El Punto Función mide el software mediante la cuantificación de la funcionalidad que el sistema le brinda al usuario basado fundamentalmente en el diseño lógico. Es independiente del lenguaje de computación, de la metodología de desarrollo, de la tecnología utilizada y de la capacidad del equipo de trabajo para desarrollar la aplicación.

El Análisis del Punto Función es un método estándar de medición de desarrollo de software desde el punto de vista del usuario. Su objetivo es medir el software basándose en la cuantificación de la funcionalidad brindada al usuario partiendo fundamentalmente de diseños lógicos. La cuenta de Punto Función para proyectos de desarrollo mide las funcionalidades que se le proporcionan al usuario conjuntamente con la primera instalación del software producido cuando el proyecto es terminado.

En el mundo del Punto Función los sistemas son divididos en componentes. Los primeros tres componentes son: Entradas (EI = External Input), Salidas (EO = External output) y Consultas (EQ = External Queries).
Cada una de estas transacciones actúa sobre ficheros agregando, modificando, eliminando, obteniendo o procesando información. Los otros dos componentes son los ficheros o entidades del sistema, denominados Ficheros Lógicos Internos (ILF = Internal Logical File) y Ficheros Lógicos Externos (ELF = External Logical File).

Frontera de la aplicación

La frontera de la aplicación indica el límite entre el software que está siendo medido. Frecuentemente los sistemas computacionales interactúan con otros sistemas computacionales y con usuarios o dispositivos.

Es crucial tener bien identificada la frontera de la aplicación a ser medida para luego recién enfocarse en la descomposición de sus componentes. La frontera de la aplicación debe ser dibujada en función del punto de vista del usuario.
En resumen, la frontera marca el borde entre el proyecto o aplicación a ser medido y las aplicaciones externas o el dominio del usuario. Una vez establecida la frontera, los componentes pueden ser clasificados y contados.

Actúa como una membrana a través de la cual los datos procesados por transacciones (EI, EO, EQ) cruzan la frontera hacia adentro y/o hacia fuera de la aplicación.

Determina los datos lógicos mantenidos por la aplicación (ILF) así como por consiguiente facilita la identificación de cuáles datos lógicos son referenciados por la aplicación pero mantenidos por otro sistema (ELF).
Las siguientes reglas deben aplicarse para el establecimiento de la frontera:

1. La frontera es determinada basándose en el punto de vista del usuario. Se focaliza en qué el usuario puede entender o describir.

2. La frontera entre aplicaciones relacionadas está basada en áreas funcionales separadas siempre desde el punto de vista del usuario y no en consideraciones técnicas.

3. La frontera establecida por una aplicación existente que está siendo modificada no se influencia por el alcance de la cuenta.

La localización de la frontera de la aplicación entre el software que está siendo medido y otras aplicaciones de software suele ser algo subjetivo y difícil de determinar dónde una aplicación termina y la otra comienza.

Es importante que la frontera de la aplicación sea dibujada con cuidado dado que todo aquel dato que atraviesa la frontera puede potencialmente ser incluido en el alcance de la cuenta.

Mecanismo

Para cada uno de los 5 tipos de elementos (EI, EO, EQ, ILF y ELF), identificar aquellos que surgen de los requerimientos o funcionalidades que proveerá el sistema. Luego analizar la complejidad de cada uno, que puede ser Alta, Media o Baja. Con estos datos, y utilizando los pesos correspondientes a cada uno de los elementos identificados es posible calcular el total de Puntos Función**.**

Ficheros Lógicos Internos (ILF)

Un ILF es un grupo de datos relacionados lógicamente o información de control reconocida por el usuario y mantenida dentro de la frontera de la aplicación. El objetivo fundamental de un ILF es manejar los datos mantenidos a través de uno o más procesos elementales (o acciones) de la aplicación que está siendo contada.

Reglas para identificarlos:

1. El grupo de datos o información de control es un grupo de datos lógico identificable por el usuario que cubre de manera completa requisitos específicos de este.

2. El grupo de datos es mantenido dentro de los límites de la aplicación.

3. El grupo de datos es mantenido o modificado por medio de un proceso elemental de la aplicación.

4. El grupo de datos identificado no se ha contado como ELF de la aplicación.

Ficheros Lógicos Externos (ELF)

Un ELF es un grupo de datos relacionados lógicamente o información de control reconocida por el usuario y referenciada pero mantenida dentro de la frontera de otra aplicación. El objetivo principal de un ELF es manejar los datos referenciados mediante uno o más procesos elementales (o acciones) dentro de la frontera de la aplicación contada.

La principal diferencia entre un archivo lógico interno ILF y una interfase de archivo externa ELF es que el ILF es mantenido dentro de la frontera de la aplicación mientras que el ELF es mantenido por otra aplicación. El ELF es referenciado por la aplicación que se está contando pero no mantenido por ella.

Reglas para identificarlos:

1. El grupo de datos o información de control es un grupo de datos lógico identificable por el usuario que cubre de manera completa requisitos específicos de este.

2. El grupo de datos es referenciado y es externo a la aplicación que está siendo contada.

3. El grupo de datos no es mantenido por la aplicación que está siendo contada.

4. El grupo de datos se ha contado como ILF en al menos otra aplicación.

5. El grupo de datos identificado no ha sido contado como un ILF para la aplicación.

Nota:

No relacionar un ILF o ELF con un archivo físico, sino como una entidad. Si por ejemplo se identifica la entidad factura, la misma probablemente se registre en dos archivos o tablas: cabecera y detalles, sin embargo, estos dos archivos corresponden a una misma entidad lógica por lo que se cuenta únicamente un ILF o ELF.

Entradas (EI)

Un EI es un proceso elemental o una acción que procesa datos o información de control que vienen desde afuera de la frontera de la aplicación hacia adentro. Los datos pueden venir desde otra aplicación o desde una pantalla de ingreso de datos. El objetivo fundamental es mantener uno o más archivos lógicos internos (o ILF de Internal Logical File) y/o alterar el comportamiento del sistema.

Se aplican las siguientes reglas:

1. Los datos se reciben desde fuera de los límites de la aplicación.

2. Los datos mantienen un ILF a través de un proceso elemental de la aplicación.

3. El proceso es la unidad más pequeña de actividad que es significativa para el negocio del usuario final.

4. El proceso es auto contenido y deja la aplicación que está siendo contada en un estado consistente.

5. El proceso identificado debe verificar alguna de estas reglas:
    a. Su lógica de proceso es única respecto de otras entradas externas de la aplicación.
    b. Los elementos de datos identificados son distintos a los de las otras EI de la aplicación.
    c. Los ficheros lógicos referenciados son diferentes

Salidas (EO)

Un EO es un proceso elemental o una acción que envía datos o información de control hacia fuera de la frontera de la aplicación. El objetivo fundamental es presentar información al usuario a través del procesamiento lógico de los datos o de la información de control obtenida. El procesamiento lógico debe contener al menos una fórmula o cálculo matemático o crear datos derivados de los obtenidos. Un EO podría mantener uno o más ILF y/o alterar el comportamiento del sistema.

Se aplican las siguientes reglas:

1. El proceso envía datos información de control.

2. Los datos o información de control se envían a través de un proceso elemental de la aplicación.

3. El proceso es la unidad más pequeña de actividad que es significativa para el negocio del usuario final.

4. El proceso es auto contenido y deja a la aplicación en un estado consistente.

5. El proceso identificado debe verificar alguna de estas reglas:
   Su lógica de proceso es única respecto de otras salidas externas de la aplicación.
   Los elementos de datos identificados son distintos a los de otras EO’s de la aplicación.
   Los ficheros lógicos referenciados son distintos.

6. Debe cumplirse al menos una de las siguientes condiciones:

El proceso elemental contiene al menos una fórmula matemática o cálculo.
El proceso crea datos derivados
El proceso que genera la salida mantiene algún ILF
El proceso que genera la salida altera el comportamiento del sistema.

7. La transferencia de datos a otras aplicaciones se cuenta como salidas

8. Los informes escritos y online se cuentan como salidas independientes.

9. Los gráficos se cuentan como una salida cada uno.

Consultas (EQ)

El EQ es un proceso elemental o una acción que envía datos o información de control hacia fuera de la frontera de la aplicación. El objetivo principal de un EQ es presentar información al usuario a través de la mera obtención del dato o de la información de control. A diferencia del EO, el procesamiento lógico no debe contener ninguna fórmula o cálculo matemático, ni tampoco debe crear datos derivados de los obtenidos.

Por otra parte ningún ILF es mantenido mientras se procesa la acción de un EQ ni tampoco el comportamiento del sistema se ve alterado.

Es una combinación de entrada/salida que se obtiene de una búsqueda de datos, no actualiza ficheros lógicos y no contiene datos derivados (aquellos que requieren un proceso distinto a búsqueda, edición o clasificación).

Se aplican las siguientes reglas:

- Una petición entra dentro del límite de la aplicación.

- Un resultado sale del límite de la aplicación

- Hay recuperación de datos

- Los datos recuperados no contienen datos derivados.

- El proceso lógico no contiene fórmulas matemáticas o cálculos

- El proceso que genera la consulta no mantiene ningún ILF ni altera el comportamiento del sistema

- La petición de entrada y el resultado de salida juntos, hacen del proceso la unidad de actividad más pequeña que es significativa para el negocio del usuario final.

- El proceso es auto contenido y deja a la aplicación que está siendo contada en un estado consistente.

- El proceso no actualiza ILF's.

- El proceso verifica alguna de estas dos reglas:
  La lógica del proceso sobre la entrada y la salida es única respecto a otras consultas de la aplicación
  Los elementos de datos que forman la entrada y la salida son distintos a los de las otras consultas de la aplicación.
  Los ficheros lógicos referenciados son distintos.

- El proceso no actualiza ILF's.

- El proceso verifica alguna de estas dos reglas:
  La lógica del proceso sobre la entrada y la salida es única respecto a otras consultas de la aplicación.
  Los elementos de datos que forman la entrada y la salida son distintos a los de las otras consultas de la aplicación.
  Los ficheros lógicos referenciados son distintos.

Puntos De Función sin ajustar (PFSA)

Los Punto Función sin ajustar refleja las específicas funcionalidades contabilizadas que se le proporcionan al usuario por el proyecto o la Aplicación. Las funcionalidades de usuario específicas son evaluadas en términos del *Qué* es lo que proporcionan y no en el *Cómo* es proporcionado.
Cada punto de función sin ajustar debe dársele un valor de 0 a 5.

Valor	Significado del valor
0	Sin influencia, factor no presente
1	Influencia insignificante, muy baja
2	Influencia moderada o baja
3	Influencia media, normal
4	Influencia alta, significativa
5	Influencia muy alta, esencial

Tabla 7valores escala puntos de función sin ajustar

Los puntos de función sin ajustar son:

¿El sistema requiere respaldo y recuperación confiables?
¿Se requieren comunicaciones de datos especializadas para transferir información a la aplicación, u obtenerla de ella?
¿Hay funciones distribuidas de procesamiento?
¿El desempeño es crítico?
¿El sistema se ejecutará en un entorno existente que tiene un uso pesado de operaciones?
¿El sistema requiere entrada de datos en línea?

¿La entrada de datos en línea requiere que la transacción de entrada se construya en varias pantallas u operaciones?
¿Los ILF se actualizan en línea?
¿Las entradas, las salidas, los archivos o las consultas son complejos?
¿Es complejo el procesamiento interno?
¿El código diseñado será reutilizable?
¿Se incluye la conversión e instalación en el diseño?
¿Está diseñado para instalaciones múltiples en diferentes organizaciones?
¿La aplicación está diseñada para el cambio y para que el usuario lo use fácilmente?

Pasos para totalizar los puntos de función

Se toman los cinco elementos EI, EO, EQ, ILF y ELF, se discriminan sus funcionalidades por el nivel de complejidad, se suman y se multiplican por su ponderación

Parámetro	Complejidad	Cantidad	Ponderación	Total = cantidad * ponderación
EI	Baja		3	
	Media		4	
	Alta		6	
EO	Baja		4	
	Media		5	
	Alta		7	
EQ	Baja		3	
	Media		4	
	Alta		6	
ILF	Baja		7	
	Media		10	
	Alta		15	
ELF	Baja		5	
	Media		7	
	Alta		10	
Total				

Tabla 8 totales por EI ,EO, EQ, ILF y ELF

Luego de esto se totalizan los puntos de función sin ajustar (PFSA).

Finalmente la formula para los puntos de función quedaría así

Puntos de función = Total * [0.65 + (0.01 * PFS)]"[18]

[18] Instructivo para la cuenta de puntos de función [sitio en Internet], disponible en http://www.dinacyt.gub.uy/pdt/files/6.2.5%20-%20puntosFuncion.pdf Último acceso: Noviembre 20 de 2007.

4.3.5 Métricas de proyecto

Permiten que un gestor de proyecto de software:

- Valore el estado de un proyecto en curso.
- Rastree los riesgos potenciales.
- Descubra área de problemas antes que se vuelvan críticas.
- Ajuste el flujo de trabajo o las tareas.
- Evalúe la habilidad del equipo del proyecto para controlar la calidad del equipo del proyecto para controlar la calidad de los productos de trabajo de software.

Plantilla para las métricas del proyecto

Con el fin de llevar las métricas del proyecto de una manera sencilla se puede alimentar la siguiente tabla:

ID proyecto	Titulo	Modulo	Prioridad	Orden ejecución	Fase del proyecto	Estado	%de avance	Complejidad	T desarrollo	T real desarrollo

Tabla 9 Datos básicos para las métricas del proyecto

La tabla de las métricas del proyecto se debe alimentar de la tabla de las métricas llevada por cada desarrollador.

Con esta sencilla tabla podemos saber el estado actual de las personalizaciones del proyecto y tomar medidas al respecto si se llegaran a necesitar.

Plantilla para las métricas del programador

Para saber el estado y las actividades de cada personalización se deben llenar las siguientes tablas

ID proyecto	
Titulo	
Descripción	
Prioridad	
Modulo	
Orden ejecución	
Fase del proyecto	
Estado	
% avance	
Funcional	
Complejidad	
T desarrollo	

Tabla 10 datos básicos métricas programador

ID Proyecto	
Fecha	
Labor realizada	

Tabla 11 Actividades del desarrollador en una personalización

Estas tablas o formatos, nos ayudan a medir el rendimiento de cada desarrollador y el estado de sus personalizaciones, a la vez que son la fuente alimentadora de las métricas del proyecto en general.
Como recomendación, estas métricas se deben llevar en formato electrónico en un portal web de manera tal que las métricas del proyecto se vallan llenando automáticamente, cuando los desarrolladores ingresen la información de sus actividades. Varias empresas para esto usan Sharepoint Server y si es posible se debería integrar con Project Server para tener unas métricas y cronogramas más afinados.

Plantilla propuesta especificaciones

Con el fin de asegurar una especificación funcional y técnica adecuada, que de pie a contratar una fábrica de software si fuese necesario y un efectivo control de cambios sin traumatismo, se hace la siguiente propuesta. **Ver anexo 2**.

4.4 Pruebas

Con el fin de asegurar la calidad en el desarrollo realizado, se deben hacer varios tipos de pruebas.

4.4.1 Pruebas del programador

A medida que se va realizando el código del desarrollo, el programador puede ir chequeando los segmentos de programa que va desarrollando.
Estas pruebas son a nivel de:

- Funciones.
- Clases.
- Formularios.
- Reportes.
- Consultas SQL.
- Segmentos de código.
- Ampliaciones (user exit, customer exit, badis y frameworkde ampliaciones).
- Jobs.

Para verificar que los ítems del listado anterior, cumplan con las especificaciones funcionales podemos ejecutar el debugger del sistema.

Con la base de datos además de usar el debugger para saber si está cumpliendo con lo requerido se puede usar el la transacción ST05 saber que consultas se están ejecutando, en que parte del código y tiempo de ejecución.

Además de verificar que el código sí cumple con las especificaciones dadas, se debe verificar el programa realizado con el "Code inspector" ->transacción SCI para garantizar que el código sea seguro y posea buenas prácticas de programación que garantice que no se está desperdiciando recursos por la forma de programar.

Además se debe mirar el rendimiento de la aplicación con respecto a los recursos que utiliza, esto se hace en el menú Utilidades -> Más utilidades->Run time analysis.

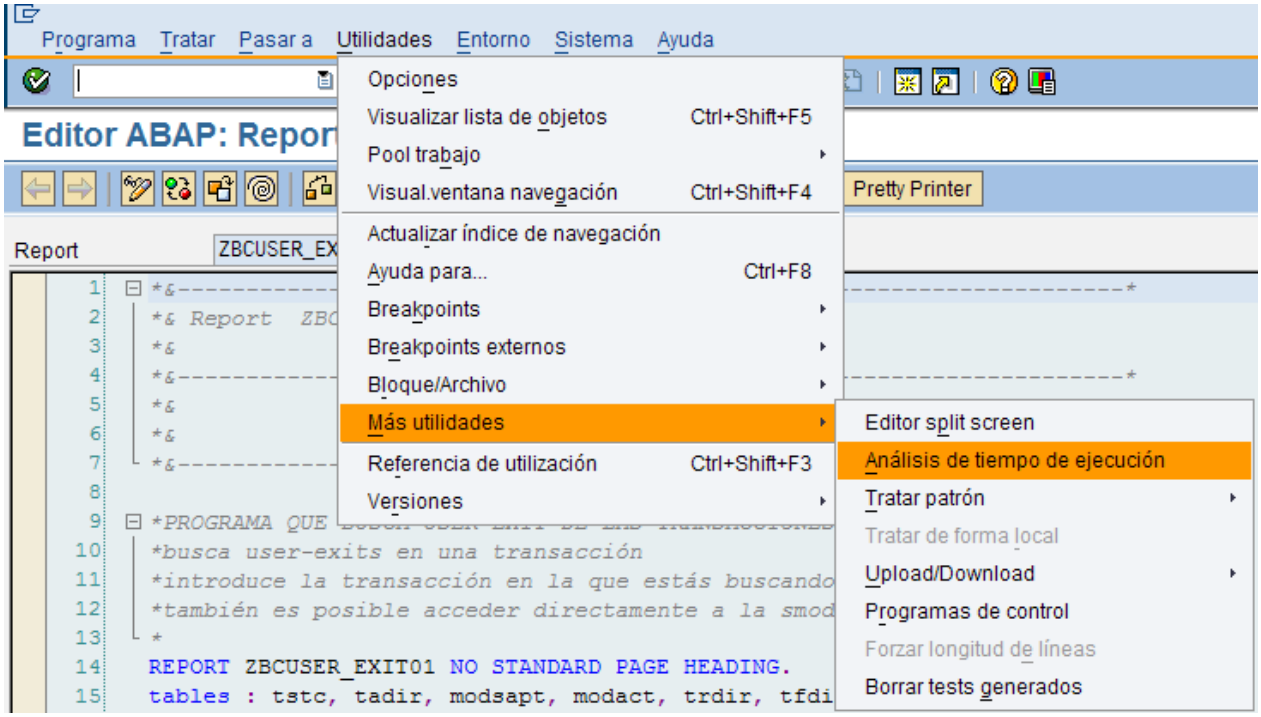


Ilustración 22 ingreso run time analysis

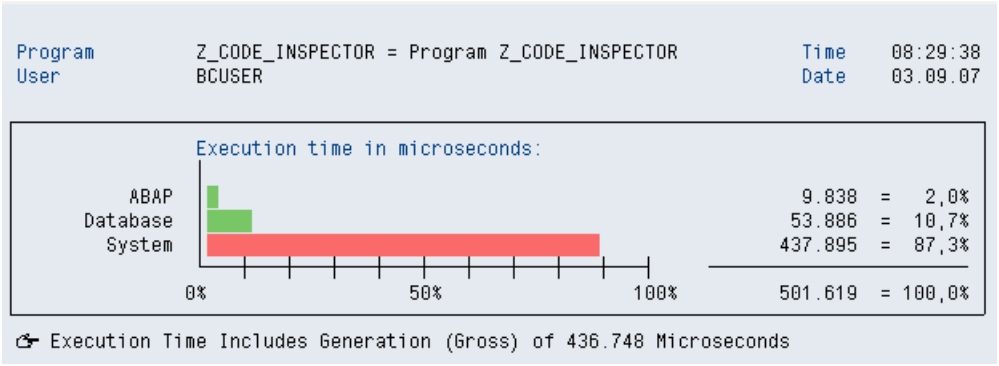


Ilustración 23 Resultados gráficos del runtime analysis

Nota:
Como punto de enfoque El programador debe centrarse en las especificaciones dadas, para diseñar su programa y hacer las pruebas de una forma organizada.

Pre-condiciones:

Se debe tener un mínimo de datos de prueba, lo suficientemente robusto para poder hacer todas las pruebas que se consideren necesarias en la realización de las pruebas.

Post_condiciones:

El desarrollo se ha probado con los datos y ha funcionado con el resultado esperado, en caso contrario se hacen las correcciones del caso**.**

4.4.2 Pruebas unitarias

Son las pruebas realizadas, usualmente por el usuario líder, funcional de ese módulo o por defecto alguien que está con la responsabilidad del desarrollo.

Estas pruebas deben estar previamente diligenciadas en el formato de pruebas y son las encargadas de verificar que el programa funcione correctamente y coexista con los demás programas del modulo para el que fue desarrollado.

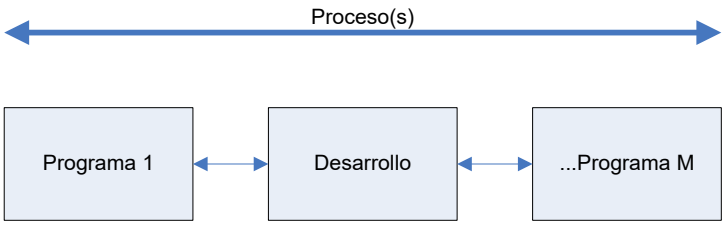


Ilustración 24 Flujo de datos en pruebas unitarias

En las pruebas unitarias, los desarrollos están en un sistema que está compuesto de varios aplicativos y estos pertenecen a uno o varios procesos del negocio y a un módulo. Hay un flujo de datos que debe funcionar de forma correcta.

Pre-condiciones:

Se debe tener un mínimo de datos de prueba, lo suficientemente robusto para poder hacer todas las pruebas que se consideren necesarias en la realización de las pruebas.

Post_condiciones:

El desarrollo se ha probado con los datos y ha funcionado con el resultado esperado y se firma que las pruebas se han cumplido satisfactoriamente**.**
En caso de existir errores se debe diligenciar el reporte de errores y notificarle esto al desarrollador.

4.4.3 Pruebas integrales

Son las pruebas realizadas, usualmente por los usuarios líderes, funcionales de los módulos o por defecto varias personas que están con la responsabilidad de los desarrollos.

Estas pruebas deben estar previamente diligenciadas en el formato de pruebas y son las encargadas de verificar que los programas funcionen correctamente como en las pruebas unitarias y coexista con los demás programas que pertenecen a otros módulos del sitema.

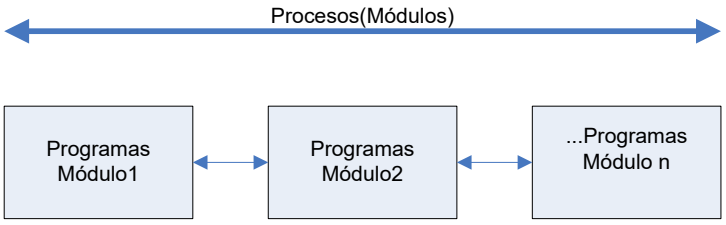


Ilustración 25 Flujo de datos en pruebas integrales

En las pruebas integrales, los desarrollos están en un sistema que está compuesto de varios aplicativos y usualmente entre estos aplicativos que pertenecen a uno o varios procesos del negocio (módulos del sistema). Hay un flujo de datos que debe funcionar de forma correcta.

Pre-condiciones:

Se debe tener un mínimo de datos de prueba, lo suficientemente robusto para poder hacer todas las pruebas que se consideren necesarias en la realización de las pruebas.

Post_condiciones:

El desarrollo se ha probado con los datos y ha funcionado con el resultado esperado y se firma que las pruebas se han cumplido satisfactoriamente**.**
En caso de existir errores se debe diligenciar el reporte de errores y notificarle esto al desarrollador.

Ingreso información formato de pruebas

Es un archivo en Excel “Formato de pruebas.xls” y consta básicamente de 2 hojas, la primera hoja “Casos de prueba” (CP) y la segunda hoja es la “Bug tracker”. **Ver Anexo 3.**

Hoja Casos de prueba

Es la hoja donde se guarda detalladamente todo lo concerniente a los casos de prueba y consta de los siguientes campos:

- **Desarrollo ID:** Identificador único de desarrollo
- **ID CP:** Identificador único caso de prueba para ese desarrollo

- **Titulo CP:** Título caso de prueba
- **Propósito:** Que va hacer este caso de prueba
- **Datos de Prueba**
- **Pasos:** paso a paso de cómo se ejecuta el paso de prueba
- **Notas y Preguntas:**
- **Estado CP:** Pasó, Falló, No se pudo ejecutar y Eliminado
- **# Bug:** Identificador único de Bug para ese desarrollo
- **Estado Bug:** Abiertos, Asignados, Solucionados, Disponible para pruebas, Verificados, No Aplica, Duplicado, No Se Puede solucionary Próxima Version.

Con fines control se lleva el número y porcentaje de:

- **Total CP Creados**
- **Total CP Ejecutados**
- **Casos de prueba Pasaron**
- **Casos de prueba Fallaron**
- **Total CP eliminados**

Hoja Bug tracker

Si se han encontrado errores en los casos de prueba, estos se registran en la hoja bug tracker y consta de los siguientes campos:

- **# Bug:** Identificador único del bug para ese caso de prueba.
- **ID CP:** Identificador único caso de prueba para ese desarrollo.
- **Titulo:** Titulo del bug.
- **Descripcion Bug:**
- **Pasos reproduccion Bug:** Paso a paso y con que datos se dio el error.
- **Descripción sln:** Descripción de la solución, cuando el desarrollador la implemente.
- **Estado:** Abiertos, Asignados, Solucionados, Disponible para pruebas, Verificados, No Aplica, Duplicado , No Se Puede solucionar y Próxima Version
- **Fecha de Registro:** Fecha en que se detecto el error.
- **Asignado a:** Quien hizo la prueba
- **Fecha de asignación:** Fecha de asignación de corrección del error.
- **Fecha posible solución:** Fecha aproximada de la corrección del error
- **Fecha de solución:** Fecha real de la corrección**.**
- **Responsable de solución.**

Con fines control se lleva el número y porcentaje de:

- Abiertos
- Asignados
- Solucionados
- Disponible para pruebas
- Verificados
- No Aplica
- Duplicado

- No Se Puede solucionar
- Próxima versión

Nota: El estado de un error en **"**Próxima versión" únicamente se podrá poner cuando haya un consenso entre el usuario líder, funcional y desarrollador.

Calidad en las pruebas unitarias e integrales

Cuando se ingresen los datos en las plantillas de las pruebas unitarias e integrales se debe realizar una revisión por pares, esto quiere decir que se escoge a alguien que esté en la capacidad de entender el problema y sea capaz de entender las plantillas de cómo se deben hacer las pruebas unitarias e integrales, en caso de duda o error se debe poner de una forma más clara las ideas en estas pruebas.

4.5 Gestión del cambio

"**La *Gestión del Cambio*** es una disciplina que apoya directamente el desarrollo y mantenimiento del software, mediante la conservación de la integridad del producto antes y después de su puesta en producción.

El **cambio** es inevitable cuando se construye software, **aumenta el grado de confusión** entre los ingenieros de software que trabajan en un proyecto. La confusión surge cuando los cambios:

- • No se analizan antes de realizarlos,
- • No se registran antes de implementarlos,
- • No se reportan a quienes deben saberlo,
- • No se controlan de forma que mejore la calidad y reduzca el error.

Las principales razones para la realización de cambios en la infraestructura TI son:

- Solución de errores conocidos.
- Desarrollo de nuevos servicios.
- Mejora de los servicios existentes.
- Imperativo legal.

Los pasos al realizar un cambio son:

RFC (Request for change): Cuando se va a realizar un cambio, se debe realizar una petición de cambio, esta obedece a:

- Corrección de errores.
- Innovación y mejora en los servicios.
- Cumplimiento de nuevas normativas legales.

Registro: La RFC debe quedar correctamente registrada, para poder hacer un seguimiento completo al proceso de cambio, el registro debe incluir:

- Identificador único de la RFC.
- Descripción detallada del cambio y sus objetivos.
- Estatus (Aprobado, Rechazado, Aplazado).

Clasificación: En caso de ser aprobado el cambio se debe tener en cuenta:

- **Prioridad:** Importancia del RFC, esto determinará el calendario del cambio
- **Categoría:** Impacto y dificultad de cambio, determinará la asignación de recursos y plazos previstos.

Urgencia: En ocasiones que el cambio no puede esperar debido a interrupciones en el sistema, el cambio se debe aprobar inmediatamente

Planificación: Una vez aprobado el cambio se debe realizar lo siguiente

- Elaboración de calendarios realistas de cambio.
- Cumplimiento de los objetivos indicados.
- Minimización de incidencias secundarias derivadas del cambio.

Roll-out: Aunque en la gestión de cambios no es la responsable directa, deber coordinar y garantizar el entorno de desarrollo, pruebas e implementación

Back out: Los planes del back out son imprescindibles para evitar interrupciones graves del servicio: sus objetivos principales son:

- Volver en el menor tiempo posible a la última configuración estable anterior al cambio.
- Impedir que se pierdan datos e información valiosa durante los procesos de implantación del cambio.

Cierre:

Tras la implementación se debe evaluar el cambio

¿Se cumplieron los objetivos previstos?
¿Cual es la percepción de los clientes y usuarios?

Si la valoración es positiva se termina de documentar y cerrar el cambio, en caso contrario se llevan a cabo los planes de back out.

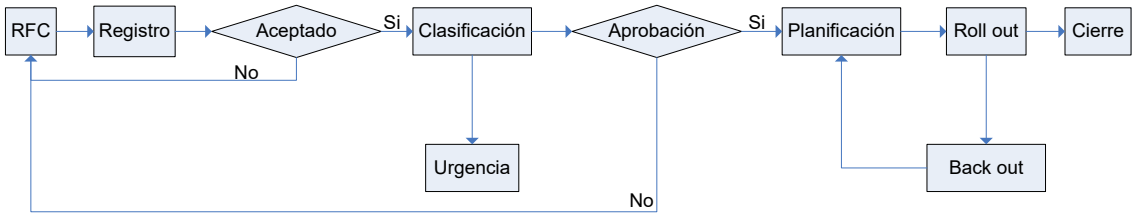


Ilustración 26 Flujograma de una petición de cambio

Los principales beneficios derivados de una correcta gestión del cambio son:

- Se reduce el número de incidentes y problemas potencialmente asociados a todo cambio.
- Se puede retornar a configuraciones estables de manera sencilla y rápida en caso de que el cambio tenga un impacto negativo en la estructura TI.
- Se reduce el número de "back-outs" necesarios.

- Los cambios son mejor aceptados y se evitan "tendencias inmovilistas".
- Se evalúan los verdaderos costes asociados al cambio y por lo tanto es más sencillo valorar el retorno real a la inversión.
- Se desarrollan procedimientos de cambio estándar que permiten la rápida actualización de sistemas no críticos.

La implementación de una adecuada política de gestión de cambios también se encuentra con algunas serias dificultades:

- Los diferentes departamentos deben aceptar la autoridad de la **Gestión de Cambios** sobre todo en lo que respecta al cambio, independientemente de que este se realice para solucionar un problema, mejorar un servicio o adaptarse a requisitos legales.
- Los encargados de la **Gestión de Cambios** no conocen a fondo las actividades, servicios, necesidades y estructura TI de la organización incapacitándoles para desarrollar correctamente su actividad.
- Los Gestores del Cambio no disponen de las herramientas adecuadas de software para monitorizar y documentar adecuadamente el proceso.
- No existe el compromiso suficiente de la dirección por implementar rigurosamente los procesos asociados.
- Se adoptan procedimientos excesivamente restrictivos que dificultan la mejora o por el contrario el proceso de cambio se trivializa provocando una falta de estabilidad necesaria para la calidad del servicio."[19]

Formato sugerido control de cambios: Ver ANEXO 4

[19] Gestión del cambio con ITIL [sitio en Internet], disponible en http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/vision_general_gestion_de_cambios/vision_general_gestion_de_cambios.php Último acceso: Noviembre 20 de 2007.

4.6 Biblioteca de software

Los procesos de negocio en cada empresa tienen sus particularidades, cada una es como una persona que posee sus individualidades.

Así como cada empresa tiene sus procesos en particular, existen procesos de negocio que son sumamente parecidos.

Los procesos que son más similares en una empresa, son los que son impuestos por ley como lo son facturación, procesos contables e impuestos como ICA, Retención en la fuente, etc.

Con el fin de agilizar las personalizaciones, se debe poseer un repositorio de software con el propósito de usar objetos altamente parametrizables y recortar tiempo sustancialmente a los desarrollos. Esta idea no es nueva y usualmente cada desarrollador de forma individual a medida que va pasando el tiempo y enfrenta nuevos desarrollos van formando un repositorio de sus programas realizados, pero esto no trae grandes aportes de manera colectiva si no únicamente individual, lo que no trae un aporte interesante al proyecto. Por otro lado si usamos los conceptos de fábrica de software los cuales nos hablan de programación por componentes altamente parametrizables tendríamos beneficios muy buenos para los proyectos, ya no de forma individual si no colectiva, porque estos componentes estarían a disposición de todos los integrantes del equipo de desarrollo.

La constitución de esta biblioteca de software para hacerla viable, debe hacerse de forma gradual si la empresa que provee los desarrolladores no tiene montado una fábrica de software debe hacer lo siguiente:

- Detectar que programas y funcionalidades a través de la experiencia son los más comunes e ir almacenando estos códigos en un repositorio común de público acceso para los miembros del equipo de desarrollo.

- Los programas y funcionalidades más comunes, se deben reescribir de forma genérica para futuras utilizaciones de forma más efectiva.

- Tener un manejo de versiones y documentación de los códigos.

- Para proveer una plataforma que soporte manejo de versiones y documentación del código se sugiere el manejo de un CVS o de Share Point

- Se sugiere que el repositorio este organizado por módulo al que pertenece el código y tipo de objeto (reporte, dympro, función, clase, sapscript, smarfrom, etc).

Al principio esto requeriría un esfuerzo adicional con respecto a lo que estamos familiarizados en las implantaciones actuales, pero nos brindaría grandes ahorros de tiempo a futuro.

A modo de consejo y hacer los componentes de software más reutilizables se aconseja concientizar a los desarrolladores en una mejor utilización de los objetos en la plataforma SAP .

- **SAP SOLUTION MANAGER**

El sap Solution Manager es una herramienta que da soporte a través de todo el ciclo de vida desde el blue print hasta la configuración del sistema en producción. Provee un acceso central a las herramientas, métodos y contenidos preconfigurados los cuales se pueden usar durante la evaluación, implementación y procesamiento operacional del sistema en productivo.

Algunas características

Implentación y actualizaciones de soluciones SAP

Acceso central a todas las herramientas del proyecto
- Administración del proyecto
- Blue print del negocio
- Configuración
- Pruebas

Gestión central de toda información del proyecto
1 Ruta del proyecto
2 Estado del sistema
3 Documentación del proyecto

Comparación y sincronización de diferentes componentes SAP

Solución de monitoreo
1. Administración central del systema
2. Anáisis del sistema con servicio de reportes
3. Monitoreo en tiempo real del sistema
4. Monitoreo del proceso del negocio.

Servicio de ayuda

Soporte de soluciones usando workflow para solucionar y procesar los mensajes de problemas.

Gestión del cambio

Manejo de peticiones de cambio, usando workflow para poder tener un rastreo y auditar los cambios y transportes en el sistema.

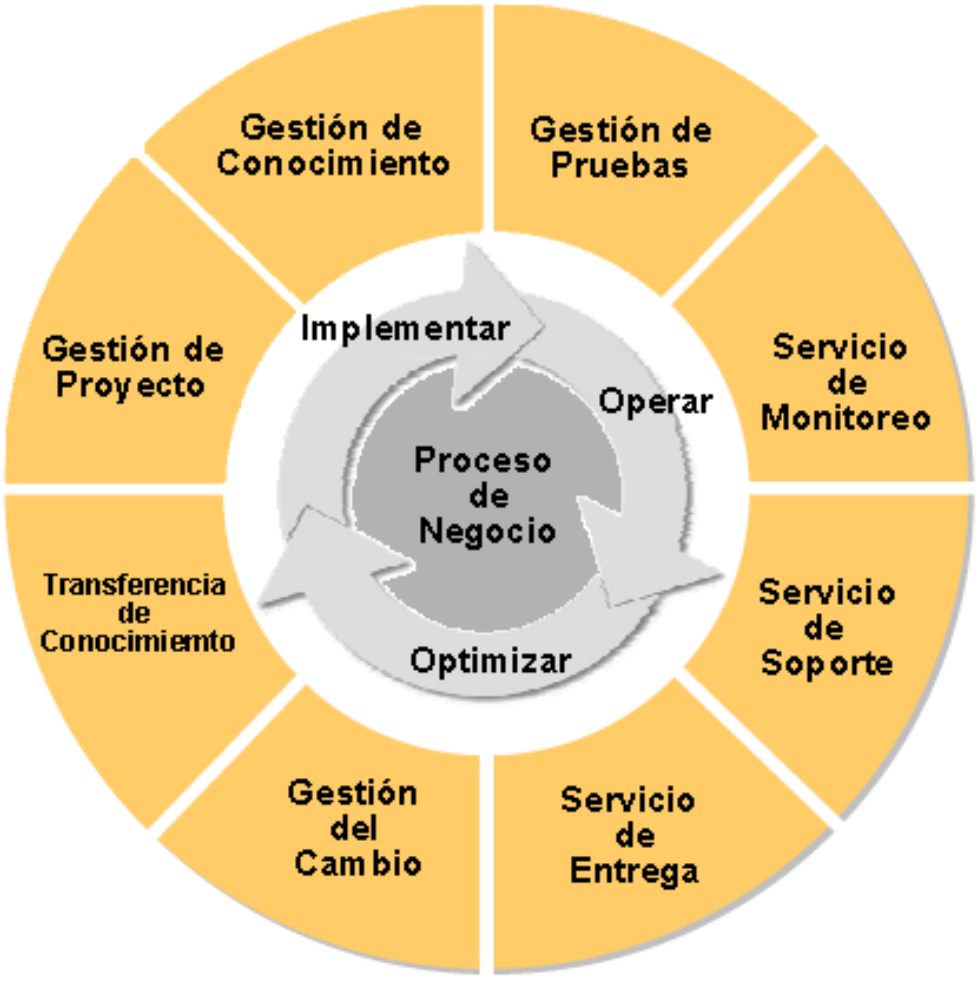


Ilustración 27 Sap solution manager

El Sap Solution Manager es un set de herramientas, servicios y documentación que garantizan el desarrollo completo de una implementación con SAP, si la organización posee la infraestructura para montarlo, es lo ideal al no tener que instalar un conjunto de componentes de software heterogéneo que no necesariamente garantizan una solución integral que si nos da el Solution Manager.

4.7 Calidad

4.7.1 ¿Que es la calidad del software?

La calidad del *software* es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad.

La calidad del *software* es medible y varía de un sistema a otro o de un programa a otro. Un *software* elaborado para el control de naves espaciales debe ser confiable al nivel de "cero fallas"; un *software* hecho para ejecutarse una sola vez no requiere el mismo nivel de calidad; mientras que un producto de *software* para ser explotado durante un largo período (10 años o más), necesita ser confiable, mantenible y flexible para disminuir los costos de mantenimiento y perfeccionamiento durante el tiempo de explotación.

La calidad del software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se detectan problemas deriva dos de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del *software*.

4.7.2 Aseguramiento de la calidad

Para garantizar la calidad, en la metodología ASAP, se deben verificar absolutamente todos los entregables respectivos a cada etapa.

Básicamente hay 6 actividades de mucha importancia y estas son:

- Levantamiento de los procesos de negocio de la organización.
- Mapeo de los procesos de la organización al mundo SAP (blue prints).
- Especificaciones de las personalizaciones.
- Manejo de riesgos.
- Control de cambios.
- Métricas del proyecto.

Levantamiento de los procesos de negocio de la organización:

Dependiendo del tamaño del proceso, intervienen un número de personas en este. Es prácticamente imposible que una persona contenga todo el conocimiento del proceso de un área de la empresa, para evitar que se presenten visiones no completas, este levantamiento se hace en conjunto y el documento (entregable) que sale de este proceso es también revisado en conjunto.

Mapeo de los procesos de la organización al mundo SAP (blue prints):

Cuando está listo el mapa de procesos de la organización, el funcional del módulo correspondiente a esos procesos, procede a especificar los blue prints. Estos blue prints, son revisados por el usuario líder para garantizar que si cubran todas las funcionalidades expresado en el mapa de procesos de la organización.

Especificaciones de las personalizaciones:

Las especificaciones de las personalizaciones son realizadas**,** por el funcional y el usuario líder. Una vez hechas las especificaciones estas deben ser revisadas por un desarrollador, ya que deben ser los suficientemente claras para que cualquier desarrollador las pueda realizar, en caso tal que se presenten inconsistencias el desarrollador le indica al usuario líder y al funcional las partes por clarificar.

La claridad de las personalizaciones es muy importante, porque es una herramienta para que a la hora de realizar el desarrollo se haga de una forma más ágil y en caso tal que se requiriera contratar a personal externo o interno extra, los desarrollos se hagan de mejor manera.

Control de cambios:

Se lleva a cabo, la misma gestión de calidad que en las personalizaciones.

Manejo de riesgos:

Periódicamente el desarrollador líder, revisará el estado del entregable de manejo de riesgos, concerniente a la parte de las personalizaciones, quien es el responsable de velar por determinado riego y su estado actual. Esto lo puede hacer a manera personal mediante una entrevista a la persona encargada de determinado riesgo (siempre y cuando en la organización exista una cultura de responsabilidad muy grande) o en su defecto de manera escrita.

Si el riesgo no está siendo bien manejado, el desarrollador líder tomará las medidas correspondientes según el caso.

Métricas del proyecto:

Periódicamente, el desarrollador líder mira el avance individual de su equipo de desarrollo basándose en los reportes de avance del desarrollador, para saber en general como va el proyecto en general y los ajustes que deben realizar para cumplir los objetivos del cronograma. Así mismo el entregable del estado de las personalizaciones en general es revisado por el gerente del proyecto y este según ese informe tomará las medicas correspondiente para que el proyecto siga según lo estipulado.

Estándares y buenas costumbres de programación:

Con el fin de hacer programas altamente mantenibles en el tiempo, las personalizaciones se deben regir por un estándar en el nombramiento de todos los objetos necesarios para el desarrollo de las aplicaciones. Pero no basta con sólo ésto, también se deben tener buenas prácticas de programación para poder garantizar el perfomance de las personalizaciones.

Para poder garantizar ésto, se debe tener el compromiso del equipo desarrollador y periódicamente se haría una verificación por pares, para garantizar que se esté llevando a cabo efectivamente.

4.8 ESTÁNDARES DE DESARROLLO

Introducción

Este documento define todos los estándares y guías necesitadas para el desarrollo de programas en el ambiente SAP ECC. Contiene los lineamientos generales que permiten manejar el mismo idioma entre los desarrolladores y entregar productos de buena calidad.

4.8.1 Convenciones para Nombres de Objetos

Restricciones para modificación de objetos estándar SAP ECC

Ampliaciones y Parametrización: antes de pensar en una modificación al sistema estándar, se debe pensar en la posibilidad de implementar una ampliación, de igual forma una ampliación debe ser considerada únicamente cuando se comprueba que la funcionalidad requerida no es posible hacerla por parametrización o por alguna herramienta funcional.

Copias a objetos estándar SAP ECC: antes de pensar en una modificación al sistema estándar, se debe considerar la posibilidad de hacer una copia del programa estándar en un programa Z o Y al cual se le puedan aplicar los cambios requeridos.

Modificaciones a objetos estándar SAP ECC: según política SAP, ningún programa estándar debe ser modificado. Esto solo es posible hacerlo cuando es recomendado en una nota OSS o se llega a la conclusión de que el cambio no puede ser hecho usando alguna herramienta que provea el sistema y además sea crítico para el negocio. En este caso se deberá conseguir una aprobación formal por parte de los gerentes de sistemas de la compañía y documentar los cambios como sea pertinente.

Convenciones para Nombres de Objetos ABAP

SAP ha reservado rangos especiales para los nombres de los objetos no estándar. Estos nombres aseguran que sus objetos no serán reemplazados o borrados cuando SAP haga un mantenimiento o una actualización del sistema. Todos los objetos creados en el sistema deben tener en su primer carácter la letra “Z” ó “Y” para pruebas.

El primer identificador que se debe definir para un programa ABAP es el **Identificador de Aplicación (AA)** que indica quien es el módulo dueño del objeto. Algunos de los valores son los siguientes:

CO	Control (Controlling)
FI	Finanzas (Finance)
MM	Materiales (Material Management)
PP	Producción (Production Planning)
SD	Ventas (Sales and Distribution)

WM	Manejo de Bodegas (Warehouse Management)
BW	Business Warehouse
SE	SEM - Manejo Estratégico (Strategic Enterprise Management)
CX	Varias aplicaciones (Cross Application)
PS	Sistema de proyectos

Tabla 12 Algunos módulos de SAP ECC

No están incluidos todos los identificadores posibles de módulos o aplicaciones SAP ECC, por lo tanto si se necesita un nuevo identificador éste debe ser incluido por el líder de desarrollo o encargado de soporte del sistema

El segundo identificador es el **Tipo de Desarrollo (TT)** que indica la función técnica del objeto, es decir que hace el objeto dentro del sistema: es un reporte, una interface entre dos sistemas, etc.:

RE	Reporte
IE	Interface de Entrada
IS	Interface de Salida
CO	Conversión de Datos
ME	Mejora o Enhancement
FS	Formulario SapScript
FF	Formulario SmartForm
IN	Include
GE	Propósito General

Tabla 13 Tipos de desarrollo en las personalizaciones

El tercer componente es un **Número Consecutivo (nnnn)** de 4 dígitos que se usa para distinguir entre los objetos que tienen los primeros componenentes iguales. Este consecutivo debe ser manejando por módulo.

Nomenclatura para Programas (TT)

Los programas tipo “report” pueden usarse para el desarrollo de todo tipo de objetos como interfaces, mejoras o reportes. El tipo Report / OnLine solo describe el tipo programa desde el punto de vista del workbench

En el caso que un desarrollador haga una copia de un programa estándar de SAP ECC para ser usado como base, se debe usar el estándar de nombre que se define en esta sección.

Formato de Nombre: el máximo permitido por SAP son 128 caracteres.
Para crear objetos nuevos, SAP requiere que estos comiencen con las letras “Z” o “Y”, y en nuestro caso usaremos la letra “Z” para objetos formales. El formato de nombre se define de la siguiente manera:

ZAATTnnnn

Para programas de prueba que no serán transportados al ambiente de producción, se usará el prefijo YP_xxxxxxxxx como nombre de programa. Estos programas deberán ser creados como objetos locales y no deben ser asociados a órdenes de transporte (Change Request)

Ejemplos:
ZMMIE0001: interface de Entrada.
ZFICO0002: conversión de Datos del modulo de Finanzas
ZSDFS0001: formulario tipo SapScript del módulo de Ventas
YP_EJEMPLO: programa de prueba

Restricciones: son las mismas especificadas en la sección 2.1

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo →SE38 - Editor ABAP

Nomenclatura para Includes (TT)

Los includes son tipos de programas que permiten la organización y modularización de las líneas de código dentro de los desarrollos. Se recomienda su uso únicamente si el programa sobrepasa las 1000 líneas de código. El tipo de include se identifica con la letra I y puede tener las siguientes variaciones:

T	Variables y tipos de datos
F	Rutinas y subrutinas
E	Eventos
X	Otros

Tabla 14 Tipos de Include

Formato de Nombre: el máximo permitido por SAP son 128 caracteres, sin embargo para tener una mejor armonía entre los nombres usaremos solamente 11 caracteres. Esta sintaxis aplica para programas propios, en el caso de las sustituciones, se deberá mantener el nombre del include original.
El formato de nombre se define de la siguiente manera:

ZAATTnnnnI

Ejemplos:
ZMMIEPKM0001T: variables para la interface de Entrada proveniente del sistema PKMS
ZFICO0002F: rutinas para la conversión de Datos del modulo de Finanzas

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo →SE38 - Editor ABAP

Los datos claves al momento de crear un programa son: título, tipo (programa ejecutable) y posteriormente asociar a una clase de desarrollo

Nomenclatura para Programas (Tipo Module Pool y Dynpros)

Un module pool es un tipo de programa ABAP revisa y procesa las entradas del usuario durante una transacción. Este tipo de programas están compuestos por un conjunto de INCLUDEs. La mayor parte de SAP ECC esta construido con este tipo de programas, por ejemplo la creación de un material usando la transacción MM01 usa un programa tipo module pool con un conjunto de pantallas y funcionalidades.

Formato de Nombre: el máximo permitido por SAP son 10 caracteres en el siguiente formato:

SAPMZAAnnn

SAPMZ es un prefijo estándar definido por SAP y que es obligatorio.

Las pantallas en SAP son conocidas como DYNPROS (Dynamic Programs) y su identificación consta de un programa ABAP más un número de 4 dígitos. El rango de números reservados para pantallas desarrolladas y que están anexas a un programa estándar de SAP va del 9500 al 9999 (con incremento de uno en uno). Para pantallas anexas a programas desarrollados, el rango reservado es del 0001 al 0999 y del 9500 al 9999 (con incremento de 10 en 10). Por estandarización se manejaran la numeración de las pantallas comenzando desde la 0100.

Como fue dicho en el principio de esta sección un programa module pool esta compuesto por varios includes que cumplen diferentes funciones. El desarrollador no tiene que preocuparse por los nombres de estos includes pues SAP los crea basados en el nombre inicial dado al programa.

Ejemplos:

SAPMZCO001: programa principal creado por el usuario
MZCO001TOP: include creado por SAP para las declaraciones de datos
MZCO001O10: include creado por SAP para los módulos PBO (se incrementa de 10 en 10)
MZCO001I10: include creado por SAP para los módulos PAI (Se incrementa de 10 en 10)
MZCO001F10: include creado por SAP para las subrutinas (Se incrementa de 10 en 10)
MZCO001H10: include creado por SAP para los módulos POH (Se incrementa de 10 en 10)
MZCO001V10: include creado por SAP para los módulos POV (Se incrementa de 10 en 10)

Pantallas: 0100, 0110 y 0120

Restricciones: se debe tener en cuenta que la generación automática de los nombres solo es exitosa si el programa principal fue creado siguiendo el estándar de nombre previamente definido. Recuerde que otros objetos dependen del nombre de programa que se crea: las pantallas o dynpros, los elementos de texto, la documentación, los menús y los títulos entre otros.

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Resumen →SE80 - Object Navegator

Nomenclatura para Grupos de Función

Un grupo de función es un objeto que concentra uno o más módulos de función que tiene algo en común para compartir.

Formato de Nombre: la máxima longitud para un grupo de funciones es de 26 caracteres, sin embargo para propósitos de estandarización solo usaremos 6 caracteres. El formato es el siguiente:

ZAAnnn

Donde AA es el código de la aplicación y nnn es un número consecutivo. La creación de un grupo de funciones resulta en la generación automática de un programa llamado SAPLZAAnnn, donde ZAAnnn es el nombre del grupo de funciones.

Ejemplos:

ZFI001: grupo de funciones para conversiones de Controlling (CO)
ZMM002: grupo de funciones para reportes de Materiales (MM)

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo →SE37 - Biblioteca de Funciones

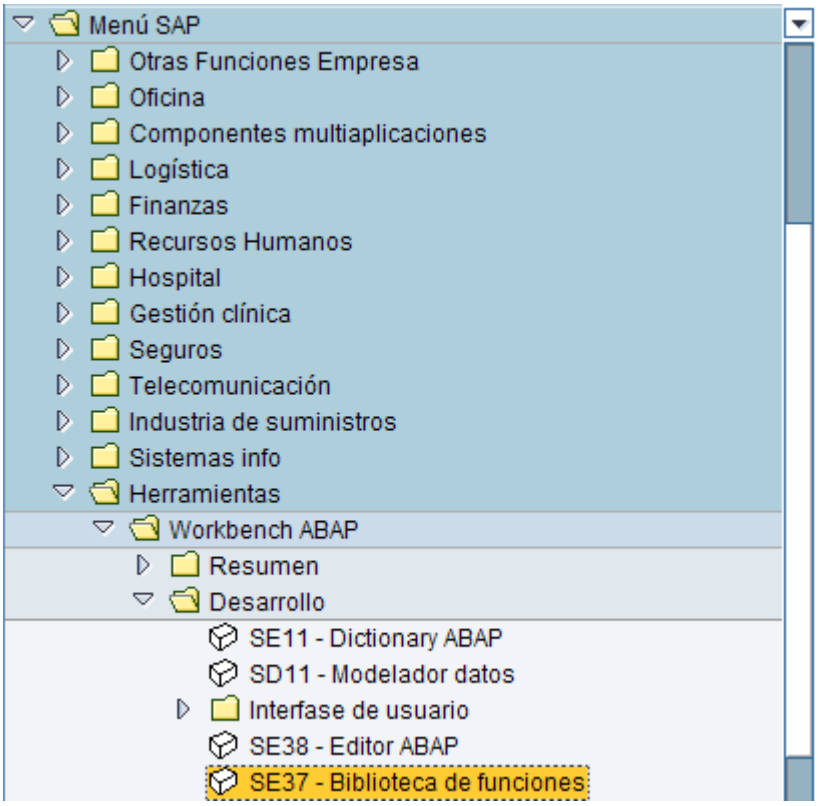


Ilustración 28 Ingreso Biblioteca de funciones

Una vez en la pantalla de biblioteca de funciones, se busca la opción de crear grupo por el menú principal: Pasar a → Gestión gr. funciones → Crear Grupo

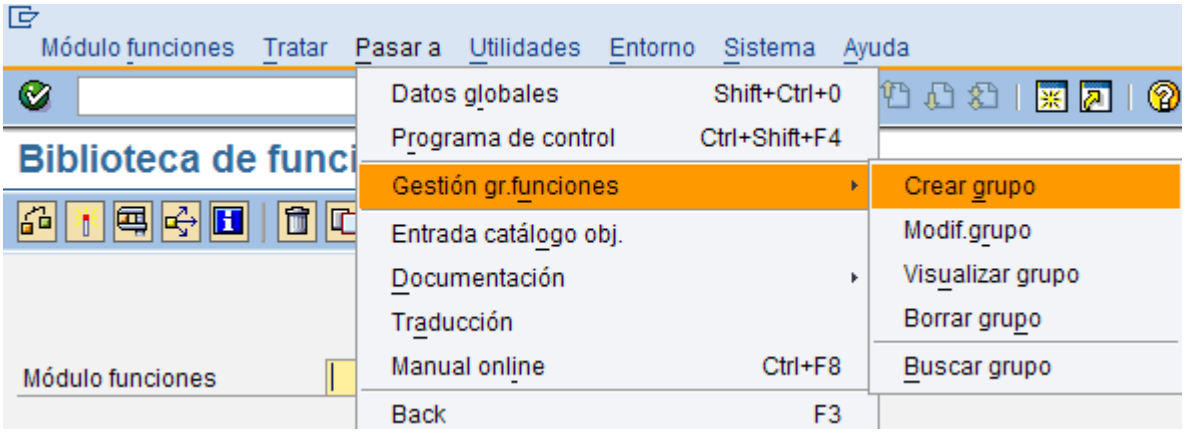


Ilustración 29 Creación grupo de funciones

Nomenclatura para Módulos de Función

Un módulo de función es un programa escrito en ABAP que conforma una unidad lógica y que permite ser usado para diferentes propósitos en diferentes programas. Son una forma de modularizar la programación que comúnmente se conocen únicamente como “Funciones”.

Formato de Nombre: la máxima longitud para un modulo de funciones es de 30 caracteres, de los cuales los primeros 5 “Z_AA_” están reservados, los restante 25 puedes ser usados libremente. El formato es el siguiente:

Z_AA_xxxxxxxxxxxxxxxxxxxxxxxxx

Donde AA es el código de la aplicación y xxxxxx… está abierto para caracteres alfanuméricos y especiales

Ejemplos:
Z_SD_CONSULTAR_PROVEEDOR
Z_CX_CONVERTIR_FECHA_PROVEEDOR

Restricciones: los módulos de función son programas que además de poder ser llamados desde los programas ABAP, pueden ser llamados desde otras aplicaciones a través del sistema operativo usando “remote function calls” (RFC). Un modulo de función solo puede pertenecer a un grupo de funciones, pero un grupo de funciones puede tener n módulos de función.

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo →SE37 - Biblioteca de Funciones

Nota: Debe documentarse el módulo de función ya que es un componente reutilizable y otros desarrolladores podrían utilizarlo. También deben documentarse los parámetros de la función.

Nomenclatura para Códigos de Transacción

Las transacciones son el objeto más común dentro de la lógica SAP ECC. Todas las funcionalidades del sistema poseen una transacción para lograr ejecutar los programas de una manera más ágil (MM01, F-02, etc.). La transacción llama el programa especificado para que el usuario pueda interactuar con la aplicación. Existen 5 tipos de transacciones que pueden ser creadas dependiendo de la necesidad:

- Programa y Dynpro (transacción de diálogo): usadas para programas tipo dynpro o module pool
- Programa e imagen de selección (transacción de report): para objetos tipo report
- Método de una clase: para programación orientada a objetos
- Transacción con variantes (transacción de variante)
- Transacción con parámetros (transacción de parámetros)

Formato de Nombre: la máxima longitud para transacciones es de 20 caracteres, para el caso del proyecto solo usaremos 6. El formato es el siguiente:

ZAAnnn

Donde AA es el código de la aplicación y nnn es el número concecutivo para esa aplicación. La asignación del concecutivo se hace revisando desde la transacción SE93, el concecutivo actual y sumando uno (1).

Ejemplos:
ZFI001: transacción para un programa de finanzas
ZSD010: transacción para un programa de ventas

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → Otras Herramientas → SE93 - Transacciones

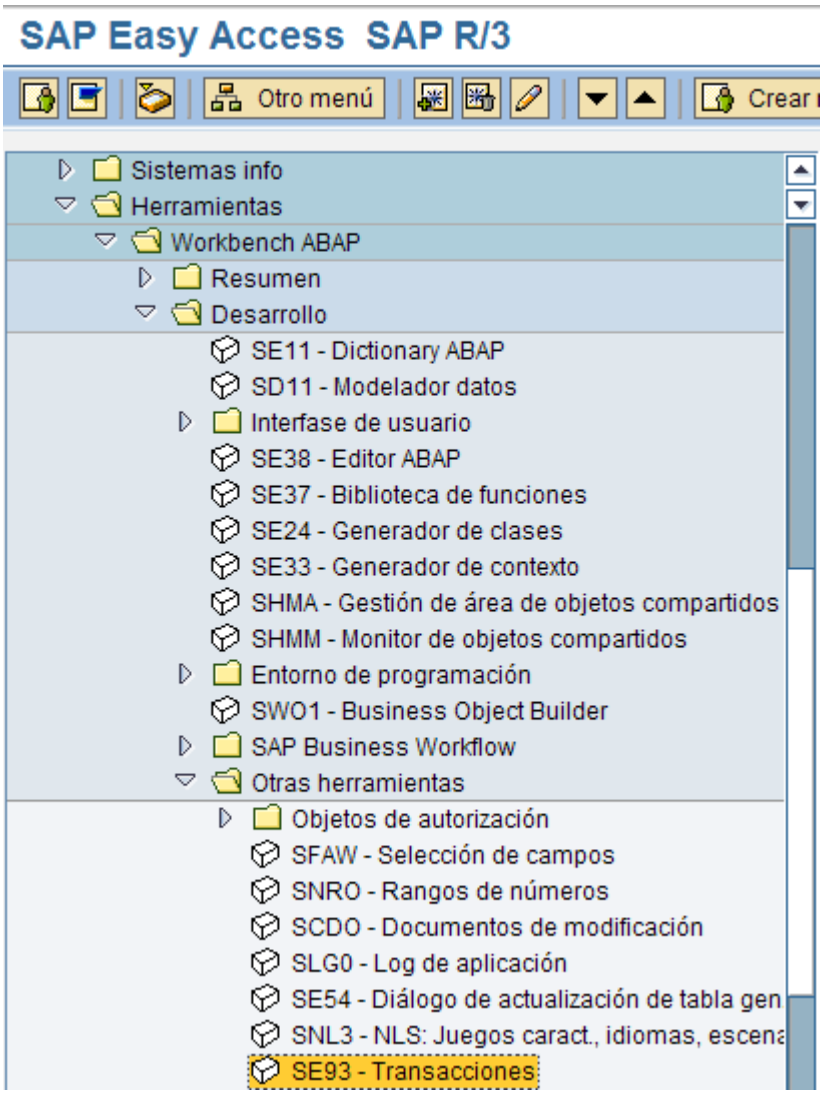


Ilustración 30 Creación de transacciones

Nomenclatura para Paquetes

Un paquete es un objeto utilizado para colocar juntos objetos de desarrollo que tienen algo en común y que son mutuamente dependientes uno del otro. El paquete es usada para corregir y transportar estos objetos de desarrollo juntos como una unidad, por ejemplo un programa que tiene tablas Z, transacciones y un modulo de función debe ser condensado en una sola clase de desarrollo.

Formato de Nombre: se debe definir con un máximo de 3 caracteres de la siguiente manera:

ZAA

Un ejemplo de los paquetes sería:

ZCO	Control (Controlling)
ZFI	Finanzas (Finance)
ZMM	Materiales (Material Management)
ZPP	Producción (Production Planning)
ZBW	Business Warehouse
ZSD	Ventas (Sales and Distribution)
ZWM	Manejo de Bodegas (Warehouse Management)
ZCX	Varias aplicaciones (Cross Application)
$TMP	Para objetos locales (no transportables)

Tabla 15 Ejemplos de paquetes

Restricciones: los paquetes son previamente creadas por el coordinador de Basis y ABAP, los desarrolladores NO tienen que crear nuevos paquetes. Los objetos que no serán transportados a ambientes de calidad y producción deben ser creados como locales bajo la clase $TMP.

Nomenclatura para Variantes

Una variante es un conjunto de valores usados para la ejecución de un programa. Estos valores son colocados en los parámetros y posteriormente grabados como variante. Las variantes tienen múltiples usos, sin embargo son comúnmente usadas para ejecutar programas más rápidamente y para trabajar con los Jobs.

Formato de Nombre: se debe definir con un máximo de 10 caracteres de la siguiente manera:

ZAAVSS_nnnn

Donde SS es el tipo de programa al que se le crea la variante:

IE	Interface de Entrada
IS	Interface de Salida
RE	Reporte
PE	Programa Estándar

Tabla 16 Tipo de programa al que se le crea la variante

Las variantes pueden ser transportadas como cualquier objeto del Workbench.

Ejemplos:

ZSDVR_0001, variante de reporte de SD
ZMMVP_0001, variante de programa estándar de MM

Navegación: las variantes son creadas directamente en la ejecución del programa o en el editor de ABAP (SE38).

Nota: Los usuarios líderes y usuarios del sistema en productivo, son los que deben conocer el estándar de nombramiento de las variantes, ya que son los que finalmente las van a usar.

Nomenclatura para Jobs (Trabajos de Fondo)

Los trabajos de fondo o jobs permiten la ejecución en fondo de programas (internos SAP ECC o externos) ya sea por una vez o de manera repetitiva. Para crear un Job es necesario definir los siguientes datos:

- Descripción del Job
- Prioridad del Job (Job Class)
- Servidor en el cual será ejecutado el Job (en el caso de existir mas de uno)
- Numero de pasos que tendrá el Job. Un paso significa un programa a ejecutar.
  - Nombre del programa
  - Nombre de la variante
  - Usuario autorizado
- Definir si el programa es externo o interno
- Cuando y que tan seguido se debe ejecutar el Job
- También se pueden definir dependencias entre Jobs

Formato de Nombre: se debe definir con un máximo de 9 caracteres de la siguiente manera:

ZAASSFnnnn

Donde SS es el tipo de programa que se ejecutará con el Job.
Donde F es la frecuencia con que se debe ejecutar el Job:

H	Cada Hora
D	Cada Día
S	Cada Semana
M	Cada Mes
C	Cada Trimestre (3 meses)
A	Cada Año
N	Cuando se necesite
Z	Otros

Tabla 17 Frecuencia de ejecución de un JOB

Ejemplos:
ZFIRED0001, reporte de FI con ejecución diaria
ZMMISM0001, interface de salida de MM con ejecución mensual

Navegación: Sistema → Servicios→ Job → Definición Job (SM36)

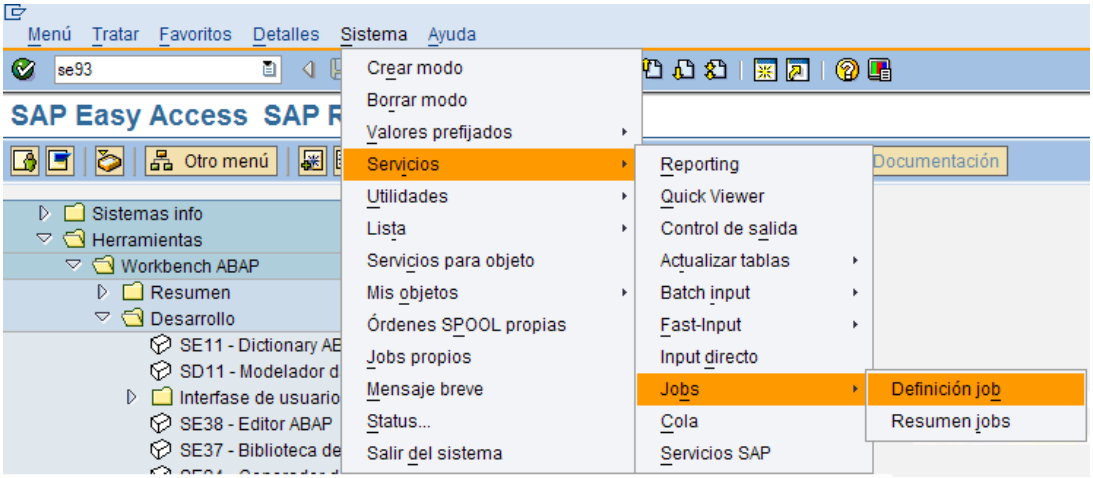


Ilustración 31 Ingreso definición de Job

Nomenclatura para Clases e Interfaces/ Clases Proxy

Las clases son declaraciones o abstracciones de objetos, lo que significa, que una clase es la definición de un objeto. Cuando se programa un objeto y se definen sus características y funcionalidades, realmente se programa una clase.

Formato de Nombre: la máxima longitud para una clase es de 30 caracteres, de los cuales los primeros 5 “ZCL_” están reservados, los restantes pueden ser usados libremente. El formato es el siguiente:

ZCL_AA_xxxxxxxxxxxxxxxxxxxxxxx

Donde AA es el código de la aplicación y xxxxxx… está abierto para caracteres alfanuméricos y especiales.

Para clases de excepción, se deberá cambiar el prefijo “ZCL” por “ZCX”:

Ejemplos:

ZCL_SD_CALCULO_PRECIOS
ZCX_MM_MVTO_INVENTARIO

Para interfaces, se deberá cambiar el prefijo “ZCL” por “ZIF”:

Ejemplos:

ZIF_SD_PRECIOS
ZIF_ INVENTARIO

Nomenclatura para Grupos de Puntos de Verificación

Para simplificar el debugging y el análisis de programas complejos, se pueden implementar breakpoints y aserciones en programas que están inactivos por defecto. Para ello, los breakpoints y las aserciones se asignan a un grupo de puntos de verificación mediante el cual pueden ser activados en caso necesario. Las parametrizaciones de activación del grupo de puntos de verificación se transfieren al comando BREAK-POINT o ASSERT.

El nombre del grupo de puntos de verificación deberá comenzar con Z o Y, ya que debe estar en el espacio de nombres del cliente.

Formato de Nombre: la máxima longitud para un grupo es de 30 caracteres. El formato es el siguiente:

NOMBRE_PGM_BRKnn

Donde:
NOMBRE_PGM: Nombre del programa ABAP, o función que contiene los puntos de verificación.
BRK: Constante para indicar que es un grupo de Breakpoints.

nn: consecutivo

Ejemplo:

ZMMIE0001_BRK01: Grupo de Breakpoints 01 para el programa Interface de Entrada, del modulo Materiales

ZFICO0002_BRK02: Grupo de Breakpoints 02 del programa de Conversión de Datos del modulo de Finanzas

Nomenclatura para Vistas de Actualización y Cluster de Vistas

Al crearse vistas de actualización o mantenimiento para tablas o vistas del diccionario de ABAP, el sistema nos solicita un grupo de función en donde se crearán los programas, funciones y dynpros necesarias para el mantenimiento de dicha tabla.

Las vistas de actualización son programas que el sistema crea en el grupo de función que le definamos al momento de su creación. Debe indicarse mantenimiento en pantalla sencilla en la mayoría de los casos. Solo para casos en que el registro tiene demasiados columnas (mas de 10), se puede realizar un mantenimiento en dos dynpros.

Debe crearse un grupo de función para cada vista de actualización, es decir, en el mismo grupo de función no pueden existir más de una vista de actualización, esto con el fin de minimizar los esfuerzos e inconvenientes durante el mantenimiento de las aplicaciones.

Formato de Nombre: El nombre del grupo de función para la vista de actualización de una tabla o vista, debe ser el mismo nombre de la tabla o vista:

NombreTabla o NombreVista

Ejemplos:

ZTFI0001: grupo de funciones para la vista de actualizacion de la tabla ZTFI0001
ZVMM0002: grupo de funciones para la vista de actualización de la vista ZVMM0002

Los cluster de vistas son la agrupación de dos o más vistas de mantenimiento con el fin de centralizar el mantenimiento de tablas muy relacionadas (en lo posible relacionadas con Foreign Key). Estos objetos son creados por la transacción SE54.

Formato de Nombre: El nombre de las vistas cluster es el siguiente:

ZCVnnnn

Ejemplos:

ZCV0001: Cluster vista para tablas de la solucion de etiquetas
ZCV0002: Cluster vista para tablas de bancos

Nomenclatura para Proyectos de Ampliación y Productos BTE

Para implementar ampliaciones al estándar de mySAP usando Customer Exit o User exit , es necesario crear un proyecto de Ampliación (transacción CMOD) que contenga la ampliación correspondiente de mySAP, la nomenclatura para este tipo de objeto es la siguiente.

Formato de Nombre:

ZAAPnnnn

Donde AA es el código de la aplicación, P constante para indicar que es un Proyecto de ampliación, y nnnn es un número consecutivo para Proyectos de Ampliación y otro consecutivo para Productos BTE.

Ejemplos:

ZFIP01: Proyecto de ampliación número uno del módulo FI
ZMMP02: Proyecto de ampliación número dos del módulo MM

Para implementar ampliaciones por medio de BTE (Business Transaction Events), se debe crear un Producto por la transacción FIBF asociándole la o las BTE's que requerimos implementar. La nomenclatura de nombre para el producto BTE es la misma de Proyectos de ampliación.

Nomenclatura para Proyectos LSMW

Como norma en el proyecto, todas las conversiones se realizarán por la herramienta LSMW (Legacy System Migration Workbench), los objetos creados en esta herramienta están organizados jerárquicamente en proyecto, subproyecto y objeto de migración. Esta estructura permite la organización de las conversiones, agrupando los objetos por módulo, funcionalidad o proceso.

Los nombres definidos para cada uno de los item de la estructura del LSMW son los siguientes:

- **Proyecto LSMW**

Agrupa a todas las migraciones y conversiones del proyecto
Nombre: "Nombre del proyecto"
Descripción: "Migración de Datos Nombre del proyecto"
Este proyecto será utilizado para las cargas por LSMW creadas por el equipo de Desarrollo.

- **Subproyectos LSMW**

Agrupa todas las conversiones de cada módulo. Los subproyectos creados son los siguientes:

CO Conversiones Módulo CO
FI Conversiones Módulo FI
MM Conversiones Módulo MM
PP Conversiones Módulo PP
SD Conversiones Módulo SD
WM Conversiones Módulo WM

- **Objetos LSMW**

El objeto de conversión, el identificador de ese objeto es el mismo ID del requerimiento. Y en la descripción debe ir el titulo del requerimiento.

Formato de Nombre:

IDRequerimiento

NOTA: Generación de Juego de Datos

Como regla, las conversiones de cualquier tipo (LSMW, Report, etc), deberán generar Juegos de Datos (Batch Input Session) para su ejecución por la transacción SM35. El Juego de datos deberá ser creado como hold, es decir, que no se elimine cuando sea ejecutado, sino que permanezca para verificar su log y su información de carga, además el juego de datos deberá tener el mismo nombre del objeto LSMW.

Nomenclatura para Implementación de BADI

Si el desarrollo demanda la ampliación de la funcionalidad estándar de mySAP por medio de una BADI, se deberá crear la implementación de la BADI.

El nombre de la BADI implementada deberá corresponder con la siguiente nomenclatura:

Formato de Nombre:

ZAA_BADI_nn

Donde AA es el código de la aplicación, BADI es el nombre de la badi que se está implementado, y nn es un número consecutivo. El máximo tamaño para el nombre de implementación de una BADI es 20 caracteres, si el nombre da mayor de 20, se deberá recortar al nombre de la BADI.

Ejemplos:

ZFI_ACEPS_BAPIPRE_01: Implementación 01 de la BADI
ACEPS_BAPIPREDOC_MOD, del módulo FI
ZSD_BADI_SD_CM_02: Implementación 02 de la BADI_SD_CM del módulo SD

Nomenclatura para Formularios (SAPScript y SmartForms)

Los formularios son un tipo de desarrollo utilizado para la impresión de documentos empresariales especiales (diferentes a reporte) y que en la mayoría de los casos son invocados automáticamente desde programas estándar cuando se presenta una salida de un tipo de mensaje específico.

Los formularios pueden ser implementados usando dos herramientas: SAPScript y SmartForms. Smartforms es una herramienta más nueva y más completa que SAPScript y los nuevos formularios diseñados por SAP están siendo diseñados con base en esta herramienta; sin embargo, en el estándar de SAP todavía existen muchos formularios estándar implementados en SAPScript y que deben seguir siendo soportados.

Formato de Nombre: el máximo permitido por SAP son 128 caracteres, sin embargo para tener una mejor armonía entre los nombres usaremos solamente 10 caracteres. Para crear objetos nuevos, SAP requiere que estos comiencen con las letras “Z” o “Y”, y en nuestro caso usaremos la letra “Z” para objetos formales. El formato de nombre se define de la siguiente manera:

Para los SapScript

ZAAFSnnn

Para los SmartForms

ZAAFFnnn

Ejemplos:

ZMMFS001 Formulario de Orden de Compra del modulo MM

Nota: Se hace una diferenciación en los nombres de los SapScripts y los Smartforms, para evitar malos entendidos ante la posibilidad de poner el mismo nombre a un formulario hecho en SapScript y otro en Smartform.

4.8.2 Convenciones para Objetos de Diccionario de Datos

Tablas

Se recomienda que todas las tablas sean definidas como tablas transparentes, sin embargo dependiendo de la problemática que se tenga se debe analizar la posibilidad de definir tablas pool o cluster. Se deben tener en cuenta las recomendaciones SAP sobre el tipo de tabla a usar, tamaño, si soporta memoria intermedia, entre otros. Además se deben usar en lo posible dominios y elementos de datos estándar SAP ECC. Sino estos deben ser definidos según los estándares que se verán en esta sección.

Formato de Nombre:

ZTAAnnnn

Donde AA es el código de la aplicación y T es una constante que indica que el objeto es una Tabla. Los 4 caracteres nnnn son el identificador numérico consecutivo.

Ejemplos:

ZTFI0001: tabla transparente de FI
ZTCO0001: tabla transparente de CO

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → SE11 – Dictionary ABAP

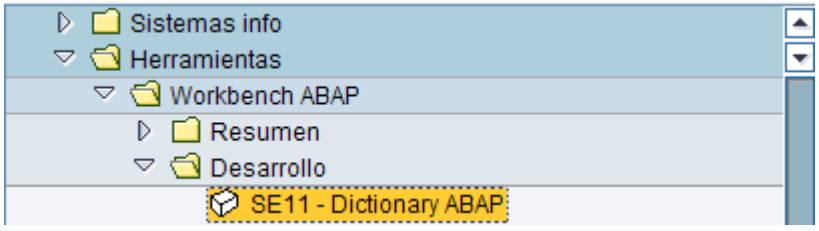


Ilustración 32 Ingreso Diccionario

Algunas tablas Z podrán tener Vista de Mantenimiento, de acuerdo al requerimiento de la Especificación funcional.

Vistas

Vistas son tablas transparentes que contienen datos de múltiples tablas, tal y como trabaja el concepto de “join”.

Formato de Nombre:

ZVAAnnnn

Donde AA es el código de la aplicación y V es una constante que indica que el objeto es una Vista. Los 4 caracteres nnnn son el identificador numérico consecutivo.

Ejemplos:

ZVMM0001: vista de MM
ZVSD0001: vista de SD

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → SE11 – Dictionary ABAP

Estructuras

Una estructura se define para ser usado como tipo de datos. No contiene datos y no se define físicamente en la base de datos. Sin embargo su construcción es muy similar a la de una tabla, exceptuando los datos técnicos e llaves. En lo posible se deben usar dominios y elementos de datos estándar SAP ECC. Sino estos deben ser definidos según los estándares que se verán en esta sección.

Formato de Nombre:

ZEAAnnnn

Donde AA es el código de la aplicación y E es una constante que indica que el objeto es una Estructura. Los 4 caracteres nnnn son el identificador numérico consecutivo.

Ejemplos:

ZEFI0001: estructura de FI
ZESD0001: estructura de SD

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → SE11 – Dictionary ABAP

Dominios

Un dominio puede ser definido como los datos técnicos de un campo en una tabla. Contiene los datos que pueden ser asignados a ese valor, por ejemplo tipo carácter de 1. También se pueden definir valores para ese campo, como “X”, “Y” y “Z”. Un dominio puede ser usado en múltiples tablas y campos.

Formato de Nombre:

ZZxxxxxxxxxx

Donde xxxxxxxxxx es abierto y puede contener hasta 10 caracteres.

Ejemplos:

ZZTIPOTRANS: dominio para un campo tipo transporte (Carácter de 4)
ZZCOLOR: dominio para un campo código de color (Carácter de 2)

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → SE11 – Dictionary ABAP

Elementos de Dato

Un elemento de dato o dominio semántico describe el papel que juega un dominio en una tabla, llevando la información semántica o simplemente la descripción

Formato de Nombre:

ZZxxxxxxxxxx

Donde xxxxxxxxxx es abierto y puede contener hasta 10 caracteres.

Ejemplos:

ZZTIPOTRANS: elemento de datos con descripción “Tipo de Transporte”
ZZCOLOR: elemento de datos con descripción “Color de Producto”

Restricciones: son las mismas especificadas en la sección 2.1

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → SE11 – Dictionary ABAP

Nota: Todos los elementos de datos Z que se creen, deberán tener etiquetas o labels y documentación que será usada para la funcionalidad de F1.

Campos

Elemento principal de una tabla. Deben ser descriptivos y no exceder los 10 caracteres, es formato abierto.

Formato de Nombre:
xxxxxxxxxx

Si son campos de append structure, deben tener el formato especial:

Formato de Nombre:

ZZxxxxxxxxxx

Debe coincidir con los nombres definidos para Elemento de Datos y Dominio.

Ejemplos:

Elemento de Dato: ZZTIPOTRANS
Campo: TIPOTRANS

Restricciones: no aplican restricciones especiales

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → SE11 – Dictionary ABAP

Índices

Los índices son un elemento que acelera la búsqueda de información ya sea en una tabla estándar o una tabla Z.

Formato de Nombre:

Znn

Donde nn es un número consecutivo dependiente de la tabla.

Ejemplos:

Tabla LIPS →

Z01, para campo MATNR
Z02, para campo VGBEL

Restricciones: los índices no son recomendados en todos los casos, un análisis detallado debe ser hecho previamente. La ayuda del equipo de Basis puede ser requerida.

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → SE11 – Dictionary ABAP (Botón de Índices)

Ayudas o Search Help

Antiguamente conocidos como “match codes” los search help son objetos que proveen una ayuda al momento de introducir un dato en un campo. Presionando F4 o el triangulo al final del campo, el sistema muestra un grupo de opciones que limitan el valor que puede ser digitado en el campo. El usuario se limita a dar doble click o Enter en valor deseado.

Existen dos tipos de objetos de ayuda, elementales y colectivos. El elemental es el concepto básico del objeto, el cual provee información de ayuda para ser introducida en un campo. Un search help se define con: método de selección (como el objeto obtendrá el valor (de una tabla, una vista, etc.), la interface, es decir los parámetros que le darán entrada de datos a la búsqueda y por último el comportamiento en línea del objeto. El tipo colectivo se define como un conjunto de search help elementales donde el usuario puede escoger varias alternativas que le ayuden a encontrar la información deseada.

Formato de Nombre:

ZHAAxxxxxx

Donde xxxxxxxxxx es abierto y puede contener hasta 26 caracteres. Generalmente tiene correspondencia con el nombre de la tabla base de la cual extrae los datos.

Ejemplos:
ZHMMTIPOTRANS

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → SE11 – Dictionary ABAP

Table Types

Los tipos de tabla globales son utilizados para definir tipos globales de tabla interna, con el fin de declarar data objects de tipo tabla interna en los programas ABAP. En el Tipo

Tabla se especifica internamente la llave y su tipo asi como la cantidad inicial y el tipo de tabla interna (STANDARD, SORTED o HASHED).

Formato de Nombre:

ZTTAAnnnn

Donde AA es el código de la aplicación y TT es una constante que indica que el objeto es un Tipo Tabla. Los 4 caracteres nnnn son el identificador numérico consecutivo.

Ejemplos:

ZTTMM0001: Tipo tabla interna para datos de MM
ZTTSD0001: Tipo tabla interna con datos de SD

Navegación: Menu SAP → Herramientas→ Workbench ABAP → Desarrollo → SE11 – Dictionary ABAP

Objeto de Bloqueo

Los objetos de bloqueos son necesarios para realizar bloqueos lógicos a registros de tablas de la base de datos. Estos bloqueos no realizan un bloqueo físico a los registros, por lo tanto se deberá consultar si un registro o conjunto de registros están siendo bloqueados por otro usuario y otra aplicación antes de intentar trabajar con él.

Para esto se generan dos módulos de función automáticamente al momento de activar el objeto de bloqueo.

Formato de Nombre:

EZ_AAnn

Donde AA es el código de la aplicación y nn 2 dígitos para el consecutivo.

4.8.3 Convenciones Generales para Programación en ABAP

El propósito de esta sección es definir las guías generales y nomenclatura que se usaran como lineamientos dentro del lenguaje de programación ABAP, especialmente para desarrollos tipo “online” o “report”. Más adelante se definirán los lineamientos para programación en module pool (dynpros).

Atributos del Programa

Al crear un programa se deben definir los atributos generales para dicho programa. Estos atributos son obligatorios y dan dirección al programa en función de su objetivo:

***Tipo*:**

L: Reporte (ejecutable)
I: Include (no ejecutable)
M: Module Pool
V: Update report

***Status*:**

Siempre usar K (Programa Cliente Productivo)

***Aplicación*:**
F: Finanzas
C: Costos
L: Logística
S: Ventas
B: Basis

***Aritmética de Punto Fijo*:**

Se debe activar esta opción a menos que solo se trabaje con números enteros

***Documentación*:**

Todos los programas deberán tener documentación explicita a nivel del objeto, con el fin que la descripción, proposito, prerequisitos, etc, aparezcan cuando el usuario de clic sobre el icono  cuando ejecute el programa. Esta documentación deberá copiarse de la especificación funcional y/o técnica.

Variables y Tablas Internas

Las variables se definen principalmente con la sentencia Data, sin embargo también pueden ser definidas como parámetros con la sentencia Parameters o Select Options. Los siguientes tipos de variables serán diseccionados en esta sección: constantes, variables de trabajo, parámetros, select-options y rangos.

Se debe tener cuidado con el uso de los caracteres "_" (guión bajo) y "-" (guión), el guión bajo se usa para separar palabras de una misma variable, el guión se usa para separar una tabla de sus campos y hacer referencia a ellos.

Variable	Ambito + "_" + Tipo + "_" + libre Ej: "G_E_CANTIDAD"	Variables de propósito general. Ver los tipos y ámbito en el siguiente cuadro
Tabla Interna	"G_TI_" + Libre "L_TI_" + Libre	Tablas Internas para guardar grupos de datos (tipo registro)
Estructura	"G_ES_" + libre "L_ES_" + libre	Estructura para almacenar un solo registro de datos
Parámetro	"PA_" + libre	Variables para entrar datos antes de la ejecución programa. Permiten máximo 8 caracteres
Select-Option	"SO_" + libre	Variables (tipo tabla interna) para entrar datos de rango antes de la ejecución del programa. Permite máximo 8 caracteres
Field Symbol	"<G_FS_" + libre + ">" "<L_FS_" + libre + ">"	
Constante	"G_CTE_" + libre "L_CTE_" + libre	Variables tipo constante no pueden cambiar durante la ejecución de los programas
Variable de referencia	"G_R_" + libre "L_R_" + libre	Variable de referencia a objetos de clases.
Tipo Local	"G_TP_" + libre "L_TP_" + libre	Tipo de datos global Tipo de datos local
Ranges	"RA_"+ libre	Variables (tipo tabla interna) para entrar datos de. Permite máximo 8 caracteres

Tabla 18 Estandar en las definiciones

Los Tipos son los siguientes:

C	Carácter
F	Fecha
U	Punto Flotante
E	Entero
N	Texto Numérico
P	Número empaquetado

H	Hora
X	Byte (Hexa)
R	Rango

Tabla 19 tipo de datos

Los Ambitos son los siguientes:

L	Local
G	Global
I	Instancia
A	Argumento

Tabla 20 Ambito de los datos

Ejemplos:

Variable tipo fecha y ámbito global:
G_F_FECHA_CREACION

Variable tipo carácter y ámbito local:
L_C_PLANTA

Tipo de Datos Global:
G_TP_ORDEN

Parámetro:
PA_ARCHIVO

Select-Option:
SO_FECHA_COMPRA

Field Symbol local:
<L_FS_DOCUMENTO>

Constante local:
L_CTE_AE10

Tabla Interna global:
G_TI_CLIENTES_KNA1

Module Pool (Dynpros)

A continuación se definen los estándares de nombres para los objetos propios de la programación en Module Pool y Dynpros:

Text Field	“TF_” + libre
Group Box	“GB_” + libre
Radio Button	“RB_” + libre
Button	“BT_” + libre
Sub-Screen	“SS_” + libre
TabStrip	“TS_” + libre
Table Control	“TC_” + libre
GUI Status	“GS_” + Numero consecutivo (4)
GUI Title	“GT_” + Numero consecutivo (4)

Tabla 21 Estandares elementos Modul Pool y Dynpros

Parámetros en Rutinas y Funciones

Cuando se use modularización de código con rutinas / subrutinas y módulos de función, los parámetros que se pasados deben tener una nomenclatura especial:

Rutinas Internas	USING → “P” + Tipo + “_” + libre CHANGING → “P” + Tipo + “_” + libre TABLES → “P” + Tipo + “_” + libre
Funciones	IMPORT → “I_” + libre EXPORT → “E_” + libre CHANGING → “C_” + libre TABLES → “T_” + libre

Tabla 22 Paso de parametros

Ejemplos:

```
PERFORM calculo_precio USING g_e_cantidad
                             g_c_nombre
……
……
……
FORM calculo_precio USING p_e_cant type i
                          p_c_descrip type c.
……
……
ENDFORM.
```

El nombre de las rutinas y las funciones no tiene una nomenclatura especial. Solo deben ser nombres explícitos que informen la función del objeto.

NOTA: Es requerido y obligatorio tipificar todos los parámetros de las subrutinas y los módulos de función.

Clases de Mensajes

Todos los mensajes utilizados en las aplicaciones del cliente, deberán ser creados en clases de mensajes Z, deberá crearse una clase de mensaje por cada módulo o aplicación del sistema. Los mensajes creados deberán reutilizarse lo máximo posible entre los diferentes desarrollos.

Formato de Nombre:

ZAAnn

Ejemplos:

ZMM01: Clase de Mensaje 01 para el módulo MM
ZCO02: Clase de Mensaje 02 para el módulo CO

Los tipos de mensaje son definidos en el **apéndice de Mensajes**. La numeración para los mensajes tiene 3 caracteres numéricos únicos por cada clase de mensaje nnn (del 000 al 999).

De ser necesario, se deberá extender el texto del mensaje con una explicación más completa, quitando la marca de “Autoexplicable” e ingresando la descripción correspondiente al mensaje.

Anexos:

Ver anexos del 5 al 8

4.8.4 Guía de Programación ABAP

En esta sección se definen las guías y reglas de programación en cuanto a orden, estructura, manejo de includes, entre otros.

Estructura del Programa

Todos los programas deben tener una estructura determinada (esqueleto) para guardar la uniformidad y el orden.

Orden del Código

Las principales reglas acerca del ordenamiento por secciones en un programa son las siguientes:

- El orden general del programa es el siguiente:
  - Encabezado General del Programa
  - Declaración de Variables
    - Constantes
    - Tipos locales
    - Estructuras Tables
    - Estructuras
    - Tablas
    - Variables
    - Parámetros y Select-Options
    - Ranges.
  - Eventos
  - Subrutinas
- Los eventos usados deben ser resaltados para que puedan ser fácilmente identificables (START-OF-SELECTION, END-OF-SELECTION, etc.). Los eventos no usados deben ser eliminados del programa.
- Todas las subrutinas creadas (FORMs), deben ser ubicadas al final del programa después del evento END-OF-SELECTION, en la sección SUBRUTINAS.

Comentarios

Los comentarios hacen parte de la documentación de un programa haciendo más fácil la lectura e interpretación de las líneas de código que lo componen.

Los comentarios deben ser usados en las siguientes condiciones:

- Antes de cada sección de código, ya sea grande o pequeña
- Para clarificar lógica que sea compleja o confusa
- Para modificar código ya existente
- Al principio de programa como encabezado general para explicar aspectos como desarrollador, fecha, identificador, propósito, etc.
- Al principio de cada subrutina o modulo de función para explicar el su propósito

Formateo de Sentencias

El editor del lenguaje de programación ABAP da la opción de hacer un formateo de sentencias automático conocido como “Pretty Printer” (Shift-F1). Este hace cambios de mayúsculas a minúsculas y viceversa según el caso, además de organizar la identificación de las líneas de código.

Manejo de Includes

Los includes son tipos de programas que permiten la organización y modularización de las líneas de código dentro de los desarrollos. Se recomienda su uso únicamente si el programa sobrepasa las 1000 líneas de código.

Mensajes

Todos los mensajes que arroja un programa deben ser manejados a través de clases de mensajes ya sea estándar o creadas por el programador. Para mayor información ver el apéndice E.

Formateo de Reportes

Las líneas en blanco dentro de un reporte deben ser creadas usando la sentencia SKIP en lugar de usar la sentencia WRITE /. Los textos fijos deben ser creados usando numeración que hace que dichos textos sean lenguaje independiente, ejemplo:

WRITE: ‘Nombre del Cliente’ (001).

El texto 001 debe ser definido como ‘Nombre del Cliente’ (dando doble clic sobre el número) y puede ser traducido a diferentes lenguajes según sea la necesidad.

El formateo de reportes se reduce sustancialmente si se usa ALV (Abap List Viewer), sin embargo esto depende del perfil de los desarrolladores, de su conocimiento de esta herramienta, y del tipo de layout del reporte.

SQL Nativo

El lenguaje de programación nos da la posibilidad de trabaja con SQL nativo, sin embargo esto solo es recomendado bajo condiciones especiales. Contacte a su coordinador del equipo de desarrollo antes de usar este tipo de instrucciones.

Subrutinas y Módulos de Función

Cuando un programa usa una función, carga en memoria el grupo de funciones a la que pertenece ocupando memoria del sistema en proporción al tamaño del grupo de funciones. Por tal razón cuando se creen funciones nuevas se debe analizar si esta debe o no incluirse en un grupo de funciones existente. También se debe considerar la creación de includes.

El uso de variables locales en subrutinas es recomendado para evitar la creación de estas variables en la memoria global del programa y para que esta memoria se libere cuando se finalice la ejecución de la subrutina.

Como regla general, todas las subrutinas deben tener una declaración de parámetros de entrada y salida (interface) usando las instrucciones USING y CHANGING que permiten el paso de parámetros desde y hacia la subrutina, asi como tipificarlos completamente. También debe especificarse si los parámetros son pasados por valor, por referencia o por valor y resultado.

Objetos de Autorización

El uso de autorizaciones depende en su totalidad del programa que se esté desarrollando. La autorización más común es la que esta inherente a la ejecución del programa. Por ejemplo un reporte que muestra información no necesita autorizaciones internas en el programa sino que se controla con el perfil asignado a nivel de transacción.

Frecuentemente los programas que necesitan manejo interno de autorizaciones son las mejoras que pueden tener funcionalidades diferentes como Visualizar y Modificar, o para restringir visualización de información. Para estos casos “Authorization Check” deberá ser implementado según la necesidad.

El desarrollador deberá ser el responsable de crear el objeto de autorización correspondiente, con su definición y campos de autorización si fueran requeridos; y realizar el chequeo respectivo en el programa utilizando para ello la sentencia AUTHORITY-CHECK.

El desarrollador deberá informar el nuevo objeto de autorización para que los administradores del sistema y seguridad creen las autorizaciones respectivas y las asignen a los perfiles.

Objetos de Bloqueo

Los objetos de bloqueo deben ser usados cuando se requiera, especialmente cuando se trata de tablas Z y objetos que no pueden ser “compartidos” por varios usuarios.

También deberá utilizarse los objetos de bloqueo del estándar en los casos que se desee manipular objetos de negocio y se requiera de manera exclusiva.

Siempre el bloqueo debe ser usado a través de las funciones Enqueue y Dequeue.

Seguimiento (Debugging)

Existen dos formas de detener los programas usando sentencias desde los mismos programas, una de ellas es BREAK-POINT que detiene el programa sin importar que usuario lo esté ejecutando y activa el modo debugging. El segundo es BREAK <usuario> que detiene el programa únicamente para el usuario <usuario>. En los dos casos estas sentencias deben ser usadas temporalmente y se deben remover ANTES de promover el programa a otros ambientes como calidad y producción.

Otra posibilidad es crear grupos de puntos de verificación BREAK-POINT ID XXXX, con el fin de poder activar o desactivar los grupos en conjunto y de servir como herramienta de pruebas. Estos tipos de puntos de verificación no se necesitan remover del código fuente del programa, y podrían pasarse a otros ambientes.

Con la transacción SAAB se pueden activar los grupos de puntos de verificación.

Usando la variable SY-SUBRC

La variable SY-SUBRC es un código de retorno que permite saber si el proceso fue exitoso o no. Siempre se debe leer y analizar esta variable (con un CASE o con un IF) después de sentencias como SELECT, OPEN, READ, TRANSFER, MODIFY, INSERT o CALL TRANSACTION. Ejemplo:

```
SELECT * FROM BKPF
INTO TABLE t_bkpf.

IF sy-subrc <> 0.
 WRITE 'No hay registros en la Base de Datos'.
ENDIF.
```

Nota: en general hay muchas variables del sistema que pueden ser usadas.

Ver ANEXO 9.

Sentencia DATA

DATA nos permite definir espacios de memoria (variables), el complemento LIKE da la posibilidad de definir variables usando variables ya existentes en la aplicación. Esta práctica es muy recomendada pues ahorra tiempo en el mantenimiento.

4.8.5 Guía de Eficiencia y Rendimiento

Existen algunas prácticas de programación que hacen más eficientes los programas y mejoran el rendimiento de los mismos.

Acceso a la Base de Datos

Minimizar el tráfico de la red

Se debe proveer el máximo número de criterios disponibles en el momento de hacer una lectura a la base de datos, para esto se puede forzar al usuario a dar la mayor cantidad de parámetros de entrada y select-options (obligatorios) en el programa que ayuden a restringir las búsquedas y por consiguiente los valores de salida.

Se debe tratar de traer los datos en un solo bloque desde la base de datos y no hacer accesos independientes cada vez que se necesite información.

Uso del NOT en WHERE

Evite el uso del NOT en la cláusula SELECT WHERE. Es menos eficiente que usar una lógica positiva.

Evitar el SELECT – ENDSELECT

Este tipo de sentencia debe ser solo utilizada si el conjunto de registros a obtener de la base de datos, debo procesarlos en un solo recorrido y no se necesitan posteriormente en la aplicación. Si esta es la situación, el uso de SELECT-ENDSELECT es más recomendado que el uso Array fetch (uso de tabla interna), ya que evita consumo de memoria interna del work process que está ejecutando el programa.

Recuerde que dentro de este bloque loop no se permiten las siguientes sentencias:

COMMIT WORK
ROLLBACK WORK
MESSAGE

Si requiere realizar este tipo de sentencias, deberá realizar una carga de los registros a una tabla interna y posteriormente recorrerla con un LOOP ENDLOOP.

Nota: Cuando se va a procesar un gran volumen de datos, se puede usar esta sentencia ya que de otra forma el sistema cancela el proceso.

Order by vs SORT

Los datos que se traen de la base de datos pueden venir en cualquier orden, existe la posibilidad de ordenarlos durante la ejecución de la lectura usando el complemento ORDER BY, sin embargo esto no es recomendado porque puede intensificar el esfuerzo de la lectura. Se recomienda ordenar el contenido de la tabla interna después de la lectura usando un SORT.

Evitar el SELECT *

Usar el SELECT * significa que se traerán a memoria todos los campos de la tabla por lo tanto siempre se deben seleccionar las columnas que se necesitan de la(s) tabla(s) en el mismo orden en que se encuentran definidas. Si una lectura trae el doble de campos que son necesarios, la lectura demorará el doble de tiempo.

Uso del SELECT SINGLE

Al usar la sentencia SELECT intente proveer la mayor cantidad de campos clave a la lectura. Si tiene todos los campos clave en su instrucción o si solo está interesado en un solo registro de la tabla use un SELECT SINGLE.

Evitar SELECT anidados

El uso de instrucciones SELECT dentro de SELECT no es una práctica recomendable en la lectura de tablas. Para suplir esto se tienen varias alternativas que mejoran el rendimiento:

1. Buscar una vista en la base de datos que contenga la relación de tablas que se necesita
2. Usar un SELECT con el complemento JOIN o OUTER JOIN
3. Subir la información en instrucciones SELECT independientes a tablas internas y después hacer LOOPs anidados.
4. Se deben seleccionar las tablas o vistas por los campos claves.

Tablas Internas

Proceso registro a registro

Se debe usar la sentencia LOOP AT en lugar de DO. Esto porque el LOOP AT da la posibilidad de colocar una cláusula restrictiva WHERE Usar un CLEAR después de cada sentencia APPEND es recomendado para evitar inconsistencias.

Cláusula OCCURS

La cláusula OCCURS <n> debería apuntar cercanamente al número de registros que contendrá la tabla interna durante su utilización. Esto porque SAP ECC reserva el espacio en memoria (roll area) en función del <n> número de filas. Si el número de registros se incrementa, SAP ECC reservará un espacio igual en otro espacio de

memoria lo cual degradará el acceso durante la manipulación de la tabla. Si usted no está seguro de la cantidad de registros que contendrá la tabla, use OCCURS 0 y SAP ECC asignará dinámicamente el valor incrementando el espacio de 8k en 8k.
NOTA: Usar la nueva sintaxis de declaración de tablas internas en vez de occurs.

Leyendo un registro sencillo

Al leer un registro sencillo de la tabla interna estándar, use la instrucción SORT para ordenarla y posteriormente use la READ TABLE WITH KEY BINARY SEARCH. Si la tabla tiene n entradas la búsqueda lineal toma O(n), mientras que la búsqueda binaria toma O(log2(n)).

Ordenamiento

Siempre que se ordene una tabla interna se debe proveer el campo o los campos por los cuales se quiere hacer el ordenamiento. SORT <tabla> <campo 1> <campo 2> es más eficiente que usar simplemente SORT <tabla>.

Número de Entradas

Para conocer el número de entradas de una tabla no es necesario hacer un recorrido de la misma con un LOOP e incrementar una variable. Para esto se debe usar la sentencia DESCRIBE TABLE <tabla> LINES <var>, la función LINES(ti), o simplemente la variable del sistema SY-TFILL.

Eliminado Entradas

Para eliminar entradas de una tabla, se debe usar la sentencia DELETE <tabla> WHERE campo = <valor>. Como en el caso anterior, no es necesario el uso de sentencias cíclicas como el LOOP.

Copiando Tablas Internas

La forma más eficiente de copiar el contenido de una tabla interna en otra tabla interna es usando una asignación directa [] de la siguiente manera tabla1[] = tabla2[]. No es necesario usar LOOPs para recorrer las tablas.

LOOP WHERE vs LOOP CHECK

Se recomienda usar el WHERE en que es un complemento de LOOP en lugar de la sentencia CHECK que debe ir dentro del LOOP. Esto evita una sentencia extra en la lógica.

<u>Se debe usar:</u>

```
LOOP AT tabla WHERE campo = valor.
        ……
ENDLOOP
```

No se debe usar:

```
LOOP AT tabla.
        CHECK campo = valor.
        ……
ENDLOOP.
```

Cláusula INTO TABLE

1. Se debe usar la cláusula INTO TABLE cuando se carga una tabla interna desde la base de datos. Esto es más eficiente que mover uno a uno los registros e insertarlos usando APPEND:

Se debe usar:

```
SELECT * FROM LFA1 INTO TABLE T_LFA1.
```

No se debe usar:

```
SELECT * FROM LFA1.
        MOVE LFA1 TO T_LFA1.
        APPEND T_LFA1.
ENDSELECT.
```

2. Recuerde evitar el uso del SELECT / ENDSELECT, cuando sea posible use un INTO TABLE para guardar la información:

Se debe usar:

```
SELECT * FROM LFA1 INTO TABLE T_LFA1.

LOOP AT T_LFA1.
        WRITE: / T_LFA1-LIFNR.
ENDLOOP.
```

No se debe usar:

```
SELECT * FROM LFA1.
        WRITE: / LFA1-LIFNR.
ENDSELECT.
```

Cláusula Appending Table

En el caso de tener una segunda o más consultas y retornar los mismos campos, se puede usar esta instrucción y reducir el número de tablas internas a utilizar.

Ej:

```
SELECT campo1 campo2 campo 3
FROM TABLE1 APPENDING TABLE itab1
WHERE fld1 = 'AA'.
```

Lógica

Uso de CHECK, STOP, EXIT y REJECT

Usar estas sentencias cuando se necesite suspender un proceso o saltar procesos innecesarios. Esto mejora el rendimiento al no tener que ejecutar ciertas instrucciones.

Uso de CHECK

Cuando sea posible use la instrucción CHECK en lugar de tener IF anidados.

CASE vs IFs

El uso de CASE en ciertos casos es mejor y más claro que los mismos IFs. Cuando se tienen más de 5 instrucciones IF anidadas, se recomienda el uso del CASE que es más rápido y legible.

Manipulación de Textos y Cadenas

Operadores de Cadenas

Se deben usar los operadores estándar CO, CA, CS en lugar de crear lógicas propias. Ejemplos:

x CA y (contiene alguno), x contiene al menos un carácter de Y
x CO y (contiene únicamente), x contiene un único carácter de Y
x CS y (contiene string), x contiene una cadena de carácter de Y

Manipulación de Cadenas

En versiones anteriores la manipulación de cadenas se hacia a través de módulos de función, ahora existen instrucciones que mejora el rendimiento de estas operaciones:

CONCATENATE, para concatenar dos cadenas
SPLIT, para separar dos cadenas
STRLEN(), para conocer la longitud de una cadena

SHIFT

Para eliminar caracteres de una cadena, se debe usar la sentencia SHIFT LEFT/RIGHT en lugar de crear lógicas propias.

Recomendaciones Varias

"Limpiar" código

Evite dejar código muerto dentro de los programas, remueva todos los objetos (variables, constantes, tipos, rutinas, tablas internas) que no son usados. Se recomienda hacer un chequeo extensivo para revisar el programa:

Transacción SE38 – Editor ABAP y por el menú principal: Programa → Verificar → Verificación de programas ampliada

Escoger las mejores tablas

Siempre se debe intentar usar la mejor tabla disponible para recuperar datos. Existen tablas que tienen los mismos datos pero son menos pesadas, por ejemplo se deben encontrar alternativas para tablas como VBAP y BSEG

Sentencia AT

Es más eficiente usar la sentencia AT (AT NEW, AT END, etc.) para hacer sumas y rompimientos, que usar sentencias como ON CHANGE OF por que estas se basan en la jerarquía de la tabla interna y permiten operaciones automáticas. Sin embargo se debe tener en cuenta que las sentencias AT no permiten ver el contenido de los campos individuales.

Tipos de Datos y Conversiones

Trate de eliminar conversiones de datos. Definir las variables de origen y de destino con el mismo tipo es la mejor solución para este problema. Para conocer la forma como se están ejecutando sus conversiones puede ejecutar la transacción SE30 (Runtime Analisys).

Sentencia WRITE

Se deben usar la menor cantidad de WRITEs como sea posible, pues cada uno es un llamado a una sentencia diferente:

<u>Se debe usar:</u>

WRITE: / 'esta es una prueba', MATNR,

/ ‘nueva línea’, 20 ‘comienza en posición 20’.

No se debe usar:

WRITE: / ‘esta es una prueba’, MATNR.

WRITE: / ‘nueva línea’, 20 ‘comienza en posición 20’.

5 GLOSARIO

ABAP(*Advanced Business Application Programming*): Es un lenguaje de cuarta generación lenguaje de cuarta generación, propiedad de SAP, que se utiliza para programar la mayoría de sus productos.

ARTEFACTO: Es una información que es utilizada o producida mediante un proceso de desarrollo de software.

ASAP(AcceleratedSAP): Metodología creada por SAP para las implantaciones de su sistema.

BLUE PRINT: En una reunión de Revisión Ejecutiva, se discutirán las metas del grupo, la estructura organizacional y los procesos de negocio de alto nivel.

CODE INSPECTOR: Programa que sirve para verificar la integridad del código generado en las personalizaciones.

ESTÁNDAR: Guías y normas de cómo hacer algo en particular.

FRAMEWORK: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un *framework* puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

FUNCIONAL: Persona que posee una formación muy especializada acerca de un módulo en SAP, tanto en su parametrización como en su funcionamiento.

INTERFACE: Programa que se utiliza para comunicar 2 sistemas.

ITERACIÓN: Periodo de tiempo en el desarrollo en el cual se realizan ciertas actividades que ya se tenían planeadas.

MÉTRICAS: Métodos para poder medir algo.

PERFOMANCE: Rendimiento de una aplicación.

PERSONALIZACIÓN: Modificación de un sistema para darle el comportamiento que se desea.

PUNTO DE FUNCIÓN: Métrica ampliamente usada para estimar cuanto tiempo se requiere para hacer un programa.

REFACTORIZACIÓN: Mejorar un código ya existente.

REQUERIMIENTOS: Especificaciones de lo que debe hacer algo.

ROLES: Conjunto de perfiles, conocimientos, destrezas y actividades que son hechas por personas para realizar determinadas funciones.

RUP(Rational United Process): Metodología para el desarrollo de software propuesta por IBM ampliamente usada en la industria.

SCRUM: Metodología ágil de desarrollo de software.

SOFTWARE EN PRODUCCIÓN: Software en el ambiente productivo.

TRANSACCIÓN: Nombre abreviado de un programa en SAP para acceder a él.

TRANSPORTES: Sistema propio en SAP, para manejar los desarrollos y diferentes objetos del sistema en los ambientes de desarrollo, calidad y productivo.

TRASABILIDAD: Facultad de hacerle un seguimiento, a un desarrollo en todo su ciclo de vida.

USUARIO LIDER: Persona experta en un área del negocio en particular.

XP(Programación extrema): Metodología ágil ampliamente usada en la industria.

6 CONCLUSIONES

- Es de vital importancia adoptar una metodología o varias metodologías que se consideren la más apropiadas, según las necesidades y características del proyecto como son su rigurosidad, complejidad, tamaño, tecnología a emplear y tiempo que se dispone para poderlo realizar. Con esto no se tiene una guía exacta de qué hacer en cada situación, ya que cada organización posee muchas singularidades, pero si es una gran ayuda que da a las personas, al indicar el orden del cómo se deben hacer las diferentes actividades que conlleva un proyecto, además las diferentes metodologías que hay tienen comunidades de personas, expertos y material bibliográfico que garantizan que se le esté sacando el máximo de provecho a las metodologías que se estén usando.

- Es muy importante saber diferenciar muy bien las diferentes metodologías que hay, cuáles son sus características y para qué caso en particular nos pueden ser de utilidad y no adoptar una metodología simplemente por el hecho que está de moda.

- Las metodologías agiles que están más en boga ahora son XP y SCRUM, en particular con la última metodología considero que es una metodología demasiado liviana que sólo nos habla de los roles y la parte administrativa del proyecto, pero no se hace mención de estándares de programación, gestión de riesgos, levantamiento de unas buenas especificaciones y gestión del cambio que son factores muy importantes para poder realizar un proyecto. El SCRUM si es mezclado con otras metodologías como el XP, puede potenciarse más.

  Por contraparte está muy claro que el tamaño del proyecto de por sí va exigiendo de forma natural, más documentación y procedimientos para poder llevarlo acabo exitosamente, las metodologías agiles no están de acuerdo en la excesiva documentación y procedimientos extras en un proyecto, van más a la parte de programación y prácticamente dejan de ser efectivas cuando un proyecto es mediano o grande.

- Una de las metodologías más usadas en el medio es el RUP, es una metodología muy completa, robusta y flexible, que puede se puede usar en pequeños y grandes proyectos. Lo único es no caer en los extremos de poseer pocos entregables o artefactos por ser facilistas o en un exceso de ellos que entorpecerían la debida ejecución de un proyecto.

- La metodología por excelencia en las implantaciones de sistemas SAP es ASAP. Esta metodología fue creada por la casa de software SAP AG y nos garantiza que el proyecto se ejecute en el orden adecuado y salga exitosamente, pero no hace mención explícita de cómo se deben gestionar las personalizaciones. Para ello podemos usar practicas clásicas de la ingeniería de software y gestión de proyectos de software y así poder garantizar una armonía con el proceso de implantación y las personalizaciones del sistema implantar.

- Todo elemento tecnológico que ayude a la realización exitosa de un proyecto, úselo!! ,en especial si son de uso en equipo. Por ejemplo para la realización y visualización de cronogramas esta el Project server, rastreo de bugs el mantis bug tracker, para repositorios de código común hay muchos CVS. Si se necesita un repositorio de documentación se está usando mucho en la industria el Share Point y así sucesivamente cada necesidad que tenga el proyecto, procure usar las herramientas más adecuadas para poder realizar mejor las actividades que se requieran.

- Es de vital importancia una excelente especificación de los requerimientos en las personalizaciones. Esto garantiza que en realidad se va a construir el software que se requiere y da la posibilidad de hacer estos desarrollos vía remota, algo que es muy poco común en las implantaciones con SAP.

- Si se posee una buena especificación de requisitos, como una excelente documentación del requerimiento ya elaborado, estaremos garantizando a futuro una gestión de cambios efectiva y medible.

- En las implantaciones de SAP con respecto a las personalizaciones es crucial poseer métricas y tener una estimación más real de cuánto tiempo llevan su realización. Usualmente esta calculo se hace por la experiencia, criterios muy personales y subjetivos lo cual conlleva usualmente a prorrogas y extensión de tiempos del proyecto reflejándose inmediatamente en costos de más que no se habían previsto antes.

- La gestión de la calidad es vital en todo proyecto de software y debe reflejarse en toda actividad que se haga. Con esto garantizamos un desenvolvimiento más sano del proyecto. Una práctica por excelencia es la revisión por pares y con esto garantizamos que los artefactos o entregables cumplan con las normas que se hayan establecido.

- Para el mantenimiento de las personalizaciones de una manera efectiva además de tener una buena documentación del requisito y la especificación técnica de la solución. Sé deben respetar estándares de programación para su mayor fácil comprensión.

- A todos los miembros del equipo de una implantación se les debe dar una explicación en términos sencillos de cómo es control de riesgos, manejo de métricas, la importancia de la documentación y buenas prácticas que se deben hacer, ya que sin esto, los integrantes del equipo tienen la sensación de estar haciendo muchas actividades extras que no aportan mucho al proyecto cuando en realidad si lo son.

7 ANEXOS

 Evaluación de riesgos.xls **Anexo 1 Plantilla de manejo de riesgos**
 especificación.doc **Anexo 2 Propuesta especificación funcional y técnica**
 Formato de pruebas.xls **Anexo 3 Formato de pruebas**
 F:\TESIS\Escritos' Solicitud de cambi **Anexo 4 Formato sugerido control de cambios**
 F:\TESIS\Escritos' Estructura Estánda **Anexo 5 Estructura Estándar para Programa tipo "Report"**
 F:\TESIS\Escritos' Estructura Estánda **Anexo 6 Estructura Estándar para Módulo de Función**
 F:\TESIS\Escritos' Cabecera para Re **Anexo 7 Cabecera para Reportes**
 F:\TESIS\Escritos' Códigos de Tipo d **Anexo 8 Códigos de Tipo de Objeto**

Listado de variables
del sistema.doc

Anexo 9 Listado variables del sistema

8 BIBLIOGRAFÍA

Marco Teórico:

Baird Stewart. Teach Yourself Extreme Programming in 24 Hours USA; Sams Publishing, 2002.

Introducción a RUP [sitio en Internet], disponible en https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc Último acceso: Noviembre 20 de 2007.

La metodología de implementación Accelerated SAP (ASAP) [sitio en Internet], disponible en http://www.sap.com/argentina/partners/howtochoose/index.epx Último acceso: Noviembre 20 de 2007.

Metodologías Ágiles en el Desarrollo de Software [sitio en Internet], disponible en www.willydev.net/descargas/prev/TodoAgil.Pdf Último acceso: Noviembre 19 de 2007.

Metodología de desarrollo de software [sitio en Internet], disponible en http://www.um.es/docencia/barzana/IAGP/Iagp2.html Último acceso: Noviembre 19 de 2007.

Módulos De Sap R3 [sitio en Internet], disponible en http://www.mundosap.com/foro/showthread.php?t=281 Último acceso: Noviembre 19 de 2007.

Principles behind the Agile Manifesto [sitio en Internet], disponible en http://www.agilemanifesto.org/principles.html Último acceso: Noviembre 19 de 2007.

¿Qué es Scrum? [sitio en Internet], disponible en http://www.baufest.com/spanish/scrum/scrumconference2006/Que_es_scrum.pdf Último acceso: Noviembre 20 de 2007.

SAP R/3 [sitio en Internet], disponible en http://usuarios.lycos.es/cblsap/sisr3.html Último acceso: Noviembre 19 de 2007.

Schwaber Ken Beedle Mike. Agile Software development with Srcum USA; Prentice Hall Hill, 2002.

Scrum in five minutes [sitio en Internet], disponible en http://www.softhouse.se/Uploades/Scrum_eng_webb.pdf Último acceso: Noviembre 20 de 2007.

Utilización del sistema SAP R/3 [sitio en Internet], disponible en http://www.tirant.com/libreria/detalle?articulo=8484681831 Último acceso: Noviembre 19 de 2007.

What Is Agile Software Development? [sitio en Internet], disponible en http://www.agilealliance.org/show/2 Último acceso: Noviembre 19 de 2007.

Propuesta metodológica:

Brand Hartwing. Implementación técnica mediamte ASAP España; Gestión 2000, 2001.

Diplomatura construcción de software, Módulo gerencia de proyectos, impartido por la universidad EAFIT.

Gestión del cambio con ITIL [sitio en Internet], disponible en http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/vision_general_gestion_de_cambios/vision_general_gestion_de_cambios.php Último acceso: Noviembre 20 de 2007.

Instructivo para la cuenta de puntos de función [sitio en Internet], disponible en http://www.dinacyt.gub.uy/pdt/files/6.2.5%20-%20puntosFuncion.pdf Último acceso: Noviembre 20 de 2007.

Pressman Roger S. Ingeniería de software Un enfoque práctico Sexta edición . Cap gestión del riego México;Mc Graw Hill, 2006

Pressman Roger S. Ingeniería de software Un enfoque práctico Sexta edición . Cap gestión del riego México;Mc Graw Hill, 2006

Schafer Marc O Melich Matthias Sap Solution Manager Primera edición Alemania; Sap Press 2007.

Licencia Creative commons 2.5 para Colombia

Reconocimiento-No comercial-Sin obras derivadas 2.5 Colombia

Usted es libre de:

copiar, distribuir y comunicar públicamente la obra

Bajo las condiciones siguientes:

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

No comercial. No puede utilizar esta obra para fines comerciales.

Sin obras derivadas. No se puede alterar, transformar o generar una obra derivada a partir de esta obra.

- Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.
- Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor
- Nada en esta licencia menoscaba o restringe los derechos morales del autor.

Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.

Esto es un resumen fácilmente legible del texto legal (la licencia completa).