Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de Ingeniero Informático

Modelo Funcional de la Factoría de Software de la UCI para la línea Carrefour

Autores:
Yanosky Ríos La Hoz
Marbys Marante Valdivia

Tutor:
Ing. Yaimí Trujillo Casañola

Ciudad de la Habana, Diciembre de 2005

Dedicatoria

A mi familia, especialmente a:
mami, papi y manita, que son mí
vida y porque este es su sueño.

A mis primos: Ari, Eli, Abe y Nelson
Por ser mi luz en los momentos
de dudas.

A tío Tingo y Santiaguito, que son mi ejemplo.

A mi abuelo Servi y a la memoria de mi abuela Eloisa, que sus historias me guian.

A mima y papi Wil por el incondicional cariño que me han dado y ser mis segundos papaces.

Marbys

A mis padres: Mami y Papa, sin ustedes no hubiera podido llegar hasta aquí, su amor y ayuda en todo momento han sido fundamentales durante este largo trayecto, son mi guía y ejemplo, ustedes se merecen esto y mucho más.

A mi hermanita: Yami ahora te toca a ti, no ha sido fácil llegar hasta aquí, pero se que tu podrás y estoy seguro que lo harás mejor que yo.

A mis abuelos, en especial a caca que siempre ha estado a mi lado y me ha consentido en todo.

A Glorita: Mi amor has estado presente desde el comienzo de mi carrera, me has dado tu apoyo y cariño en todo momento, también es para ti.

A Marisol y Octavio por lo buenos que han sido conmigo.

A mi familia en general, por ser una familia excelente y muy unida.

Yanosky

Resumen

En el mundo, donde la competencia rige las actividades comerciales, es casi imprescindible la rápida aplicación de las innovaciones. La tendencia actual muestra que las entidades para ser más eficaces, eficientes y competitivas deben aplicar la informatización en todas las esferas. Informatización que a su vez tiene que estar industrializada, regida por procesos que organicen y coordinen los objetivos a cumplir.

En las empresas de software sigue existiendo un alto porcentaje de soluciones artesanales. Este fenómeno unido al creciente interés de usar el recurso humano intelectual como principal fuente económica; ha traído consigo la creación de variadas estrategias con el fin de elevar la producción de software cubano.

En el presente trabajo se desarrolla el modelo funcional de la "Factoría de Software" de la Universidad de las Ciencias Informáticas (UCI) para la línea de aplicaciones Carrefour; garantizando un adecuado flujo de procesos, lo cuales permiten mejorar el trabajo en equipo; obtener productos de mejor calidad, y dar un punto de partida a la construcción de una metodología propia. El Modelo que se propone se basa en las tendencias actuales, en el entorno donde se va aplicar y en la experiencia que se ha acumulado en los trabajos realizados por la empresa Desoft y la UCI.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Se viven tiempos de transición, la era industrial poco a poco deja espacios a una nueva que se caracteriza por el uso intensivo de la información y el conocimiento. Se muestran cambios en todos los patrones tradicionales de la actividad humana, apoyados en el conocimiento y las Tecnologías de la Información y las Comunicaciones (TIC). Las cuales se potencian mutuamente con el objetivo de obtener cada vez, más y mejores programas. Estableciendo con ello nuevos estándares de desempeño y creando nuevos mercados y oportunidades económicas.

En el entorno empresarial del software siguen existiendo un alto porcentaje de empresas que ofrecen soluciones artesanales y a medida. Escapar al dinamismo de la competencia es un reto para cualquier empresa que se dedique a la producción de software. Pero el futuro pasa por evolucionar a centros de desarrollo en los que las mejoras del producto, la disminución del costo y el tiempo de desarrollo son el verdadero objetivo.

Una solución a tan alto reto se dirige hacia la producción organizada y estructurada de software, en entidades responsables, que ofrezcan servicios diferenciados, orientadas a incrementar la calidad del producto final. Las *factorías de software*, buscan industrializar el desarrollo de sistemas, rescatar ese proceso del ámbito del cliente y trasladarlo a un medio donde se establezcan unos procedimientos y métodos definidos que cuenten con herramientas que ayuden en su implantación.

El concepto de factorías de software se acuñó para describir un proceso de software repetible, donde se reutilizan códigos ya desarrollados. Según el European Software Institute, hasta un 60 por ciento del código de programas es reutilizable. La idea de reutilización de componentes de software fue concebida por el ingeniero y matemático McIlroy en 1968, durante una Conferencia de la OTAN sobre Ingeniería de Software celebrada en Alemania.

Las factorías de software se perfilan, en la unión del conocimiento y la metodología, en la que se acumule todo lo desarrollado, lo que permite conseguir altos porcentajes de reutilización. La industrialización del proceso de software facilita la evaluación, medición y control del proceso, y con ello, su mejora y adaptación al cambio, no sólo en el análisis de los procesos internos, sino en la investigación de nuevas tecnologías, herramientas y métodos. La ardua tarea de las empresas que apostan por este perfil consiste en aportar una metodología para la mejora del desarrollo de software, definiendo una serie de prácticas técnicas y de gestión que deben adoptarse en los proyectos informáticos para

asegurar el cumplimiento de los plazos, los presupuestos y las exigencias de calidad de los clientes.

Actualmente el mundo se dirige hacia un nuevo modelo económico basado en el conocimiento. Las empresas y la economía en su conjunto se enfrentan a una transformación global de sus planteamientos, consecuencia del desarrollo de las TIC, estos avances han contribuido a que el conocimiento sea considerado como un nuevo recurso, generador de importantes ventajas competitivas.

Cuba es un país donde el desarrollo de software es aun incipiente. Es por ello que una de las principales tareas del Gobierno Cubano es desarrollar la Industria del Software. No solamente con el fin del desarrollo de sistemas para la informatización de la sociedad sino también por los beneficios de insertarnos en el mercado de software a nivel mundial por su perspectiva económica.

La constante investigación y el desarrollo científico de la ingeniería y gestión de software proponen profundos avances en la resolución de problemas en todos los campos. Sin embargo, la dinámica del mercado y la tradición en la gestión de ofertas de software no fomentan la mejora de la calidad de los procesos de trabajo o de sus productos.

La infraestructura cubana no escapa a estos problemas, en la actualidad el enfoque teórico y científico de las universidades e instituciones importantes, no ha sido capaz de llegar a las empresas, las cuales hoy tienen la mayor responsabilidad en la esfera productiva y el aporte económico. A las cuales se les dificulta mucho la investigación pues están a expensas de una competencia cada vez más exigente en tiempo, costo y calidad.

Nuestro país no está ajeno a las tendencias que evidentemente propician facilidades para la competencia entre las empresas y el desarrollo de las mismas en todos los sentidos. La mayoría de las Universidades del país realizan proyectos de producción dadas las necesidades de las entidades.

Con el objetivo de lograr un vinculación fuerte entre la Universidad y las Empresas se creó, hace tres años la Universidad de las Ciencias Informáticas (UCI).; introduciendo un nuevo concepto basado en la relación Universidad-Empresa, una Universidad Productiva. En esta universidad la producción es un problema social, político, y económico, donde el cien por ciento de los estudiantes y profesores esté vinculado a la producción.

La necesidad de este trabajo es aportar el "Modelo Funcional de la Factoría de Software de la Universidad de las Ciencias Informáticas para la línea Carrefour". El trabajo se parte de los resultados alcanzados en esta rama en los últimos años, el cual ha sido documentado por especialistas en estas materias. Se ha documentado la experiencia adquirida durante el

tiempo que se ha trabajado en la Factoría de Software de la Universidad de la Ciencias Informáticas, en un proyecto de conjunto con la empresa DESOFT, donde el equipo de desarrollo se vio forzado a madurar y resolver problemas que se pueden evitar en futuros trabajos.

Este trabajo tiene como principal destinatario a empresas que aceptan el reto de implantar enfoques de factorías de software o se propongan absorber proyectos externos de otras factorías de software ya estandarizadas. El conocimiento adquirido, está siendo utilizado en la Factoría de la UCI.

El trabajo fue dividido en cinco capítulos, a continuación se presentará el nombre del capítulo y su objetivo en un contexto global.

Capítulo 1: Marco Teórico. El propósito de este capitulo es la formalización de todos los conceptos asociados al tema y que son necesarios para la comprensión de lo que se describe en el resto del trabajo, como la descripción de todos los requisitos asociados al modelo de Factoría de Software, la descripción, importancia y utilidades de los Frameworks, dando una breve introducción a los procesos fabriles industriales. Se describe la misión de la UCI como institución responsable del desarrollo de la industria de software cubana y de Matchmind como empresa establecida en el mercado, que contrata el servicio de la Factoría de la UCI.

Capítulo 2: Evaluación de Modelos Existentes. Se pretende facilitar la descripción de algunos modelos de factorías de software. Tiene como objetivo tomar estos modelos como base y seleccionar elementos de los mismos hasta llegar al modelo propuesto.

Capítulo 3: Situación actual de la Factoría de Software de la UCI en la línea Carrefour. Tiene como objetivo la descripción de lo que se ha hecho hasta el momento en la factoría objeto de estudio en la línea Carrefour, identificando los procesos existentes, la manera de interactuar con el cliente y deficiencias presentadas.

Capítulo 4: Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour. Se presentan las entidades por la que está compuesta de dicho modelo, sus relaciones y características; con el fin de optimizar el funcionamiento actual.

Capítulo 1 - Marco Teórico.

1.1 Introducción

En el presente capítulo se abordarán un grupo de conceptos importantes relacionados con la obtención del Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour. Primeramente en el mismo se tratarán cuestiones como: ¿Que es una factoría?, ¿Qué es la línea de producción y montaje?, ¿Qué es una Factoría de Software y cuales son sus objetivos?, ¿Qué es Matchmind?, ¿Qué es la UCI? Además se definirá que son las aplicaciones empresariales y cuales son los aspectos que las hacen complejas y difíciles de construir, además de abordar un conjunto de elementos y tecnologías relacionados al desarrollo de este tipo de aplicaciones como son: Framework, Componente de software, y la influencia de la plataforma Java 2 Enterprise Edición (J2EE) en la programación para empresas. Se definirán las características del framework Carrefour y que arquitectura presentan las aplicaciones construidas con el mismo.

1.2 Factoría

Se denomina así, de forma genérica, a cualquier tipo de fábrica o industria, es decir, a cualquier tipo de instalación en la cual se produce la transformación de materias primas o productos semiterminados en otros productos, bien para otras industrias, bien para su uso o consumo final. Por extensión se está aplicando esta palabra para designar determinadas actividades en las cuales no se produce consumo y transformación de materias y que tienen como objeto final la obtención de productos intangibles: factoría de comunicación, factoría de cine, factoría de software.

1.3 Línea de producción y montaje

La línea de producción y montaje o cadena de montaje es una técnica utilizada en el proceso fabril tradicional. Esta es una técnica que si bien no fue Henry Ford el inventor, si fue el principal responsable de que estas prácticas se generalizaran en el desarrollo industrial. Consiste en instalar una cadena de montaje a base de mecanismos de transmisión y guías de deslizamiento que van desplazando automáticamente las unidades en montaje por los puestos de trabajo donde sucesivos grupos de operarios van realizando tareas especificas hasta logar la integración del producto final. Este sistema de piezas

intercambiables fue el aplicado por Ford en la construcción de automóviles, lo que revolucionó la industria automovilística, y a partir de entonces ha sido aplicado en la industria en general, es una técnica que reduce los costos de producción por la vía de la estandarización de productos y procesos.

Con la aplicación de esta técnica la empresa Ford llegó a ser líder en el sector automovilístico, esta forma de producir le permitía una parcelación precisa de las tareas y una asignación de tiempo a cada una de ellas. Estas innovaciones se tradujeron en un importante aumento de la productividad en la empresa.

En la línea de producción y montaje Henry Ford vinculó importantes principios de de administración científica y organización racional del trabajo:

División del trabajo, especializando cada trabajador en una tarea específica.

La standardización de los métodos de trabajo, herramientas, materias primas y componentes con el objetivo de aumentar la eficiencia.

Principio de la productividad consiste en aumentar la capacidad de producción del hombre en el mismo período.

"Construiré un automóvil para la gran multitud, con los mejores materiales y por los mejores hombres contratados, siguiendo los diseños más simples que la ingeniería moderna pueda idear [...] de precio tan bajo que cualquier hombre que gane un buen salario será capaz de poseer – y disfrutar con su familia la bendición de horas de placer con su en los grandes espacios abiertos de la naturaleza". Declaración de Henry Ford en el inicio de su carrera como productor de automóviles.

Analizando la declaración anterior, Ford nos remite a algunas interrogantes: ¿sería posible producir software en grandes cantidades para un segmento del mercado (ejemplo de segmento de mercado: software de gestión bancaria, de gestión contable, para dispositivos móviles)?, ¿La calidad de los recursos humanos empleados en la producción de software influye en la calidad del producto?, ¿La simplicidad en el proceso de desarrollo del software influye en el precio final del mismo? La bibliografía sobre ingeniería de software en su mayoría responde de forma positiva las dos últimas cuestiones. La producción de software en grandes cantidades para un segmento de mercado (sentido de fábrica) es una cuestión que puede ser respondida por la literatura sobre factoría de software.

1.4 Factoría de Software

Existen varias definiciones sobre factoría de software:

El término factoría de software fue utilizado por primera vez en la década del 60 en Japón. Pero varias empresas asociaron el término al mero desarrollo de software.

A continuación se enuncian varios conceptos de factoría de software dado por distintos autores:

Una empresa productora de software que no responda a características como: producción de software en gran escala, estandarización de tareas, estandarización del control, división del trabajo, mecanización y automatización, no puede ser considerada una factoría de software. El desarrollo de una factoría implica que las buenas prácticas de Ingeniería de Software sean aplicadas sistemáticamente [Cusumano, 1989].

Una factoría de software debe, para ser flexible, ser capaz de producir varios tipos de productos; llevar a cabo conceptos de Ingeniería de Software (metodología, herramientas, gestión de la configuración), y también ser capaz de estudiar, diseñar, implementar y mejorar sus sistemas y procesos [Cantone, 1992].

Una organización con características de factoría de software debe poseer una estructura de construcción de software basada en componentes. Los componentes utilizados en la construcción del software pueden ser desarrollados por una unidad de producción de componentes (factoría de componentes). La factoría de componentes es la base para la implementación de una factoría de software [Basili et. al., 1992].

Una factoría de software debe poseer un conjunto de herramientas estandarizadas para la construcción de software, bases históricas para ser usadas en la dirección de proyectos, y principalmente, poseer un alto grado de reutilización de código en el proceso de desarrollo de un determinado software, apoyado en una base de componentes reutilizables [Li et. al., 2001].

Una organización fabril para el desarrollo de software debe tener claro el asunto del "software único", es decir, todo software es único, pero algunas partes de ellos se pueden repetir en varios proyectos. El proceso industrial debe contener el desarrollo, almacenamiento y montaje de partes reutilizables en un producto [Fernstrom et. al., 1992].

Una factoría de software es una organización con procesos estructurados, controlados y mejorados de forma continua, considerando principios de Ingeniería Industrial, orientados a dar respuesta a múltiples demandas de distintas naturaleza y alcance. Dirigida a la creación

de productos de software, conforme a los requerimientos documentados de los usuarios y clientes, de la forma más productiva y económica posible [Fernández y Teixeira, 2004].

Según [Fernández y Teixeira, 2004] una factoría de software puede tener varios dominios de actuación en dependencia de las fases de desarrollo del software en las que opere, desde un proyecto de software completo, hasta un proyecto físico o solo codificación del software.

Estos autores enumeran una serie de características inherentes a una estructura fabril para software:

- Proceso definido y estandarizado para el desarrollo de software.
- Interacción controlada con el cliente, alto poder de entendimiento.
- Estimación de costos y tiempo basados en el conocimiento real de la capacidad productiva, mediante métodos de obtención basados en datos históricos.
- Control de los recursos humanos involucrados.
- Planificación y control de la producción
- Control y almacenamiento en bibliotecas de componentes de software (documentos, código, métodos, etc.)
- Control del estado de ejecución de todas las demandas.
- La producción de software debe estar fuertemente basada en métodos y técnicas estandarizadas.
- Capacitación de los recursos humanos.
- Garantía de la calidad del producto.
- Mecanismos de control de costos.
- Mejora continua del proceso.
- Hardware y software ajustado a las necesidades del usuario.

La tendencia de la sociedad de la información gira en torno a la producción sistematizada de software en centros de desarrollo, que ofrezcan prestaciones diferenciadas orientadas a incrementar la calidad del producto final. Las factorías de software, encargadas de industrializar el desarrollo de sistemas, rescatar ese proceso del ámbito del cliente y trasladarlo a una "fábrica" donde se establezcan unos procedimientos y métodos definidos que cuenten con herramientas que ayuden en su implantación. Actualmente existen muchos centros en los que el desarrollo de software tiene un alto porcentaje de artesanía y trabajo a medida, por lo que la tendencia actual pasa por evolucionar hacia las factorías de

software en donde las mejoras de producto sean el verdadero negocio. Las factorías de software se perfilan, pues, como una conjunción de conocimiento y metodología, en las que se acumula todo lo desarrollado, lo que permite conseguir altos porcentajes de reutilización. La industrialización del proceso de software facilita la evaluación, medición y control del proceso, y con ello, su mejora y adaptación al cambio, no sólo en el análisis de los procesos internos, sino en la investigación de nuevas tecnologías, herramientas y métodos.

Basado en las definiciones anteriormente analizadas este trabajo asume factoría de software como una organización estructurada creada para el desarrollo de software, con procesos estandarizados, repetibles, gerenciales y principalmente mejorable continuamente. Significa esfuerzos integrados – por encima de proyectos individuales – para mejorar las operaciones relativas al software. Debe poseer un grupo de herramientas estandarizadas tanto para la construcción de software como para la gestión y administración de proyectos, automatizando gran parte del trabajo. Reducir la cantidad de trabajo promoviendo la reutilización de componentes software (desarrollo basado en componentes), arquitectura y conocimiento en el desarrollo de un determinado producto, de forma tal que se puedan obtener mejores resultados en menor tiempo y con menos costos (productividad). Debe ser una fábrica en la cual las actividades de desarrollo sean predecibles, lo que implica la existencia de técnicas para la estimación de costos, plazos y tamaño de un equipo para un determinado proyecto, basado en el conocimiento real de la capacidad productiva, para lograr que los costos estimados y compromisos de cronograma puedan ser satisfechos y confiables. Es importante tener establecida una política que garantice la calidad del software, basando el desarrollo en métodos y técnicas estandarizadas, y manteniendo capacitados sus recursos humanos. Una factoría de software tiene establecidas líneas de productos y se enfoca a segmentos de mercado, esto supone la creación de una arquitectura común para cada línea de producto y muchas veces un framework que soporte esa arquitectura. Estas son las principales características que hacen la diferencia entre desarrollar software de manera artesanal y una verdadera factoría de software.

1.4.1 Objetivos de una Factoría de Software

El enfoque de factoría de software viene a formalizar todos los procesos (etapas de producción) y sus productos, trabajando en líneas de producción, con etapas y tareas perfectamente definidas para cada tipo de profesional involucrado en el proceso, yendo de la productividad en la línea de producción a las rutinas de control de la calidad. Se busca la

especialización de los profesionales, para que cada uno garantice la productividad de la fase en la que está ocupado. Entre los principales objetivos trazados por una factoría de software están:

1- Industrializar el desarrollo de sistemas de software.
2- Producción de software a gran escala.
3- Lograr una alta productividad en el desarrollo de software.
4- Establecer una línea de producción.
5- Mejora continua de los procesos.
6- Estimación de costos y plazos extremadamente precisa.
7- Reducción de los costos de producción.
8- Lograr un buen control de la calidad.
9- Especializar al profesional en una tarea específica del proceso, concentrando sus esfuerzos en dicha tarea.

1.5 Matchmind

Matchmind es una empresa dedicada a servicios profesionales de consultoría de gestión y tecnología de la información. Cuya estrategia corporativa se basa en tres ejes:

1. Proximidad y compromiso con los clientes.
2. Búsqueda de la excelencia en la ejecución de los servicios.
3. Gestión eficaz de todo el ciclo de relación con sus profesionales.

Sustentados en sólidos cimientos:

- Una organización basada en líneas de negocio muy especializadas que aporta un alto valor añadido para sus clientes.
- Un sistema de calidad que asegura el alineamiento de su trabajo con los interese de sus clientes.
- Un riguroso proceso de selección, formación y evaluación que les permite aportar al trabajo el talento de profesionales con amplia experiencia.

1.5.1 Organización

La oferta de servicios de Matchmind está agrupada en cinco divisiones, cada una de ellas especializada en distintas líneas de negocio claramente definidas y con su propia capacidad de I+D.

1. **Consultoría orientada a la eficiencia en procesos de negocio.**

Servicio cuyo objetivo es ayudar a los clientes a generar valor, a través de la eficiencia en procesos y la mejora de rentabilidad en las operaciones. Este servicio está dirigido a:

- Compañías que quieran optimizar sus procesos de negocio.
- Áreas de negocio que quieran acometer un proceso de mejora.
- Áreas de tecnología que lideren acciones de cambio y eficiencia.

2. **Especialización en consultoría tecnológica, experiencia de usuario, soporte y arquitectura.**

Innovación tecnológica, pedagógica y arquitectura al servicio de las necesidades del negocio. Servicio dirigido a:

- Compañías que quieran desarrollar e implementar nuevos sistemas, arquitecturas y soluciones tecnológicas.
- Departamento de desarrollo informático y arquitectura de sistemas en proceso de evolución tecnológica, recursos humanos y marketing.

3. **Especialización en el desarrollo de sistemas.**

Desarrollo de sistemas bajo últimas tecnologías, eficiencia y productividad en la construcción de sistemas de información por medio de la especialización, metodología y reutilización. Servicio dirigido a:

- Compañías que quieran implementar y desarrollar nuevos sistemas estructurales de aplicaciones.
- Departamento de desarrollo informático y arquitectura de sistemas en proceso de evolución tecnológica.

4. **Implantación eficiente de soluciones de negocio basadas en software estándar.**

Implantación eficiente de soluciones empresariales ERP/RRHH, CRM y BI, basada en metodología para gestionar el cambio de procesos, tecnología y organización. Servicio dirigido a:

- Áreas de negocio y tecnología de compañías que quieran acometer un proceso de mejora por medio de la implantación de soluciones de software de gestión empresarial.

5. **Gestión de sistemas.**

Metodología y soporte a la explotación de los activos tecnológicos para aumentar la calidad y fiabilidad de los flujos de información necesarios para la gestión del negocio. Dirigido a:

- Departamentos de informática que se planteen optimizar sus procesos.

- Centros de Proceso de Datos de grandes organizaciones, en busca de mejoras de eficiencia.

1.5.2 Cifras Matchmind

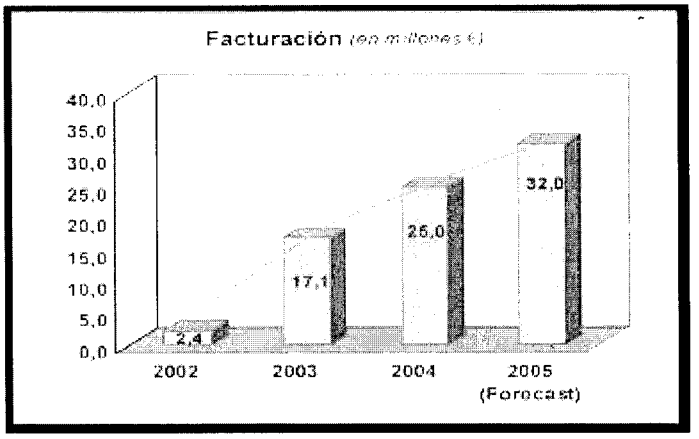


Figura 1.1 - Facturación (en millones de €)

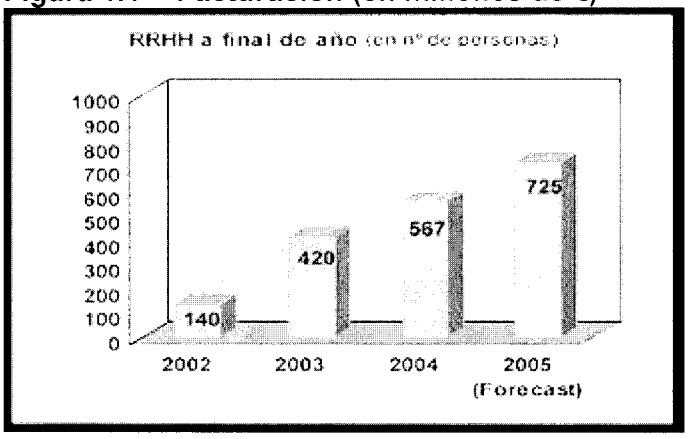


Figura 1.2 - Recursos humanos a finales del año (en número de personas)

1.6 Universidad de las Ciencias Informáticas (UCI)

Tradicionalmente la universidad genera conocimiento pero es muy difícil que la aplique, por lo que se aleja de los problemas de la producción conllevando a que las investigaciones no respondan a las necesidades de las empresas. Por otra parte a las empresas les es muy difícil investigar pues tienen que depender de un mercado cada vez más exigente en tiempo, costo y calidad.

La búsqueda de soluciones ha encontrando la solución en la vinculación Universidad-Empresa, es esta una alianza estratégica de intercambio donde la primera obtiene la facilidad de aplicar sus investigaciones y de vincular sus estudiantes y profesores al mundo

empresarial y de funcionar como una entidad empresarial, la segunda recibe el conocimiento y la innovación constante que generan las universidades.

Es por ello que las entidades han ido descubriendo que necesitan actualizarse al ritmo de estos avances, de lo contrario estarían condenadas al fracaso; se irían quedando detrás en la carrera del desarrollo. De esta forma, en muchos países, se comenzó a promover la vinculación de los estudiantes universitarios y de la universidad en general con entidades externas. Existen diferentes variantes, los contratos entre universidades, los parques científicos, y los parques tecnológicos son espacios donde se materializan estas relaciones.

La universidad de las Ciencia Informáticas (UCI) es la primera universidad surgida al calor de la Batalla de Ideas, sobre la base del nuevo concepto de universidad productiva, logrando una fuerte vinculación Universidad-Empresa. En la UCI la producción es un problema social, político, y económico, el cien por ciento de los estudiantes y profesores se vinculan a la producción participando en proyectos de alto valor tanto nacionales como para empresas extranjeras, se plasma la concepción de que la docencia se pueda realizar desde la producción.

La UCI tiene como objetivos:

- Formar profesionales altamente calificados, comprometidos con su Patria, en la rama de la informática.
- Ser la vanguardia del desarrollo de las empresas de software en Cuba y de llevar la informatización a todos los sectores de la sociedad: Salud, Educación, Cultura, Deporte, Turismo, Prensa, etc.
- Regir y propiciar un avance tecnológico y de la industria del software en Cuba.
- Convertir la industria del software en un renglón fundamental de la economía e insertarnos en el mercado internacional.

Formación Docente – Productiva:

- Las facultades se especializan según perfiles y se vinculan a proyectos productivos.
- Aprendizaje desde la producción.
- Planes de estudio diseñados de manera flexible y con posibilidades de cambio.
- Uso creciente de la Teleformación.
- Se acreditan competencias a través de los proyectos productivos.

Este nuevo concepto de Universidad Productiva implica que la producción pasa a jugar un papel tan importante como la docencia. Los estudiantes deben estar vinculados desde los primeros años a proyectos productivos. En este nuevo proyecto juega un papel fundamental la relación estudio-trabajo, o sea en la Universidad el estudiante y el profesor juegan roles importantes en el desarrollo de productos de software.

Para el logro de estos objetivos se han concentrado en la UCI: recursos humanos, tecnología de punta y conocimiento.

1.7 Tecnologías en la programación para empresas

En el presente epígrafe se describirán un grupo de conceptos relacionados a las tecnologías de desarrollo presentes en el Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour. Antes de abordar la infraestructura de software usada en la línea de producción es necesario dar respuesta a las siguientes interrogantes: ¿Qué es una aplicación Web?, ¿Qué es una aplicación de empresa o empresarial?, ¿Qué es un framework?, ¿Que es un componente? y que especifica la plataforma J2EE.

Luego de ello se puede llegar a definir la infraestructura de software y los servicios que brinda la misma.

1.7.1 Aplicaciones Web

El de Aplicación Web es seguramente el concepto más sencillo de definir. Una Aplicación Web es una pieza de software que se ejecuta en un Servidor Web, este tipo de aplicaciones los usuarios la utilizan a través de Internet o de una Intranet. Se puede considerar como una especialización de una aplicación cliente/servidor en la que el cliente es un navegador Web o, en algunos casos, otros dispositivos capaces de conectarse a Internet. El servidor de una Aplicación Web es un programa que implementa el protocolo HTTP, el cual se mantiene a la espera de peticiones HTTP llevadas a cabo por el cliente. Al recibir la petición responde con el contenido que el cliente solicitó, en forma de hipertextos o páginas Web.

1.7.2 Aplicaciones Empresariales

Esta investigación aborda la creación de aplicaciones de empresa, pero ¿A qué nos referimos exactamente cuando hablamos de una "empresa"?

Una empresa es una organización económica y las aplicaciones de empresa son aquellas aplicaciones de software que facilitan diversas actividades dentro de este tipo de organizaciones.

Las aplicaciones de empresa pueden ser aquellas que hacen concesiones a los usuarios finales vía Internet, a colaboradores vía Internet o redes privadas, a diversas unidades económicas dentro de la empresa vía distintos tipos de interfaces de usuario. En esencia, las aplicaciones de empresa son aplicaciones complejas y complicadas, aquellas que permiten a una empresa gestionar sus actividades económicas. Ejemplos de tales actividades incluyen planificación de recursos, inventarios y catálogos de productos, preparación de facturas, satisfacción de bienes y servicios prestados. Crear aplicaciones para una empresa siempre ha sido un reto, en este sector es donde esta concentrado el mayor por ciento de los software que se realizan en el mundo y es donde mas mercado hay y donde el cliente necesita mas confiar en quien lo produce.

Algunos de los factores que contribuyen a este reto y su complejidad son los siguientes:

- Diversidad de necesidades de información:

  En una empresa, la información es creada y consumida por varios usuarios en una serie de formas diferentes, dependiendo de necesidades especificas.
- Complejidad de procesos económicos:

  La mayoría de procesos económicos de empresa conllevan recabar información compleja, procesarla y distribuirla. Con mucha frecuencia, se enfrentan a una lógica compleja para recabar y procesar información.
- Diversidad de aplicaciones:

  Debido a la compleja naturaleza de los procesos económicos de empresa, es frecuente encontrar una empresa con un gran número de aplicaciones, cada una de ellas creada en distintos momentos para satisfacer las necesidades de distintos procesos económicos. Uno de los retos a los que se enfrentan las empresas hoy en día es la necesidad de conseguir que dichas aplicaciones se comuniquen entre si de modo que los procesos económicos puedan llevarse a cabo sin interrupciones.

Estos factores son muy comunes y las empresas incurren en enormes costes para crear y gestionar aplicaciones que hagan frente a estos retos.

Durante los últimos años, estos retos han alcanzado dimensiones monstruosas. Gracias a Internet y al reciente crecimiento del comercio electrónico, los dispositivos de información de una empresa son ahora más valiosos. Este cambio hacia una economía de la información está forzando a muchas compañías a reformular incluso sus prácticas económicas más básicas. Con el objetivo de mantener un margen competitivo, la adopción de nuevas tecnologías para cubrir las necesidades del momento rápidamente se ha convertido en un factor clave en la habilidad de una empresa para explotar al máximo sus

dispositivos de información. Fundamentalmente, la adaptación de estas nuevas tecnologías para que funcionen en combinación con los sistemas legados ya existentes es ahora uno de los requisitos principales de la empresa.

Uno de los lugares donde estos cambios se han percibido de un modo más intenso ha sido en el ámbito del desarrollo de aplicaciones. A lo largo de los últimos años, los fondos y el tiempo invertidos en el desarrollo de aplicaciones se han visto reducidos, mientras que las demandas para crear complejos procesos económicos han aumentado. Sin embargo, ambos son obstáculos que los promotores pueden superar, pero también se encuentran con los siguientes requisitos que cumplir:

- Productividad de programación:

  La adopción directa de nuevas tecnologías es insuficiente si estas no son utilizadas adecuadamente aprovechando su potencial al máximo y si no son integradas apropiadamente con otras tecnologías relevantes. De este modo, la habilidad para desarrollar y después desplegar las aplicaciones tan rápida y efectivamente como sea posible es muy importante.

- Fiabilidad y disponibilidad:

  En la economía actual de Internet, el tiempo de inactividad puede resultar fatal para una empresa. La habilidad para organizar y poner en marcha sus operaciones de base Web, y mantenerlas activas, es crucial para lograr el objetivo con éxito. Y por si no fuera suficiente, también debe ser capaz de garantizar la fiabilidad de sus transacciones económicas de modo que sean procesadas por completo y con exactitud.

- Seguridad:

  Internet no solo ha aumentado exponencialmente el número de usuarios potenciales sino también el valor de la información de una compañía, de modo que la seguridad de esta información se ha convertido en una preocupación de primer orden. Es mas, a medida que las tecnologías son más avanzadas, las aplicaciones más sofisticadas y las empresas más complejas, poner en marcha un modelo de seguridad efectivo es cada vez más difícil.

- Reajustabilidad:

  La capacidad de una aplicación para crecer y ajustarse a una nueva demanda, tanto en su operación como en su base de usuario, es esencial.

- Integración:

  Aunque la información ha crecido hasta convertirse en un dispositivo económico clave, mucha de esta información existe como datos en sistemas de información viejos y

obsoletos. Para maximizar la utilidad de esta información, las aplicaciones deben poder ser integradas en los sistemas de información ya existentes, lo cual no es necesariamente una tarea fácil. La habilidad para combinar las nuevas y las viejas tecnologías es fundamental para el éxito del desarrollo de las empresas de hoy en día.

Lo que las empresas están buscando es una tecnología y una infraestructura permisiva que simplifique algunos de los temas técnicos complejos.

Ninguno de estos ámbitos problemáticos es especialmente nuevo para los creadores de programas de una empresa. Pero resolverlos de un modo rentable sigue siendo crucial. Puede que sea consciente de que han existido diversas tecnologías para resolver una o más de las demandas anteriores. Sin embargo, lo que muchas veces ha faltado ha sido una plataforma con una rica infraestructura y numerosas posibilidades de arquitectura que promueva al mismo tiempo un rápido entorno de desarrollo.

1.7.3 Frameworks y Componentes de Software

En la bibliografía consultada se encontraron las siguientes definiciones de framework, aplicadas al ámbito del desarrollo del software:

- Conjunto de clases que cooperan y forman un diseño reutilizable para un tipo específico de software. Un framework ofrece una guía arquitectónica partiendo el diseño en clases abstractas y definiendo sus responsabilidades y sus colaboraciones. Un desarrollador personaliza el framework para una aplicación particular mediante herencia y composición de instancias de las clases del framework [Gamma et al., 1995].
- Infraestructura software que crea un entorno común para integrar aplicaciones e información compartida dentro de un dominio dado [SEMATECH, 1998].
- Es una aplicación semicompleta que contiene componentes estáticos y dinámicos que pueden ser personalizados para obtener aplicaciones de usuario específicas [Fayad et al., 1999].

Tras leer estas definiciones se puede concluir que un framework para el desarrollo de aplicaciones Web y/o empresariales, es un conjunto de clases que cooperan y forman un diseño reutilizable formando una infraestructura que facilita y agiliza el desarrollo de las aplicaciones, según ciertas normas establecidas por el framework.

Si el concepto de framework plantea problemas en su definición, estos son todavía más complejos al intentar definir el concepto de componente. El significado común de la palabra es pieza o parte. Pero el sentido que se aplica en la Ingeniería del Software da pié a largas discusiones, sobre todo si intentamos diferenciar entre componentes, objetos y módulos.

Una definición ampliamente aceptada es la debida a Clemens Szyperski, uno de los padres de la programación basada en componentes:

Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio [Szyperski y Pfister, 1997].

Es decir, un componente es una unidad de composición, que puede estar formada por recursos heterogéneos y que puede ser desarrollado y desplegado de forma independiente al resto del sistema del que va a formar parte.

1.7.4 J2EE y la programación para empresas.

En la actualidad, Java es uno de los lenguajes de programación más elaborados y más utilizados para la creación de software de empresa. La evolución de Java, que ha pasado de ser un medio de desarrollo de applets para ser ejecutados en navegadores a un modelo de programación capaz de manejar las aplicaciones de una empresa de hoy en día, ha sido extraordinaria. Con el paso de los desafíos, Java ha desarrollado tres ediciones de plataformas diferentes, cada una de ellas destinada a cubrir un conjunto diferente de necesidades de programación:

- La plataforma Java 2, Standar Edition (J2SE): Es la plataforma más utilizada dentro de Java, consistente en un entorno de tiempo de ejecución y un conjunto de varios API para crear una amplia variedad de aplicaciones que abarca desde applets, pasando por aplicaciones independientes ejecutables en distintas plataformas, hasta aplicaciones de cliente para diversas aplicaciones de empresa.
- La plataforma Java 2, Enterprise Edition (J2EE): J2EE es una plataforma para crear aplicaciones de servidor o aplicaciones para el ámbito empresarial. Como se ha explicado en el epígrafe anterior, las aplicaciones de empresa tienen requisitos adicionales durante la fase de desarrollo. J2EE proporciona la infraestructura necesaria para satisfacer estas necesidades.
- La plataforma Java 2, Micro Edition (J2ME): Esta edición, la ultima en agregarse a la familia Java, permite la creación de aplicaciones Java para "micro-dispositivos" (dispositivos con un apoyo de pantalla y memoria limitado, como teléfonos móviles, PDAs, etc.).

J2EE es una de las mejores soluciones que existen para satisfacer las demandas de las empresas de hoy en día. Especifica tanto la infraestructura para gestionar sus aplicaciones, como los servicios API para construir sus aplicaciones.

La Plataforma J2EE es esencialmente un entorno distribuido aplicación-servidor, un entorno Java que ofrece:

- Un conjunto de varios API de extensión Java para construir aplicaciones. Estos API definen un modelo de programación para aplicaciones J2EE.
- Una infraestructura de periodo de ejecución para albergar y gestionar aplicaciones. Este es el periodo de ejecución en el que residen sus aplicaciones.

Las aplicaciones que puede desarrollar con estos dos elementos pueden ser desde sencillas aplicaciones Web hasta complejas aplicaciones de empresa, que pueden estar conformadas por componentes para implementar transacciones complejas de bases de datos, Web Services todos ellos distribuidos por la red.

1.8 Framewok C4J

En todo desarrollo, y sobre todo cuando se trata de una aplicación distribuida, aparece la necesidad de definir una arquitectura basada en la plataforma y/o lenguaje que se vaya a utilizar. Esta arquitectura debe proporcionar al desarrollador todos los servicios necesarios: control de excepciones, mecanismo de log, acceso a datos, presentación.

Además de la arquitectura, casi siempre se repiten las mismas necesidades: cómo descargar un fichero desde el cliente al servidor, cómo gestionar las excepciones, como enviar un correo electrónico o incluso cómo realizar un listado de resultados parcial. Todos estos problemas se pueden resolver siempre de una manera u otra. La manera en la que lo resolvamos, hará que la solución sea más o menos costosa, más o menos óptima e incluso más o menos mantenible. Por ello, se definen los denominados patrones, donde para cada problema se define su solución.

En la plataforma J2EE estas necesidades se traducen en lo que se denomina un Framework J2EE, un ejemplo de ello es el utilizado en la línea de desarrollo de la Factoría de la UCI, el Framework C4J (Carrefour for Java).

C4J es un framework para el desarrollo de aplicaciones J2EE resultado de la aplicación del patrón de arquitectura MVC y del uso de patrones de diseño y patrones J2EE. En el se integran varias soluciones OpenSource para brindarle un grupo de servicios al programador que le faciliten el desarrollo de aplicaciones de comercio electrónico dándole solución a las

necesidades anteriormente mencionadas. Este framework lo usa la empresa Machmind, para la cual trabaja actualmente la Factoría de Software de la UCI.

1.8.1 Arquitectura de las aplicaciones con el Framework C4J

Cuando se inicia un Framework se plantean 2 posibilidades: comenzar desde cero o bien partir de un Framework base y extenderlo. C4J extiende a uno de los frameworks con mayor aceptación en la comunidad Java, Struts. Struts posibilita la creación de aplicativos Web basado en el patrón de diseño Modelo Vista Controlador que permite la separación de las distintas capas de la aplicación (presentación, lógica de negocio y datos), proporciona una serie de facilidades que permiten que nos podamos concentrar en la lógica y el control del aplicativo, y dejar de lado otros aspectos más mecánicos que ya tiene implementados. C4J tiene a Struts como base para poder proporcionar todos los servicios requeridos, de igual manera utiliza patrones de diseño que nos permiten estructurar la aplicación por capas de la mejor manera.

Las aplicaciones construidas con C4J tienen la siguiente arquitectura:

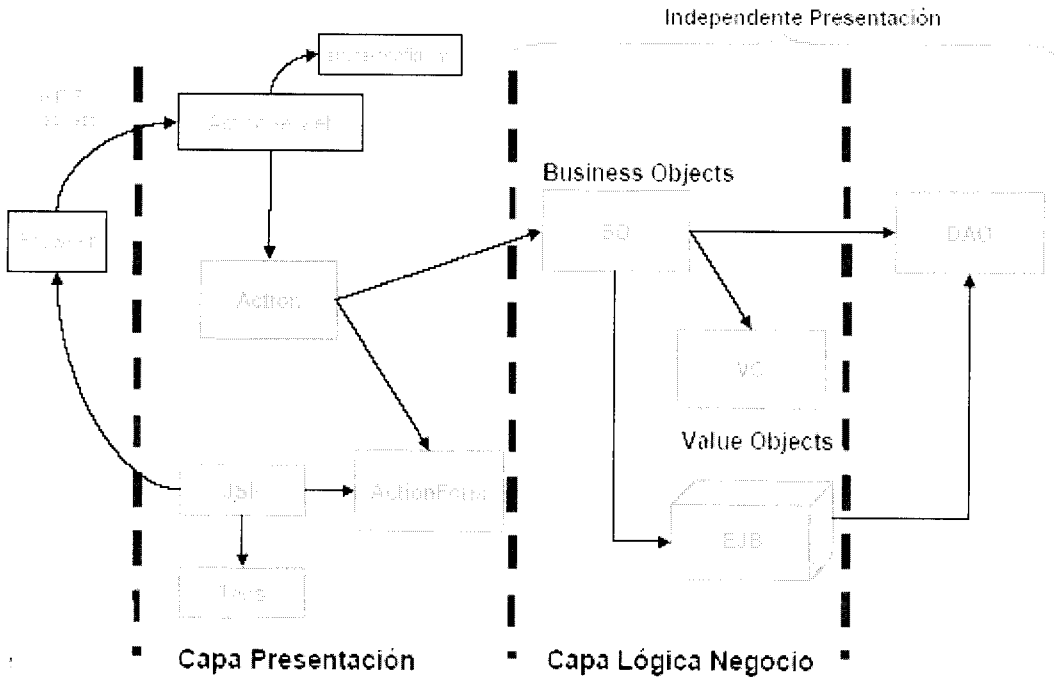


Figura 1.3 - Arquitectura de las aplicaciones desarrolladas con el Framework C4J.

La Capa de Presentación se define básicamente por la arquitectura de Struts.

1. ActionServlet: Representa el controlador de la aplicación, es el que recibe todas las peticiones.

2. ActionForm: Objeto java de formulario asociado a la petición, contiene los datos de la misma.
3. Action: Clase que resuelve la petición, representa la acción a ejecutar.
4. JSP: Páginas que representan las diferentes vistas de la aplicación.

La Capa de Negocio se define mediante 3 componentes básicos:

1. Business Objects (BO). Son los objetos de proceso de la lógica de negocio con java puro, de manera que la capa de presentación sea independiente de la lógica. Esta separación permite que por futuras tendencias hacia otros frameworks o tecnologías, la lógica de negocio se mantenga invariable.
2. Value Objects (VO). Son los objetos que mantienen los datos de la lógica de negocio.
3. EJBs. En caso necesario se puede requerir del uso de EJBs

La Capa de Acceso a Datos se define principalmente por los Data Access Object (DAO), objetos que encapsulan toda la lógica de acceso a datos.

1.8.2 Servcios del Framework C4J

El Framework C4J posee servicios o utilidades tales como: Servicio de Acceso a Datos, Servicio de Alertas, Servicio de Planificación de Tareas, Servicio de Colas, Servicio de Configuración, Servicio de Excel, Servicio de Excepciones, Servicio de Ficheros, Servicio de Generación de Reportes, Servicio de Logs, Servicio de Mailing, Servicio de Acceso Lotus Notes, Servicio de Presentación Usuario, Servicio de Presentación Usuario Arquitectura Básica, Servicio de Presentación Usuario Listados, Servicio de Presentación Usuario Multiidioma, Servicio de Presentación de Pantallas, Servicio de Presentación Usuario Tags, Servicio de Presentación Usuario Upload, Servicio de Presentación Usuario Validación, Servicio de Pruebas Unitarias, Servicio de Seguridad Usuario, Servicio XML Usuario.

Los detalles de los servicios se encuentran en el Anexo 1.

1.9 Conclusiones

En este capítulo se han definido conceptos fundamentales a conocer para la obtención del Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour. Después de analizar las definiciones abordadas fue posible percatarse de que las Factorías de Software son las encargadas de industrializar el desarrollo de software, llevar ese proceso del ámbito artesanal al industrial o sea a una "fábrica" donde se establezcan procedimientos y métodos definidos que cuenten con herramientas que ayuden en su implantación. Además de llegar a la conclusión de que las Factorías de Software como las fábricas tradicionales tienen establecido líneas de producción. Teniendo en cuenta que el modelo funcional que se propone es para la línea de aplicaciones Carrefour, se puede concluir que las características y peculiaridades de este tipo de aplicaciones, y la arquitectura del Framework C4J utilizado en la producción en la factoría, inciden notablemente en el modelo.

Capítulo 2 - Evaluación de Modelos Existentes

2.1 Introducción

En el presente capítulo se presentara una selección de los modelos de Factoría de Software más representativos encontrados en la bibliografía consultada durante la investigación realizada. Los mismos servirán como base para elaborar el Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour, a partir de los elementos más importantes identificados en cada uno de ellos. Se abordarán cinco modelos, los mismos son:

- Modelo basado en la norma ISO 9001 y CMM.
- Modelo Eureka.
- Modelo Clasificatorio.
- Modelo propuesto por Basili.
- Modelo Replicable.

2.2 Modelo basado en la norma ISO 9001 y CMM

En este modelo se hace una división de los elementos fundamentales de una Factoría de Software en cinco entidades bien definidas.

Tabla 2.1 - Entidades del modelo

Entidad	Descripción
1. Técnicas	Comprende el contexto de las técnicas que sirven de soporte al proceso de desarrollo, técnicas para la reutilización de software, para el desarrollo basado en componente y otras técnicas utilizadas por la factoría.
2. Proceso	Representa el proceso de desarrollo de software, los flujos de trabajo y actividades que componen el mismo.
3. Trabajadores involucrados	Personas que actúan directamente en el desarrollo de software.
4. Gestión de la factoría	Define la estructura organizacional de la Factoría de Software, el proceso fabril y la gestión de calidad.
5. Activos del proceso, herramientas y	Entiéndase como activos del proceso modelos, patrones, algoritmos utilizados como artefactos en el proceso. Los

componentes de código	activos del proceso también pueden ser denominados como componentes de infraestructura, componentes de valor en el proceso.

Este es un modelo en el que cualquier Factoría de Software puede adaptar las entidades que componen el mismo de acuerdo a sus características y necesidades.

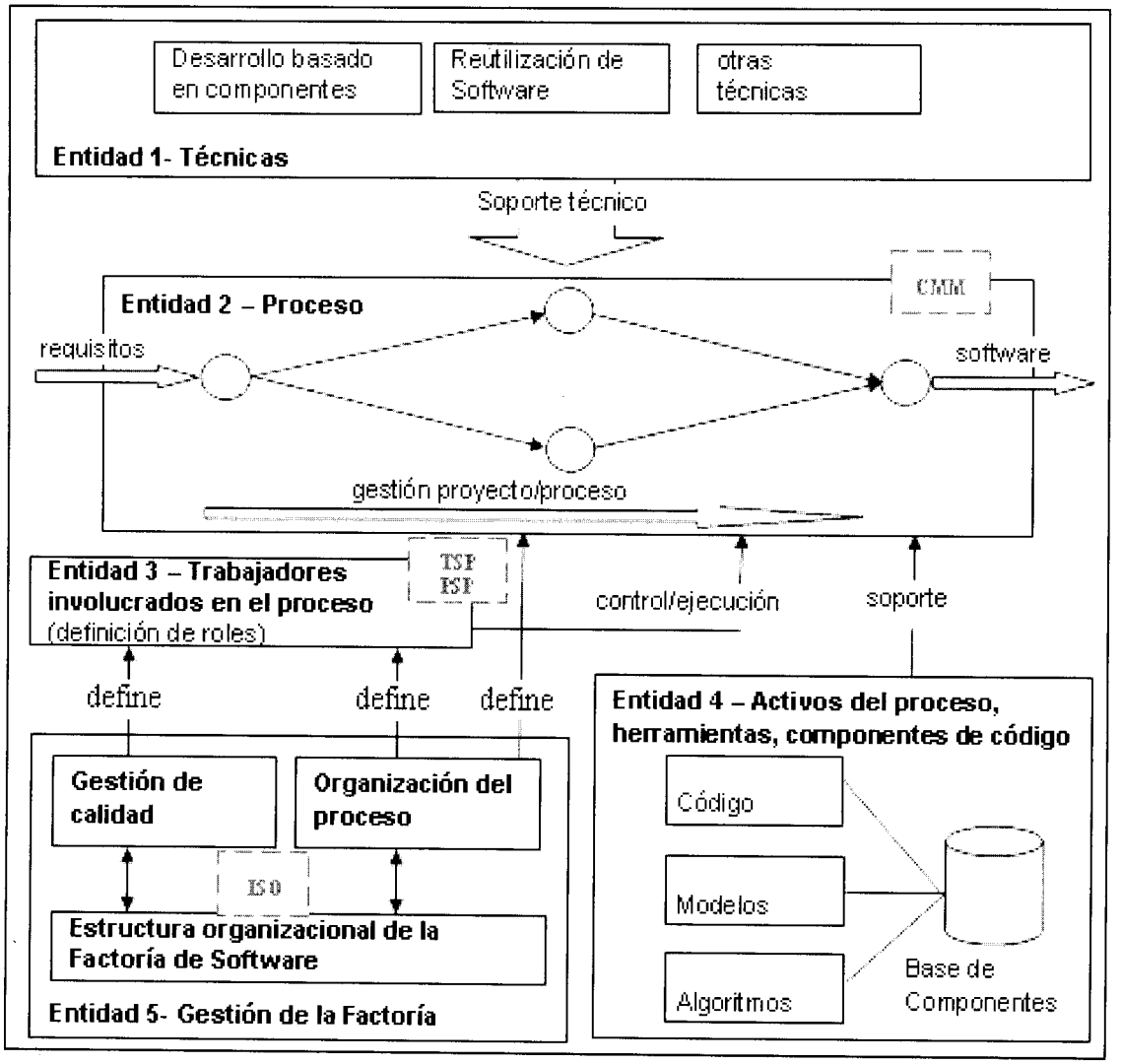


Figura 2.1 - Modelo basado en la norma ISO 9001 y CMM

La arquitectura propuesta por el modelo se puede ver en la Figura 2.1. En la misma se observa que la entidad Técnicas provee el soporte técnico y conceptual para la definición del proceso. Este es guiado por el estándar de calidad CMM, los requisitos de calidad para la organización de la factoría son definidos por la norma ISO 9001. El modelo toma la

norma ISO 9001 como un estándar utilizado en el contexto industrial cuyo enfoque está en el sistema de calidad organizacional, propone un conjunto de principios probados para mejorar la calidad final del producto mediante mejoras en la organización de la empresa. CMM es designado para la industria del software, de este modo las áreas claves proveen detalles importantes para la evaluación y mejora del proceso de desarrollo, su propósito es guiar a las organizaciones en la selección de estrategias de mejora determinando la madurez del proceso actual e identificando los puntos importantes que se deben estudiar y trabajar para mejorar tanto el proceso como la calidad del software.

La entidad Gestión de la Factoría define, a través de las sub-entidades Gestión de Calidad y Organización del Modelo de Proceso, los trabajadores involucrados en el proceso de desarrollo de software y sus roles. La sub-entidad Organización del Modelo de Proceso define características y organización del proceso de desarrollo de software. Los Trabajadores Involucrados son guiados por los modelos PSP (Personal Software Process) y TSP (Team Software Process).

Los activos del proceso, las herramientas y los componentes de código dan soporte al proceso de desarrollo de software.

Lo que más aporta este modelo es la definición de los elementos o entidades que forman el Modelo factoría de Software y las relaciones que se establecen entre ellas, así como la aplicación de normas y técnicas de calidad usadas hoy en el mundo del software.

2.3 Modelo Eureka

El modelo Eureka surgió como el proyecto Eureka Software Factory. El objetivo del proyecto es crear un mercado para productos CASE. En el mismo participan un conjunto de compañías europeas, tales compañías actúan en las siguientes áreas: manufactura de computadoras, instituciones de investigación, producción de herramientas CASE y desarrollo de sistemas.

El modelo fabril propuesto por el proyecto Eureka está compuesto de proceso, reglas, herramientas, información, trabajadores y equipamiento (computadoras). La arquitectura del mismo puede verse en la Figura 2.2.

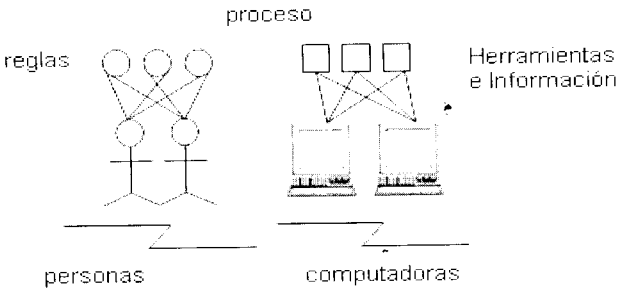


Figura 2.2 - Modelo Eureka

En la Figura 2.2 se percibe que el proceso de desarrollo está compuesto por reglas, las que son definidas por las personas involucradas en el ambiente de desarrollo de software y constituyen patrones a seguir, algoritmos, métodos de desarrollo de software. Las herramientas e información almacenada, soportan la automatización del proceso de desarrollo.

El modelo posee características giradas al proceso de desarrollo de software distribuido, en el mismo se sigue el enfoque software bus. Enfoque que estipula reglas de conexión de componentes en la construcción de un software (ver Figura 2.3).

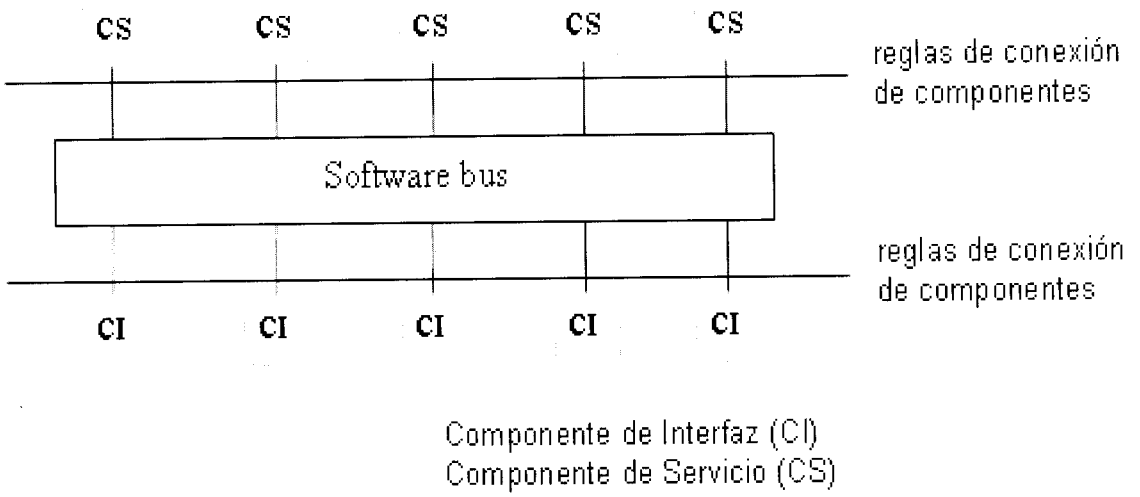


Figura 2.3 - Enfoque Software bus en el Modelo Eureka

En la Figura 2.3 se percibe que el bus conecta los componentes de servicios (CS) y los de interfaz (CI). Esos componentes pueden ser desarrollados por diversas Factorías de Software en localidades diferentes, y su conexión se realiza de acuerdo a reglas y lineamientos establecidos. Las técnicas de distribución permiten que las factorías que usen ese modelo puedan compartir los componentes para la construcción del software.

En el modelo se clasifican los componentes en CS y CI, reflejándose un desarrollo de software por capaz.

Existe una semejanza con el concepto de línea de montaje en el proceso industrial, destacándose la integración de piezas (componentes) a través del software bus.

El aporte de este modelo está en el desarrollo distribuido de software, da una visión de cómo se puede distribuir la construcción de un producto software entre diferentes factorías, y después realizar la unión de los componentes elaborados por cada una para formar el producto final.

2.4 Modelo Clasificatorio

El Modelo Clasificatorio propuesto por Fernandes y Teixeira [Fernandes y Teixeira, 2004] está dirigido a clasificar las factorías de acuerdo al alcance o ámbito de funcionamiento que tienen a lo largo del proceso de desarrollo de software. Una Fábrica de Software puede ser clasificada como:

1. Factoría de Proyectos Ampliada.
2. Factoría de Proyectos de Software.
3. Factoría de Proyectos Físicos.
4. Factoría de Programas.

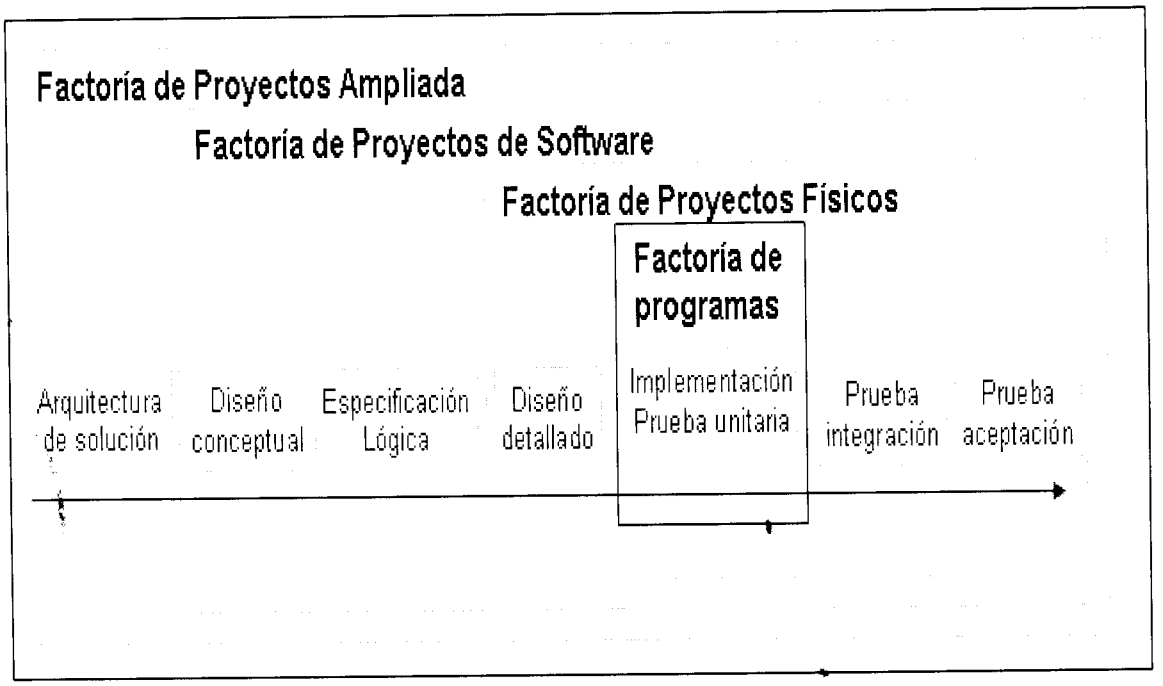


Figura 2.4 - Modelo Clasificatorio

En la Figura 2.4 se puede observar que una *Factoría de Proyectos Ampliada* comprende el concepto de arquitectura de solución. La arquitectura de solución es una etapa anterior al diseño conceptual del software, la cual se ocupa en proyectar una solución en la que el software está formado por los componentes más significativos arquitectónicamente, se definen los principios que orientan el diseño y evolución del software. La arquitectura de solución puede contener, además del software, definición de procesos, definición de equipamiento, infraestructura de redes, plataforma de desarrollo, patrones a seguir.

La *Factoría de Proyectos de Software* abarca todo el ciclo de vida sistémico para la realización del software, correspondiente al análisis, diseño, implementación, prueba e implantación. En este tipo de factorías se tiene un conocimiento al detalle del negocio a automatizar.

La *Factoría de Proyectos Físicos* se abstrae del enfoque sistémico del software, se dedica al diseño, implementación y prueba. No se tiene un pleno conocimiento del negocio.

La *Factoría de Programas*, considerada la menor de las entidades, tiene como objetivos desarrollar componentes de código para la construcción del software. Esta factoría no se preocupa del contexto sistémico ni del diseño, se ocupa de producir código según las especificaciones del diseño. Posee como entrada la especificación del diseño de una parte del software y su salida es un componente de código que formará parte del software a desarrollar.

El mayor aporte de este modelo radica en que permite clasificar la Factoría de Software de la UCI de acuerdo al alcance de esta en el proceso de desarrollo, la misma se clasifica como una Factoría de Proyectos Físicos ya que en estos momentos se realiza diseño detallado, implementación y prueba.

2.5 Modelo propuesto por Basili

El presente modelo divide una factoría de software en dos grandes entidades: organización basada en proyectos y factoría de componentes. El autor plantea que una organización con características de Factoría de Software debe poseer una estructura de construcción de software basada en componentes. Los componentes utilizados en la construcción del software pueden ser desarrollados por la factoría de componentes.

Como muestra la Figura 2.5, el modelo se divide en organización basada en proyectos de software (unidad de producción de software), y factoría de componentes (unidad de producción de componentes). La Organización basada en proyectos realiza las solicitudes

de productos (componentes para la construcción del software), de datos (estadística para la estimación de costo y plazos) y de planos (modelos, métodos para el análisis y diseño de software) a la factoría de componentes. La factoría de componentes posee una base de componentes reutilizables, de la cual se apoya para dar respuesta a las solicitudes hechas por la unidad de producción de software. En respuesta a la solicitud la organización basada en proyectos recibe los modelos y componentes para la construcción del software, además de estadísticas y datos históricos que se encuentran en la base de componentes.

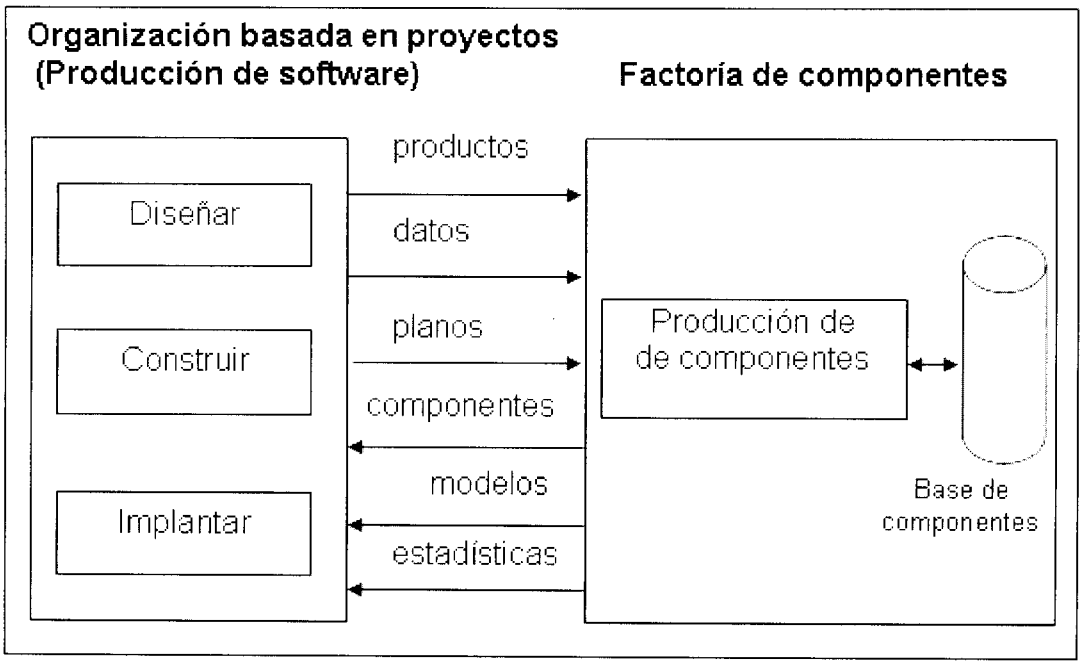


Figura 2.5 - Modelo propuesto por Basili

Este es un modelo que puede ser adaptado a las características de una determinada Factoría de Software, las actividades diseñar, construir e implantar, no son únicas y necesarias en la formación del proceso de producción de una Factoría de Software. Una factoría puede configurar sus unidades de producción de software con estas y otras actividades. Esa premisa también vale para la unidad de producción de componentes.

Comparando el presente modelo con la idea de línea de producción y montaje del proceso industrial promovida por Ford, es posible notar también una fuerte presencia de intercambio e integración de piezas, en el caso de la línea de producción de software, esas piezas son denominadas componentes de software.

El mayor aporte de este modelo es la división de la factoría en dos unidades, y también se enfoca en la reutilización durante el desarrollo, para esto se propone tener una base de componentes reutilizables.

2.6 Modelo Replicable

El presente modelo fue desarrollado para ser replicado en una factoría de software. Se puede adaptar a las necesidades y recursos de una factoría determinada. Reúne las características más importantes de los modelos anteriores, es el que mas se desarrolla dado el alcance que tiene.

2.6.1 El modelo Fabril

Este modelo plantea que una factoría de software debe poseer:

- Un modelo de organización de la producción.
- Una unidad de producción de componentes y una unidad de producción de software.
- Tanto la unidad de producción de componentes como la de software poseen un proceso.
- El proceso es guiado por un modelo de calidad de software.
- El proceso es compuesto de actividades que son compuestas de tareas.
- Las tareas utilizan los componentes, y estos son clasificados en infraestructura (o activos del proceso) y código.
- Las tareas usan un conjunto de herramientas para la automatización de las mismas.
- Por último el proceso puede ser aplicado al desarrollo de software o al desarrollo de un componente.

Los elementos que componen el modelo de Factoría de Software están presentes en la Figura 2.6.

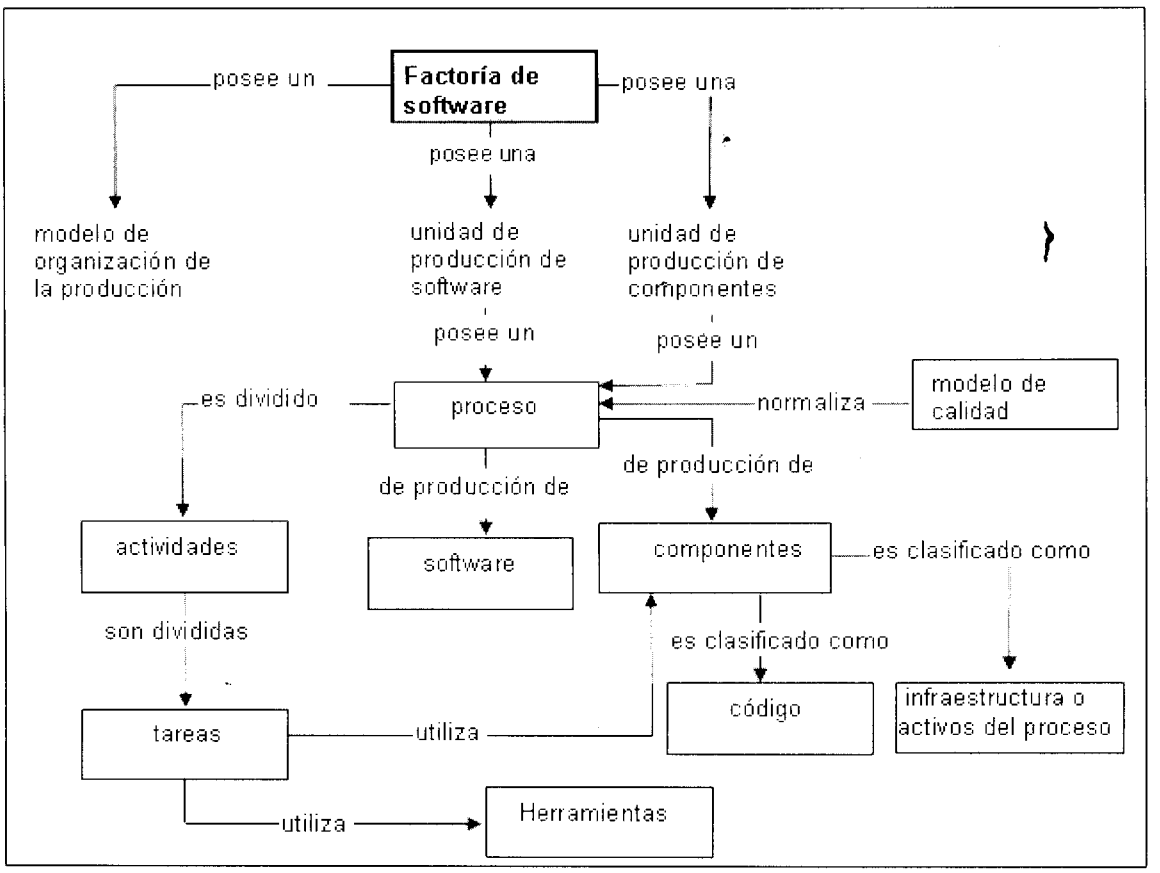


Figura 2.6 - Elementos del Modelo Replicable

2.6.1.1 Organización de la Producción

En el presente modelo se define la organización de la producción de una Factoría de Software dividiendo la misma en cinco áreas. Estas son:

1. Área de producción de análisis de sistema o modelado de negocio: Esta área es la encargada del estudio de un área de trabajo o algún negocio, llevando al análisis y especificación del sistema a desarrollar.
2. Área de producción de diseño de software: Es el área encargada de realizar el diseño a partir de la especificación proveniente del análisis del sistema. Un software automatiza un sistema o negocio, por lo que esta área se encuentra incluida en el área de análisis de sistema o modelado de negocio.
3. Área de construcción de software: Esta área es la encargada de construir el producto software de acuerdo a las especificaciones desarrolladas en el área de diseño. Es importante destacar que en una Factoría de Software la construcción se basa en la unión o ensamblado de componentes previamente desarrollados.

4. Área de producción de componentes de infraestructura o activos del proceso: Como que una Factoría de Software posee una unidad de producción de componentes, y estos pueden ser clasificados como componentes de infraestructura o de código, por lo tanto esta área tiene como principal responsabilidad producir componentes de infraestructura o activos del proceso solicitados por el área de construcción de software.
5. Área de producción de componentes de código: Esta área tiene la función de producir componentes de código solicitados por el área de construcción de software.

La unión de las áreas de análisis de sistemas, diseño de software y construcción de software forman el ámbito de negocio del modelo de producción. El ámbito de negocio incluye la interacción entre el cliente y la fábrica de software.

Las áreas de producción de componentes de infraestructura y componentes de código forman el ámbito interno del modelo, el que es transparente a los ojos del cliente de la factoría. Este ámbito es el responsable de los subproductos creados, componentes para la construcción del sistema.

2.6.1.2 Definición de las actividades en la unidad de producción de software y en la unidad de producción de componentes.

El presente modelo plantea que una Factoría de Software posee una unidad de producción de software y una de producción de componentes. La unidad de producción de software utiliza componentes en la construcción del sistema, que son elaborados por la unidad de producción de componentes.

El proceso en ambas unidades está dividido en actividades, y estas están basadas en la norma IEEE std 1220-1998, (ver Anexo 2).

La unidad de producción de software posee las siguientes actividades: analizar, diseñar, construir, probar, implantar y revisar. Para la unidad de producción de componentes son las siguientes: diseñar componentes, construir, probar, almacenar y distribuir componentes. En la Figura 2.7 se muestran ambas unidades y las actividades que poseen.

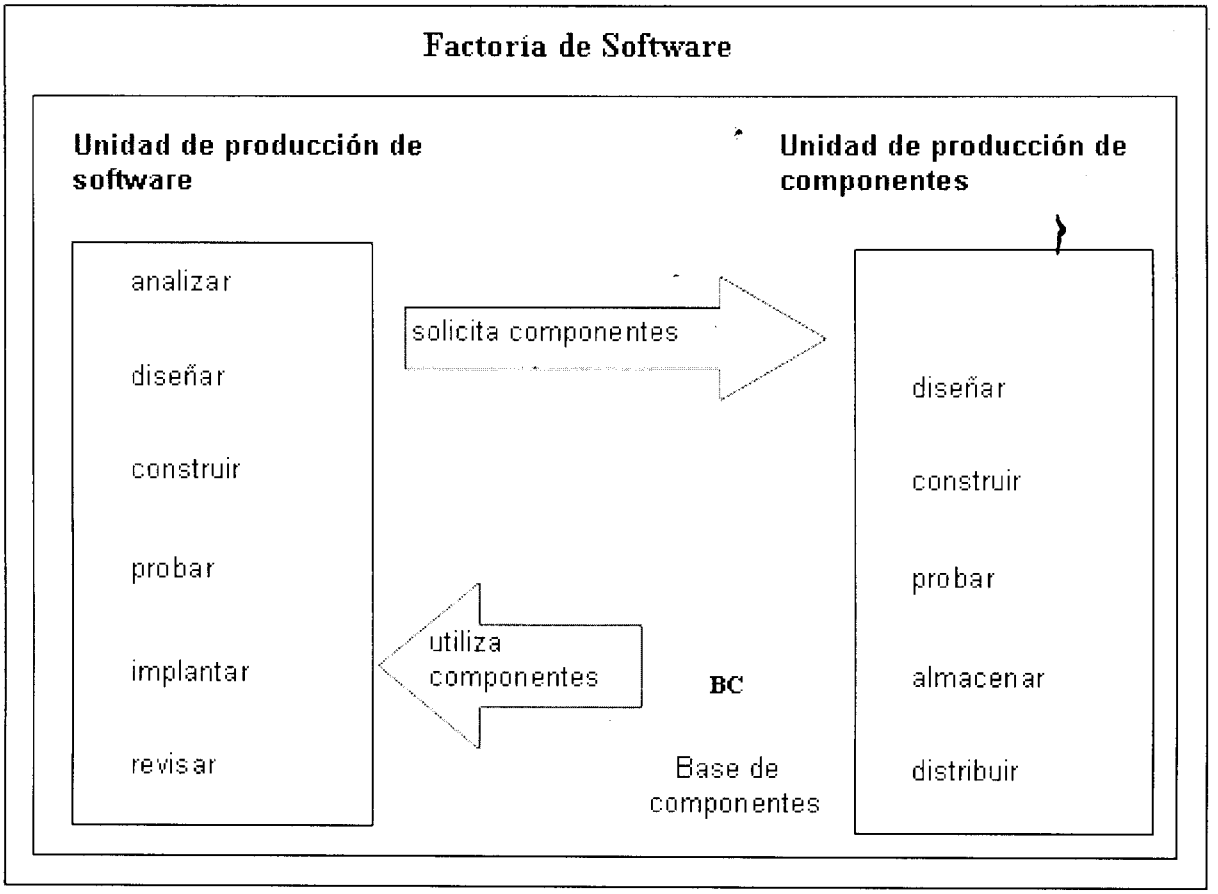


Figura 2.7 - Unidades de la Factoría de Software.

Este modelo plantea realizar la construcción del software mediante la integración de componentes producidos por la unidad de producción de componentes, lo que se asemeja a la línea de producción y montaje del proceso industrial tradicional.

En el Anexo 3 se puede ver una representación gráfica de las actividades y tareas del proceso de desarrollo en la unidad de producción de software y en la de producción de componentes.

2.6.2 El proceso Industrial para el desarrollo de software.

El proceso de desarrollo de software es un conjunto de actividades y resultados asociados que llevan a la producción de un producto software. A continuación se presentan las actividades y tareas del proceso propuesto por el presente modelo.

2.6.2.1 Definición de tareas de la actividad Analizar.

La actividad Analizar tiene como objetivo definir los eventos sistémicos y modelar el negocio del cliente que solicita la construcción del software. Las tareas pertinentes a esa actividad son: definir el alcance del sistema o negocio; definir información sobre los actores;

definir eventos sistémicos; modelar eventos sistémicos; estimar costo, plazos y recursos; interactuar con el cliente (presentación de los eventos sistémicos para la aprobación); definir bases tecnológicas y gestionar la actividad.

Definición del alcance del sistema o negocio:

En esta tarea los analistas deberán establecer un contacto con los clientes para definir:

- La información de la organización: capturar los datos que definen la organización.
- Los objetivos del sistema o negocio: capturar cual es la misión de la organización, su principal actividad, actividades secundarias.
- La frontera entre la organización y sus clientes: capturar los clientes primarios, cuales son las relaciones que la organización mantiene con esos clientes.
- Las reglas del sistema o negocio: información inherente a reglas de operación de negocio. Ejemplo: leyes y regalamientos impuestos al negocio.
- Las personas claves: identificación de actores.
- Los principales problema presentados por la organización.

Obtener información de los actores:

En esta tarea los analistas deberán establecer un contacto con cada actor presente en el negocio y utilizando técnicas de recopilación de información obtener:

- Responsabilidad de los actores en la organización.
- Acciones o tareas que los actores desarrollan en el sistema de negocio.
- Relación entre los actores.
- Herramientas utilizadas por esos actores en el ambiente de trabajo.
- Principales problemas enfrentados por los actores dentro de la organización.

Definir eventos sistémicos:

Para definir los eventos sistémicos se utiliza la definición del alcance del sistema y la información de los actores.

Suponiendo, en un sistema cualquiera, que un actor externo X realiza las siguientes tareas: atender clientes y almacenar sus datos. X se relaciona con el actor externo Y solicitando la aprobación del cliente en el momento de efectuar una compra. El actor X posee la siguiente restricción: almacenar solamente los datos de los clientes aprobados por Y. Concluyéndose entonces que:

Existen tres eventos sistémicos relacionados a X: atender cliente, almacenar los datos del cliente y solicitar aprobación del cliente en el momento de la compra. También es posible inferir un evento sistémico relacionado a Y: aprobar cliente en el momento de la compra.

Por último, el cliente se relaciona con el evento sistémico comprar productos. Los eventos sistémicos se registran como se muestra en la siguiente tabla.

Evento sistémico	Actor externo	Restricciones
Atender cliente	X	
Almacenar datos del cliente	X	Para clientes aprobados
Solicitar aprobación del cliente	X	
Aprobar cliente	Y	
Comprar productos	Cliente	

Tabla 2.2 - Definición de eventos sistémicos

Modelar Eventos Sistémicos:

Esta tarea los analistas la pueden desarrollar haciendo uso de la Tabla 2.2.

Este modelo de Factoría propone que en el modelado de los eventos sistémicos los analistas pueden utilizar cualquier notación definida en la literatura, por ejemplo: diagrama de flujo de datos, IDEF-0 (orientación a proceso) y casos de uso (orientación a objetos).

Estimar costos, plazos y recursos:

Después de modelar los eventos sistémicos, es necesario estimar costos, plazos y recursos utilizando herramientas del área de métricas o informaciones almacenadas en bases históricas de proyectos de la fábrica.

Interactuar con el cliente:

Después de realizadas las tareas de definir el alcance del sistema o negocio; obtener informaciones de los actores; definir y modelar eventos sistémicos; y estimar costos, periodos y recursos se hace necesario presentar lo realizado al cliente de la Factoría de Software. Esta tarea debe ser documentada formalmente con la siguiente información: número de interacción; datos, hora de inicio y terminación de la interacción, personas involucradas, recursos necesarios, eventos sistémicos que serán transformados en software, definir las bases tecnológicas y las alteraciones necesarias o consideraciones del cliente en relación al modelo propuesto.

El hecho de priorizar los eventos sistémicos (debido al proceso incremental) que serán implementados, es una de las acciones pertinentes de la interacción con el cliente.

Definir bases tecnológicas:

En esta tarea es donde se definen cuestiones técnicas importantes para el desarrollo del sistema, tecnologías sobre las que el sistema va ser implantado como son: sistemas operativos en los que va a funcionar, tecnología de redes, protocolos de comunicación; y demás aspectos técnicos que guíen el proyecto.

Gestión de la actividad:

La Factoría de Software debe gestionar la actividad de análisis en relación al tiempo, costo y recursos humanos. La gestión es establecida por proyecto, por ejemplo: la actividad de análisis del proyecto P de la empresa X dura H horas, consumirá A analistas y el costo por horas de cada analista es R.

2.6.2.2 Definición de tareas de la actividad Diseñar

Uno de los principales objetivos de esta actividad es definir la arquitectura del software. En este modelo se trata en la arquitectura cuestiones como: definición de funcionalidades (un evento sistémico puede poseer varias funcionalidades), tamaño del software, definir componentes, distribución física del software, desempeño y selección entre las alternativas para el desarrollo del proyecto, patrones.

Para el desarrollo del documento de arquitectura de software se utiliza la definición de las bases tecnológicas y la definición de eventos sistémicos que serán automatizados (eventos de software), definiciones realizadas en la actividad anterior.

Esta actividad está dividida en 6 tareas: modelar funcionalidades, modelar estructura de datos, modelar interfase de acceso de los usuarios, definir el tamaño del software con mayor rigor, definir el modelo de implantación, interacción con el cliente, definir la composición de los componentes que integrarán el sistema, y gestionar la actividad.

Modelar funcionalidades:

El modelo funcional reúne un conjunto de acciones, derivadas de un evento de software, que trabajan en conjunto para alcanzar un determinado objetivo. Tales acciones deben ser modeladas según el método de modelado escogido. Por ejemplo: suponiendo que el evento sistémico comprar productos de la Tabla 2.2 se ha definido como un evento de software. Suponiendo también que ese mismo evento se ha compuesto de 4 funcionalidades (acciones): abrir recibo de compra, almacenar los ítems de compra (los productos), fechar recibo de compra y dividir la compra (la unión de esas actividades resultan el evento de software comprar productos). Se percibe entonces, que esas funcionalidades supuestas deberán ser modeladas utilizando alguna técnica, por ejemplo: el concepto de caso de uso, diagrama de actividades de UML, o un diagrama de flujo.

Modelar estructura de datos:

La estructura de datos define la configuración y la relación de los datos que serán manipulados por el software. El modelado de la estructura de datos puede ser hecho por medio de un diagrama de clases (orientado a objetos) o un diagrama de entidad relación, o ambos. Es importante relacionar el modelo funcional y el modelo estructural de datos en el desarrollo de la arquitectura.

Modelar interfases de acceso de los usuarios:

El modelo de interfase de acceso de usuario tiene como objetivo obtener el diseño de las pantallas (funcionalidades que constituye la entrada de información) y los reportes (funcionalidades que constituyen la salida de información) que serán desarrollados por el software.

Definir tamaño del software:

Después de realizar el modelo de interfase, el modelo de datos y el modelo funcional es posible aplicar técnicas para la definición del tamaño del software. En este modelo son consideradas técnicas de análisis de puntos de función, COCOMO y Use Case Point.

Cocomo: Éste método predice basado en ecuaciones matemáticas que permiten calcular el esfuerzo a partir de ciertas métricas, como el Análisis de Puntos de Función y las líneas de código fuente.

Use Case Point: Puede predecir a partir de las características de sus requisitos, expresados en los casos de uso.

Estimado el tamaño del software es posible prever costo, periodos y recursos con un mayor grado de certeza.

Definir el modelo de implantación:

Efectuadas las previsiones, citadas en el párrafo anterior, se hace necesario definir el modelo de implantación de software. Esta tarea utiliza el diagrama de despliegue y el de componentes de UML. Este guiará la configuración de las estaciones de trabajo que recibirán el software.

Interacción con el cliente:

Esta tarea posee gran semejanza a la interacción de la actividad de análisis, la diferencia está en los artefactos que serán presentados. La interacción en esta actividad presenta el modelo funcional, el modelo de datos, el modelo de interfase y el modelo de implantación al cliente. El cliente aprueba si esta de acuerdo con los modelos presentados.

Definir la composición de los componentes que integrarán el sistema:

Esta tarea ocurre solamente si los modelos presentados en la interacción fueran aprobados por el cliente. En la misma el diseñador de software diseña para cada funcionalidad los componentes de código que serán utilizados en la construcción del software. Es de extrema importancia que el diseñador tenga conocimiento sobre los componentes de código almacenados en la base de componentes de la factoría y reutilizar lo más que pueda.

Gestionar la actividad:

La factoría también debe gestionar la actividad de diseño en relación al tiempo, costo y recursos humanos.

La actividad diseñar proporciona dos salidas: Documento de arquitectura y orden de montaje para el software.

En la unidad de producción de componentes, la actividad diseñar posee solamente las tareas: modelar funcionalidades, modelar estructura de datos, modelar interfases y gestionar la actividad.

En esa unidad, en la tarea modelar funcionalidades se diseña y presenta la(s) funcionalidad(es) de un componente de código o de infraestructura. Modelar estructura de datos define cuales son los datos que serán manipulados por los componentes (infraestructura y código). Modelar interfase debe definir los procedimientos de acoplamiento entre componentes que se unen. La especificación de acoplamiento del componente en el sistema software también debe ser definida en esa tarea. La tarea de gestionar la actividad debe estimar el costo, tiempo y recursos utilizados para el desarrollo del componente.

2.6.2.3 Definición de las tareas de la actividad Construir.

La actividad construir está compuesta de las tareas implementar, montar y gestionar. En esta actividad, los programadores de la unidad de producción de software, de poseer la orden de montaje (proporcionada por la actividad Diseñar), integran componentes y dirigen el software ensamblado para la actividad de prueba.

En la unidad de producción de componentes, los programadores desarrollan componentes de código que fueron solicitados por los analistas y diseñadores. Para realizar esa tarea los programadores, deben seguir un documento de convención de código.

El flujo de desarrollo de la actividad construir, para la unidad de producción de software, posee como entrada el documento de arquitectura y la orden de montaje, ambas generadas por la actividad diseñar. La salida de la actividad construir (unidad de producción de software) está formada por una unidad de software.

La entrada de la actividad construir, para la unidad de producción de componentes, es caracterizada por el diseño de arquitectura de componente. La salida es un componente de código.

2.6.2.4 Definición de tareas de la actividad Probar.

El ambiente de prueba para una Factoría de Software está compuesto por dos acciones: verificar y validar.

La acción de validar corresponde con supervisar que la Factoría de Software está construyendo el producto correcto. Concluyéndose que, en el momento de la interacción con el cliente (actividades de analizar y diseñar), la factoría está validando el producto software.

La acción verificar dice si la factoría está construyendo el producto cierto. Esa acción debe ser desarrollada por medio de un proceso formal de prueba de software (conjunto de procedimientos utilizados para probar funciones, métodos, componentes), también puede ser a través de un laboratorio de prueba preparado para el efecto. La verificación supervisa el desarrollo del código.

La acción validar abarca las actividades analizar, diseñar e implantar. La acción verificar abarca solamente la actividad de prueba. En la actividad de prueba es necesario realizar las pruebas de componentes, las de software (responsable por la integración de los componentes que forma una unidad de software), las de integración de requisitos (verifica si la unidad de software desarrollada está de acuerdo a las funcionalidades y los eventos de software definidos en las actividades anteriores), y la prueba de aceptación.

La prueba de aceptación mezcla las acciones verificar y validar, la verificación es hecha en presencia del cliente, por medio de una interacción (validación).

La actividad de prueba está dividida en 4 tareas: diseñar caso de prueba, preparar los casos de prueba, ejecutar programas con los datos de prueba y comparar los resultados obtenidos. La actividad de prueba debe ser gestionada de forma semejante a la actividad diseñar. En esta actividad se emplean las metodologías de pruebas: caja negra y caja blanca.

Las unidades de software (derivadas de la integración de componentes) deben pasar la prueba de integración. Ese proceso incluye construir el software (uniendo componentes de código) y la prueba en cuanto a posibles problemas que surjan a partir de esa unión.

Después de la prueba de integración, la unidad de software debe ser probada en relación a la integración de requisitos, en ese momento es verificado si esa unidad da respuesta a los requisitos del usuario del sistema.

La actividad prueba presentada en este modelo responde a definiciones de las normas IEEE Std 829-1998 y IEEE Std 1008-1997.

2.6.2.5 Definición de tareas de la actividad Implantar

La actividad implantar posee las siguientes tareas: verificar las bases tecnológicas que están disponibles, configurar el software en las estaciones de trabajo, entrenar a los usuarios y gestionar la actividad.

Verificar las bases tecnológicas:

En esta tarea se utiliza el diagrama de implantación definido en la actividad diseñar, y se verifica que el cliente dispone de todos los equipamientos necesarios para que el software pueda entrar en operación, también puede basarse en determinados requerimientos no funcionales definidos previamente.

Configuración del software en las estaciones de trabajo:

La configuración del software en las estaciones de trabajo provee al cliente de la instalación del mismo.

Entrenar a los usuarios:

Esta tarea es de mucha importancia, en ella el cliente recibe todo el entrenamiento necesario en el software y se les proporcionan los manuales del mismo. La definición de esa tarea debe estar explícita en el contrato de elaboración del software.

Gestionar la actividad:

Esta tarea posee las mismas características que la gestión de las actividades analizar, diseñar, construir, y probar.

2.6.2.6 Definición de tareas de la actividad Revisar.

En esta actividad el software pasa periódicamente por optimizaciones. Optimización no es sinónimo de corrección sino de mejoría. Ejemplo de optimización: sustitución de un componente que está funcionando bien por uno nuevo que dará un mejor desempeño al software.

Cuando una nueva versión de un componente es almacenado en la base de componentes, el administrador de la base de componentes debe verificar cuales son los clientes que utilizan la versión anterior para comunicárselos. En caso de que el cliente solicite una sustitución de componente, el administrador debe solicitar a un diseñador o a un programador la sustitución de ese componente. En esa actividad solamente la tarea de sustituir es gestionada. Cuando un componente es sustituido la actividad de prueba debe ser disparada.

2.6.2.7 Definición de tareas de la actividad Almacenar Componentes.

Esta actividad posee tres actividades: documentar el componente, almacenarlo en la base de componentes y comunicar a los involucrados que el mismo fue almacenado. La solicitud de desarrollo o evolución de un componente constituye la información de entrada para esa actividad. La salida es la comunicación a los involucrados en el proceso, del desarrollo o evolución de un componente.

Documentar componente:

Esta tarea tiene como objetivo definir los datos de un componente de código o infraestructura. Esa descripción en detalle sirve de ayuda para que el administrador de la base de componentes la gestione con calidad. Por lo tanto un componente debe poseer los siguientes datos:

- Datos de creación del componentes, por ejemplo fecha y hora de creación, proyecto solicitante del componente, tiempo de desarrollo del componente.
- Datos sobre la seguridad del componente. Los permisos de acceso de los componentes deben ser almacenados, pues cuando el componente fue solicitado, el administrador de componentes debe tener conocimiento sobre ello y verificar si el solicitante tiene derecho de acceso al componente.
- Datos sobre la utilización del componente. Esta información quedando almacenada correctamente, libera al desarrollador de componentes de tener que responder cuestiones de los desarrolladores de software relacionadas al manejo del componente (por ejemplo, como acoplar el componente a uno u otro).
- Datos generales de un componente (código, nombre, palabra clave para la búsqueda).
- Tener el conocimiento si el componente x está siendo utilizado. Por ejemplo, el componente x no se utilizará debido a una evolución tecnológica.

Almacenarlo en la base de componentes y comunicar a los involucrados:

Cuando el componente esté documentado se pasa a su almacenamiento. Para realizar esta tarea el administrador de la base debe poseer una herramienta que le facilite el trabajo. Esta herramienta también debe comunicar a los involucrados en el desarrollo del almacenamiento o evolución del componente.

2.6.2.8 Definición de tareas de la actividad Distribuir.

La actividad distribuir componentes ocurre cuando el mercado externo desea adquirir un componente almacenado en la base, o las otras actividades en la factoría solicitan un componente de código o infraestructura. Esta actividad posee solamente una tarea, verificar si el componente responde a la solicitud del cliente.

2.6.2.9 Definición de tareas de la actividad Gestión de Proyecto

Un proyecto es un emprendimiento temporal con el objetivo de crear un producto o servicio único. Temporal significa que cada proyecto tiene un comienzo y un fin bien definidos. Único significa que el producto o servicio producido es de alguna forma diferente de todos los productos o servicios semejantes.

La actividad gestión de proyecto en este modelo posee 3 sub-actividades: planificar, ejecutar y controlar.

Sub-actividad Planificación:

Tiene como finalidad definir los objetivos del proyecto y seleccionar las mejores alternativas para alcanzar los objetivos definidos. Posee las siguientes tareas: definir el alcance del proyecto, definir las actividades que necesita el proyecto para cumplir sus objetivos, secuenciar las actividades, definir cronograma, recurso, costo y riesgos.

En la tarea definir el alcance, el jefe de proyecto debe definir las justificaciones para el desarrollo del proyecto, describir los productos y subproductos que serán generados. La tarea definir actividades tiene como objetivo identificar las actividades que deben ser realizadas para producir los diversos subproductos del proyecto. La identificación y documentación de las dependencias entre las actividades guiarán la tarea secuenciar actividades. La tarea definir cronograma tiene como objetivo crear el cronograma del proyecto a partir del análisis de la secuencia de actividades, sus duraciones, y las necesidades de recursos. La planificación de recursos (personas, equipamiento, preparación, materiales) es responsabilidad de la tarea definir recursos. La definición de costo tiene como objetivo estimar, aproximadamente, el costo global del proyecto. Y por último, la actividad de administración de riesgo identifica los riesgos eminentes al proyecto.

La información definida en la planificación puede ser utilizada en la definición del contrato de prestación de servicio entre la Factoría de Software y el cliente.

Sub-actividad Ejecutar:

En esta sub-actividad se realizan las actividades correspondientes al plan del proyecto, definidas en la sub-actividad de planificación. Es de extrema importancia que durante la ejecución del proyecto existan preocupaciones con la garantía de calidad (evaluar

regularmente el desempeño general del proyecto) y con el desarrollo del equipo (desarrollar nuevas habilidades en las personas involucradas en el proyecto).

Sub-actividad Controlar:

En esta sub-actividad se supervisa regularmente el proyecto, verificando su desempeño, y las desviaciones, en relación a la planificación, deben ser analizadas y corregidas. En la medida en que son identificados desvíos significativos (aquellos que colocan en riesgo el objetivo del proyecto), se realizan ajustes al plan a través de la repetición de la sub-actividad de planificación.

Es importante señalar que la actividad gestión de proyecto acompaña todo el proceso de desarrollo de la Factoría de Software.

2.6.2.10 Roles

En esta sección el modelo hace una propuesta de roles para las actividades del proceso de desarrollo. En la Tabla 2.3 se relaciona cada actividad del proceso con el(los) rol(es) que participa(n) en dicha actividad. Esta definición de roles es solo una propuesta, puede adaptarse a las necesidades y recursos de la factoría en la que se aplique el modelo.

Actividad	Rol(es)
Gestionar proyecto	Gestor de proyecto
Analizar	Analista
Diseñar	Diseñador
Construir	Programador
Probar	Probador
Implantar	Analista, diseñador
Revisar	Analista
Almacenar	Administrador de la base de componentes
Distribuir	Administrador de la base de componentes

Tabla 2.3 - Roles en el proceso de desarrollo

2.6.3 Herramientas, métodos y mecanismos para la automatización del proceso

Este modelo para la automatización del proceso propone un conjunto de herramientas, métodos o mecanismos tales como:

- Herramienta de gestión de proyecto de software: destinada al soporte a jefes de proyecto en el desarrollo de software, suministrando información para el control de los proyectos durante el proceso (recursos necesarios para su desarrollo), de la mano de obra (asignación de personal por proyecto), de los proveedores y de riesgos del proyecto.
- Método de gestión de proyecto, para un determinado proyecto de software: dar soporte a los ingenieros de software, analistas de sistemas, desarrolladores y arquitectos, para controlar la información inherente a la gestión de requisitos y a las actividades del proceso (dirección del proceso).
- Método o mecanismo para el almacenamiento de información de los componentes: Este método tiene como objetivo proveer un mecanismo para almacenar la información inherente a los componentes.
- Herramienta para modelado UML: provee utilidades para la especificación de diagramas de caso de uso, de clases, de transición de estados, de actividades, de secuencia, de colaboración y de componentes. Después de la especificación del software, con esta herramienta es posible generar el esqueleto de las clases y métodos para el desarrollo de los componentes de código.
- Herramientas para modelar: ofrece utilidades para especificación de diagramas de flujo de datos, diagrama entidad relación, IDEF0.
- Herramientas para el desarrollo de componentes de código.
- Método para metrificar software: tiene como objetivo, automatizar el cálculo del tamaño del software a ser desarrollado por el proyecto. Definiendo el tamaño es posible calcular el costo y estipular el periodo de desarrollo. Este método trabaja en conjunto con la herramienta de administración de proyecto y es utilizado después de la definición de la arquitectura del software.
- Herramienta para el control de versión y gestión de la configuración: esta herramienta tiene como objetivo proveer un control automatizado de las versiones del software y los componentes. La gestión de la configuración también debe ser considerada en esta herramienta.

2.7 Análisis comparativo de los modelos

La selección de los modelos descritos anteriormente para ser analizados y llegar a la propuesta del Modelo funcional de la Factoría de Software de la UCI, responde a que se ha encontrado en los mismos un grupo de características fundamentales que debe tener una

Factoría según las definiciones de Factoría de Software analizadas en la bibliografía consultada.

El Modelo basado en la norma ISO 9001 y CMM, es un modelo que abarca un grupo de elementos esenciales en el desarrollo de una estructura fabril para software, y se puede adaptar a las características y necesidades de una Factoría. El modelo también posee una característica que está presente en la mayoría de las definiciones de Factorías de Software analizadas en este trabajo, la necesidad de tener un repositorio de componentes reutilizables para apoyar el desarrollo basado en componentes. Dentro de los elementos más importantes que influyeron en su selección se encuentra, que el mismo está basado en ISO 9001 y CMM que son estándares de calidad reconocidos y muy utilizados a nivel mundial. CMM destinado a la evaluación y mejora de procesos, ISO 9001 ayuda a mejorar los aspectos organizativos de una empresa. Otra característica importante de este modelo es la utilización de PSP y TSP en la entidad actores involucrados en el proceso, PSP orientado al proceso personal y TSP al trabajo en equipo. Además define las entidades que forman el modelo Factoría de Software y las relaciones entre ellas.

En cuanto al Modelo Eureka la importancia radica en que es utilizado por un conjunto de empresas Europeas, teniendo en cuenta que Europa presenta un alto nivel en el desarrollo de software. También presenta características que están presentes en la mayoría de las definiciones de Factoría de Software como son: la utilización de herramientas para la automatización del proceso de desarrollo, desarrollo basado en componentes. El modelo da una visión de cómo se puede desarrollar un producto en partes manejables y después la unión de estas para formar el producto final.

El Modelo clasificatorio constituye un modelo a tener en cuenta ya que el mismo realiza una clasificación de las Factorías de Software de acuerdo a las actividades que realizan durante el proceso de desarrollo de software. Ayuda a identificar de qué tipo es la Factoría de Software de la UCI y hacia donde se puede ir avanzando en este enfoque, ya que en un futuro se pudiera pasar a una factoría de mayor o menor alcance.

El Modelo Propuesto por Basili presenta esa característica importante que es el desarrollo basado en componentes, y la reutilización durante el proceso de desarrollo.

En cuanto al Modelo replicable, la importancia del mismo radica en que incluye los principales elementos de los demás modelos abordados.

2.8 Conclusiones

Después de analizar los modelos presentados y haciendo una comparación con el modelo fabril convencional se puede llegar a la conclusión de que la mayoría de los modelos acogen el concepto de línea de producción y montaje del proceso fabril tradicional y lo vinculan al desarrollo de software. Esto se ve en el proceso de producción dividido en actividades (división del trabajo) y en la integración de componentes para la construcción del software (intercambio de piezas), lo que indica un desarrollo basado en componentes, soportado por un alto porcentaje de reutilización. Se puede concluir además que el Modelo replicable y el Modelo basado en la norma ISO 9001 y CMM constituyen los principales a tener en cuenta para la obtención del Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour.

Capítulo 3 - Situación Actual de la Factoría de Software de la UCI en la Línea Carrefour

3.1 Introducción

En este capítulo se describe como se dieron los primeros pasos en la formación del personal de la Factoría de Software y el proceso de desarrollo de un proyecto en la misma. Los principales problemas detectados y la situación problémica. Basado fundamentalmente en los proyectos que se han realizado en la factoría. Teniendo como objetivo principal detectar todas las fallas existentes en el proceso de desarrollo con el fin de obtener soluciones viables.

3.2 Formación brindada por cliente.

La empresa Matchmind brinda entre otros servicios el de factoría de software, la cual esta estandarizada con nivel de CMM 3, para dar respuestas a las demandas ha abierto recientemente nuevas instalaciones en Ávila para ampliar la capacidad de su servicio de factoría de software. Pero estos no son suficientes para satisfacer los diferentes usuarios. Por lo que el cliente solicita el servicio de factoría de software a la empresa Centersoft con la cual mantenía relaciones en entorno Cobol. Esta a su vez solicitó la inserción de la empresa Desoft y la UCI con el objetivo de aunar fuerzas que respondieran a los intereses de Matchmind. Para capacitar el personal que asumiría los proyectos de la Factoría un especialista del área de formación de la empresa impartió un curso con un tiempo de duración de cuatro semanas sobre las tecnologías de desarrollo en dos frameworks. Una de esas tecnologías se nombra Carrefour (C4J), framework personalizado con el objetivo de la automatización de una cadena francesa de supermercados y la otra Banksphere, framework desarrollado por la cadena bancaria Banesto, el cual sigue el objetivo de la automatización de su red de bancos, ambas son basadas en la plataforma J2EE.

Toda la documentación y tele conferencias del curso de formación se encuentra digitalizados y con acceso público, brindando la posibilidad de contar con una bibliografía ampliada para la formación de nuevos recursos humanos y que todo el personal interesado pueda acceder a medida de sus necesidades.

El cliente con vistas a evaluar la preparación del personal, solicitó la realización de un proyecto de prueba para seis personas con el fin demostrar la capacidad del recurso

humano para asumir tareas de igual o mayor envergadura. El prototipo fue enviado vía correo electrónico, el cual estaba compuesto de dos documentos: uno sobre el análisis técnico y otro sobre el análisis funcional de la aplicación. A partir de estos documentos se comenzó a planificar el proceso productivo de la aplicación.

3.3 Información que se maneja

Los documentos que se manejan son el análisis técnico y el análisis funcional de la aplicación. A partir de estos documentos se comienza a planificar el proceso productivo de la aplicación.

- El análisis técnico consiste en la descripción de los requisitos técnicos de la aplicación: "Cuadro de Mando de flujo tenso". El documento describe el modelo de datos creado para la carga de los datos recibidos desde las interfaces de fichero enviados por sistemas externos, los procedimientos de lectura de los ficheros y relleno de las tablas del modelo de datos, y por ultimo la lectura e interpretación de estos por la interfaz Web. Este documento va dirigido al equipo técnico encargado del desarrollo de la aplicación.
- El análisis funcional es usado para la definición de indicadores que reflejan el funcionamiento de la aplicación "Cuadro de Mando de flujo tenso". Mediante estos se centraliza toda la información de forma que sea posible hacer comparativas a todos los niveles con otras plataformas, sectores, secciones, etcétera.

3.4 Objeto de Estudio

El proyecto de prueba consistió en el desarrollo del aplicativo Flujo Tenso de acuerdo a las especificaciones contenidas en los documentos Análisis Técnico Cuadro Mando FLT v1.3 y el documento Análisis Funcional Cuadro de Mando Flujo Tenso. Este es un aplicativo típico de desarrollo con el framework Carrefour.

Para el desarrollo del mismo, el equipo de trabajo fue formado por compañeros a partir de sus resultados obtenidos en una prueba elaborada, y por la disponibilidad de estos en otros proyectos estratégicos de la universidad. El equipo lo formaron seis integrantes, cuota establecida por el cliente y no evaluando por la parte del equipo en la UCI, la viabilidad de la decisión. Quedando constituido por dos especialistas de Desoft, tres profesores y un estudiante de la UCI en busca de una variedad en la formación del personal. Este equipo se concentro en la UCI creándose las condiciones que demandaba el cliente en cuanto a recurso de maquinas y vía de comunicación.

En las primeras reuniones se establecieron las actividades a tener en cuenta para el desarrollo del proyecto. Se elaboró el cronograma representado que técnicamente se creyó que se debían ejecutar para llevar a cabo el proceso de desarrollo del aplicativo.

Para la estimación y duración de las mismas se puso como objetivo llevar a cabo el proyecto en el tiempo que propuso el cliente sin tener establecidos indicadores de estimación por parte del equipo de desarrollo. El plazo fue de cuatro semanas, las cuales fueron estimadas por el cliente, sin existir ninguna propuesta por parte del equipo. A las actividades se le dieron prioridades por el nivel de complejidad y se estimo sin tener en cuenta ningún indicador.

Las comunicaciones con el cliente se establecieron vía correo y por chat en dependencia de las necesidades de información de ambas partes. Los integrantes del equipo de desarrollo además de poder comunicarse por las vías antes mencionadas tenían contactos periódicos de intercambio de criterios y definiciones, los cuales no eran programados ni organizados, surgían sobre la marcha y en dependencia de la necesidad.

En la primera reunión para organizar el proyecto se identificaron roles, responsabilidades y relaciones de reporte en el proyecto. Se eligieron dos desarrolladores con el fin de elaborar las páginas Web correspondientes a la presentación, dos desarrolladores para trabajar en la configuración de ficheros XML de configuración del framework y dos responsables de la lógica de negocio, el acceso a datos y modelación de la aplicación. No se definió claramente un responsable del diseño de la base de datos pues esta estaba especificada en el documento referente al análisis técnico. Responsabilidades pertenecientes a la administración de redes en los temas: seguridad de la información, configuración SGBD, CVS, instalación de las herramientas, configuración de framework, conexiones de redes, entre otras fueron correspondidas de manera aleatoria en dependencia de la prioridad de la tarea en ese instante de tiempo.

En cuanto a la calidad se decidió en este punto que la aplicación debía seguir los estándares de calidad que especifica el framework Carrefour.

Al estar establecidas todas las actividades de planificación, el equipo pasa a configurar el Framework y el gestor de base de datos para comenzar el desarrollo. Dado que no existía ninguna experiencia en el desarrollo de este tipo de aplicaciones fue necesario montar un proyecto piloto de ejemplo que se había dejado junto con la documentación en la etapa de capacitación, para que el equipo se guiara por el mismo durante la producción. El mismo no se pudo poner a funcionar ya que estaba incompleto, faltaban librerías entre otros componentes propios del framework. Además era necesario hacer un grupo de

configuraciones las cuales no se conocían pues durante el curso no se llegó a realizar un ejemplo práctico que cumpliera con estas necesidades. Dado estos problemas surgidos fue necesario que un especialista en el framework asesorara al equipo en la instalación y configuración del mismo, no quedando registrados los cambios en ningún documento oficial. Al quedar configurada la plataforma de trabajo se pudo comenzar la construcción del aplicativo. Durante el desarrollo hubo inestabilidad dado que no había un flujo de procesos definido, y los roles no estaban definidos claramente, los desarrolladores no tenían claro que hacer en cada momento ni en que orden ejecutar las tareas. En ocasiones se perdía tiempo realizando una actividad, mientras existían otras con mayor prioridad de ejecución. Finalmente a pesar de los inconvenientes surgidos se logró cumplir con el proyecto orientado por el cliente pero no en el tiempo establecido, ni con la mejor calidad.

3.5 Situación Problémica

La definición de roles y responsabilidades que se realizaron no responden al enfoque de trabajo en factoría, afectándose la eficiencia, la calidad, y el tiempo de desarrollo de un producto. Esto empeorará con el aumento de la fuerza de trabajo y de la demanda del cliente.

Al no existir un flujo de procesos definido el desarrollador se siente desorientado y no sabe que hacer en cada momento ni a quien dirigirse, llevando a la desorganización de la producción afectando la productividad.

La planificación del trabajo tanto personal como a nivel de equipo no es la mejor, no se sigue estándares establecidos en la Ingeniería de Software, afectándose la efectividad del equipo de desarrollo.

El insuficiente dominio de las herramientas de trabajo provoca que la gestión de las dudas con los clientes sea un tema crítico e imprescindible. El proceso de identificación de las dudas es lento. No hay un control estricto de las mismas emitidas al cliente, de la fecha tanto de envío como de recepción de las respuestas. La demora en responder a las dudas aumenta el tiempo de entrega del producto al cliente, y a la hora de acordar los plazos no se tiene en cuenta el tiempo de intercambio de dudas.

Debemos tener una estrategia de estimación del tiempo de entrega y el costo de un trabajo determinado basado en el conocimiento real de la capacidad productiva, partiendo del principio de que el proceso de revisión es crítico e imprescindible y que permita poder dar tiempo al trámite de las dudas. Logrando así tener un ambiente que nos permita la entrega en tiempo y con calidad. Así como una gestión de cambio.

Los componentes realizados no se han almacenado en un repositorio donde se encuentren clasificados y documentados, esto dificulta su reutilización en futuros proyectos.

3.6 Problema

Luego de la observación e investigación del proceso actual de desarrollo de los proyectos de la factoría de software de la UCI, se han detectado los siguientes problemas:

1. No existe un orden lógico de procesos definidos desde que llega un proyecto hasta que se entrega.
2. No hay definición de roles organizados estructuralmente.
3. No existe una buena planificación del trabajo personal así como de equipo. Tanto en la producción, en la formación, coma la investigación para la búsqueda de mejoras en los procesos de desarrollo de software en la factoría.
4. No existe una buena organización de la producción.
5. No se logra estabilidad en el trabajo.
6. Insuficiente dominio de la herramienta de trabajo.
7. No existen métricas o mecanismos para la estimación de costos y plazos para el desarrollo de un producto.
8. No existe gestión de la comunicación y cambio.
9. No se cuenta con un repositorio de componentes reutilizables.

3.7 Problema Científico

La no existencia de un Modelo funcional en la Factoría de Software de la UCI para la línea Carrefour es uno de los factores que afecta la eficiencia de los procesos de desarrollo de software en la factoría en cuanto a calidad, tiempo y costo.

3.8 Hipótesis

Si se logra obtener y aplicar un modelo de Factoría de Software para la línea Carrefour que responda a las características de nuestro entorno basado en las mejores prácticas aplicadas en empresas de marcada experiencia a nivel internacional entonces posibilitará el aumento de la calidad en los procesos realizados y en el producto obtenido y disminuirá el tiempo y costo de desarrollo.

3.9 Objetivos

Objetivo general:

- Obtener el Modelo funcional de la Factoría de Software de la Universidad de las Ciencias Informáticas para la línea Carrefour.

Objetivos específicos:

- Realizar un estudio de las tendencias y tecnologías actuales del campo de la Informática.
- Realizar un estudio de los modelos de factorías de software existentes.
- Identificar elementos esenciales que definen un modelo de factoría de software.
- Identificar las deficiencias existentes en el modelo actual y sus vías de solución.
- Definir las entidades del modelo para la Factoría de Software de la UCI.
- Especificar las entidades a la línea Carrefour.
- Establecer mecanismos de control de la Calidad que permitan valorar el producto a través de su desarrollo en la Factoría de Software de la UCI.

3.10 Conclusiones

En el presente capitulo se describió desde los inicios hasta la actualidad el desarrollo de la factoría de software en la línea Carrefour. Se pudo comprobar que la formación impartida por el cliente sobre el framework Carrefour no cumplió con todo lo necesario para empezar la producción en esta línea de desarrollo en la factoría, la práctica lo demostró, con los problemas presentados durante el comienzo del aplicativo de prueba orientado. Pero no solo esto influye negativamente en el desempeño productivo de la factoría, se han identificado un grupo de problemas existentes que se le deben dar solución para poder funcionar como una verdadera factoría de software. Dentro de los problemas más críticos que este trabajo pretende dar solución se encuentran la no existencia de procesos y roles definidos adaptados a la línea de desarrollo Carrefour que guíen a los desarrolladores durante la producción y permitan la especialización de cada uno en una tarea específica del proceso.

Capítulo 4 - Modelo Funcional de la Factoría de Software de la UCI para la Línea Carrefour.

4.1 Introducción

Según la situación problémica planteada anteriormente y en base al análisis hecho de los modelos presentados en capítulos anteriores, se propone el Modelo Funcional de la Factoría de Software de la UCI para la línea Carrefour, con el fin de darle solución a las deficiencias encontradas. Para esto se tomaron los aspectos positivos de los modelos anteriores y se adaptaron al contexto de la factoría en esa línea de aplicaciones específicamente. Buscando resolver el problema de funcionamiento debido a que es donde más carga de trabajo se tiene en estos momentos.

Inicialmente en el capítulo se identifican los elementos fundamentales del modelo, hasta llegar entidades que lo conforman. Luego se define cada entidad y se refleja en cada una de ellas, las cuestiones que pueden ser aplicadas a otras líneas de producción de la factoría.

4.2 Identificación de elementos del modelo

Hoy día la factoría de software de la UCI se dedica a la producción de partes manejables de grandes software empresariales; que la empresa MatchMind es responsable ante el mercado.

La factoría de la UCI no tiene el enfoque sistémico del producto, solo se dedica a la construcción de subsistemas cuyas especificaciones provienen de Matchmind, la cual luego de obtener los entregables ensambla el producto final. Por lo que el proceso actual de desarrollo que sigue la factoría abarca las fases de desarrollo que más recursos humanos necesitan. Lo cual trae consigo que este para su correcto funcionamiento debe estar enmarcado en los siguientes aspectos:

- Organización de la producción, definición y especificación de las entidades por las que está compuesta la Factoría.
- Reutilización de componentes de código o componentes de infraestructura (activos del proceso). Los cuales son almacenados; gestionados mediante técnicas definidas para la clasificación y recuperación, y herramientas que automatizan esta tarea.
- Herramientas que automatizan el proceso de desarrollo en la línea Carrefour (de modelado, de programación, de gestión de configuración, entre otras).

- Estándares de calidad junto a técnicas y mecanismos que deben apoyar el proceso de desarrollo y la organización de los implicados.
- Roles que ejecuten y sean guiados por el proceso de desarrollo.
- Lograr la especialización en cada role definido.
- Producción de componentes de alto nivel o subsistemas, que siga un proceso de desarrollo estandarizado.

Este grupo de aspectos claves mencionados anteriormente quedan agrupados en cuatro entidades y sus relaciones, para conformar el modelo funcional anteriormente mencionado:

1. Técnicas y Herramientas: Comprende el contexto de las tecnologías, herramientas para dar soporte y automatización al proceso de desarrollo.
2. Proceso de desarrollo: Comprende el conjunto de actividades que conforman el flujo de trabajo, el cual consta de: recepción de especificaciones, diseño, implementación, prueba, almacenamiento y entrega.
3. Participantes: Comprende el recurso humano involucrado con el proceso de desarrollo de software, la estructura organizativa y los roles que ocupan, está dividida en dos sub-entidades: Gestores de la Factoría y Grupo de desarrollo.
4. Repositorio de componentes: Activos del proceso y componentes de código. Entiéndase como activos del proceso formularios, documentos, patrones, algoritmos utilizados como artefactos en el proceso. Los activos del proceso también pueden ser denominados como componentes de infraestructura, componentes de valor en el proceso.

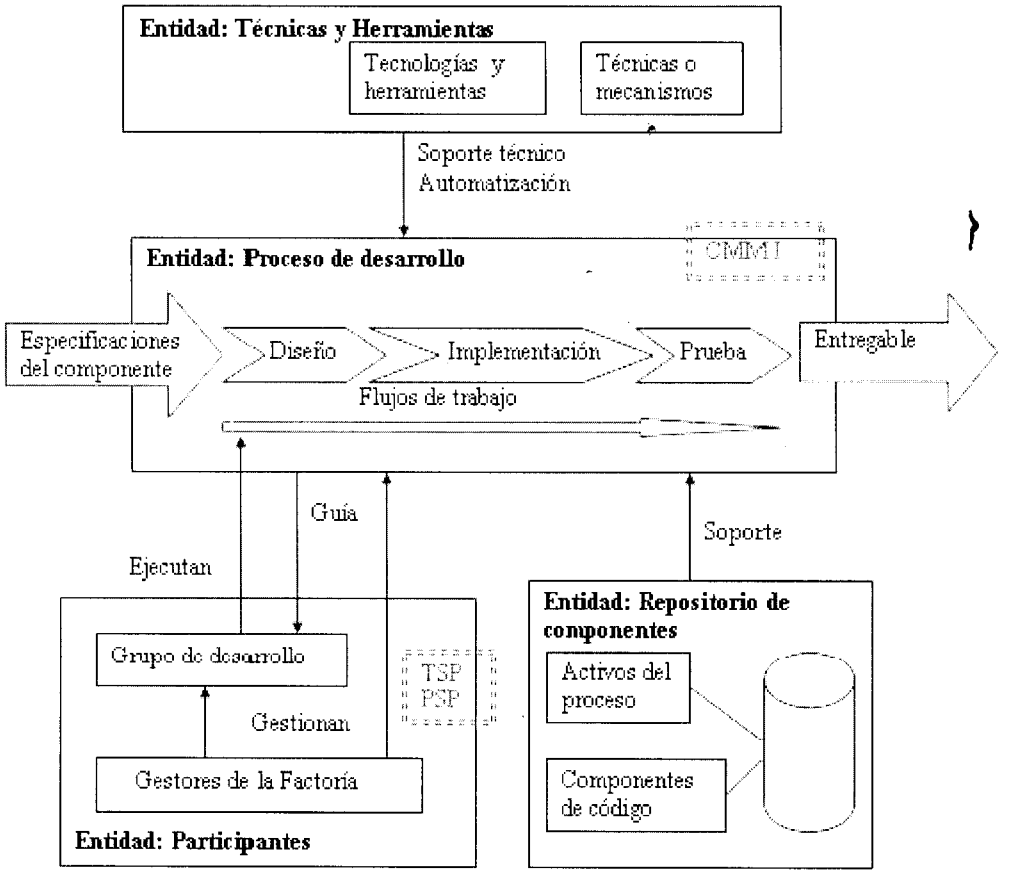


Figura 4.1 - Modelo propuesta de Factoría de Software para la Línea C4J.

En la figura 4.1 se observa la estructura y composición del modelo funcional propuesto por este trabajo. El resultado final de un proyecto en la factoría software es un producto, en este caso un subsistema o componente de alto nivel, que toma forma durante su desarrollo gracias a la intervención de variados tipos de personas, estas personas están representadas en el modelo mediante la sub-entidad Grupo de desarrollo y Gestores de la Factoría. El equipo de desarrollo lo forman las personas involucradas directamente en el proceso, las cuales son quienes ejecutan las actividades o flujos de trabajo (recepción de especificaciones, diseño, implementación, prueba, almacenamiento y destino), a su vez este equipo es guiado por el proceso de desarrollo de software, representado en el modelo mediante la entidad Proceso de desarrollo.

La sub-entidad Gestores de la Factoría comprende el equipo de dirección de la misma, encargados del control y gestión del grupo de desarrollo y del proceso. El proceso es automatizado y soportado por diversas tecnologías y herramientas, representados en la entidad "Técnicas" en el modelo. La reutilización tiene efectos muy positivos en el desarrollo

de software, entre estos efectos están el aumento en la productividad y calidad así como la reducción del tiempo de desarrollo, para dar soporte al proceso en este sentido la factoría cuenta con una base de componentes reutilizables, representada en la entidad "Repositorio de Componentes".

El proceso es regulado por normas de calidad dando los primeros pasos, hacia CMMI ya que éste es un modelo de calidad integrado para la industria del software que provee detalles importantes para la evaluación del proceso de desarrollo y la gestión de proyectos.

El trabajo del grupo de desarrollo es entregar productos software de alta calidad en un plazo determinado. Hay así, tres aspectos que hacen efectivo el trabajo de ingeniero de software: obtener productos de alta calidad, hacer el trabajo a los costes esperados y completar el trabajo de acuerdo con la planificación establecida. Para hacer un trabajo efectivo el desarrollador de software necesita:

1. Planificar su trabajo.
2. Hacer su trabajo de acuerdo con el plan.
3. Esforzarse en producir productos de máxima calidad.

El modelo propone que la entidad Participantes haga uso de PSP (Personal Software Process) y TSP (Team Software Process).

PSP muestra como aplicar métodos avanzados de ingeniería a las tareas diarias de cada individuo, proporciona métodos detallados de planificación y estimación, muestra a los implicados en el proceso como controlar su rendimiento y explica como los procesos definidos guían su trabajo.

TSP es un modelo o proceso de trabajo en equipo enfocado a aminorar varios de los problemas, tanto técnicos como administrativos, que se presentan en el desarrollo de software. El TSP provee un esquema de trabajo donde cada desarrollador tiene perfectamente definido sus roles, sus actividades, y sus responsabilidades. Asimismo, el TSP incluye procedimientos para la mejora continua del proceso de desarrollo, para mejorar la calidad del software producido, para mejorar la estimación del tiempo de desarrollo, para la disminución de defectos en el producto y para promover la integración del equipo de desarrollo. Es decir, el TSP apoya tanto al equipo de desarrollo como a los administradores del proyecto para la culminación a tiempo y dentro del presupuesto de los proyectos de desarrollo de software. Basado en el enfoque planteado de TSP es que se desarrolla este trabajo, en cada una de sus entidades, por lo que no se exponen instrumentos específicos de este tipo de modelo.

4.3 Repositorio de componentes reutilizables

La reutilización de componentes de software es un enfoque de desarrollo que trata de maximizar el uso recurrente de componentes existentes, en las distintas etapas del desarrollo. En el enfoque de Factorías de software la reutilización juega un papel fundamental en el proceso de desarrollo, dada las ventajas que trae consigo:

- Aumento de la productividad.
- Incremento de la calidad (Cuanto más se reutiliza un componente menor es la probabilidad de encontrar errores en el).
- Disminución del tiempo de entrega.
- La confiabilidad de un sistema se ve incrementada.
- El riesgo es reducido.

Constituye un requerimiento poder encontrar componentes para la reutilización apropiados en una base o repositorio de componentes, el repositorio constituye el almacén de componentes reutilizables de la Factoría y debe ser mantenido y gestionado constantemente.

El repositorio de componentes reutilizables puede contener dos grandes grupos:

- Componentes de código.
- Activos del proceso.

Los componentes de código pueden ser:

- Una clase
- Un procedimiento o función
- Un módulo
- Un subsistema
- Una aplicación

Los activos del proceso pueden ser:

- Patrones de diseño
- Algoritmos
- Un esquema de base de datos
- Manuales y documentación.
- Un modelo

Para poder efectuar la reutilización no basta con que los componentes se encuentren en el repositorio, además deben estar bien clasificados, documentados y fáciles de comprender para poder encontrar uno de acuerdo a necesidades específicas.

Para reducir el costo de encontrar los componentes adecuados en el repositorio existen las técnicas de clasificación y recuperación de componentes en un repositorio, las cuales son llevadas a un sistema que automatice los procesos.

La búsqueda y selección puede ser hecha mediante la visualización de una jerarquía de componentes en la biblioteca, lo que hemos dado en llamar hojear; o siguiendo un conjunto de enlaces hipertexto (o hipermedia), o por recuperación lineal.

- Hojear (Browsing):

  Por lo general, el proceso de hojear comienza con el conjunto de candidatos recuperados a partir de la consulta del usuario, permitiendo al usuario acceder al resto de la existencia a medida que se lo ordena al visualizador, los cuales se exponen en cantidades elegidas por el mismo usuario. Solo es recomendable en repositorios pequeños.

  Existen propuestas de mecanismos basados únicamente en visualización rápida [Goldb84], la mayor parte sobre la base de niveles de jerarquía. La principal desventaja de estas técnicas es que requieren que sea el propio usuario el encargado de manejar el proceso de navegación, lo cual puede resultar confuso y difícil en grandes repositorios.

- Hipertexto

  Son varias las propuestas para recuperación sobre la base de la tecnología hipertexto [Cybul93]. En tales sistemas la información se organiza como una red de nodos - unidades de información- que se interconectan por medio de enlaces y relaciones. El usuario puede navegar por la red siguiendo los enlaces preestablecidos, y guiándose en este proceso por la semántica de cada enlace [Savoy93].

  El principal problema de esta técnica es que el desarrollo y mantenimiento de un repositorio en un ambiente hipertexto requiere de una inversión muy grande en recursos humanos.

  Los enlaces podrían pueden ser generados automáticamente a partir de información adicional que se introduce en la descripción del componente, por ejemplo: el atributo *usa a* <nombre de componente> permite establecer un enlace entre dos componentes (el que *usa* y el *usado*) [Allan95]; el problema ahora consiste en que hay que añadir a las descripciones de los componentes nuevas relaciones, que si no están bien diseñadas ocasionarán la "generación automática de una red inútil".

- Recuperación Lineal

  En el campo de la investigación existen técnicas que utilizan la recuperación lineal como su principal mecanismo, es decir, recuperar un conjunto de componentes potencialmente reutilizables a partir de una especificación del usuario sobre lo que se desea. En el campo de la investigación existen varias técnicas o mecanismo basados en temas de la inteligencia artificial, estos son muy interesantes pues se acercan bastante a la técnica de "clasificación/recuperación" que siempre nos proporcione el mejor componente.

  En la factoría aun hay muy pocos elementos que reutilizar, el método *Hojear* resuelve el problema y es sencillo de implementar. A medida que se vayan elaborando nuevos elementos hay que elaborar un estudio para mejorar el sistema continuamente; buscando una filosofía donde el mantenimiento a la mejora no sea engorroso.

4.4 Participantes

Hay personas implicadas durante todo el proceso de desarrollo de software. La Factoría presenta una estructura organizativa donde cada persona implicada ocupa un rol determinado dentro de la misma en dependencia de sus conocimientos y capacidades. Las personas constituyen un factor importante en el éxito de un proyecto de software por lo que su organización es fundamental.

4.4.1 Grupo de desarrollo.

En la Factoría las personas involucradas directamente a la producción deben estar organizadas en equipos de trabajo para lograr la potenciación de las habilidades individuales en función de lograr un objetivo común. Esto significa esfuerzos integrados por encima de proyectos individuales, logrando mejorar las operaciones relativas al software.

4.4.1.1 Organización del grupo de desarrollo

Existen tantas estructuras de organización del personal para el desarrollo de software como organizaciones se dedican a ello; por lo que lograr la estructura ideal es algo complicado, no todo grupo es equipo y no todo equipo es eficiente. Hay que tener en cuenta varios factores que influyen tanto positiva como negativamente en la estructura a aplicar. La mejor estructura de equipo depende de las características de la Factoría, el número de personas que compondrá el equipo, la preparación que posean sus integrantes y la dificultad de las tareas asignadas al mismo.

Pressman (2002) hace un análisis de varios organigramas para equipos de desarrollo de software, estos son: Descentralizado Democrático (DD), Descentralizado Controlado (DC) y Centralizado Controlado (CC).

Descentralizado Democrático (DD): Este tipo de equipo no posee un jefe permanente. Se nombran coordinadores de tareas a corto plazo y se sustituyen por otros para diferentes tareas. Las decisiones sobre problemas y los enfoques se hacen por consenso del grupo. La comunicación entre los miembros del equipo es horizontal.

Descentralizado controlado (DC): Este tipo de equipo tiene un jefe definido que coordina tareas específicas y jefes secundarios que tienen responsabilidades sobre subtareas. La resolución de problemas sigue siendo una actividad del grupo, pero la implementación de la solución se reparte entre subgrupos por el jefe de equipo. Aunque la comunicación dentro de los subgrupos es básicamente horizontal, también hay comunicación vertical a lo largo de la jerarquía de control.

Centralizado Controlado (CC): Tienen un jefe de equipo que se encarga de la resolución de problemas de alto nivel y la coordinación interna del equipo, mientras la comunicación entre el jefe y los miembros del grupo es vertical.

Cuando se planifica el organigrama de equipos en el desarrollo de software deben considerarse algunos factores que influyen significativamente, estos son:

- La dificultad del problema a resolver.
- El tamaño del programa(o de los programas) resultante(s).
- El tiempo de vida del equipo.
- El grado en que el problema puede ser modularizado.
- La calidad requerida y fiabilidad del sistema que se desea construir.
- Grado de comunicación requerida.

Analizando estos factores y teniendo en cuenta que la Factoría a la que se le define el modelo funcional esta ligada al desarrollo de aplicaciones empresariales en la plataforma J2EE, puede identificarse la influencia de los mismos en la Factoría. El desarrollo de componentes o aplicaciones empresariales generalmente es un problema complejo por las características de este tipo de aplicación. La Factoría trabaja la plataforma J2EE con el Framework C4J, este es un framemework Modelo Vista Controlador, lo que permite la división del trabajo por capaz y en componentes desarrollables. Las aplicaciones empresariales requieren de una calidad y seguridad alta debido al tipo de información que manipulan y los servicios que brindan. La Factoría lleva poco tiempo de creada por lo que

no tiene la experiencia suficiente en el desarrollo de este tipo de aplicaciones, la cual requiere de una buena comunicación durante el desarrollo del producto software.

La estructura centralizada puede realizar las tareas más rápidamente, pero debe ser aplicada a la solución de problemas sencillos en los cuales el equipo domina el problema a resolver, las tecnologías y herramientas a utilizar. Teniendo en cuenta el estado en que se encuentra la Factoría esta no sería la estructura mas adecuada para aplicar pues no existe una basta experiencia en el uso de las tecnologías y herramientas con que se trabajan, en un futuro no muy lejano cuando el equipo esté un poco más familiarizado con las mismas puede optarse por aplicar esta estructura de equipo en el desarrollo.

Los grupos descentralizados generan más y mejores soluciones que los individuales, por lo que estos equipos tienen más probabilidades de éxito en la resolución de problemas complejos.

Partiendo de que el rendimiento de un equipo es inversamente proporcional a la cantidad de comunicación que se debe entablar se puede plantear que los proyectos muy grandes son guiados mejor por grupos con estructura CC o DC, donde se puedan formar subgrupos fácilmente.

Cuando la modularidad es relativamente baja se aplica mejor el organigrama DD y cuando es posible una modularidad alta funcionan bien los organigramas CC o DC, además de que producen menos defectos que los equipos DD.

Hay que tener en cuenta que los equipos descentralizados requieren más tiempo para completar un proyecto que los centralizados y al mismo tiempo son mejores cuando se precisa una gran cantidad de comunicación.

Considerando los diferentes organigramas de equipos abordados anteriormente y teniendo en cuenta que el modelo propuesto es para la línea de aplicaciones C4J, se propone la creación de equipos siguiendo el organigrama Descentralizado Controlado (DC).

Su desarrollo requiere de múltiples conocimientos, por lo que se hace necesario reunir a personas con los conocimientos y habilidades adecuadas y organizarlas en equipos de manera que puedan enfrentar los retos del desarrollo de este tipo de aplicaciones con éxito.

Al conformar la estructura de los equipos de desarrollo debe lograrse la integración de las personas en grupos pequeños, esto posibilita un mejor desempeño del grupo de trabajo y aumenta la cohesión del mismo.

Un grupo pequeño debe poseer de 8 a 10 personas para logra una buena interacción entre los integrantes y que cada persona pueda comunicarse con los demás cara a cara. Cuando los integrantes empiezan a comunicarse abiertamente entre si, esto da lugar a mayor

confianza e interacción dentro del grupo, también las discusiones comienzan a centrarse más específicamente en las tareas de resolución de problemas y en el desarrollo de estrategias para cumplir las tareas.

Teniendo en cuenta que la factoría puede trabajar en varias líneas de producción a la vez (Línea de desarrollo de aplicaciones empresariales con el Framework J2EE C4J y otras en un futuro), por cada línea de desarrollo existirán varios equipos de desarrollo en dependencia de la carga de trabajo que posea la Factoría en esa línea, y cada equipo estará compuesto por un grupo de diseño, un grupo de implementación y uno de prueba. Lo que en además de la organización exigirá de estilos de dirección.

4.4.2 Estilo de dirección

Un estilo de dirección es la manera en que los gerentes interactúan con las personas bajo su mando. Existen tres estilos básicos que figuran en la bibliografía más usual de gestión de proyectos: el autocrático, el permisivo y el democrático.

Para hacer un análisis de estos estilos y obtener sus diferencias, ventajas y desventajas se analizan en función de los flujos de información.

1. Estilo autocrático: En este estilo los gestores de proyecto no se preocupan por procesar la información proveniente de afuera ni la retroalimentación que viene del equipo. En ocasiones sostienen una política de puertas abiertas, pero solo en la medida en que buscan la información del exterior, porque después no la tienen en cuenta, son autocráticos.

   Ventajas: Este estilo es adecuado para proyectos de bajo riesgo, en los que el equipo se limita a llevar el plan como fue establecido, en este caso la retroalimentación no es tan decisiva como en un proyecto de alto riesgo. Puede resultar eficaz cuando es preciso tomar decisiones rápidamente.

   Desventajas: Puede llevar a la desmoralización del equipo, pues los desarrolladores se ven excluidos al no poder contribuir a la toma de decisiones. Puede llevar a tomar decisiones erróneas, producto de que el jefe no obtiene suficiente información del exterior.
2. Estilo permisivo: En este estilo la toma de decisiones es muy difusa, se basa en "dejar hacer". Existe un escaso flujo de información o un flujo aleatorio, que no se canaliza adecuadamente. En un estilo permisivo el equipo de proyecto debe ser capaz de realimentar al equipo de gestión pero los directivos no suelen obrar adecuadamente con respecto a esta realimentación. El estilo autocrático y el permisivo comparten un rasgo

común: en ninguno de los dos la información fluye desde el equipo de trabajo hacia los directores.

Ventajas: Puede ser eficaz en proyectos innovadores, en los que se promueve la creatividad. La libertad de acción suele levantar la moral del equipo de desarrollo.

Desventajas: Puede llevar a la desorganización por falta de dirección y resulta desastroso cuando se precisa una toma de decisión rápida.

3. Estilo democrático: En este estilo los gestores antes de tomar decisiones, tratan de recibir información originada en el equipo. Generalmente este estilo es el más eficaz.

   Ventajas: Posibilita la toma de buenas decisiones, porque refleja varios puntos de vista. Aumenta el compromiso del equipo, ya que se saben partícipes en la toma de decisiones.

   Desventajas: Puede ser inútil cuando se precisa la toma de decisiones rápidas. Difícilmente podrá ser aplicado en sitios en los que ni siquiera conocen la faceta humana y política de la democracia.

En la factoría existen normas de funcionamiento basadas en trabajos investigativos realizados, donde los directivos son responsables del control de su estricto cumplimento. El modelo propone que el estilo de dirección adoptado para el control de estas normas sea autocrático, pues las mismas son las que guían a los participantes de la factoría a cumplir los objetivos estratégicos planteados.

Dentro de las tareas a realizar por parte de cada role definido en la factoría se debe implantar un estilo democrático, los participantes deben tener libertad de generar ideas e intercambiar opiniones, esto unido a la juventud de cada uno de sus miembros puede acaudalar una gran creatividad; de la cual se pueden retroalimentar todos a su vez, incluyendo los directivos; los cuales no tienen experiencia laboral alguna.

El estilo de dirección empleado tiene una gran influencia sobre los resultados a obtener, la clave está en saber cual aplicar según las circunstancias. Esa decisión depende en gran medida del sentido común del director y de su capacidad para evaluar correctamente las circunstancias en que se encuentra, además de algunas características de su personalidad.

La estructura de dirección estaría compuesta por: un equipo de asesoría y servicio a la producción; líneas de desarrollo definidas, y equipos de desarrollo software, donde cada equipo tiene un responsable. El responsable del equipo de asesoría y servicio, junto a los responsables de las líneas de producción se subordinaría a la dirección general de la factoría; y el responsable del equipo de desarrollo se subordinaría al jefe de su línea

correspondiente. Podrían existir tantos equipos de desarrollo como demanda de trabajo haya por parte del cliente.

Estos estilos y estructura jerárquica de dirección, unidos a la organización de los grupos de desarrollo podrían ser utilizados no solo en la línea de aplicaciones Carrefour, sino en la factoría en sentido general.

4.4.3 Distribución de roles en la factoría de software

Dada la estructura del equipo de dirección y del equipo de desarrollo se han identificado los siguientes roles, los cuales pueden ser ocupados por una o varias personas:

- *Gestor de la Factoría*: Es el encargado de dirigir y apoyar las distintas líneas de desarrollo de la factoría. Debe ser capaz de en conjunto con la dirección de cada línea de desarrollo estimar plazos, costos y riesgos para la realización de un determinado producto software, estableciendo las relaciones necesarias con el cliente. Además controlar el desarrollo de los distintos productos, debe verificar y tener noción de la fase en que se encuentran. Administrará con economía y eficacia los distintos recursos y procesos de la factoría, debe organizar el trabajo de forma tal que las personas bajo su mando cumplan su rol adecuadamente.

Por cada línea de desarrollo dentro de la factoría habrá un equipo de dirección de la misma donde se identifican los siguientes roles:

- Jefe de línea de desarrollo: Es el encargado de dirigir el desarrollo de productos software en una línea determinada dentro de la factoría. Tiene la tarea de velar por que los productos en su línea de desarrollo se realicen a tiempo y según las especificaciones establecidas por el cliente, realizando la recepción de un producto a desarrollar, controlando su estado de ejecución, enviando el entregable al cliente y almacenándolo segundas reglas establecidas. Tiene bajo su mando a uno o varios equipos de desarrollo, es el intermediario entre estos y el jefe de la factoría, constituye el canal a través del cual fluye la información desde los directivos hacia el equipo de desarrollo.

Además la Factoría contará con un grupo de soporte que tendrá como función brindar asesoría y servicio a las distintas líneas de desarrollo, en este se identifican los siguientes roles:

- Jefe de grupo de asesoría y servicio: Guía y apoya al grupo de soporte de la Factoría, responde por este grupo ante la dirección de la misma. Es el encargado de identificar

las necesidades de cada línea de desarrollo con el fin de poner a los especialistas del grupo en función de las mismas.

- Administrador del repositorio de componentes reutilizables: Es el responsable de gestionar el repositorio de componentes reutilizables de la Factoría.
- Administrador de base de datos: Es el encargado de la administración del sistema gestor de base de datos utilizado. Es responsable de la creación de la base de datos, y de mantener la seguridad e integridad de la misma.
- Investigador: Encargado de realizar cualquier investigación, búsqueda de información en un tema determinado solicitado por cualquier línea de desarrollo. Desarrollar herramientas de apoyo a la programación y a los procesos de desarrollo con el objetivo de aumentar la productividad y la calidad en el trabajo, automatizando y optimizando los procesos.

Los roles perteneciente a cada equipo de desarrollo, se definen en la entidad "*Proceso de Desarrollo para la línea Carrefour*" que se verá más adelante y estos si serán específicos en la línea Carrefour.

4.4.4 Instrumentos PSP que apoyan a los participantes.

Existen instrumentos de PSP que pueden ayudar en gran magnitud el trabajo organizativo de cada individuo en la factoría; teniendo en cuenta que el personal de la misma es joven e inexperto en desarrollo de software. Además de tener otras responsabilidades adicionales al desarrollo de software. Estos instrumentos pueden ser usados en todas las líneas de desarrollo de la factoría.

Consecuentemente se proponen algunos modelos referentes a este tema, sin pretender englobar todo el trabajo organizativo individual, teniendo como objetivo inicial incentivar y dar a conocer la importancia de estas herramientas entre los implicados, así como dar los primeros pasos en la planificación personal.

4.4.4.1 Cuaderno del ingeniero.

Uno de los principales modelos a poner en práctica es el cuaderno del ingeniero, mediante el cual los implicados podrán registrar tiempos, guardar cálculos, tomar notas de las fases del desarrollo, y registrar además cualquier otro tipo de información que afecte el horario de laboral en la factoría. Teniendo en cuenta que quienes trabajan en ella tienen responsabilidades adicionales al desarrollo de software.

El cuaderno podrá ser utilizado como evidencia del trabajo realizado, evidencia importante para la defensa de la factoría en cuanto a responsabilidad legal de un producto.

Pudiéndose mediante este demostrar las negligencias de la contrapartida como del trabajado de la factoría propiamente. Por el contrario, al registrar las buenas ideas en cualquiera de las fases del desarrollo, los directivos y asociados al equipo podrán reconocer el merito alcanzado.

El diseño del cuaderno se puede observar en el Anexo 4.

4.4.4.2 Cuaderno de registro de tiempo.

El cuaderno de registro de tiempo sirve para registrar diariamente las actividades realizadas en cada instante del día. Cuaderno muy útil para contabilizar y describir las interrupciones ocurridas durante la jornada laboral, y dejando almacenado el historial de las mismas. Un beneficio útil es que ayudaría al individuo a controlar la duración y frecuencia de las interrupciones. Además los directivos tendrían posibilidades de observar los comportamientos y realizar análisis al respecto. Intentado evitar el aumento de los tiempos de interrupciones, las cuales no son solamente un despilfarro de tiempo, sino que rompen el ritmo de pensamiento de los individuos de la factoría llevándolos a la ineficiencia y al error. El cuaderno de registro es una manera de hacerles comprender como son de interrumpidos, ayudándolos de ese modo a mejorar la calidad y eficiencia de su trabajo.

El diseño del cuaderno se puede observar en el Anexo 5.

4.4.4.3 Cuaderno de resumen semanal de actividades.

Los directivos de la factoría necesitan planificar actividades relacionadas con los horarios laborales establecidos; la capacidad de trabajo y profesional de cada miembro del equipo; la cual es variable en equipos de gran cantidad de integrantes; como es el caso de los equipos dedicados al desarrollo de aplicaciones empresariales.

Para este propósito, los registros de tiempos son muy detallados por lo que se necesitan reunir los datos de una forma más útil. Esto se hace mediante el modelo que propone PSP nombrado resumen semanal de actividades, en el cual se agrupan las actividades por días, obteniendo totales de tiempo relacionados con los trabajos efectuados diariamente y de los trabajos efectuados en la semana en general.

Al modelo básico que propone PSP se le agregarán varios aspectos para cumplimentar los objetivos de planificación de la factoría. Dentro de los que se encuentra:

- Clasificación: Aspecto que define y agrupa las actividades relacionadas en el cuaderno de registro de tiempo en dependencia de los objetivos perseguidos al planificar. Los cuales podrían ser: vacaciones, incapacidad, diseño, implementación, pruebas, contingencias, comunicación, soporte, capacitación y reuniones. Estos son los esenciales, a medida que se experimenten en la práctica pueden ir surgiendo otros.

- Descripción: Existe otro aspecto a agregar al modelo propuesto por PSP, el cual es la descripción breve de la actividad realizada en el modelo de registro de tiempo.
- Identificador de requerimiento: Estaría formado por el número del equipo, en caso de que existan varios y el número en orden ascendente del componente a elaborar. Ejemplo: 03001, se refiere al primer componente del equipo 3.

Finalmente el modelo quedaría tal y como está representado en Anexo 6.

4.4.4.4 Los cuadernos de trabajo.

Este cuaderno se diseñó para registrar datos de tiempos estimados y reales. Este es un documento de planificación de producto, a diferencia del Cuaderno de registro de Tiempos y el Resumen Semanal de Actividades que contienen datos de planificación de períodos.

Con los datos almacenados en el cuaderno de trabajo se puede calcular la velocidad media de un individuo para una tarea dada, de esa forma los trabajadores de la factoría pueden comenzar a estimar su velocidad mínima, media y máxima ante los tareas a elaborar en dependencia de su tipo. Aunque en los trabajos iniciales tengan que predecir; ya luego para otras ocasiones tendrán almacenados un grupo de datos por los cuales orientarse. Los cuales estarían representados de una forma concisa y clara para acceder a ellos, lo que es clave para hacer estimaciones exactas y esto a su vez es importante para elaborar planificaciones. Luego de conocer la velocidad individual de cada individuo, el alcance de los componentes y acumular cierta experiencia; los directivos podrían hacer planificaciones de tareas de mayor magnitud, componentes, componentes de alto nivel o subsistemas. Pudiendo estimar y dar informaciones referentes a las fechas de entrega a los clientes y otros implicados.

El modelo está representado en el Anexo 7.

Existen muchas otras técnicas de PSP que pueden soportar el desempeñó de los implicados en la factoría en general, pero inicialmente lo pretendido en el modelo productivo planteado es el aprendizaje de la estimación del tiempo necesario para lograr objetivos. Una vez logrado esto todas las demás cuestiones de planificación se lograrán con mayor facilidad.

4.5 Técnicas y Herramientas.

El proceso de desarrollo está basado en herramientas, métodos y mecanismos. En el momento de seleccionar las herramientas para trabajar en una fase del desarrollo, se debe ser bien cuidadoso, al igual que al diseñar las metodologías que optimicen los procesos.

Es conocido que no se necesita la última generación de ordenadores para desarrollar software de alta calidad, pues las herramientas CASE (Computer Aided Software Engineering - Ingeniería de Software Asistida por Computadora), son un elemento indispensable para la optimización de cualquier proceso productivo. Por lo tanto, el mejoramiento continuo en el uso de las mismas por cada trabajador de la factoría es muy importante para lograr aumentar la productividad continuamente. Cada vez que a la factoría se incorpore un nuevo integrante, deberá estar ya relacionado con el uso de las herramientas y las normas establecidas en el proceso.

Es bien sencillo equivocarse al suponer que como ya se tienen las técnicas, mecanismos y herramientas definidas, todos asimilarán su uso con igual facilidad. Se deben implantar mecanismos para evaluar el verdadero conocimiento de los trabajadores de la factoría y así evitar falsas suposiciones que pueden traer inconsistencias en los tiempos de entrega, por diferencias en la velocidad del aprendizaje.

En la organización de una factoría de software debe conocerse fundamentalmente que utilidad presenta cada herramienta, y consumo de recursos en la estación de trabajo. Además se debe crear una cultura de cumplimiento a totalidad de todas las normas disciplinarias y técnicas especificadas por los directivos.

4.5.1 Herramientas de Gestión Proyecto.

Parta esta actividad la herramienta usada es: Microsoft Project, una aplicación de Microsoft que ayuda al usuario a crear planes de proyectos, comunicarlos a otros usuarios y adaptarse a los cambios a medida que éstos se van produciendo. Es un sistema de planificación de proyectos versátil y fácil de utilizar. Provee servicios tales como la introducción de las tareas del proyecto y sus duraciones, la organización de tareas en estructura jerárquica y la vinculación de tareas para diferentes tipos de relaciones están sumamente simplificadas. También se incluyen prestaciones para crear hipervínculos, para aprovechar los servicios de Internet y paginas Web.

Con un procesador Pentium || de 200 Mhz. y 32 MB de memoria RAM, el programa se desenvuelve normalmente.

4.5.2 Tecnologías y herramientas de Construcción.

Cada tecnología de construcción tiene asociado un gran cúmulo de herramientas y metodologías que soportan su desarrollo. Un principio indispensable de cualquier factoría de software es que debe tener establecido para cada línea de producción cual es la plataforma o lenguaje de programación y el grupo de herramientas a utilizar. Por lo tanto se

considera en este grupo de tecnologías de construcción: Frameworks J2EE, herramientas de Diseño, herramientas de construcción, herramientas para pruebas y de gestión de bases de datos.

- Plataforma de desarrollo: Framework Carrefour(C4J). El correcto uso de su arquitectura MVC, basada en Struts y los servicios presentes, traen consigo la elaboración de componentes reutilizables y aplicaciones robustas.

- *Construcción:* para esta actividad se propone IBM WebSphere Studio Application Developer (WSAD) Integration Edition version 5.1. Entorno integrado de desarrollo para el diseño rápido, construcción, pruebas, despliegue de aplicaciones, Servicios Web, incluyendo desarrollo de portales. Esta nueva herramienta de uso sencillo incluye nuevas capacidades para análisis de código y análisis de rendimiento de las aplicaciones desarrolladas. Exige de hardware las siguientes restricciones:

- Pentium III 500 MHz o Pentium IV recomendado.
- 512 MB RAM como mínimo o 768 MB de RAM recomendado.

- *Diseño y Especificación:* IBM Rational XDE Developer Plus para Java. Esta es una herramienta que posee los artefactos específicos para el correcto modelado de aplicaciones J2EE, la cual permite una perfecta integración con WSAD. Exige de hardware las siguientes condiciones:

- Pentium III a 500 MHz mínimo o Pentium IV.
- 512 MB de RAM recomendado; mas memoria generalmente mejora responsabilidades.

- *Pruebas: IBM Rational XDE Tester.* Herramienta de fácil uso para técnicos de pruebas y analistas, que introduce innovadoras técnicas de creación y ejecución de pruebas para reducir los efectos de los cambios de software. Permite crear pruebas manuales robustas y fiables para verificar la calidad de las aplicaciones antes de la instalación.

- *Gestor de Base de Datos: Oracle 9i.*Permite almacenar gran cantidad de información y su posterior manejo de forma rápida y segura, la herramienta que utiliza ORACLE para acceder a la base de datos es el lenguaje no procedural SQL, y este lenguaje es *relacionalmente completo*, es decir, implementa prácticamente toda la funcionalidad y características del modelo relacional teórico. Exige de hardware las siguientes condiciones:

- 512 MB de RAM recomendado; mas memoria generalmente mejora.
- Pentium IV.

4.5.3 Herramientas de Gestión de Configuración.

Debido al gran número de herramientas de gestión de configuración existentes en la actualidad y la ausencia de un concepto claro para elegir cual es la óptima para una tarea específica; es que los organizadores de grupos de trabajo tienen que elegirlas en función del trabajo que se vaya a realizar, la experiencia en su uso y su capacidad de adquisición en ese momento. Una herramienta muy popular es el CVS (Concurrent Versions System). CVS es un estándar para el control de versiones que posee implementaciones en todos los sistemas operativos y es considerada libre (freeware), además es la herramienta recomendad por IBM pues logra una perfecta integración con WSAD y de esta manera el entorno de trabajo del desarrollador queda cómodo pues se logra una integración entre el cúmulo herramientas fundamentales usadas por el. Estas cualidades hacen que sea muy utilizado en el desarrollo de sistemas distribuidos a través de Internet.

4.5.4 Gestión de la comunicación.

Las comunicaciones están establecidas en dependencia de las necesidades de información y del role que tenga quien la necesite. Son los directivos de la factoría los de mayor protagonismo al establecer comunicación con el medio externo, pues son estos los que poseen la visión estratégica de los proyectos emprendidos. Esta comunicación se ha definido a través de una cuenta de correo electrónico a la cual solo tiene acceso el gestor de la factoría y niveles de dirección superiores a el.

La vía telefónica posibilita rapidez en las comunicaciones pero es bien sencillo piratear los diálogos establecidos. Además se vuelve engorroso dejar copias, imposibilitando la exigencia de retrasos.

El uso de los sistemas de Chat o correo electrónico por parte de los integrantes del equipo solo se usaría en los momentos de dudas durante el desarrollo y con la autorización del gestor de la factoría; dejando definido los niveles de encriptación por parte del equipo de soporte técnico de la infraestructura productiva.

Una solución a la gestión de la información la resuelve Lotus Enterprise Integrator, ofreciendo una probada experiencia de mensajería y colaboración de gran calidad. Amplia las aplicaciones colaborativas a aplicaciones Web basadas en estándares con un nuevo elemento de diseño de servicios Web; almacenamiento de datos de Lotus Notes en una base de datos relacional DB2 y permite crear aplicaciones que combinan la colaboración

con el lenguaje SQL. En MatchMind es ampliamente usado este sistema, quienes como empresa consultora lo recomiendan a todas sus factorías.

4.5.5 Mecanismos de Control de la calidad.

Los productos terminados deben cumplir las especificaciones de calidad que se le propone al framework, denominadas lista de comprobación. Mediante la cual se especifican una serie de puntos a evaluar. En todas las fases del desarrollo se deben tener en cuenta estos puntos en función de que al finalizar cada tarea haya que hacer el menor número de correcciones posibles. Entre estos puntos se encuentran: Recepción, Convenciones de código y de Diseño J2EE, Diseño Arquitectura Lógica Genérico, Pruebas Básicas, Inspección guía de estilos, Desarrollo seguro con J2EE.Todos estos puntos se pueden observar en detalle en el Anexo 8.

4.5.5.1 Utilización de plugin de validación y control de la calidad.

Existen plugins que se deben usar para facilitar el desarrollo de la implementación cumpliendo ciertos estándares de calidad, definidos por SUN. Aunque algunas veces los puedan agregar estándares propios. Sean cuales sean los criterios, los plugins siempre proporcionarán una utilidad fundamental al desarrollador.

Concretamente se utilizarán dos plugins: Checkclipse y PMD Utiles.

Ambos tienen una licencia gratuita y su uso no supondrá ningún problema legal.

El uso va dirigido a todos aquellos que desarrollen proyectos utilizando Java, J2EE, aunque puede hacerse extensible al resto de proyectos y plataformas (como .NET, C/C++...).

- CHECKCLIPSE: Este plugin se centra en la sintaxis del código, javadoc y comentarios y servirá para el chequeo de código Java, J2EE.
- PDM: Este plugin a parte de la sintaxis y comentarios Javadoc, también nos ayuda a detectar errores de código como variables que no se usan, bloques de código repetidos y anidaciones de bucles indebidas.

El uso y configuración de los plugin se puede observar en el Anexo 9.

4.5.5.2 El cuaderno de registro de defectos.

El cuaderno de registro de defectos esta diseñado para ayudar a reunir datos de defectos. Mediante este cuaderno se deben registrar los defectos en dependencia de su tipo, principalmente los que tienen que ver con la calidad de componentes implementados. Con el fin de tenerlos presentes a la hora de realizar nuevas correcciones a ese mismo componente o a otro diferente. El diseño del cuaderno puede observarse en el Anexo 10.

4.5.6 Mecanismo de seguridad de la información.

En toda organización es importante establecer medidas que posibiliten la seguridad de la información que fluye tanto internamente como externamente a la factoría, pero primeramente es necesario clasificarla en grupos que posibilitan su mejor manipulación. Como son:

- Plantillas y técnicas que ordenan el proceso, ya sean enfocadas a los servicios de administración y seguridad de la información, al control de la calidad, al formato y estilo de los archivos a recibir y de los entregables.
- Documentación asociada a la plataforma J2EE, metodologías de desarrollo de software y a los servicios del Framework, sujeta a los cambios del mismo.
- Instaladores necesarios y repositorio de componentes de software reutilizables.

Para mantener la información es necesario establecer políticas de seguridad de la información (PSI), las cuales surgen como una herramienta organizacional para concienciar a cada uno de los miembros de la factoría sobre la importancia y sensibilidad de la información.

Estás políticas describen accesos como: la información básica, software que se deben instalar, garantía del producto en uso, restricciones sobre los dispositivos de acceso a la estación de trabajo y del personal ajeno a la factoría. Incluyendo medidas que se deben tomar para traspasar información a través de Internet. Estas se pueden observar en el Anexo 11.

4.5.7 Técnicas de configuración de las herramientas.

Como se ha comentado anteriormente el framework posee un grupo numeroso de servicios y utilidades, ahora este es una tecnología propia de los clientes, que para nada tiene que ver con las herramientas de construcción. Por eso se necesitan una serie de técnicas de configuración.

Dentro de las técnicas de configuración se encuentran las establecidas respecto a la arquitectura de los componentes; la introducción del Framework en las herramientas de construcción (WSAD y SGBD) y la selección y configuración del servidor de aplicaciones a utilizar. Luego de estar refinados todos estos mecanismos, el desarrollador solo tiene que concentrarse en las especificaciones técnicas y funcionales, ganando mucho más en productividad a partir de ese momento.

Las técnicas de configuración están establecidas en el Anexo 12.

Finalmente se podría agregar que solo las técnicas que intervienen en la construcción, control de la calidad y configuración del framework ; son las propias de la línea Carrefour, o sea el resto de las técnicas, mecanismos o herramientas expuestas pueden ser utilizadas en el desempeño de la factoría en sentido general.

4.6 Proceso de desarrollo para la línea Carrefour.

El desarrollo de software actual esta siendo orientado por estándares y metodología que se adaptan a las condiciones de las organizaciones. Mantener bien organizado y localizado cada elemento de la metodología es una tarea muy importante. En sistemas pensados para elaborarse en entidades empresariales diferentes esto se vuelve una necesidad vital. Pues a través de normas negociables se organizaría eficientemente la comunicación y se lograría un proceso consistente.

Elaborar normas, definir roles, artefactos y herramientas que guíen en cada fase el proceso productivo, para el desarrollo de subsistemas, es el objetivo fundamental de este epígrafe. Particularmente persigue elabora flujos de trabajo basadas en RUP, las cuales sinteticen ese gran proceso a los aspectos claves que pueden ser útiles en el desempeño de la factoría en la línea C4J, donde las fases serían: Recepción de Especificaciones, Diseño, Implementación, Pruebas y Entrega. Explicándose para cada uno de ellos la necesidad de su creación, así como los elementos que deben garantizar un avance estable. Este proceso debe estar coordinado por un *Jefe de Equipo de Desarrollo* quien es el encargado de dirigir y guiar al equipo en su conjunto durante el desarrollo, controlar que cada integrante cumpla con su trabajo en el tiempo establecido además de atender sus necesidades. Podría a su vez jugar el role de jefe del equipo de diseño.

4.6.1 Recepción de Especificaciones.

En todo aplicativo a implementar es necesario en el comienzo del ciclo de desarrollo capturar algún tipo de información que especifique los requerimientos en general del producto. En el proceso productivo en la factoría, comienza en la recepción de las especificaciones del componente a implementar; estas elaboradas por la empresa cliente. En muchos de los casos no vienen de igual forma esas especificaciones, trayendo consigo que los desarrolladores tengan que adaptarse a disímiles situaciones, lo que provoca descensos en la productividad. El objetivo de la fase inicial es la creación de normas que rijan la "Recepción de Especificaciones" de los componentes a elaborar.

Los componentes en todos los casos tienen dos rasgos describibles sobre los cuales se debe hacer énfasis: Análisis funcional y Análisis técnico. A continuación se muestra una propuesta de cada uno de estos análisis; con el objetivo de tomarlos como plantillas para la recepción de las especificaciones.

4.6.1.1 Análisis funcional

En el análisis funcional se describen las prestaciones que brindará el componente a la organización, describiendo textualmente cual es el alcance y las fronteras del mismo. La descripción no abarca el enfoque sistémico del proyecto, sino la definición específicamente del componente a elaborar. Dejando claro la descripción de los usuarios que interactuarán con la aplicación y a que niveles de información pueden acceder.

Para esto es recomendable usar modelos de Casos de Uso los cuales en su expansión logren un detalle de la definición de cada funcionalidad así como del componente en general, apoyados en diagramas de actividad en casos complejos. Además debe incluir a modo de anexos todos los datos secundarios relacionados con las funcionalidades descritas.

Estos modelos pueden ser recibidos en documentos de textos los cuales estén bien especificados; sin la necesidad de los ficheros con extensión de modelado.

Tal como podría observarse en el Anexo 13.

Se deben agregar al análisis funcional un segundo documento con la definición de requerimientos; los cuales describan el alcance; fecha de puesta en funcionamiento de la aplicación; y el entorno tecnológico en el que se ejecutará. Además de la especificacion de los requerimientos con un orden lógico.

4.6.1.2 Análisis Técnico

El análisis técnico consiste en la descripción de los requisitos técnicos del componente a implementar. El documento debe exponer y describir claramente la maqueta del proyecto, la cual estará formada por las interfaces gráficas de usuarios, y estas a su vez por componentes de presentación de páginas del framework C4J. Además debe describir la interactividad de la misma. Tal como se muestra en el Anexo 14.

Al análisis técnico debe venir adjunto un documento donde se especifique el modelo de datos del componente a implementar, o los ficheros de modelado efectuados mediante una herramienta CASE como Embarcadero ER/Studio o Power Designed.

Finalmente al llegar la especificación del producto el jefe de la línea de producción, debe verificar que se cumplan los requisitos planteados por los documentos explicados

anteriormente, en caso positivo se pasa al jefe de equipo que se estime conveniente, en caso contrario se devuelve a los clientes.

4.6.2 Fase de Diseño de Subsistema.

En un desarrollo entre varias entidades empresariales a medida que la relación gana en seriedad, el grado de las tareas asignadas va aumentando paulatinamente. Cuando los componentes son de alto nivel o la cuantía de los requerimientos es numerosa, los documentos de especificaciones se vuelven muy extensos. Esto trae consigo que se dificulte el trabajo de definición de los componentes de código y de infraestructura a utilizar. Además se hace necesario un conocimiento detallado de los servicios brindados por el framework y su arquitectura; teniendo como base los recursos existentes en el repositorio de componentes, en cuanto a clasificación y utilidad que pueden prestar. La factoría necesita de una organización que posibilite al trabajador(s) encargado, coordinar una serie de actividades simultáneamente, para garantizar un desempeño óptimo en la obtención de un producto con calidad en la etapa de construcción. Debido a que esta es la fase que más recurso humano se necesita a lo largo del proceso fabril.

Específicamente se recomienda una fase de complemento al diseño, pues en los documentos recibidos, se especifica solamente el modelo de datos y las interfaces gráficas del aplicativo. Lo que completaría y facilitaría el desarrollo de la implementación es el diseño y estructura que deben soportar los subsistemas, componentes de código y de infraestructura; logrando transformar los requerimientos en los esqueletos de componentes a implementar, aprovechando el código generado por la herramienta de modelado. De esa forma se podrá incluir en el equipo de implementación más personal, que requiera solamente conocimientos básicos de implementación.

4.6.2.1 Subsistemas de Diseño

Un subsistema de software es un artefacto de software manejable, auto-contenido y claramente identificable, que puede ser utilizado con diversos objetivos, tales como:

- Ejecución de funciones específicas.
- Integración con otros sistemas a través de una interfaz clara que posee.
- Sustitución independientemente, sin romper otras partes de los sistemas.
- Capacidad de desarrollo y mantenimiento independiente; mientras mantengan las interfaces.
- Desplegados independientemente en un conjunto de nodos computacionales distribuidos.

- Dividir el sistema en unidades con acceso restringido sobre los recursos claves.
- Representación de productos existentes o sistemas externos en el diseño.

Lo expuesto anteriormente demuestra que los subsistemas pueden ser usados para encapsular componentes, que son más que un archivo de código. Por eso el diseño en el proceso se enfoca al diseño de subsistemas implementados en C4J.

4.6.2.2 Artefactos utilizados

Luego de tener definido lo que se va a modelar se deben establecer los artefactos fundamentales que serán usados en esta fase. Garantizando que se reduzca el número de estos expuesto por RUP y que se estandarice el uso de los mismos durante la fase.

✓ Modelo de Diseño:

Esta es la fase de desarrollo donde se modela el resultado de los documentos expuestos en la fase de Recepción de Especificaciones; teniendo en cuenta el framework y tecnologías asociadas que serán utilizadas. Esto será a través de la creación de un modelo de diseño. La ingeniería de software plantea que las labores de arquitectura de una aplicación es una tarea de prioridad; la cual debe ser orientada a establecer la estructura y organización del modelo de diseñó. Mediante esta se muestra la vista arquitectónica, y los desarrolladores de cuestiones específicas pueden tener una noción lógica de todo el desarrollo.

La propuesta, está orientada al desarrollo de subsistemas o componentes de alto nivel, basados en otros componentes de diferente funcionalidad. Constituyendo a su vez aplicaciones empresariales multicapas. Tal como se muestra en el Anexo 15.

Además de las capas de la aplicación, la estructura agrega realización de casos de uso; el cual serán explicados a continuación.

✓ Realización de los casos de uso:

Cada realización de Caso de Uso es un Caso de Uso con el mismo nombre del caso de especificado en los documentos recibidos en la etapa de recepción. Un ejemplo de los artefactos que tiene una realización, es mostrada en el Anexo 16.

- Entre los artefactos de diseño que participan en la elaboración del caso de uso, se proponen los diagramas de interacción, ya sean: diagrama de secuencia o de colaboración, con el flujo básico y cada uno de los flujos alternativos. Teniendo en cuenta que ya estos están más asociados a la fase de implementación. Se tiene además una descripción del flujo de eventos textual.

- En la realización se creará un diagrama de clases referenciado a cada uno de los elementos de análisis asociados a el, teniendo en cuenta las asociaciones entre ellos.

A través de los diagramas, se describe el comportamiento del componente a implementar, completando el caso de uso a través de cada uno de los elementos que interactúa. En los diagramas de clases se deben dividir y detallar cuidadosamente las clases según su role en el framework, ya sean en clases de lógica de presentación, de lógica de negocio, de entidades del negocio, o de acceso a datos.

En la capa de presentación se deben modelar los elementos que participan en la realización de los Casos de Uso; además de los accesos con otras capas interiores al sistema, cuya realización se detalla en los subsistemas correspondientes.

O sea, en la capa de presentación deben quedar ubicadas las clases de interfaz de usuario y las clases relacionadas con la lógica de presentación de las interfaces, mientras que el resto de las clases se especifican en las capas más internas.

✓ Flujo de sucesos del diseño:

Este flujo es una descripción textual que explica y complementa a los diagramas y a sus etiquetas. El propósito fundamental de esta descripción es detallar las funcionalidades que se expondrán sobre los XMLs de configuración del framework, los cuales tienen la responsabilidad de conectar las clases con los servicios del mismo.

La herramienta de modelado no expone esta funcionalidad para diseño, pues esta es propia del framework C4J para implementación. Con el uso de etiquetas en el modelado o simulación de objetos en el caso de los diagramas de secuencias se puede representar el comportamiento que se le da a la aplicación con el uso de los XMLs de configuración, pero cuando estos sean numerosos el modelado puede oscurecerse con demasiadas etiquetas; por lo tanto mediante la descripción representada en el "Flujo de Modelo de Diseño" se especificaría con más detalle la funcionalidad de dichos componentes.

Otra utilidad importante de esta descripción es para la representación de múltiples diagramas de interacción, o para diagramas de interacción complejos. Este es el caso de las aplicaciones J2EE, las cuales integran componentes y subsistemas para ensamblar los subsistemas finales.

Lo que trae consigo que el flujo de sucesos de diseño de una realización de caso de uso no sea local a un determinado diagrama de secuencia. Por tanto puede utilizarse para describir la relación entre varios diagramas.

- ✓ Registro de planificación y distribución del diseño: En este registro se especifica la planificación del diseño de cada capa en específico, dejando claro la manera que se relacionan los componentes de una capa con las capas relacionadas y a que nivel de detalle se deben diseñar.

4.6.2.3 Roles que participan en la fase de diseño.

Algunos de estos roles son definidos en la entidad de "Participantes" con ámbitos globales dentro de la factoría; en esta sección se especifican un poco más el alcance de cada uno de los que participan en la etapa de diseño, definiendo además relaciones con otros roles y que artefactos y herramientas usan para desempeñar su función.

- ✓ Jefe de diseño: Es el que debe llevar la dirección del grupo de diseño, debe tener una visión global de toda la funcionalidad y forma del sistema. Debe apoyar todas las tareas con el fin de conseguir el máximo de creatividad en esta etapa.
- ✓ Arquitecto: en esta etapa el arquitecto es responsable de, la estructura del modelo de diseño. Debido a que el trabajo en las factorías es regulado por líneas de producción y tecnologías de implementación bien definidas (framework C4J en este caso), la estructura de la arquitectura es estándar en la mayoría de los subsistemas. El trabajo en ese sentido es enfocado principalmente al control de los artefactos utilizados en las diferentes partes de la estructura, teniendo en cuenta los componentes reutilizables existentes en el repositorio, los cuales exigen de un lugar en esa estructura dada su clasificación, su estilo de adaptación e integración. Además el modelo de diseño debe adaptarse a los eventuales cambios que sufre esta tecnología de implementación.
- ✓ Diseñador de caso de uso: el diseñador de casos de uso es el responsable de la integridad de una o más realizaciones de casos de uso de diseño y debe garantizar que se cumplen los requisitos que se esperan de ellos. Esto incluye hacer legibles y adecuadas para su propósito todas las descripciones textuales y todos los diagramas que describen la realización del caso de uso.
- ✓ Diseñador de componente general: El diseñador de componentes es el que agrupa los casos de uso por subsistema y a su vez los divide por capas, dejando especificado que componentes pertenecientes a un caso de uso determinado debe ser

implementado en cada capa y las relaciones que vana tener con los las capas relacionadas.

- ✓ Un diseñador de componentes en sentido general es el que define, mantiene las operaciones, métodos, atributos, relaciones y requisitos de implementación de una o más clases, garantizando que cada clase de diseño cumpla con los requisitos que se espera de ella; según la realización del caso de uso al que pertenezca. En la práctica estas son funcionalidades básicas que deben tener todos los diseñadores, por lo que hay que agregar funcionalidades adicionales en dependencia de la capa a la que se vaya a diseñar.
- ✓ Diseñador de componentes de la capa presentación: en la capa de presentación se deben diseñar las JSPs y las clases de lógica de presentación. En el caso de las páginas Web, vienen especificadas en el análisis técnico recibido desde los clientes, pero se deben especificar los XML de conexión con los servicios de presentación, detallando además que servicio se usarán. En cuanto a las páginas de lógica de presentación se deben detallar estando acorde con las especificaciones del framework, y al papel que juegan dentro de la capa de presentación.
- ✓ Diseñador de componentes de la capa de lógica de negocio: es el responsable de diseñar la funcionalidad del subsistema, utilizando los componentes especificados por el framework y guiando a los programadores sobre que responsabilidades y relaciones entre las clases deben establecerse.
- ✓ Diseñador de componentes de la capa de acceso a datos: es el responsable de decidir que tipo de acceso a datos de los especificados por el framework será implementado; según los requisitos del subsistema. Además debe dejar modelado los diagramas correspondientes y especificados de manera tal que los programadores solo tengan que concentrarse en traducirlos a líneas de códigos del framework.
- ✓ El diseñador de componentes puede también mantener la integridad de uno o más subsistemas. Asegurando que sus contenidos son correctos, al igual que las dependencias de otros subsistemas y/o interfaces son correctas. El diseñador de componentes que es responsable de un subsistema, es el diseñador que se debe convertir en jefe de implementación en la fase de Implementación.

4.6.2.4 Flujo de trabajo de diseño

Inicialmente el arquitecto debe elaborar el modelo de diseño, garantizado que este se rija por las especificaciones del framework. Identificando que subsistemas y componentes reutilizables pueden ser esbozados, dejando especificado sus interfaces. Después los diseñadores de casos de uso, elaboran su realización con el fin de comenzar a modelar los diagramas que le dan soporte y dejar especificado un enfoque global de la funcionalidad del sistema. Este a su vez delega la responsabilidad de cada subsistema especificado y realizado a un diseñador de componentes en dependencia del alcance del subsistema más general. Luego cada diseñador de componente general agrupa los casos de uso por subsistema y capas de la arquitectura en las que interactúan estos casos de uso; para así asignárselas a los diseñadores responsables de cada capa, esto lo registra en el Registro de planificación y distribución del diseño. Estos tienen la responsabilidad de detallar los elementos contenidos dentro de los subsistemas pertenecientes a su capa de arquitectura. Dejando descrito el flujo de sucesos del diseño, para mayor aclaración de los implicados.

4.6.2.5 Actividades principales a realizar dentro del flujo de trabajo de Diseño

Las actividades representan las principales acciones que debe llevar a cabo cada role para lograr los objetivos planteados en la fase del proceso.

- ✓ Diseñar la arquitectura: El principal objetivo de la arquitectura en esta etapa inicial del desarrollo, es la creación del modelo de diseño, el cual debe estar basado en las especificaciones del framework C4J, siguiendo una estructura multicapa orientada a subsistemas. En esta etapa es donde se decide que componentes o librerías brindan mejor servicios en el desarrollo, considerando los servicios que brinda el repositorio. Además se debe especificar que artefactos se deben utilizar para la modelación en la fase, así como los elementos que se deben usar.
- ✓ Identificar Subsistemas: Una clase de análisis compleja, se puede hacer corresponder con un subsistema de diseño, si la misma encierra un comportamiento que no puede ser responsabilidad de una única clase de diseño actuando por sí sola. Una clase de diseño compleja puede también convertirse en un subsistema, si es mejor implementarlo como un conjunto de clases que colaboran entre sí.

  Los subsistemas son además un buen mecanismo de identificación de partes del sistema que serán desarrolladas independientemente por un equipo separado. Si los elementos de diseño que colaboran entre sí pueden estar contenidos completamente en un paquete, solo con sus colaboraciones, un subsistema puede proveer un

mecanismo más robusto de encapsulamiento que el proporcionado por un simple paquete.

✓ Modelación de Subsistema de Diseño: La modelación de los subsistemas puede ser: como Subsistemas o Componentes UML. Como ya se ha explicado estas construcciones proporcionan capacidades de modelado similares. Pero los subsistemas incluyen explícitamente las nociones de "especificación" y "realización". Esto puede ser expresado con componentes, creando modelos de componentes separados para la especificación y la realización.

- Los artefactos de especificación, los cuales son usados para brindar una especificación abstracta del comportamiento ofrecido por los elementos de realización. Además actúan como contrato definiendo todo lo que los clientes necesitan conocer para usar el subsistema.
- Elementos de realización para modelar el comportamiento interno del sistema y es el diseño interno detallado destinado a guiar a los programadores.

La representación de estos artefactos como Subsistemas UML o Componentes UML es una decisión propia, se recomienda el uso de Subsistemas UML, estructurados como se explica a continuación.

La estructura propuesta para modelar un subsistema es la que muestra el Anexo 17. El cual refleja la Organización Interna de un Subsistema de Diseño.

Es evidente, el subsistema de diseño se divide en varios paquetes, uno o varios con la realización y otro con la especificación.

La realización de un subsistema contiene diagramas que referencia los elementos que intervienen en la misma. O sea, las clases y subsistemas de los niveles inferiores que se relacionan para dar solución a cada una de las operaciones de las interfaces. Para cada operación que muestra el subsistema en su interfaz, se construye una realización, con los mismos artefactos definidos para las realizaciones de Casos de Uso.

En las dependencias de la realización, se introducen las referencias a componentes reutilizables que se usen para aumentar la productividad. El uso de este componente es un detalle que no necesita ser expuesto a los clientes del subsistema. Estas dependencias adicionales deberán ser capturadas en diagramas separados como parte de la realización.

En la especificación del subsistema de diseño, se referencia mediante diagramas las interfaces del subsistema. Una especificación debe definir además sus dependencias. Estas son las interfaces y los elementos visibles de otros subsistemas y paquetes que deben estar disponibles en todas las realizaciones compatibles del subsistema.

- ✓ Identificar clases de diseño: en esta tarea se debe identificar las clases del diseño que se necesitan para realizar al caso de uso; para esto se deben estudiar la funcionalidad y los requisitos descritos por los subsistemas, tratando de diferenciar conceptos y asociaciones. Esto debe estar apoyado del documento que describe el análisis técnico, el cual refleja la navegación de la aplicación. Luego se deben identificar las clases de diseño que realizan estos requisitos y definen que componentes del framework la soportarán, especificando la capa de arquitectura correspondiente.
- ✓ Interacción entre Objetos de Diseño: Esta tarea se debe llevar a cabo para describir como interactúan los objetos los objetos del diseño. Esto se hace mediante diagramas de secuencias que contienen las instancias de los actores, los objetos del diseño y las transmisiones de mensaje entre estos. Si los casos de uso poseen varios flujos entonces deben crear un diagrama para cada flujo garantizando así su reutilización.

  Este paso se debe comenzar a partir de las especificaciones en el análisis técnico y con el apoyo del diagrama de clases. Decidiendo paso a paso que nuevos objetos deben aparecer y que instancias de actores son necesarias. Describiendo de esa manera el flujo del caso de uso.

Finalmente en la etapa el Anexo 18 se refleja una tabla resumen con el flujo anteriormente mencionado; la cual se divide en tres cuestiones fundamentales:

1. ¿Quién debe entregar especificaciones y qué debe entregar a un role determinado?
2. ¿Qué tarea debe ejecutar el role con esas especificaciones y qué herramientas debe usar?
3. ¿A quién debe enviar el role su entregable y qué debe enviar, para así comenzar el ciclo nuevamente?

4.6.3 Fase de Implementación

En esta fase se parte de los resultados del diseño y se implementa relacionando diversos elementos entre los que se encuentran archivos de código fuente, archivos scripts, archivos ejecutables, bibliotecas de terceros y similares. La arquitectura de las aplicaciones desarrolladas en la factoría es especificada por el framework C4J; siendo responsabilidad de su construcción la fase de implementación en función del desarrollo del sistema como un

todo. En este flujo de trabajo no se guiará el proceso por modelos sino que se convertirá el modelo de diseño directamente a la estructura de directorios en la que se basa la arquitectura.

Específicamente en la etapa de implementación se persigue:

- Revisar el modelo de diseño detalladamente.
- Agrupar los componentes por capas y por tipos de componentes, en función de implementarlos individualmente.
- Implementar componentes tales como: páginas web, interfaces, clases, XMLs, ficheros de recursos y bases de datos, que conformen componentes o subsistemas especificados durante el diseño.
- Probar funcionalidad de los componentes por capas y seguidamente integrarlos en la aplicación como un todo, antes de ser enviados a la fase de prueba.

4.6.3.1 Artefactos utilizados

Es considerado artefactos, los elementos utilizados por los roles para llevar a cabo las tareas que posibilitan el cumplimiento del la presente fase.

- ✓ Arquitectura básica de los proyectos C4J: Es la forma en la que están distribuidos los directorios y archivos de configuración en los proyectos a implementar.
- ✓ Descripción del plan de ensamblado: el propósito de este artefacto es dejar plasmado las partes manejables del subsistema, y dentro de cada parte manejable que componentes C4J se deben implementar. Además de especificar que componentes no pertenecientes al framework es necesario utilizar. En este documento además debe quedar descrita la segmentación de la realización de los casos de uso de diseño por capas en la fase de implementación. Para así orientar a los desarrolladores que componentes específicamente deben implementar y como conectarse con componentes de otras capas en el momento adecuado.
- ✓ Subsistemas de implementación: son partes manejables de sistemas de mayor alcance. Estos a su vez están conformados por interfaces, componentes de código, o de infraestructura y otros subsistemas. Implementan y proporcionan las interfaces que representan la funcionalidad que contienen en su interior.
- ✓ Solicitud de Repositorio: esta consiste en la descripción formal del recurso a utilizar del repositorio, ya sean componentes de código, patrones, algoritmos, formularios o documentación.

4.6.3.2 Roles que participan en la fase de implementación.

Al igual que en secciones anteriores, algunos de estos roles son definidos en la entidad de "Participantes" con ámbitos globales dentro de la factoría; en esta sección se especifican un poco más el alcance de cada uno de los que participan en la etapa de implementación, definiendo además relaciones con otros roles y que artefactos y herramientas usan para desempeñar una función determinada.

- ✓ Arquitecto: Durante la fase de implementación, el arquitecto es responsable de la construcción de la estructura de directorios del subsistema a implementar y asegura que la estructura como un todo sea correcta y se rija por las especificaciones del framework, incluyendo las relaciones entre capas. Se considera que la estructura es correcta cuando implementa la funcionalidad descrita en el modelo de diseño y en los requisitos adicionales especificados en esa fase. Además debe colocar en la estructura de directorio los componentes reutilizables especificados en el diseño. El arquitecto debe garantizar la sincronización entre el modelo de diseño y la estructura de implementación; asegurando que cada cambio o nueva idea arquitectónica generada y aprobada en la implementación, sea correspondida en el diseño.
- ✓ Responsable de calidad: Además es responsable de controlar la calidad en ciertos momentos del proceso de desarrollo. Se encarga de controlar que cada desarrollador cumpla con las normas establecidas, estableciendo mecanismo que logren que la calidad sea una tarea de todos.
- ✓ Jefe de implementación: El Jefe de implementación necesita garantizar que los contenidos y elementos dentro de los subsistemas de diseño (por ejemplo las JSPs y las Clases) se corresponden con los elementos físicos de la estructura de implementación; que sus dependencias con otros subsistemas o interfaces son correctos, y que implementan correctamente las interfaces que proporcionan. Debe planificar la implementación y dejarla registrada en la planificaron de ensamblado.

  Los Jefe de implementación deben tener responsabilidades en la fase de diseño del subsistema y de sus contenidos, para en la fase de implementación tener una visión de cada artefacto de implementación a utilizar en cada capa de la arquitectura y poder dirigir la fase con mejor claridad.

Programadores de alcance más específico:

- ✓ Programador de interfaz gráfica: Es el responsable de desarrollar la interfaz gráfica del subsistema utilizando los servicios del framework; incluyendo el mapeo de los XMLs entre las JSPs y las clases de lógica de presentación.

- ✓ Programador de lógica de presentación: Debe definir mediante código fuente uno o varios componentes, los cuales conecten la capa de presentación con la capa de lógica de negocio y establezcan el dinamismo de la presentación.
- ✓ Programador de lógica de negocio: Debe definir mediante código fuente uno o varios componentes, los cuales conecten la capa de presentación con la capa de acceso a datos y reflejen el dinamismo de la aplicación.
- ✓ Programador de la capa de acceso a datos: Debe definir mediante código fuente las clases que encapsulan y abstraen el acceso a los datos.

4.6.3.3 Flujo de trabajo de implementación.

El flujo se inicia implementando la arquitectura básica por el arquitecto del proyecto, dejando esbozado los componentes claves del modelo de diseño. Si es necesario incluir componentes del repositorio este se le solicita al gestor del repositorio, colocándolo a su ves en la arquitectura. A continuación el jefe de implementación chequea que el diseño esté elaborado completamente con el nivel de detalle especificado y que los elementos y contenidos del subsistema están acorde a las especificaciones del framework. Planifica la implementación, dejándola registrada en el plan de ensamblado, segmenta diseño y lo distribuye a los programadores responsables de cada capa. Los programadores deben implementar, probar y controlar la calidad de la capa o porción de la misma en que trabajan. Al finalizar la tarea correspondiente solicitan la revisión por parte del jefe de programación y del responsable de calidad, aprobando los mismos el ensamblado de la capa correspondiente. Luego de terminado este trabajo se le permite a otros programadores el uso de esta capa. Siguiendo un orden lógico dado la arquitectura del framework, se deben ensamblar desde las capas inferiores hacia las superiores, aunque no siempre tiene que ser así dependiendo eso de la capa que primero termine. Finalmente el jefe de programación es el responsable del ensamblado final, junto a otros programadores si es necesario. Luego se controla la calidad del subsistema como un todo y envía a la fase pruebas.

4.6.3.4 Actividades principales a realizar dentro del flujo de trabajo de implementación

Las actividades representan las principales acciones que debe llevar a cabo cada role para lograr los objetivos planteados en la fase del proceso.

- ✓ Implementación de la arquitectura: La principal cuestión que debe tener en cuenta en la implementación de la arquitectura, es la identificación de componentes significativos dentro de la arquitectura, tales como componentes ejecutables o

reutilizables. Como el framework especifica la arquitectura básica de las aplicaciones, esta se debe construirse y reutilizarse en otras construcciones, además los aspectos de configuración deben quedar bien definidos. Esto está abordado en la sección 4.2.1.8, "Técnicas de configuración de las herramientas".

Luego de obtenerse el modelo de diseño y las dependencias entre componentes, en implementación el mayor reto es crear dentro del subsistema de implementación los componentes que implementan los subsistemas de diseño correspondiente.

Existen servicios del framework que no son usados en todas las construcciones, como son los de la capa de integración. Es en esta tarea donde se deben agregar o eliminar a la configuración los ejecutables que prestan los servicios anteriormente mencionados. Posibilitando así que lo programadores solo tengan que preocuparse por planificar y traducir el modelo de diseño a líneas de código.

✓ Planificar implementación: En la planificación la construcción inicial debería comenzar a partir de de las capas intermedias y de software del sistema, las construcciones subsiguientes se expanden hacia arriba a las capas generales y específica de la aplicación. Por lo tanto se debe verificar que los componentes reutilizables estén bien diseñados, y posicionados correctamente dentro de la arquitectura; pues se hace muy difícil implementar componentes en las capas superiores antes de que estén colocados y correctamente funcionando los componentes de las capas inferiores.

Como lo que se va a implementar son subsistemas o componentes. Se deben tener en cuenta todos los requisitos modelados a través de casos de usos de diseño. Para cada caso de uso a implementar se debe hacer lo siguiente:

1- Inicialmente se debe analizar detalladamente la realización de cada caso de uso de diseño, pues en este se define la información necesaria para los programadores.
2- Identificar los subsistemas y clases de diseño que participan el la realización del caso de uso.
3- Determinar que componentes dentro del alcance del framework corresponden con los componentes de diseño definidos en el paso 2, si los componentes diseñados se van de las fronteras del framework se debe especificar que componentes del API J2EE es el que se va a utilizar en el plan de ensamblado.
4- Agrupar los componentes por sus capas correspondientes.

Los resultados deben estar especificados en el plan de ensamblado y comunicados a los programadores responsables de cada capa en el equipo.

- ✓ Ensamblar capas: El propósito de esta tarea es ensamblar el subsistema como un todo, conectando las capas según las normas de relaciones entre capas especificadas por el arquitecto.
  - Se debe estudiar el plan de ensamblado para conocer el tipo de asociación a utilizar, así como relaciones de componentes que debe efectuarse en la misma.
- ✓ Implementar una capa: La implementación de una capa consiste en la implementación de todos los componentes internos a la misma, y luego su ensamblado. Las capas poseen componentes tales como: (Java Server Pages) JSPs, archivos XMLs, Clases y Bases de Datos.
- ✓ Implementar Interfaz de Usuario: En esta se persigue principalmente elaborar las páginas web acordes a la especificación recibida en la fase de recepción de la documentación. Para esto se deben utilizar todas las tecnologías que provee el framework para la capa de presentación, dentro de las que se encuentra:
  - Uso de etiquetas personalizadas que permiten que la funcionalidad sea invocada sin tener la necesidad de escribir código java en dentro de la página Web.
  - Uso de archivos con código Java Scripts.
  - Uso de hojas de estilos.
  -Mapeo a archivos XMLs que soportan la funcionalidad de la página.
- ✓ Implementar clase: La implementación de una clase consiste en implementar una clase de diseño en un componente de fichero. Esto incluye lo siguiente:
  - Esbozo de un componente de fichero que contendrá el código fuente.
  - Generación del código fuente a partir de la clase de diseño y de las relaciones en que participa.
  - Implementación de las operaciones en las clases de diseño en forma de métodos.
  - Comprobación de que el componente proporciona las mismas interfaces que las clases de diseño.
  - Reparación de defectos de la clase, durante la pruebas de la clase.
- ✓ Implementar XMLs: Esta implementación tiene como propósito establecer los mapeos entre componentes de la aplicación y servicios del framework.
  -Implementar etiquetas y atributos especificados por el framework.
- ✓ Implementar Base de datos: Esta implementación tiene como objetivo generar la base de datos física en correspondencia con lo especificado en el análisis técnico. Para esto es necesario:

-Colocar valores en las tablas de las base de datos según los requisitos especificados para la misma.

✓ Generación de código a partir del modelo de diseño: al elaborar la estructura de un modelo de código, esta debe estar sincronizado con el modelo de diseño, teniendo en cuenta restricciones tales como las relaciones entre capas. De hecho el código fuente debe generarse a partir de la herramienta de modelado que se esté utilizando. Los componentes de tercero o los componentes desarrollados previamente por el equipo de desarrollo deben quedar referenciados en el modelo de diseño.

Hoy día se crean herramientas y tecnologías de implementación que no son soportadas por las mejores herramientas de modelado existente. Por lo general los ambientes de desarrollo integrados, unidos a framework que personalizan el trabajo de muchas organizaciones no encuentran fácilmente herramientas de modelado que les pueda generar completamente el código fuente o el esqueleto, de las aplicaciones correspondientes. En este caso se deben seleccionar las herramientas para adaptar o simular la creación de los elementos de desarrollo más apropiados; sin importar el origen del componente, intentando que el mismo refleje su asociación entre el modelo de código y el de diseño. En el caso del framework C4J en varios de los componentes de código pertenecientes a su arquitectura ocurre esta situación, es ahí donde hay que hacer uso de lo explicado anteriormente. No obstante en la mayoría de los componentes la herramienta si garantiza la generación del esqueleto del código eficientemente.

✓ Controlar Calidad: El control de la calidad en el proceso productivo estará enfocado al producto final, estrictamente al cumplimiento de las normas establecidas por el cliente. Este debe realizarse en tres momentos:

- Primer momento: cuando se implementa cada componente, antes de sincronizar con el CVS se debe chequear la calidad y registrar los errores encontrados en el cuaderno de registro de defectos. Para esto se puede apoyar en las herramientas existentes en el soporte técnico para el control de la calidad.
- Segundo momento: Cuando se ensambla una capa, se chequean los controles de calidad del "Primer Momento" y luego se controla la calidad de la capa a totalidad, estableciendo normas de calificación. Solo si la supera puede ser enviada a la fase de ensamblado de capas. En esta revisión se debe hacer una

revisión detallada basada en la lista de comprobaciones especificadas en el soporte técnico. Aquí también se deben registrar los errores.

- Tercer momento: Cuando el subsistema se integra finalmente antes de enviarlo al cliente, estableciendo normas de calificación. Solo si la supera puede ser enviado al cliente.

Es importante que en cada uno de estos momentos se evalúe también el cumplimiento de los requisitos funcionales planteados, para esto es necesario la realización de pruebas de unidad.

✓ Realizar pruebas: El propósito de realizar pruebas de unidad es probar los componentes implementados como unidades individuales. Se deben realizar pruebas de especificación y pruebas de estructura:

- Las pruebas de especificación verifica el comportamiento del componente externamente. Para esto es necesaria la implementación de clases de pruebas que contengan un conjunto de valores de entrada, estado o salida; para los que se supone que un componente se debe comportar.
- Las pruebas de estructura se realizan con el propósito de verificar que un componente funciona internamente como se quería. O sea, se debe intentar que en cada algoritmo se evalúe todos los caminos posibles a recorrer.

En el Anexo 19 se refleja una tabla descriptiva con el flujo de implementación de C4J anteriormente mencionado; en esta se puede observar como los programadores son guiados por el proceso de desarrollo.

4.6.4 Prueba

Actualmente existen muchos centros en los que el desarrollo de software tiene un alto porcentaje de artesanía y trabajo a medida. La tendencia actual pasa por evolucionar hacia las Factorías de Software en donde uno de los objetivos principales es la mejora y calidad del producto.

En el proceso de desarrollo de una Factoría están involucrados un gran número de trabajadores que participan en diferentes flujos de trabajo dentro del proceso. A pesar de que en la Factoría se busca la especialización de los trabajadores en un rol específico, las posibilidades de que se cometan errores son significativas.

En la garantía de la calidad del producto software en la factoría las pruebas juegan un papel fundamental, representan una revisión de las especificaciones, del diseño y la codificación. Para reducir los riesgos de posibles fallos de un componente o sistema se hace necesaria la creación de pruebas minuciosas y bien planificadas.

La prueba tiene como objetivos:

1. Realizar un plan de pruebas. Un buen plan de pruebas es la piedra angular y en consecuencia el principal factor crítico de éxito para la puesta en práctica de un proceso de pruebas que permita entregar un software de mejor nivel.
2. Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, los procedimientos de prueba que dicen como realizar las pruebas y si es posible componentes de prueba para automatizar las mismas.
3. Realizar las diferentes pruebas y analizar los resultados de las mismas. Las construcciones en las que se detecte error se prueban nuevamente y posiblemente devueltas a diseño o implementación para que esos errores sean corregidos.

Cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de pruebas y la depuración y evaluación de los datos resultantes.

4.6.4.1 Artefactos utilizados

Basado en RUP este trabajo propone que los trabajadores involucrados en el flujo de prueba en la Factoría construyan los siguientes artefactos:

- Plan de prueba
- Modelo de pruebas
- Caso de prueba
- Procedimiento de prueba
- Evaluación de prueba
- Reporte de defectos

✓ Plan de prueba: Representa la planificación de las pruebas, el mismo debe poseer lo siguiente:

- Descripción de aspectos generales
- Descripción de requerimientos que serán probados
- Definición de la estrategia de prueba a utilizar
- Recursos requeridos
- Calendario y plazos

- Definición de entregables
- Seguimiento y reporte de defectos
- Aprobación del plan

✓ Modelo de pruebas: El modelo de pruebas es una colección de casos de prueba, procedimientos de prueba y componentes de prueba que describe como deben ejecutarse las pruebas a aspectos específicos del software.

✓ Caso de prueba: En el caso de prueba se documenta la forma de probar el componente, incluye los datos de entrada con los que se va a probar, los resultados que se esperan y las condiciones en las que ha de probarse.

✓ Procedimiento de prueba: El procedimiento de prueba indica al probador como realizar uno o varios casos de prueba. A veces se reutiliza un procedimiento de prueba para varios casos de prueba y varios procedimientos para un caso de prueba.

✓ Evaluación de prueba: Es la evaluación del resultado de las pruebas, tales como discrepancias en la realización de las pruebas, estado de los defectos.

✓ Reporte de defectos: En el reporte de defectos es donde el jefe de prueba registra los defectos encontrados para enviarlos a los flujos anteriores (diseño o implementación).

4.6.4.2 Roles que participan en la fase de prueba.

E la fase de pruebas con solo dos roles se puede lleva a cabo esta tarea, los cuales son:

✓ Jefe de Prueba: es el que planifica, diseña y evalúa las pruebas a realizar.

✓ Probador: es quien tiene la responsabilidad de ejecutar las pruebas diseñadas por el jefe de pruebas guiándose por los procedimientos de prueba.

4.6.4.3 Flujo de trabajo de prueba

El flujo de trabajo de prueba comienza cuando llega al equipo de prueba el análisis funcional, análisis técnico, modelo de diseño y el resultado de la implementación del subsistema a probar, necesario para que el jefe de pruebas inicie la planificación de las mismas dejando elaborado el Plan de prueba. Posteriormente pasa al diseño de las pruebas dejando especificado los casos de prueba y los procedimientos correspondientes para ejecutarlas. Cuando quedan elaborados los casos y procedimientos de prueba, los probadores los utilizan para ejecutar las pruebas, detectando errores que puedan existir. Los errores que se detecten pasan al jefe de pruebas para registrarlo en el seguimiento y reporte de defectos, y enviarlos a flujos anteriores (diseño e implementación) para que sean corregidos. El jefe de prueba hace una evaluación general de la prueba, analizando los resultados obtenidos y comparándolo con el plan elaborado, los resultados quedan

documentados en la evaluación de la prueba, sirviendo como experiencia para futuras pruebas y para poder predecir el esfuerzo necesario para alcanzar un nivel de calidad aceptable en futuros componentes.

4.6.4.4 Actividades principales a realizar dentro del flujo de trabajo de prueba

A continuación se muestran las actividades que reflejan el comportamiento del flujo de pruebas. Dentrote las cuales se encuentra:

- ✓ Planificar prueba: en esta actividad el jefe de prueba realiza una planificación de las pruebas que se llevarán a cabo. El análisis funcional y técnico, modelo de diseño y modelo de implementación del subsistema, son los elementos de entrada para elaborar el plan de prueba. En la elaboración de este plan el jefe de prueba debe decidir cuales de los requerimientos registrados en el análisis funcional serán probados, ya que muchas veces es prácticamente imposible probarlos todos, en este caso debe decidir cuales son los más importantes. Debe trazar una estrategia de prueba, tipos de prueba, así como estimar los recursos requeridos, calendario y plazos para desarrollar las pruebas. La factoría desarrolla subsistemas y componentes que formarán parte de un sistema final del cual no posee el enfoque sistémico, por lo que mayormente se realizarán pruebas de integración, las cuales se utilizan para comprobar que los componentes integrados en un subsistema interaccionan entre sí adecuadamente, descartando las pruebas de sistema. Además el plan debe dejar establecido un mecanismo de seguimiento y reporte de posibles defectos que puedan detectarse durante la ejecución de la actividad.
- ✓ Diseñar prueba: En el diseño de las pruebas el jefe de prueba se encarga de elaborar los casos de prueba y los procedimientos de prueba que especifican como realizar estos casos. Para esto se basa en el análisis funcional y técnico, modelo de diseño, modelo de implementación y en el plan de prueba.
- ✓ Ejecutar prueba de integración: En esta actividad se realizan las pruebas planificadas y diseñadas en las actividades anteriores. El probador lleva a cabo las pruebas realizando los procedimientos correspondientes a cada caso de prueba, se comparan los resultados de las pruebas con los resultados esperados y se analizan los que no coincidan, además de informarle de los defectos encontrados al jefe de prueba para que este los registre y los envié a flujos anteriores para ser corregidos.
- ✓ Evaluar prueba: El jefe de prueba evalúa los resultados de las pruebas comparándolos con el plan de prueba realizado al inicio del flujo. Aquí se determina si las pruebas han cumplido su objetivo, analizando la cantidad de casos de prueba

ejecutados y de código probado. El jefe de prueba decide si es necesario realizar pruebas adicionales para localizar otros posibles defectos, puede que los resultados obtenidos y la experiencia de construcciones anteriores indiquen que el subsistema no está suficientemente maduro. El jefe de prueba documenta los resultados obtenidos y sugiere nuevas acciones si fuera necesario en el documento de evaluación de pruebas.

- ✓ Reportar defecto: En esta actividad el jefe de prueba realiza el reporte de los defectos encontrados a los flujos anteriores para que sean corregidos.

Finalmente mediante Anexo 20 se pude observar la descripción del flujo de trabajo de prueba en la línea C4J. Siguiendo el orden lógico de las tareas realizadas. Donde se puede apreciar como el proceso productivo guía a los probadores.

4.6.5 Entrega

Luego de la finalización de las pruebas se comienza la etapa de envío y almacenamiento del producto terminado.

Al terminar las pruebas el producto es enviado al jefe de línea de producción el cual es el responsable de aprobar el envío. Para el envío a los clientes se usan las técnicas de comunicación, establecidas en el soporte técnico.

Los clientes al recibir el producto; lo evalúan en sus departamentos de calidad y deciden la continuación del ciclo de vida o el retorno nuevamente.

Por otro lado el producto debe ser almacenado en el repositorio de componentes de la factoría. La fase comienza por la solicitud de almacenamiento enviada al administrador del repositorio de componentes; este documenta el almacenamiento junto al Jefe de línea de producción y de ahí en adelante es responsabilidad de dicho administrador el mantenimiento de esa información a lo largo del ciclo de vida del componente.

Dicha documentación del subsistema, se debe regir por la tabla expuesta en el Anexo 21. Esta tabla resume los aspectos esenciales de la Actividad almacenar componentes del modelo replicable.

Luego de esta se le suministra el componente al administrador del repositorio de componente, para que lo almacene o recupere en función de las técnicas expuestas en el soporte técnico.

Dicho subsistema en lo adelante es observado como un componente de software reutilizable (CSR).

4.7 Conclusiones

La propuesta del modelo funcional, da la medida de la cantidad de variables a tener en cuenta a la hora de crear proceso y luego poder optimizarlos. Esta propuesta aportó una serie de reglas por las cuales regir el funcionamiento, dejando abierto ciertos temas que deben ser refinados en el desarrollo práctico.

Los métodos referentes a cuestiones de dirección y trabajo en equipo, brindan orientaciones claras; pero aun son insuficientes pues el medio es muy dinámico y en la práctica surgen contratiempos que los jóvenes directivos deben enfrentar con solo su intuición y creatividad. Lo cual no es beneficioso en todos los casos.

La organización del recurso humano es fundamental en el desarrollo de cualquier entidad, no solo las relacionadas con productos de software. Por eso se define y profundiza en este aspecto, hasta llegar a la modelación del proceso productivo. En el cual se mostró lo que recibe cada individuo, tareas que realiza con lo recibido y destino que da al resultado de su realización.

Las ideas relacionadas con la reutilización de componentes, muestra la importancia y repercusión en la calidad y productividad; aunque en estos momentos no se tienen componentes reutilizables abundantes como para comprobar todas las variantes posibles de optimización de las técnicas expuestas.

Finalmente se logró elaborar el modelo funcional para aplicaciones C4J específicamente planteado como premisa, y se especifico en cada una de las entidades que lo componen, aspectos reutilizables a la hora de estandarizar el modelo hacia todas las líneas de producción de la factoría.

Conclusiones Generales

La realización de este Trabajo de Diploma ha aportado importantes conocimientos a los autores e implicados en la aplicación del enfoque factoría de software, brindando una guía con el fin de lograr una producción de software industrializada.

Para llevar a fin el objetivo se estudiaron las últimas tendencias y tecnologías actuales, se identificaron los cinco modelos de factoría más representativos entre los consultados.

Se logró identificar los elementos necesarios para la elaboración de un modelo funcional asi como las entidades que lo componen; basado en la situación problémica actual y en el análisis de modelos aplicables a factorías de software, que muestran cualidades atractivas y manejables.

Llegando así a la propuesta de un modelo funcional para la línea carrefour enunciada en el trabajo donde se definen las entidades y los mecanismos de control de la calidad durante todo el proceso de desarrollo.

La aplicación de ella debe llevar a que entre los integrantes de la factoría ha posibilite una mejor comunicación, así como la coordinación de las tareas a desarrollar por cada role definido. Agregando a esta la organización de los artefactos, mecanismos y herramientas que se reflejan en cada etapa. Sobre todo la comprensión de la necesidad e importancia de su creación, intentado adelantar pasos hacia la creación de un real proceso productivo industrial.

Finalmente la investigación ha servido para consolidar y normar ideas que estaban dispersas; logrando la unidad y creación de nuevos criterios entre los ingenieros de software que laboran con el objetivo de obtener productos de calidad reconocida.

Recomendaciones

Los objetivos específicos del trabajo no abarcan todos los temas relacionados con el enfoque de factoría de software; los cuales son amplios y diversos. Por lo que se considera necesario garantizar la identificación y seguimiento de elementos que pueden ser útiles en la optimización del modelo propuesto. Por lo que se propone:

- ✓ Elaborar un modelo funcional aplicable al resto de las líneas de producción existentes en la factoría.
- ✓ Elaborar modelos de formación e investigación desde la producción, en la Factoría de Software de la UCI.
- ✓ Creación de bancos de proyectos investigativos relacionados al enfoque de factoría de software. Dentro de los cuales se podría encontrar:
  - o Gestión de costo y plazo de los proyectos elaborados en la factoría.
  - o Estandarización de normas de calidad.
  - o Automatización de la gestión del repositorio de componentes reutilizables.

Por los aportes que tiene la aplicación del enfoque factoría de software en el ciclo de desarrollo de un producto de software en cuanto a la disminución de los costos y el tiempo de desarrollo así como elevar la calidad se recomienda a la infraestructura productiva el desarrollo de Modelos de Factorías de Software.

Bibliografía Referenciada

[Allan95]. ALLAN, J. Automatic Hypertext Construction, TR95-1484, Department of Computer Science, Cornell University, Ithaca, Feb 1995.

[Basili et. al., 1992]. Basili, V. R.; Caldiera, G.; Cantone, G.; *A Reference Archiecture for the Component Factory*. ACM Transaction on Software Engineering and Methodology. Vol 1. nº 1. pp 53-80. January 1992.

[Cantone, 1992]. Cantone, G. *Software Factory: Modeling the Improvement*. 'Competitive Performance Through Advanced Technology'. Third International Conference on (Conf. Publ. No. 359). Pages: 124 – 129. 27-29 Jul 1992.

[Cusumano, 1989]. Cusumano, Michael A. *Software Factory: A Historical Interpretation*. IEEE Software, (Vol. 6, No. 2). pp. 23-30. March/April 1989.

[Cybul93]. CYBULSKI, J.L. Reed: The Use of Templates and Restricted English in Structuring and Analysis of Informal Requirenment Specifications. AAITP Technical Report TR024, 1993.

[Fayad et al., 1999]. Fayad, E.M; Schmidt, D.C; Johnson, R.E. *Implementing Applications Frameworks: Object-Oriented Frameworks at Work*. Wiley & Sons, 1999

[Fernández y Teixeira, 2004]. Fernandes, Aguinaldo Aragon; Teixeira, Descartes de Souza. *Fábrica de Software:Implementação e Gestão de Operações*. Editora Atlas, 2004.

[Fernstrom et. al., 1992]. FERNSTROM, C.; NARFELT, K. H; OHLSSON, L. *Software Factory Principles, Architecture and Experiments*. IEEE Software. (Vol. 9, No. 2) pp. 36-44. March/April 1992.

[SEMATECH, 1998]. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Computer Integrated Manufacturing (CIM) Framework Specification. Version 2.0.* http://ismi.sematech.org/docubase/abstracts/1697jeng.htm (25/9/05)

[Gamma et al., 1995]. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley, 19995

[Goldb84]. GOLDBERG, *A. Smalltalk-80: The interactive Programming Environment*. Addison-Wesley, 1984.

[Li et. al., 2001]. Li, C.; Li, H.; Li, M. *A Software Factory Model Based on ISO 9000 e CMM for Chinese Small Organization*. Second Asia-Pacific Conference on Quality Software (APAQS'01).Hong Kong. December, 2001.

[Saboy93]. SABOY. J. *Effectiveness of Information Retrieval Systems Used in a Hypertext Environment*, Hipermedia, Vol. 5, Nº 1, 1993.

[Saboy93]. SABOY. J. *Effectiveness of Information Retrieval Systems Used in a Hypertext Environment*, Hipermedia, Vol. 5, Nº 1, 1993.

[Szyperski y Pfister, 1997]. Szyperski, C.; Pfister, C. *Component-Oriented Programming*. M. Muhlhauser. Special Issues in Object-Oriented Programming - ECOOP96 Workshop Reader. Dpunkt Verlag, Heidelber, 1997

Bibliografía Consultada

AHMED, K. Z. *Developing Enterprise Java Applications with J2EE and UML*. 1ra. Edición. Addison-Wesley, 2001. p. 0201738295.

ALLAMARAJU, S; Beust, C; DAVIES, J. *Programación Java Server con J2EE edición 1.3*. 1ra Edición. 2002. 1245 p. 84-415-1358-9.

Basili, V. R.; Caldiera, G.; Cantone, G.; *A Reference Archiecture for the Component Factory*. ACM Transaction on Software Engineering and Methodology. Vol 1. nº 1. pp 53-80. January 1992.

CANTOR, M. *Object-Oriented Project Management with UML*. John Wiley & Sons, 1998. p. 0471253030

D'SOUZA, D. F. and A. C. WILLS. *Objects, Components, and Frameworks with UML. The Catalysis Approach*. ADDISON-WESLEY, 1998. 735 p. 0-201-31012-0

DÍAZ, M. D. *Cómo desarrollar una arquitectura software: los lenguajes de patrones*, 2004. http://www.moisesdaniel.com/es/wri/ComoDesArqSoft.htm (15/10/05)

ELIZABETH, E. G. *¿Los equipos virtuales, son reales?*, 2002. http://www.ingrupos.com.ar/docs/equiposvirtuales.doc (30/10/2005)

Fernström, C. *The Eureka Software Factory: Concepts and Accomplishment*. Proceedings Third European Software Engineering Conference. Berlin, 1991.

FOWLER, P; Rifkin, S. *Software Engineering Process Group Guide* (CMU/SEI-90-TR-24). Software Engineering Institute, Carnegie Mellon University, 1990.

HUMPHREY, W. S. *Introducción al Proceso de Software Personal*. Madrid, Addison Wesley, 2001. p.

HUMPHREY, W. S. *Introduction to the Team Software Process*. The SEI Series in Software Engineering, 1999.

HUNTER, J. C. *La paradoja, un relato sobre la verdadera esencia del liderazgo*. Barcelona, Ediciones Urano, 2001. p. 847953365X

IEEE std 1008-1987. *An American National Standard IEEE Standard for Software Unit Testing*. IEEE Computer Society. Approved 11 December 1986 and reaffirmed 2 December de 1993.

IEEE std 1220-1998. *Standard for Application and Management of the Systems Engineering*. IEEE Computer Society. Approved 8 December 1998.

IEEE std 829-1998. *IEEE Standard for Software Test Documentation*. IEEE Computer Society. Approved 16 September 1998.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. El proceso unificado de desarrollo de software, Pearson Educación S.A., 2000.

Li, C.; Li, H.; Li, M. *A Software Factory Model Based on ISO 9000 e CMM for Chinese Small Organization*. Second Asia-Pacific Conference on Quality Software (APAQS'01).Hong Kong. December, 2001.

PRESSMAN, R. S. *Ingeniería del software, un Enfoque Práctico.* Quinta edición. Madrid, McGraw-Hill, 2002. 601 p. 84-481-3214-9

RATIONAL SOFTWARE CORPORATION. *Rational XDE Guide to Team Development,* 2003. http://WWW-136.ibm.com/developerworks/rational/library/4137.html (10/11/05)

Reynoso, Carlos B. *Introducción a la Arquitectura de Software.* http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp. (25/11/05)

Rockwell, R.; Gera, M. H. *The Eureka Software Factory CoRe: A Conceptual Reference Model for Software Factories*. Software Engineering Environments Conference, 1993. Proceedings. Pages:80 – 93. 7-9 July 1993.

SINGH, I. *Designing Enterprise Applications with the J2EE Platform*. 2da Ed. California, 2002. p. The Java Series. 0-201-78790-3.

VERRAL , M. S. *Software Bus. Architectures for Distributed Development Support Environments*. IEEE Colloquium on. Pages:4/1 - 4/3.

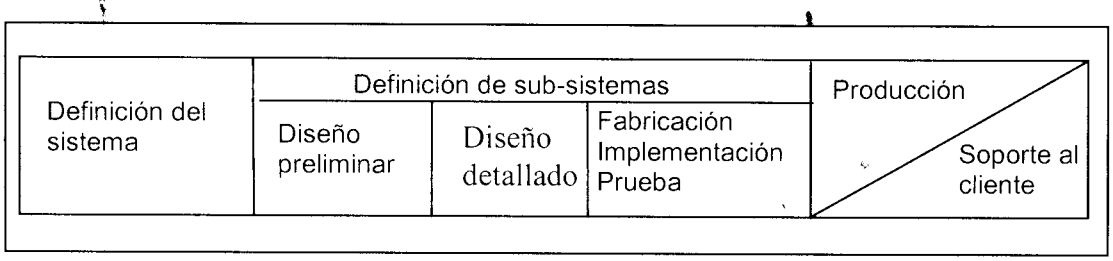
Anexos

Anexo 1: Descripción de servicios del Framework C4J

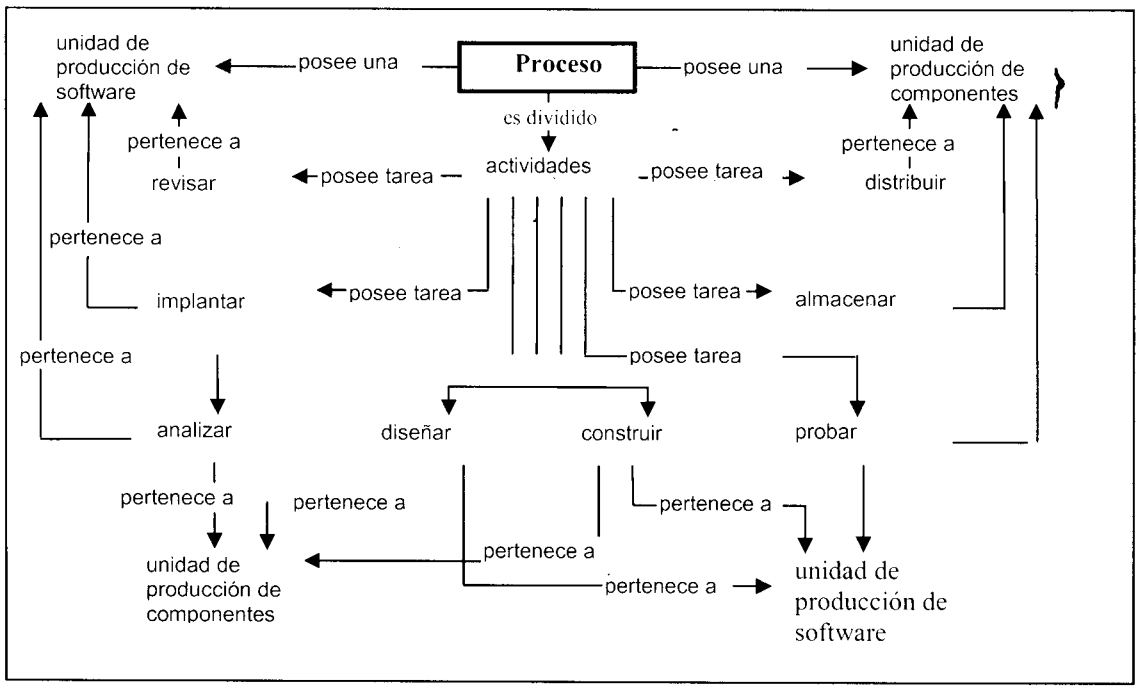
Servicios	Visión general
Servicio de Tags	Proporciona un conjunto de etiquetas personalizadas para usar en las páginas JSP. Entre las utilidades que ofrece tenemos: • Formularios y componentes con modo (una sola página para crear, editar, o consultar), de forma que se presenten editables o no dependiendo del perfil del usuario o de la acción a realizar. • Indicación de campos requeridos. • Conversión automática de datos. • Campos fecha con selección por calendario. • Independencia de codificación de las páginas si el portal es con o sin frames. • Indicación al usuario de que se está procesando la petición cuando ésta ha sido enviada al servidor. • Comprobación de posible pérdida de los cambios realizados en una pantalla si el usuario pulsa una opción que no realiza la grabación de los cambios e implica salir de esa pantalla. • Selección del valor de un textfield abriendo una nueva ventana de selección. • Ayuda asociada a los componentes. • Selects con valores especiales de selección ('<Todos>','<Seleccione un valor de la lista>')
Servicio de Multidioma	Permite la definición de aplicaciones internacionalizadas, consiguiendo como mayor beneficio el desarrollo de la aplicación sin importar que posteriormente se añadan nuevos idiomas a la aplicación.
Servicio de Pantallas	Permite la definición de layouts de presentación: layout con 4 partes (título, menú, cabecera, cuerpo y pie), layout de pestañas. Estos layouts permiten definir un look&feel genérico a toda la aplicación de forma que un cambio al look&feel de la aplicación no implique ningún cambio a las páginas jsp creadas, y solo requiera un cambio de configuración.
Servicio de Listados	Permite la definición de listados de resultados paginados, donde se permite al usuario navegación por página (adelante, atrás, primera, última) y acceso directo a número de página, ordenación ascendente / descendente por cada una de las columnas, y selección del número de filas a mostrar por cada página.
Servicio de Upload	Define cómo se puede obtener un fichero descargado por el usuario y su posterior tratamiento.

Servicio de Validación	Permite presentar al usuario errores de entrada de datos (campo obligatorio, campo entero, campo fecha, etc.) definiendo en un fichero qué tipo de validación realizar para cada formulario de la aplicación.
Servicio de Mailing	Permite el envío de correo electrónico, donde el cuerpo del mensaje puede ser un simple texto, una página HTML embebida o una serie de ficheros adjuntados.
Servicio de Alertas	Permite un mecanismo de suscripción, publicación por el cual, podemos avisar al usuario por pantalla de nuevos eventos que tengan lugar en el sistema.
Servicio OLE:Excel	Permite la lectura y tratamiento de ficheros Excel tanto leerlos, así como su creación y modificación.
Servicio de Generación de Reports	Permite la generación de informes PDF.
Servicio de Autentificación	Realiza la autentificación del usuario, actualmente contra LDAP.
Servicio de Autorización	Permite la entrada del usuario a determinadas URLs, así como los tipos de acceso permitidos en cada una de ellas. En conexión con el mecanismo de tags de presentación muestra la página en un modo de edición o consulta según estos tipos de acceso.
Servicio XML	Permite el tratamiento de ficheros XML.
Servicio de Acceso a Datos	Permite el acceso a una base de datos relacional mediante el uso de JDBC y SQL.
Servicio de Mapeo O-R	Permite el mapeo directo entre objetos y una base de datos relacional, sin que el usuario deba realizar ninguna consulta SQL en el código de la aplicación.
Servicio de Colas	Permite la utilización de colas JMS.
Servicio de ETL	Permite el tratamiento de ficheros ETL
Servicio de Ficheros	Define una API de tratamiento de ficheros (creación de ficheros, directorios, copia de ficheros).
Servicio Lotus Notes	Permite la integración con Lotus Notes.
Servicio de Planificación de Tares	Permite especificar tareas programadas que deben ejecutarse en un determinado momento (una hora, un día, cíclicamente).

Anexo 2: Actividades de la norma IEEE std 1220-1998



Anexo 3: Actividades y tareas de las unidades de producción de software y de componentes en el Modelo replicable.



Anexo 4: Diseño del Cuaderno del Ingeniero.

Se debe numerar cada página, para tener las páginas en orden y registro legal del trabajo. Se deben registrar las notas en orden cronológico y no se podrá insertar o eliminar páginas. En la portada se debe etiquetar el cuaderno con un número de cuaderno, para ordenarlos cronológicamente y referenciarlos al ser numerosos. Debe tener además nombre, número de teléfono, dirección de correo electrónico, fecha de comienzo de escritura de los datos y fecha del cierre. Por ejemplo:

Ejemplo de portada del Cuaderno de Ingeniero.

Cuaderno número: 1

Cuaderno de Ingeniería

Factoría de Software de la UCI

Nombre del Ingeniero: Juan Carlos Alonso Martinez

Teléfono/ correo electrónico: juan@uci.cu

Fecha de Apertura: 24/11/2005 **Fecha de Cierre:**

Dentro del cuaderno se deben utilizar las dos primeras páginas como índice de contenidos y cada página debe ser numerada.

Ejemplo del cuaderno contenido de ingeniería.

Contenido del cuaderno de ingeniería		1
Página	**Tema**	**Fechas**
3	Estudiar patrones	10-12/13
4	Leer especificación	12-13/10
5	Diseño del primer programa	13-17/10

Ejemplo de la página contenidos del cuaderno de ingeniería

Fecha	Descripción	3
10/9	Estudiar Arquitectura J2EE	
	Elaborar cuaderno de ingeniería	
13/9	Instalar y Configurar herramientas de desarrollo	
	Establecer arquitectura del subsistema	
15/9	Trazar la línea base de la arquitectura	

Anexo 5: Diseño del cuaderno de registro de tiempo.

El cuaderno en su cabecera debe suministrar los datos referentes al nombre del trabajador de la factoría ya sea estudiante o profesor, la fecha de inicio, y el nombre o número de clase. Cada período de tiempo se introduce una línea de la siguiente forma.

- Fecha. La fecha de realización de alguna actividad.
- Comienzo. La hora de comienzo de alguna actividad.
- Fin. La hora en que terminas de hacer una actividad.
- Interrupción. Cualquier pérdida de tiempo, debido a interrupciones.
- Δ Tiempo. El tiempo dedicado a cada actividad en minutos.
- Actividad. Nombre descriptivo para la actividad.
- Comentarios. Una descripción mas completa de lo que se está haciendo.
- C (Completado). Rellena esta columna cuando termines una tarea.
- U (Unidades). El número de unidades de una tarea acabada.

A continuación se puede observar un ejemplo:

Estudiante: _____ Fecha: _____

Profesor: _____

Ejemplo de cuaderno de registro de tiempo.

Fecha	Inicio	Fin	Tiempo I.	Δ Tiempo	Actividad	Comentarios	C	U

Anexo 6: Diseño del cuaderno resumen semanal de actividades.

El cuaderno en su cabecera debe tener el nombre del trabajador y el rango de fecha de la semana. Dentro de las actividades se debe encontrar el número secuencial de cada una de ellas, su clasificación, el Identificador de requerimiento, y la descripción de la actividad. Además de la definición de las horas usadas en los días de la semana. Finalmente los totales de tiempos por actividad y los totales de tiempo trabajados diariamente.

El aspecto Clasificación (CL), se identificará por un número que referencia el tipo de agrupación de la tabla de registro de tiempo: 1. Vacaciones, 2.Incapacidad, 3.Diseño, 4.Implementación, 5.Pruebas, 6. Problemas, 7. Comunicación, 8.Soporte, 9. Capacitación, 10. Reuniones e 11.Interrupciones.

Nombre: _____ Fecha: _____

Ejemplo del resumen semanal de actividades.

Actividades				Número de Horas							
No.	Cl.	Id. Req.	Descripción	L	Ma	Mi	J	V	S	D	Total
1	9	01001	Lectura de especificaciones del componente 1	2		1					3
2	9	02001	Lectura de especificaciones del componente 2		1	1	2				4
3	9	03001	Lectura de especificaciones del componente 3		3						3
4	3	03001	Diseño del Componente 1	1							1
5	10	03001	Trabajo en grupo de definición	2							2
6	9	03001	Estudio de los métodos de mapeos de OJB.	3	3	3					9
7	7	03001	Sincronización con el control de versiones internacional	2							2
8	3	03001	Trabajar nuevamente en el diseño del componente 1			5					5
9	7	03001	Revisión con el cliente del diseño		2						2
10	4	03001	Generar base de datos a partir del modelo de datos.	1	1	1	1				4
11	9		Estudiar acceso a datos por JDBC según la especificaron del Framework.				5				5
			Total de Horas	11	10	11	8	0			**40**

Anexo 7: Diseño del Cuaderno de trabajo.

1. El cuaderno de trabajo en su cabeceras debe poseer datos sobre nombre del trabajador de la factoría y fecha de apertura del cuaderno. Debe poseer aspectos tales como:
2. Trabajo (Tr): Numeración de las trabajos a desarrollar.
3. Fecha (Fe): Fecha del comienzo del trabajo.
4. Proceso(Pr): Descripción de la tarea.
5. Tiempo Estimado(TE): Tiempo que crees que gastaras al desarrollar esa tarea.
6. Unidades Estimadas(UE): Numero de unidades que se han desarrollado en dependencia del caso.
7. Tiempo Real(TR): Tiempo dedicado al trabajo terminado.
8. Unidades Reales(UR): Registro de las unidades reales al terminar el trabajo.

9. Velocidad Real(VR): Se obtiene mediante el tiempo Real Dividido por las Unidades Reales.
10. Tiempo hasta la fecha(TFE): Es la suma de todos los tiempos correspondientes a tareas de un mismo proceso, hasta una fecha dada.
11. Unidades hasta la fecha(UFE): Es la suma de todas las unidades correspondientes a tareas de un mismo proceso, hasta una fecha dada.
12. Velocidad hasta la fecha(VFE): Es el tiempo hasta la fecha dividido por las unidades hasta la fecha.
13. Máximo hasta la fecha dada (Max): Para encontrar la velocidad máxima para cualquier tarea, compara la velocidad real de los trabajos más recientes con el máximo hasta la fecha de los trabajos previos del mismo tipo y anota el valor.
14. Mínimo hasta la fecha dada (Min.): El valor mínimo hasta la fecha es la velocidad mínima para cualquier tarea de un tipo dado. Para determinar este valor, compara la Velocidad Real del trabajo con el mínimo hasta la fecha de los trabajos mas recientes de ese tipo y escribe el valor menor.

Nombre: _____ Fecha: _____

Ejemplo del cuaderno de trabajo.

Tr	Fe	Pr	Estimado		Real			Hasta la Fecha				
			T	U	T	U	V	T	U	V	Max.	Min.
1	9/9	Cod	100	1	158	1	158	158	1	158	158	158
	Descripción: Escribir el programa 1(minutos por programas)											
2	9/9	Texto	50	2	80	2	40	80	2	40	40	40
	Descripción: Leer los capítulos 1 y 2 del libro de texto.(minutos por capítulo)											
3	11/9	Cod	158	1	69	1	69	227	2	113,5	158	69
	Descripción: Escribir el programa 2.											
4	11/9	Texto	40	1	28	1	28	108	3	36	40	28
	Descripción: Leer capitulo 3 del libro de texto											
5	12/9	Cod	114	1	114	1	114	341	3	113.3	158	69
	Descripción: Escribir programa 3()											
6	13/9	Texto	60	1	118	1	128	226	4	56.5	118	28
	Descripción: Leer capitulo 4 del libro de texto											
7	16/9	Cod	114	1	93	1	93	434	4	108.5	158	69
	Descripción: Escribir programa 4											
8	17/9	Cod	109	1	95	1	95	529	5	105.8	158	69
	Descripción: Escribir programa 5											
9	18/9	Texto	57	1	71	1	71	297	5	59.4	118	28
	Descripción: Escribir programa 5											

Anexo 8: Lista de Comprobación

Requisitos Mínimos:

1. En el cierre de versión, ¿las etiquetas de los proyectos de la aplicación siguen la nomenclatura o al menos tienen un formato parecido? 2.7
2. ¿Se ha recibido el Manual de Instalación?
3. ¿Se genera el EAR correctamente?
4. ¿Se evita la utilización de ficheros de propiedades fuera de la aplicación y por consiguiente del EAR? 2.2
5. ¿Se encuentran fuera de los proyectos de la aplicación las librerías correspondientes a drivers de bb.dd, librerías MQSeries, librerías de terceros como JIntegra, Log4J.

Recepción:

1. ¿Se ha recibido el Código Fuente correctamente versionado?
2. ¿Se ha recibido el Javadoc de la aplicación?
3. ¿Se ha recibido el código fuente correctamente comentado sobre las modificaciones realizadas?
4. ¿Se ha recibido el Diseño Técnico compuesto por diagramas UML (diagramas de clases y de
5. secuencia) elaborado con la herramienta Rational XDE?
6. ¿Se ha recibido el Diseño Técnico en formato original de mantenimientos evolutivos?
7. ¿Se ha recibido el Modelo de Datos elaborado con la herramienta PowerDesigner?
8. ¿Se ha recibido documentación técnica escrita sobre componentes adicionales?
9. ¿Se ha recibido el Manual de Instalación?
10. ¿Se ha recibido el Manual de Explotación si hay procesos batch?
11. ¿Se han recibido la documentación de los accesos de BBDD (Online + Procesos Batch)?
12. ¿Se ha recibido la Guía Rápida paso Soporte?
13. ¿Se han recibido los scripts de Autorización?
14. ¿Se han recibido los scripts de BBDD?
15. Otras Incidencias de Recepción.

Configuración Instalable Aplicaciones Java:

1. ¿Se cumple la estructura de proyecto exigida para el Proyecto Documentación de la aplicación?
2. ¿Se cumple la estructura de proyecto exigida para el Proyecto EJB de la aplicación (si existe)?
3. ¿Se cumple la estructura de proyecto exigida para el Proyecto Java de la aplicación (si existe)?
4. ¿Se cumple la estructura de proyecto exigida para el Proyecto Web de la aplicación?
5. ¿Se evita la utilización ficheros de propiedades fuera de la aplicación y por consiguiente del ear?
6. ¿Se encuentran en el proyecto EJB los EJB's (Interfaces local y remota, bean, y código desplegado) y clases java utilizadas exclusivamente en este proyecto?
7. ¿Se encuentran en el proyecto Web, dentro de la carpeta Source, las clases propias de una aplicación web (servlets, etc.) y clases java utilizadas exclusivamente en este proyecto?
8. ¿Se encuentran en el proyecto Web, dentro la carpeta WebApplication, el resto de recursos web de la aplicación (jsp, js, css, jpg,) en sus carpetas correspondientes?
9. ¿Se encuentran en el proyecto Web, dentro del subdirectorio WEB-INF/lib, todas las librerías que se emplean únicamente en este proyecto?.(2.3)
10. ¿Se encuentran en el proyecto Web, dentro del subdirectorio WEB-INF/classes, todos los ficheros de recursos (.properties, .xml, etc.) relativos a la aplicación?
11. ¿Se encuentran en el proyecto Java las clases de utilidad específicas de la aplicación, utilizadas tanto por el módulo WEB como por el módulo EJB?
12. ¿Se encuentran en el proyecto Batch las clases específicas para los procesos batch (si existen)?
13. ¿Se encuentran fuera de los proyectos de la aplicación las librerías correspondientes a drivers de bb.dd, librerías MQSeries, librerías de terceros como JIntegra, Log4J,...?
14. ¿Se encuentra dentro del proyecto de Documentación una carpeta donde se guardan los scripts de la aplicación (autorización y bb.dd.)?
15. ¿Se encuentran los ficheros de configuración (susceptibles de ser modificados por
16. administración) por triplicado y adaptados para los entornos de Desarrollo, Preproducción y Producción?
17. ¿Se encuentran los ficheros de configuración (susceptibles de ser modificados por

18. administración) documentados en el Manual de Instalación? (2.8)

Convenciones de código y de Diseño J2EE:

1. ¿Se utilizan convenciones de javadoc para documentar clases y metodos?
2. ¿Tienen los ficheros menos de 2000 líneas?
3. ¿Siguen los nombres de los directorios la jerarquía "Estructura paquetes" (com.carrefour.general/com.carrefour.aplicacion)?
4. ¿Los nombres de las clases están compuestos por una o más palabras enteras, estando la primera letra de cada palabra en mayúsculas y el resto en minúsculas?
5. ¿Las clases que heredan de una interfaz siguen la misma nomenclatura que las clases más el sufijo "Impl"?
6. ¿Los nombres de paquetes están generados en minúscula?
7. ¿Los métodos son verbos donde la primera palabra está en minúsculas y la segunda tiene la primera letra en mayúsculas?
8. "¿Los parámetros que no son de tipo booleano de los métodos comienzan con los prefijos "a" o "an""? ¿Comienzan por "b" los parámetros de tipo booleano? "
9. En las variables, ¿está la primera palabra en minúsculas y la siguiente tiene la primera letra en mayúscula?
10. ¿Están escritas las constantes en mayúsculas y contienen "_" como separador entre palabras?
11. ¿Se mantienen los formatos para las clases establecidos en la Normativa J2EE?
12. ¿Tienen las clases la nomenclatura xxxTipoClase (xxxAction, xxxForm, xxBO, xxxDAO, xxVO)?
13. ¿Se evita el uso de import con asteriscos?
14. ¿Se evita hacer referencia en algún momento al perfil en concreto?
15. ¿Tienen las variables nombres identificativos?
16. ¿Se evitan las abreviaturas en los nombres de los identificadores?
17. Para páginas JSP, ¿se usa la nomenclatura que diferencia listados (listxxx.jsp) de
18. mantenimientos (mantxxx.jsp)?
19. ¿Se evitan constantes como literales?
20. ¿Se usan ficheros .properties para configuración e internacionalizacion?
21. ¿Se usan ficheros .properties para configuración e internacionalizacion y sino en su defecto ficheros .xml?
22. ¿Se usan ficheros .properties para constantes y sino en su defecto ficheros .xml, interfaces o en ultimo caso clases no heredables?

23. ¿Se utiliza la cláusula final en constantes y en métodos y clases que no se quiere que sean heredados?
24. ¿Se asegura la liberalización de recursos mediante el método finalize(), en clases que mantengan referencia a escuchadores de eventos?
25. Con objeto de ayudar al garbage collector a la liberación de memoria, ¿se asigna referencias a null a un objeto al finalizar su ámbito, al menos en objetos de tipo array?
26. ¿Las clases ActionForm almacenan todos los parámetros del formulario en String, String [] y boolean?
27. ¿Se accede a los parámetros del formulario en las clases Action vía ActionForm y se devuelven en un objeto ActionForward?
28. ¿Se utilizan librerías de etiquetas de struts (en el caso del Framework, las de este)?
29. ¿Se minimiza el uso de scriptles? (Solo se utiliza cuando sea justificado su uso)
30. ¿Son los DAO los que encapsulan toda la lógica de acceso a datos?
31. ¿Se evita el uso de ActionForms en la lógica de negocio?
32. ¿Se utilizan los objetos ActionError y ActionMessages para encapsular los mensajes devueltos por el método validate del ActionForm?
33. ¿Se ejecutan validaciones de entradas de usuarios en el lado servidor a través del servicio de validación Struts Validator?
34. ¿Se delega la lógica de negocio en los BO, no utilizando los Action para este fin?
35. ¿Se utiliza DispatchAction para las acciones del Action?
36. ¿Se llama directamente al componente Autenticación y Autorización (con Framework no es aplicable)?
37. ¿Se mantiene intacto el componente de Autenticación y Autorización?
38. ¿Los Business Objects encapsulan los casos de uso ofrecidos por la aplicación?
39. ¿Se utiliza las clases Value Object para encapsular los datos de lógica de negocio?
40. ¿Se ejecuta el método commit() para finalizar las transacciones y rollback en caso de error?
41. Las transacciones básicas, ¿Se implementan en el Bussines Object y no en los DAO?
42. En transacciones básicas, ¿se minimiza el número de operaciones para una transaccion?

43. En transacciones básicas, ¿se invoca en la cláusula finally el método close() del objeto Connection para asegurar que la conexión a Base de Datos se cierra y así liberar recursos?
44. En transacciones distribuidas, ¿se controla desde los métodos del BO las transacciones?
45. "En los EJB Stateless el nivel de aislamiento, ¿es ""repeateable read"" en métodos de actualización?
46. En los EJB Stateless el nivel de aislamiento, ¿es ""read commited"" en métodos de consulta?
47. ¿Los DAO son los que abstraen toda la complejidad de guardar, actualizar, recuperar y eliminar datos de un repositorio?
48. ¿No incluye lógica de negocio el DAO?
49. ¿Se cumple que en ningún caso el DAO sea responsable de la gestión de conexiones a la base de datos?
50. Por cada Base de Datos, ¿existe un fichero properties donde se encuentran los códigos de mensajes mas comunes de una Base de Datos?
51. ¿Se hace uso de "PreparedStatement" en todas las ejecuciones de sentencias SQL con parámetros?
52. En la composición de sentencias SQL, ¿se concatenan sus elementos mediante "StringBuffer" ?
53. ¿Se evitan los EJBs de entidad? (No implementan el interface EntityBean)

Manual de Explotación:

1. ¿Se utiliza la plantilla creada para este tipo de documento proporcionada por la Oficina de Proyectos?
2. ¿Existen cuadros para la identificación, el cambio, la distribución y el control del documento?
3. ¿Están debidamente completados?
4. ¿Se incluye una Introducción compuesta por Propósito, Glosario y Audiencia?
5. ¿Está actualizada la última versión del documento?
6. ¿Está disponible y accesible (en CVS) la última versión del documento a las personas implicadas?
7. ¿Se indica en Datos Generales el nombre de la aplicación, equipo de ejecución, el peticionario y el autorizador?

8. ¿Se indica en Ejecución y Características una descripción breve de los procesos, ruta y versión de cada proceso, duración estimada de los procesos, librerías externas utilizadas y usuario de ejecución de cada proceso?
9. ¿Se indica en Planificación e Incompatibilidades la periodicidad, las dependencias y/o horas de ejecución y las incompatibilidades?
10. ¿Se indica un Organigrama de ejecución de tareas?
11. ¿Se indica en Relación de Recursos los Archivos de entrada/salida, los archivos de configuración, las tablas de b.d., los archivos de log y el backup de archivos?
12. ¿Se indica en Contingencias los posibles códigos de error, las acciones a realizar por cada caso de error y los impactos sobre siguientes ejecuciones si no se soluciona el error?
13. ¿Se indican los datos correspondientes al Alta de Cuenta FTP si procede?
14. ¿Se utiliza la plantilla creada para este tipo de documento proporcionada por la Oficina de Proyectos?
15. ¿Existen cuadros para la identificación, el cambio, la distribución y el control del documento?
16. ¿Están debidamente completados?
17. ¿Se incluye una Introducción compuesta por Propósito, Glosario y Audiencia?
18. ¿Está actualizada la última versión del documento?
19. ¿Está disponible y accesible la última versión del documento a las personas implicadas?
20. ¿Es correcta la URL de acceso al sistema principal?
21. ¿Es correcta la versión a desplegar?
22. Si es una instalación inicial ¿están declarados los EJB's, métodos y nombres JNDI de la aplicación?
23. Si no es una instalación inicial ¿están declarados los EJB's, métodos y nombres JNDI modificados?
24. ¿Son correctos los ficheros de configuración críticos y los path relativos?

Guía rápida paso a soporte:

1. ¿Se utiliza la plantilla creada para este tipo de documento proporcionada por la Oficina de Proyectos?
2. ¿Existen cuadros para la identificación, el cambio, la distribución y el control del documento?

3. ¿Están debidamente completados?
4. ¿Se incluye una Introducción compuesta por Propósito, Glosario y Audiencia?
5. ¿Está actualizada la última versión del documento?
6. ¿Está disponible y accesible (en la base de datos) la última versión del documento a las personas implicadas?
7. ¿Existen los datos generales del proyecto o módulo?
8. ¿Está incluido un directorio de personas relacionadas con el sistema?
9. ¿Se indica el nombre, la función y el nombre del documento técnico relativo a cada proceso online?
10. ¿Se indica el nombre, la función y el nombre del documento técnico relativo a cada proceso batch?
11. ¿Están descritos la información, los procedimientos y protocolos de seguridad para el acceso al sistema?
12. ¿Se indican las características de la monitorización y las alertas?
13. ¿Se incluye el protocolo de actuación ante una crisis?
14. ¿Existe observaciones adicionales?
15. ¿Se incluyen la vista, el título y el documento técnico de las pantallas?

Diseño Arquitectura Lógica Genérico:

1. Si por causa de fuerza mayor se han empleado applets para esta capa, ¿está justificado su uso?
2. ¿Se utiliza un browser convencional como cliente?
3. Si la aplicación es para intranet,¿está asegurada la compatibilidad con versiones de Internet Explorer igual o superior a 5.5?
4. Si la aplicación tiene salida a internet, ¿es compatible con exploradores como MS Internet Explorer y Netscape?
5. ¿Se emplean Struts y un unico servlet como para la presentación?
6. Capa Presentación. Se emplean en esta capa únicamente JSPs, Servlets, clases y ficheros de propiedades?
7. ¿Se usan estilos CSS para la estandarización del diseño de la aplicación?

Capa Presentación :

8. Para la apertura de ventanas de diálogo, ¿se utiliza window.showModalDialog para asegurar el funcionamiento dentro de un terminal server?
9. Se Incluye <base target="_self"/> en el <head> en la página que se muestra en modo modal para evitar la navegación?
10. Capa Presentación (2.3) (NO APLICAR EN NINGÚN CASO, pendiente de borrar)
11. Se usan las tablas sólo para mostrar tablas de datos?
12. Las propiedades de presentación de los componentes se declararan mediante clases CSS (class="nombreClase"), o bien mediante el atributo style (style="width:20px").
13. ¿Se definen clases CSS comunes a toda la aplicación de forma que la
14. clases título, tabla, etiqueta, etc.. sean comunes por aplicación?
15. Se definen las clases con un nombre autoidentificativo?
16. Se utiliza Jasper Reports para la exportación de ficheros a formato PDF?
17. Se utiliza iReport para el diseño de las plantillas?
18. ¿Se emplea el API POI de Jakarta para la exportación de ficheros a
19. formato Excel?

Capa Lógica de Negocio :

20. Se utiliza un EJB Stateful para almacenar los datos de sesión del usuario si el tamaño de la información de sesión es mayor de 10kb? (Comprobar que no se abusa del método HTTPSession.putValue())
21. Se evitan las llamadas a B.D. desde los Action?
22. Se implementa el patrón DAO para encapsular toda la lógica de acceso a datos?
23. Se implementa la lógica de Negocio como Business Objects (patrón BO)?
24. Para la lógica de negocio propia de la aplicación, ¿se utilizan EJB Stateless para los procesos que requieran transaccionalidad con la base de datos, y Clases java para el resto de la lógica de negocio?
25. ¿Se implementa el patrón 'Service Locator' para acceder a los EJBs?
26. ¿Se utilizan el patrón 'Business Delegate' como intermediario entre la capa de Presentación y la capa de EJBs?
27. ¿Se utiliza Repeateable Read para métodos de actualización y
28. Read Commited para metodos de consulta?

Capa de Integración :

29. Para el intercambio de información con el resto de aplicaciones, ¿se utiliza un Middleware orientado a mensajes (MOM), implementada con Websphere MQ?
30. ¿La entrada y salida de ficheros con MQ se realiza a través de SMTF?
31. ¿Se utiliza el estándar JMS para el envío de mensajes de texto?

Capa de Conectividad :

32. ¿Se hace el acceso a BBDD a través de JDBC?
33. ¿Es independiente el acceso a BBDD de la implementación de JDBC que se utilize (Oracle,
34. Informix,...)?

Capa de Persistencia :

35. ¿Se utiliza una única clase encargada de gestionar las conexiones con la base de datos y por la que se accede a través de un pool de conexiones?
36. La persistencia se realiza con JDBC, no utilizando Beans de Entidad para conseguir dicha persistencia?

Capa de Seguridad :

37. ¿Se realiza la autenticación con el componente AuthenticatorClient (SecurityDelegate)?
38. ¿Se realiza la autorización con el componente AutorizacionDelegate?
39. La gestión de perfiles y usuarios se gestiona de forma centralizada por el departamento de seguridad con una aplicación destinada a tal fin?
40. La aplicación java se alimenta de esta información a través del componente
41. AutorizacionDelegate para ver si un usuario tiene o no acceso a determinadas opciones de la aplicación?

Ficheros de configuración :

42. Están todos los ficheros de configuración en el subdirectorio ProjectoWeb/WEBINF/classes(*.properties, *.xml,...)?
43. ¿Se evitan las referencias absolutas a los ficheros de Configuración?
44. ¿Se obtienen los ficheros de configuración con extensión *.properties mediante
45. ResourceBundle.getBundle(String,Locale) ?
46. Se obtienen los ficheros de configuración con otra extensión que no sea *.properties (xml,...) mediante Thread.currentThread().getContextClassloader().getResourceAsStream(?nombrefichero.extension?)?

Internacionalización :

47. ¿Se utiliza el API estándar i18n para la internacionalización?
48. ¿Están internacionalizados todos los literales que se muestran al usuario?
49. ¿Están internacionalizadas las trazas de la aplicación?
50. ¿Existe un fichero de recursos para cada idioma?
51. Diccionario de Datos
52. ¿Esta definido el diccionario de datos donde se deberán definir todos los recursos de la b.d. (campos y tablas) en un fichero de propiedades?

Diseño Arquitectura Lógica Genérico Framework

Capa Cliente :

1. Si por causa de fuerza mayor se han empleado applets para esta capa, ¿está justificado su uso?
2. ¿Se utiliza un browser convencional como cliente?
3. Si la aplicación es para intranet, ¿está asegurada la compatibilidad con versiones de Internet Explorer igual o superior a 5.5?

Capa Presentación:

4. Para la apertura de ventanas de diálogo, ¿se utiliza window.showModalDialog para asegurar el funcionamiento dentro de un terminal server?
5. ¿Se Incluye <base target="_self"/> en el <head> en la páginas de diálogo para evitar la navegación?
6. ¿Se utiliza Jasper Reports para la exportación de ficheros a formato PDF?
7. ¿Se utiliza iReport para el diseño de las plantillas?

Capa Lógica de Negocio:

8. Para la lógica de negocio propia de la aplicación, ¿se emplean componentes de los siguientes tipos: Value Objects (VO), Business Objects (BO) y EJB Session Stateless, siguiendo las pautas dadas por la documentación de C4J?

Capa de Integración :

9. Para el intercambio de información con el resto de aplicaciones, ¿se utiliza un Middleware orientado a mensajes(MOM), implementada con Websphere MQ?
10. ¿La entrada y salida de ficheros con MQ se realiza a través de SMTF?
11. ¿Se utiliza el estándar JMS para el envío de mensajes de texto, empleando el servicio proporcionado por C4J, Open JMS?

Capa de Conectividad:

12. ¿Se hace el acceso a BBDD a través de JDBC, empleando el driver Java proporcionado por
13. cada RDBMS?

Capa de Persistencia:

14. ¿Se emplea el servicio de mapeo objeto-relacional incluido en C4J, ObJect relational Bridge (OJB)?

Capa de Seguridad:

15. ¿La gestión de perfiles y usuarios se gestiona de forma centralizada por el departamento de seguridad con una aplicación destinada a tal fin?
16. ¿La aplicación java se alimenta de esta información a través del servicio de autenticación y autorización integrado en C4J (Servicio Seguridad) para ver si un usuario tiene o no acceso a determinadas opciones de la aplicación?

Pruebas Básicas:

1. ¿Funciona correctamente la entrada en la aplicación introduciendo el nombre de usuario, la contraseña y pulsando en el botón Entrar?
2. ¿Se impide la entrada en la aplicación cuando se introduce un usuario que no tiene permiso de acceso para esa aplicación?
3. ¿Se impide la entrada a la aplicación al introducir una contraseña incorrecta?
4. ¿Aparece un mensaje de error descriptivo al introducir un usuario o una contraseña incorrecta?
5. ¿Funciona correctamente la navegación por los diferentes módulos de la aplicación?
6. ¿Funciona correctamente la tabulación (de izquierda a derecha y de arriba hacia abajo) entre los campos de la pantalla? ¿Funciona correctamente la tabulación inversa?
7. ¿Son coherentes los tamaños de los campos en la pantalla con el tamaño de los datos que se muestran en ellos?
8. ¿Los controles indican que tienen el cursor mostrando una caja punteada?
9. Al entrar en un campo de texto, ¿se activa todo el texto contenido en el?
10. ¿Los campos deshabilitados no pueden tomar el cursor ni con el ratón ni por tabulación?
11. ¿Están ordenados alfabéticamente los valores de las listas de selección?
12. ¿Se produce un mensaje de error al introducir en un campo caracteres inválidos?
13. ¿Se produce un mensaje de error al introducir en un campo un formato inválido?
14. ¿Se produce un mensaje de error al introducir en un campo demasiados caracteres?

15. ¿Se produce un mensaje de error al no rellenar un campo obligatorio?
16. ¿Funciona correctamente la opción añadir?
17. ¿Funciona correctamente la opción buscar?
18. ¿Funciona correctamente la opción modificar?
19. ¿Funciona correctamente la opción borrar?
20. ¿Expira la sesión transcurrido un cierto tiempo?
21. Si existe, ¿Funciona correctamente la opción Imprimir?
22. Si existe, ¿Funciona correctamente la opción Exportar a Excel?
23. Si existe, ¿Funciona correctamente la opción Exportar a PDF?
24. Si existe, ¿Funciona correctamente la opción Ayuda?
25. Si existe, ¿Funciona correctamente la opción Generar Gráfico?
26. ¿Funciona correctamente la opción Salir?
27. Si la aplicación es para Internet, ¿se puede navegar en ella utilizando Netscape?
28. ¿Se evitan las faltas de ortografía, incluidos los acentos?

Inspección Guía de Estilo:

1. En la pantalla de acceso a la aplicación, ¿aparece el logotipo de Carrefour, el fondo de pantalla establecido?
2. ¿Aparece el nombre de la aplicación en la pantalla de acceso a la aplicación?
3. ¿La estructura de las ventanas consta de: cabecera, menú principal, menú secundario, área de trabajo y área lateral?
4. ¿Funciona correctamente la secuencia de funcionamiento de la aplicación: la selección de una opción principal, presentación del segundo nivel de opciones, selección de un último nivel dentro del menú secundario.
5. ¿La cabecera de la aplicación consta de: logotipo de Carrefour, nombre de la aplicación, menú auxiliar e identificador de sesión?
6. ¿Aparecen las opciones de primer nivel de la aplicación en el menú principal? ¿Aparecen (si es necesario) en la titulación del área de trabajo el menú de informes mediante iconos?
7. ¿Aparece en el extremo derecho del menú principal el scroll de menú? (OBLIGATORIO)
8. ¿Está conectada la opción activa del menú principal con el menú secundario?
9. ¿Se puede ocultar/mostrar el menú secundario?
10. ¿Aparecen las opciones del menú secundario agrupadas en diferentes niveles desplegables?

11. ¿Se marca en negrita la selección final de tarea?
12. ¿Está conectada la selección de una tarea final del menú secundario con el Área de trabajo resultante?
13. ¿Aparece el título de pantalla dentro del área de trabajo y su ruta dentro de la jerarquía del menú? ¿Aparece el título en negrita?
14. En caso de procesos secuenciales, ¿aparece el indicador de proceso?
15. La botonera del área de trabajo, ¿contiene botones de acción que afectan a toda la pantalla?
16. En caso de existir faldón y que ocupe toda la pantalla, ¿está éste al mismo nivel que la botonera del área de trabajo?
17. En caso de coexistir los siguientes tres tipos de botones, ¿aparecen los botones de vuelta a la izquierda, los botones de todo el proceso en el centro, y los botones de avance a la derecha?
18. ¿Se puede ocultar /mostrar el módulo Búsqueda – Filtro?
19. En caso de existir asistentes de búsqueda, ¿funcionan correctamente lanzando una pantalla modal?
20. ¿Aparecen asistentes Calendario en todos los campos fecha? ¿Funcionan estos asistentes correctamente?
21. ¿Funcionan correctamente las listas de selección del filtro de búsquedas?
22. ¿Están incluidos los botones de acción Buscar y Limpiar, situados en faldón del módulo Búsqueda – Filtro? ¿Es correcto su funcionamiento?
23. ¿Funciona correctamente la ordenación ascendente y descendente por columnas de la tabla?
24. ¿Se permite plegar/colapsar columnas en la tabla?
25. ¿Existe para las líneas de la tabla la posibilidad de selección de registro mediante una casilla de verificación?
26. ¿Funciona correctamente el desplazamiento del cursor entre registros a través los elementos de la barra de Paginación y número de registros?
27. ¿Aparece el contador de registros en la barra de Paginación y número de registros?
28. ¿Se expande la tabla al cerrar alguno de los módulos plegables de la pantalla?
29. ¿Funciona el scroll interno de la tabla?
30. ¿Funciona correctamente la selección de registros junto a los botones de modificación y borrado?¿Se imposibilita la ocultación de los módulos de captura de datos?

31. ¿Están marcados con un asterisco rojo los campos obligatorios en la captura de datos?¿Funciona correctamente el asistente buscar en los módulos de captura de datos?
32. ¿Tiene el asistente calendario el formato normalizado de fechas para aplicaciones Carrefour?

Desarrollo Seguro con J2EE:

1. La información confidencial (usuarios, passwords, URLs privada,...), ¿se encuentra fuera del código fuente?
2. Siempre que la lógica del programa lo permita ¿los métodos y los atributos de las clases se declaran como 'private'?
3. Las clases que implementan el interface 'Cloneable' ¿declaran su método 'clone()' como 'final'?
4. Para acceder a los atributos de una clase ¿se proporcionan métodos de acceso 'setxxx' o 'getxxx', donde xxx es el nombre del atributo al cual se accede evitando así definir los atributos como 'public'?
5. ¿Se han eliminado de la aplicación, o no existen, clases, métodos, o atributos que se hayan dejado de utilizar o no se hayan utilizado nunca?
6. ¿Se evita el uso masivo de variables estáticas 'static'?
7. Para comprobar si dos referencias están apuntando a una misma instancia ¿se utiliza el método 'getClass()' sobre los dos objetos y se comparan con el operador de comparación '= ='?
8. ¿Se inicializan todos los atributos de una clase en el constructor?
9. ¿Las variables locales a método se inicializan en su declaración?
10. ¿Se evita en la declaración de un método el uso de 'Metodo() throws Exception' lanzando en su lugar una envoltura especifca o la excepción original?
11. En caso de usar 'Runtime.exec()' ¿está totalmente justificado su uso?
12. ¿Se declara el "conjunto de caracteres" (cabecera "Content-Type") en los servlets y JSPs?
13. ¿Se utiliza en número mínimo de servlets o JSP's como puntos de acceso a la aplicación?
14. El servlet o JSP que sirva como punto de entrada a la aplicación ¿es lo mas ligero posible y delega el resto de operaciones al resto de clases de la aplicación WEB?

15. Si no se puede utilizar la infraestructura montada para el control de acceso y se tiene que hacer con servlet o JSP ¿se siguen todas las medidas descritas en la política de seguridad que son: id de usuario, password, rol de usuario, grupo al que pertenece
16. ¿La información de credenciales esta fuera del código fuente de servlets o JSP's?
17. En las aplicaciones Web que transmiten información confidencial, ¿se utiliza un protoclo seguro (SSL o TLS)?
18. ¿Se evita la inserción de datos de carácter sensible en las cookies de sesión? ¿Se capturan y controlan todas las excepciones que se puedan producir en un servlet o JSP?
19. ¿Se han eliminado todos los comandos de depuración en los servlets y JSPs?
20. ¿Todos los comentarios de las JSP's son implementados utilizando el elemento '<%--comentario --%>' en vez de '<! - - comentario - ->'?

Normativa i18n (Internacionalización):

1. En la pantalla de login de la aplicación, junto a los campos de usuario y password, ¿se presenta un menú desplegable con los idiomas soportados para que el usuario realice la selección del idioma con el que desea conectarse?
2. En la página de inicio de la aplicación ¿se muestra el idioma por defecto del navegador?
3. Una vez seleccionado el idioma e iniciada la sesión en la aplicación, ¿los contenidos de la misma se presentan en dicho idioma a lo largo de la misma sesión? (Probar con dos idiomas por lo menos).
4. Aplicaciones construidas sin struts, para determinar el locale del cliente ¿se utiliza la interface ServletRequest a través de los métodos getLocale() o getLocales() ?
5. Cuándo se utiliza ResourceBundle.getBundle("clase base",Locale) ¿se proporciona siempre la clase base sin sufijos? Con esto se consigue evitar que se lance una MissingResourceException.
6. ¿Se evita el almacenamiento de objetos String en clases de tipo ListResourceBundle (Objetos)?
7. Los objetos String se deben almacenar en clases de tipo PropertyResourceBoundle (Ficheros de
8. propiedades)
9. ¿Existe un fichero de propiedades por defecto denominado ""nombre base".properties" al cual

10. no se le concatenará ningún sufijo relacionado con el código del lenguaje y/o país?¿Existe un fichero de propiedades que contenga valores traducidos por cada Locale soportado con formato "nombre base"_códigoIdioma.properties evitando el sufijo del código del país? Ej: aplicación_en.properties - Inglés (2.2.4.3) (4.4)
11. ¿Los ficheros de propiedade se encuentran dentro del módulo web en el directorio WEBINF/classes? (2.2.4.8)
12. Si existen ficheros de clase (clases compiladas con idiomas por defecto) y ficheros de propiedades asociados a estas clases, ¿Se encuentran dentro del paquete com.carrefour.[códigoaplicación].resources dentro del módulo web? Ubicación: WEB-INF/classes/co
13. Aplicaciones construidas con struts. ¿Se accede al Locale a través del método getLocale() definido en la clase Action de Struts?
14. Aplicaciones construidas con struts, para acceder desde la capa de presentación (jsps) a los recursos de idioma ¿se utiliza el tab message? Ej: <title><bean:message key=global.title/></title>
15. Internacionalización Modelo de datos. En las tablas con campos susceptibles de ser internacionalizados ¿Existe una tabla auxiliar cuyo nombre tendrá como raiz el nombre de la tabla original y como sufijo el témino i18n?
16. Internacionalización Modelo de datos. ¿Existe una tabla Locale con dos campos (Identificador de Locale, Descripción de Locale) que contenga los idiomas soportados por la aplicación?
17. Internacionalizacion Modelo de datos. ¿Los idiomas soportados por la aplicación están reflejados en la tabla de Locale del modelo de datos?

Anexo 9: Instalación y Configuración de los Plugins.

Concretamente se utilizarán dos plugins: Checkclipse y PMD Utiles. Este apartado describe la metodología de instalación.

6.1- CHECKCLIPSE

6.1.1-El plugin se encuentra en el archivo adjunto "Plugins Chequeo.zip" en la siguiente ruta: se coge el archivo que se encuentra en la carpeta de "plugins" con el nombre: "de.mvmsoft.checkclipse_1.0.0".

6.1.2- Para instalarlo lo único que habrá que hacer es copiar ese archivo en la siguiente carpeta (Recomendable parar el entorno): \Carpeta instalación WSAD\eclipse\plugins.

NOTA: Si hubiera instalado una versión distinta del mismo plugin en esta carpeta se debería de eliminar para evitar posibles conflictos

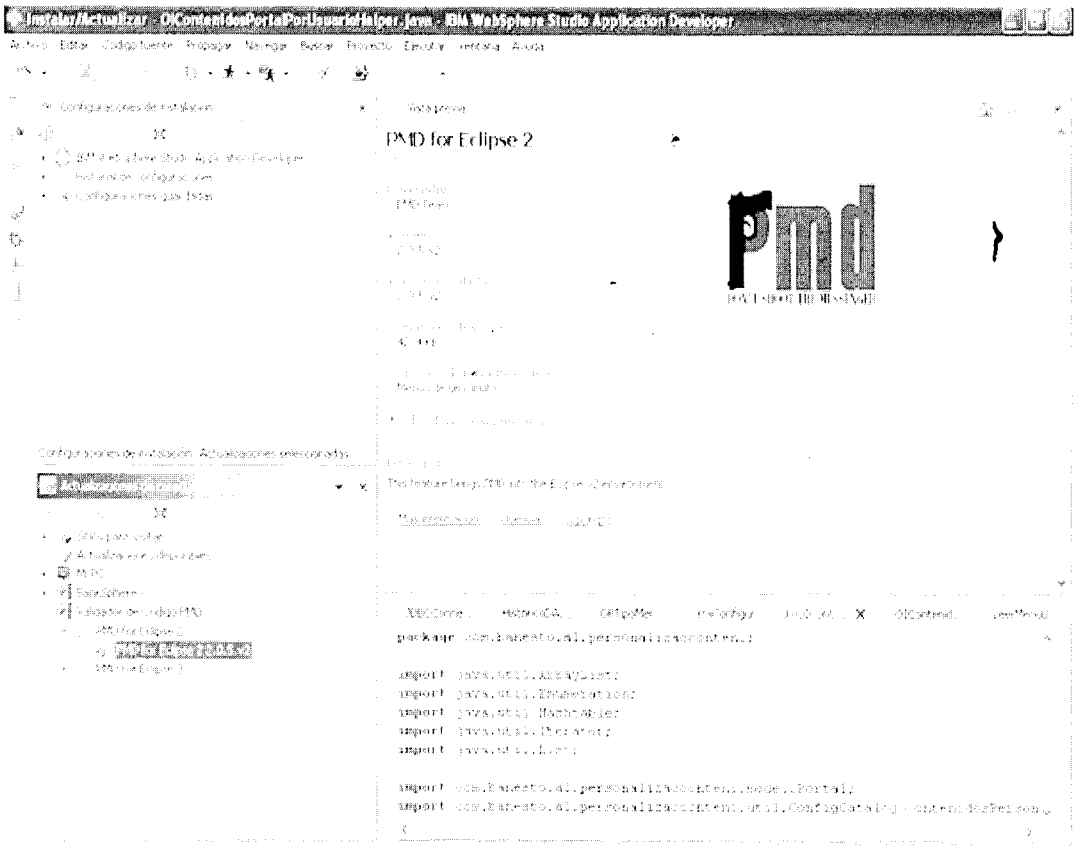
6.1.3- Es importante que se tengan las aplicaciones instaladas en las mismas carpetas ya que será más fácil recuperar datos o intercambiarlos de un equipo a otro. Las rutas de instalación de las aplicaciones están definidas en el documento "Herramientas de Desarrollo de la Factoría", que para WASAD es: D:\Archivos de programa\Ibm\WebSphere Studio.

6.2- PDM

Para instalarlo, en este caso se hará a través del actualizador incluido en WSAD o copiando los archivos adjuntos en sus carpetas respectivas.

6.2.1- Usando el actualizador de los entornos de desarrollo (No recomendado):

Una vez abierto WSAD, ir a Ayuda\actualizaciones de Software\gestor de actualizaciones. En la ventana de abajo se pulsa el botón derecho del ratón; se selecciona Nueva marca de sitio Favorito y se pone como nombre Validador PMD y como url: http://pmd.sourceforge.net/eclipse. Se despliega y se coloca PMD for Eclipse 2.0, se escoge la última versión y se le da a instalar. Queda tal y como se observa en la figura a continuación:



6.2.2- Usando los plugins que se adjuntan en el "Plugins Chequeo.zip" (Recomendado):

6.2.2.1- El plugin se encuentra en el archivo adjunto "Plugins Chequeo.zip" en la siguiente ruta: se cogen los archivos que se encuentra en la carpeta de "plugins" con el nombre "net.sourceforge.pmd.core_1.8.0.v2" y "net.sourceforge.pmd.eclipse_2.0.5.v2".

6.2.2.2- Para instalarlo lo único que habrá que hacer es copiar ese archivo en la siguiente carpeta (Recomendable parar el entorno):\Carpeta instalación WSAD\eclipse\plugins.

NOTA: Si hubiera instalado una versión distinta del mismo plugin en esta carpeta se debería de eliminar para evitar posibles conflictos.

6.2.2.3- Se cogen los archivos que se encuentra en la carpeta de "features" con el nombre: "net.sourceforge.pmd.eclipse_2.0.5.v2".

Para instalarlo lo único que habrá que hacer es copiar ese archivo en la siguiente carpeta: \Carpeta instalación WSAD\eclipse\ features.

NOTA: Si hubiera instalado una versión distinta del mismo plugin en esta carpeta se debería de eliminar para evitar posibles conflitos.

Una vez copiados estos archivos se ejecuta el entorno y si todo ha ido bien mostrará un mensaje donde confirmará que se han encontrado nuevos plugins.

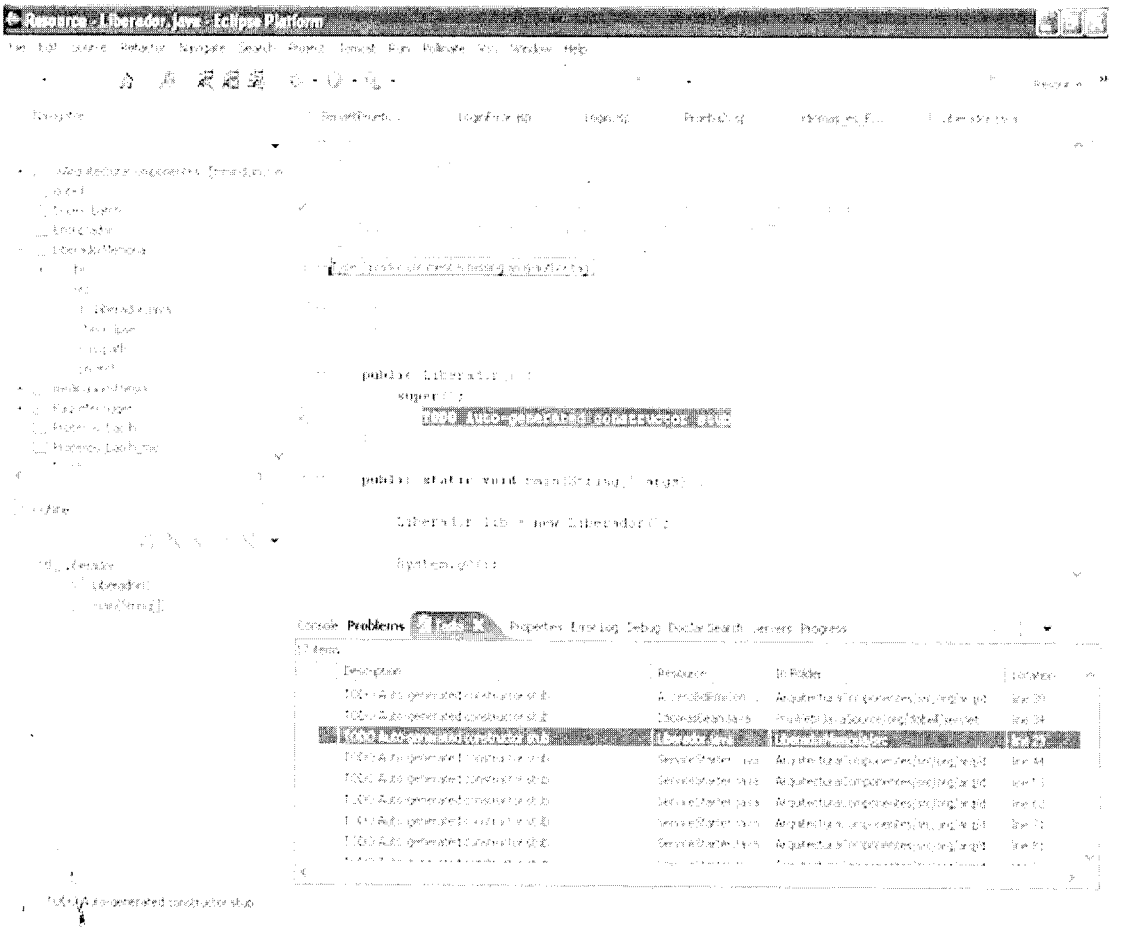
6.3 Uso de los plugins

Este apartado describe como se deben usar los plugins Checkclipse y PDM, con la herramienta WSAD.

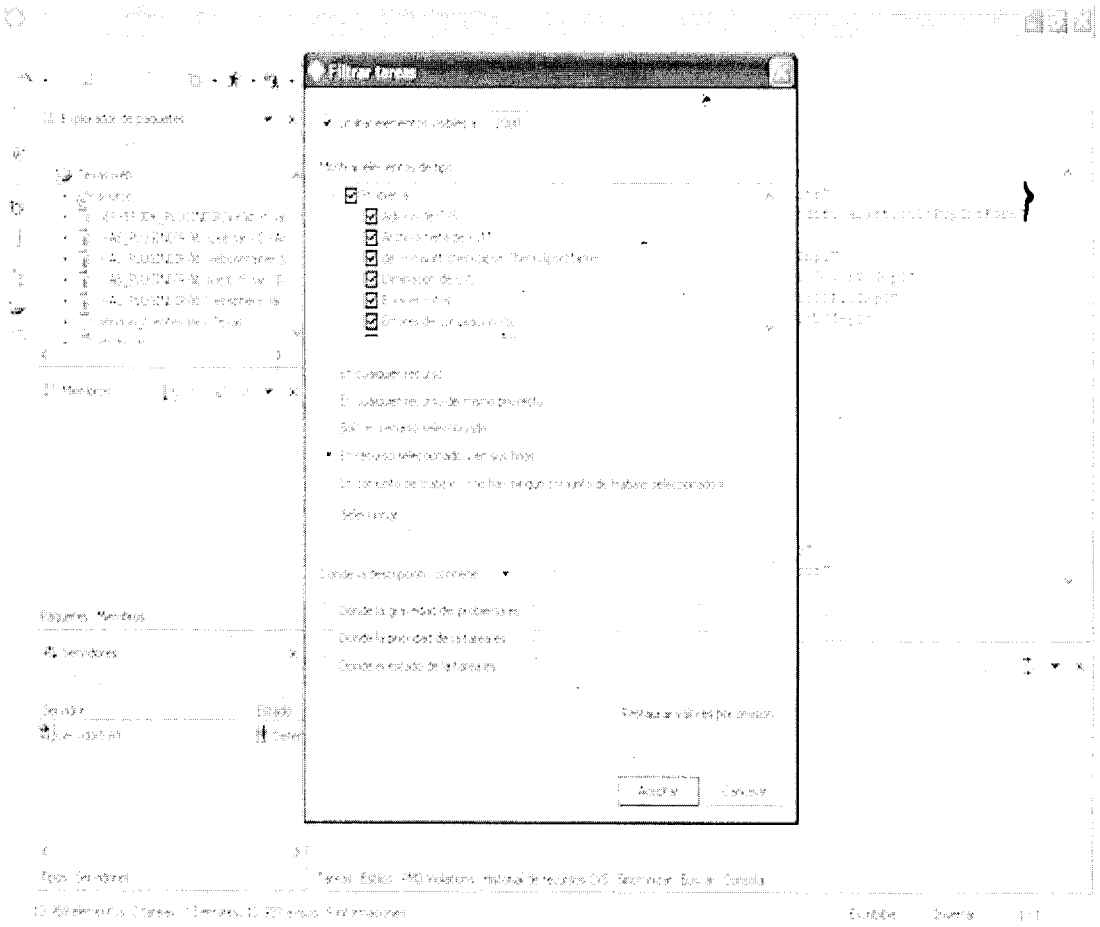
6.3.1- CHECKCLIPSE

El plugin se puede habilitar para cada proyecto. Para ello en las propiedades del proyecto, aparecerá una opción llamada checkclipse, si la habilitamos, el chequeo del código comenzará.

Una vez ha chequeado el código, podremos ver los problemas encontrados en la vista de de tareas. Si en el editor de la clase java, nos situamos encima del warning, podemos ver el error que tenemos, tal y como se observa en el siguiente ejemplo:

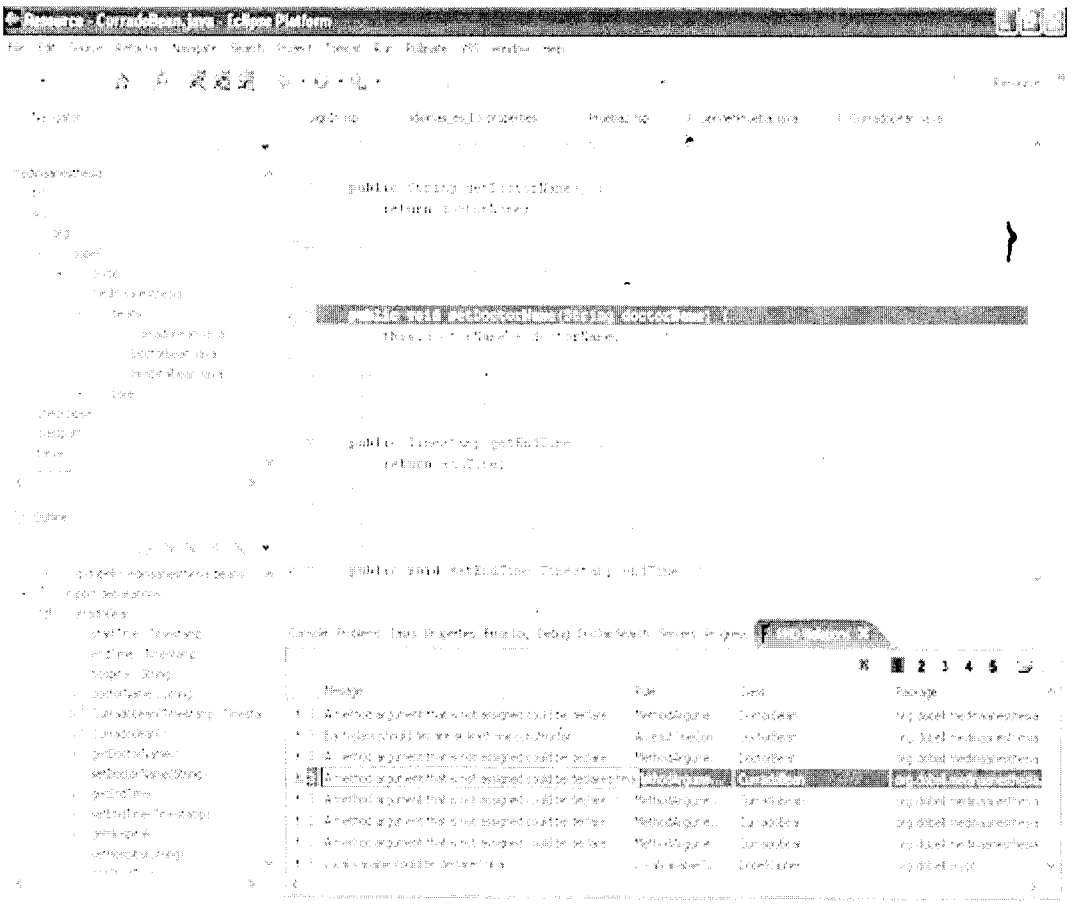


En la sección de "Mostrar elementos de tipo" se tendrá que activar la opción de "de.mvmsoft.checkclipse.CheckclipseMaker"; de la figura a continuación. A medida que vayamos corrigiendo, al guardar, los errores irán desapareciendo. Ejemplo de WSAD al corregir errores:

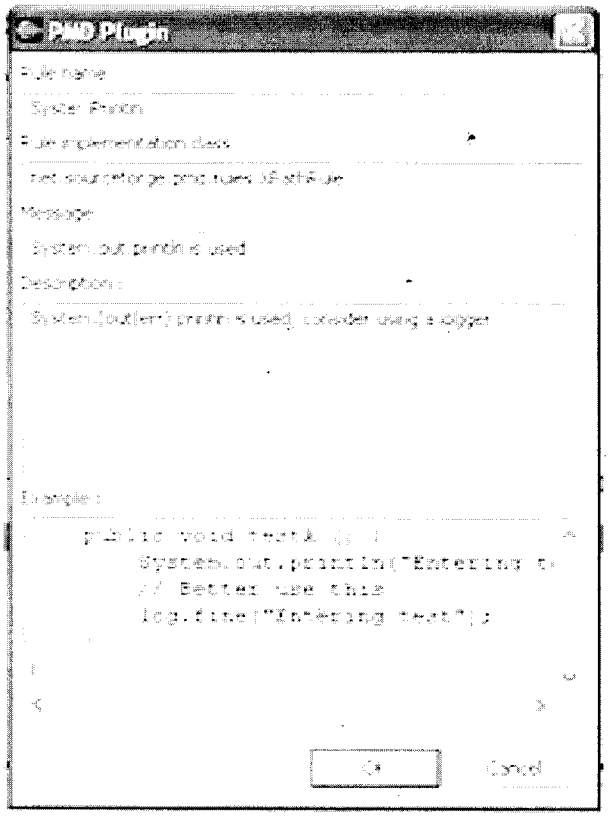


6.3.2- PDM

Para habilitar el plugin, debemos seleccionar el recurso deseado (un recurso es un proyecto, una clase java, una carpeta...) y con el menú emergente que aparece al pulsar el botón derecho del ratón, vamos a PMD y seleccionamos Check Code With PMD. En este caso podemos ver los errores como tareas de cada clase, aunque no los veremos en la vista de tareas. Para poder ver los resultados del escaneo tenemos que abrir la vista *PMD Violations*. En ella podremos ver el listado de los errores. La siguiente figura muestra el ejemplo de WSAD al escanear un recurso, mostrando errores del mismo en la vista PMD Violations.



En este caso, los errores vienen clasificados por categorías y cada categoría se puede diferenciar por el color. No debería dejarse sin corregir ningún error de color rojo ya que son errores que al cliente no pasará. Errores de este tipo son por ejemplo el uso de System.out. Si queremos ver ayuda de por qué lo que nos indica se considera un error o cómo solucionarlo, podemos pulsar sobre el error en la vista de PMD, con el botón derecho y seleccionamos *Show Rule*, se puede ver en la Figura a continuación.



Hay que tener en cuenta que la lista de errores no se actualiza solo como en el caso del otro plugin, para actualizarla hay que volver a realizar el proceso de análisis del código.

Anexo 10: Cuaderno de registro de defectos

En el cuaderno se debe anotar todas las revisiones, compilaciones y pruebas de defectos en este cuaderno. Si necesita espacio adicional utiliza espacio adicional, con otra copia de la tabla.

En la cabecera introduce los siguientes datos: nombre del trabajador de la factoría, fecha y numeración del componente.

Dentro de las instrucciones para el cuaderno de registro de defectos: se encuentra:

- Fecha: Anotar la fecha en la que se encontró el defecto.
- Número: Número para cada defecto.
- Tipo: Anota el tipo de defectos según la lista de tipos de defectos de la tabla.
- Introducido: Anotar la fase en la que se introdujo el defecto.
- Eliminado: Fecha en la que se eliminó el defecto.
- Tipo de corrección: Estima o mide el tiempo necesario para encontrar y corregir el defecto.

- Defecto Corregido: Se puede ignorar esta casilla la primera vez.
- Descripción: Escribe brevemente la descripción de un defecto.

En la siguiente tabla, se muestra el Cuaderno de Registros de Defectos.

Nombre del trabajador: _____

Fecha: _____

Del Componente: _____

Ejemplo de Cuaderno de Registro de defectos

Fecha	Número	Tipo	Introducción	Eliminado	Tipo de corrección	Defecto Corregido
Descripción:						
Fecha	**Número**	**Tipo**	**Introducción**	**Eliminado**	**Tipo de corrección**	**Defecto Corregido**
Descripción:						
Fecha	**Número**	**Tipo**	**Introducción**	**Eliminado**	**Tipo de corrección**	**Defecto Corregido**
Descripción:						

Anexo 11: Medidas de Seguridad

- El acceso a la información básica queda establecido mediante el acceso a un servidor FTP que está habilitado en la factoría. Con acceso permitido solo a las estaciones de trabajo perteneciente a la línea de aplicaciones empresariales, y usuarios específicos desde esos ordenadores. Fuera de las fronteras de factoría el servidor no tendría visibilidad.

- Se habilita a las estaciones de trabajo de la factoría programas diseñados para prevenir el acceso a no autorizados (Firewall), en conjunto con antivirus que bloqueen la entrada de programas malignos: Norton Antivirus resolvería el problema, es al antivirus que siempre gana el "Editor's Choice" de la Revista PC Magazine.
- En el caso del producto en desarrollo se tendría una copia local en todas las estaciones de trabajo y una copia centralizada en el ()CVS. Este se sincronizará periódicamente, garantizando así que permanezca en el, la versión más actualizada del componente a implementar. Luego se subiría esta copia al repositorio de componente siguiendo las normas establecidas por la administración correspondiente, garantizando el almacenamiento del mismo.
- En las unidades de DVD (Digital Versatile Disc) y de 3½ Floppy se deben establecer sellos de seguridad o retirarlas de la estación. Habilitar a su vez un sistema de instalación en las computadoras a través de mecanismos de clonación de computadoras, el cual ha sido implementado en reiteradas ocasiones por el personal de soporte técnico central en la UCI. Así se evitaría la introducción de discos piratas que irrumpan la seguridad de las estaciones de trabajo.
- El acceso al personal contratado o consultores debe ponerse especial consideración en la política y administración de sus perfiles de acceso.
- En el traspaso de información es a través de Internet. Se colocaría un CVS fuera de las fronteras de la factoría, con alcance internacional, posibilitaría que el cliente pueda sincronizar la información con los cambios efectuados por parte de la factoría y viceversa para el recibo de la información. En ningún momento la información entraría o saldría, directamente a las estaciones de trabajo de la factoría. Si las conexiones de red no son adecuadas, usar el mecanismo pero con un servidor FTP.

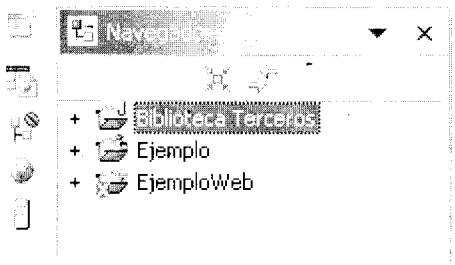
Anexo 12: Configuración y arquitectura básica de las aplicaciones C4J.

En WSAD cada que componen las aplicaciones empresariales se traducen a proyectos. O sea una aplicación empresarial estaría compuesta por proyectos Java, Proyectos Web, Proyectos EJB, Proyectos simples y Proyectos de Modelado, junto a un Proyecto de Empresa que los contiene a todos.

Los componentes deben tener la estructura básica de los módulos Web implementadas en WSAD, o sea un proyecto Web (EjemploWEB) asociado a un Proyecto Empresa (Ejemplo)

y junto a ellos un proyecto Java auxiliar (Biblioteca de Terceros),. A continuación se explicará la estructura de cada uno de ellos.

Inicialmente se debe tener un proyecto auxiliar para introducir las librerías:



Ejemplo de la Estructura Básica de los Proyectos en WSAD.

- Proyecto Java: Biblioteca Terceros

En una estructura de este tipo se deben tener las librerías del framework en un lugar accesible por todos los módulos web que se deseen construir, además de poder adaptar la aplicación a posibles cambios futuros. En ese directorio estarán las librerías comunes que no hay que incluir en el WEB-INF/lib, tal como se muestra en el ejemplo a continuación.

Ejemplo de Proyecto auxiliar con las Librerías del Framework.

Aquí el proyecto Bibliotecas Terceros incluye estas librerías, además hay que añadir los siguientes archivos:

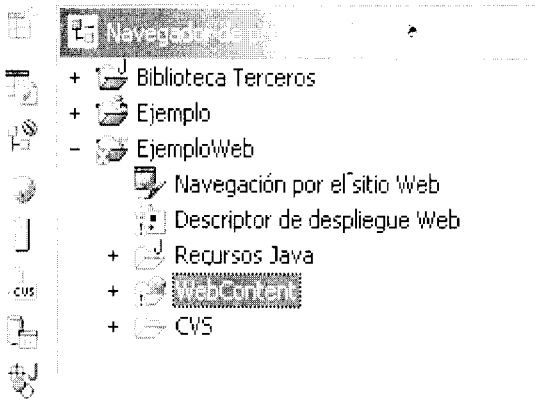
- authentication.properties
- autorizacion.properties

Es requerido que el authentication.properties tenga al final la propiedad authenticationDisable con valor true (authenticationDisable=true) y que el autorizacion.properties también tenga la propiedad autorizacionDisabled con valor true (autorizacionDisabled=true).

Proyecto Web

En el proyecto web la estructura estaría conformada por el descriptor de despliegue, el directorio de recursos (Recursos Java), el directorio de contenidos del proyecto

web(WebContent), y el directorio donde se almacena la información relacionada con el control de versiones. Tal como se muestra en el ejemplo a continuación.



Ejemplo de Estructura General del Proyecto Web.

Esta estructura es generada automáticamente por WSAD, por lo que se debe prestar más atención a la estructura interna del directorio Recursos Java y WebContent.

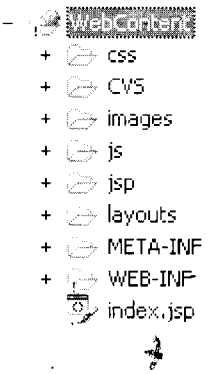
En el directorio Recursos Java, se organiza la estructura de paquetes Java por la que deben estar organizados de manera estándar todos los componentes a implementar. Mediante esa estructura se delimitan las diferentes capas de la aplicación, junto a todos los archivos XMLs de configuración del framework. Así como se representa en el siguiente ejemplo

Ejemplo de Estructura del directorio Recursos Java del Proyecto Web

En la figura se muestra como se define un paquete para los ActionServlets (com.carrefour.ejemplo.action) y otro para los ActionForms (com.carrefour.ejemplo.form), clases responsables de la lógica de presentación de la aplicación. Otros dos paquetes serian responsables de la lógica de negocio de la aplicación, dentro de los cuales se encentran los ValuesObjects (com.carrefour.ejemplo.vo) y los BusinessObjects (com.carrefour.ejemplo.bo). Los componentes de acceso a datos se definen en com.carrefour.ejemplo.dao.pb.

Adicionándole a estos un paquete donde se almacenen las clases usadas para el manejo de excepciones (com.carrefour.ejemplo.exception) y otro de elementos comunes, los cuales son accesibles desde todas las capas de la aplicación (com.carrefour.ejemplo.common).

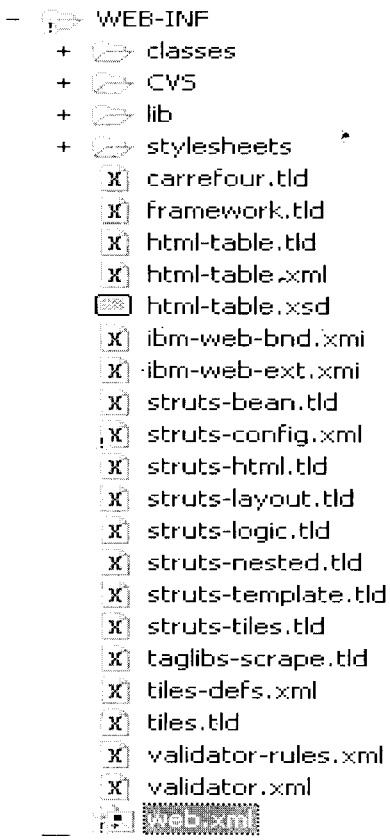
Otro directorio que merece atención especial es la estructura interna del WebContent. En este se describe la localización de lo elementos necesarios para el Proyecto Web.



Estructura del directorio WebContent del proyecto Web.

Tal como se muestra en la figura se define directorios para: hojas de estilo, imágenes, ficheros java scripts, páginas JSP, plantillas y dos directorios de mayor complejidad los cuales son el META-INF y el WEB-INF.

En el META-INF existe un fichero mediante el cual se especifican las dependencias de este módulo con otros archivos JAR existentes en una aplicación de empresa. En el directorio WEB-INF quedan establecidos todos los archivos compilados en el subdirectorio classes; las librerías específicas usadas por la aplicación en un subdirectorio nombrado lib y todas las etiquetas personalizadas que pueden ser usadas en las páginas JSP incluyendo archivos XMLs; además del archivo Web XML, donde se muestra toda la información de implementación relacionada con el proyecto Web. Se puede observar la estructura en la figura a continuación.



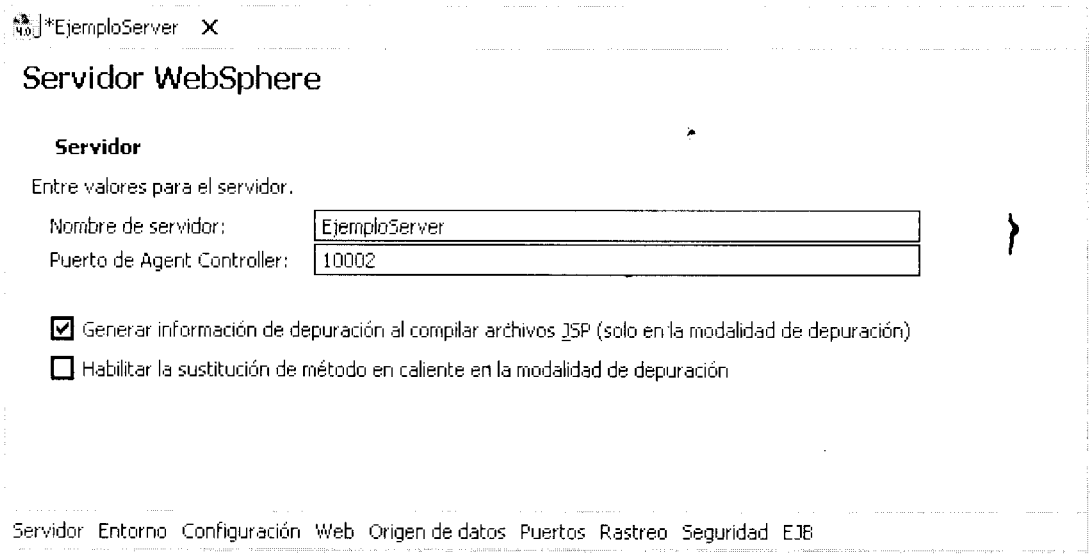
Estructura del directorio Web-INF del proyecto Web.

Técnicas de configuración del servidor de aplicaciones a utilizar:

Inicialmente el servidor a utilizar es el servidor 4.0 de WSAD.

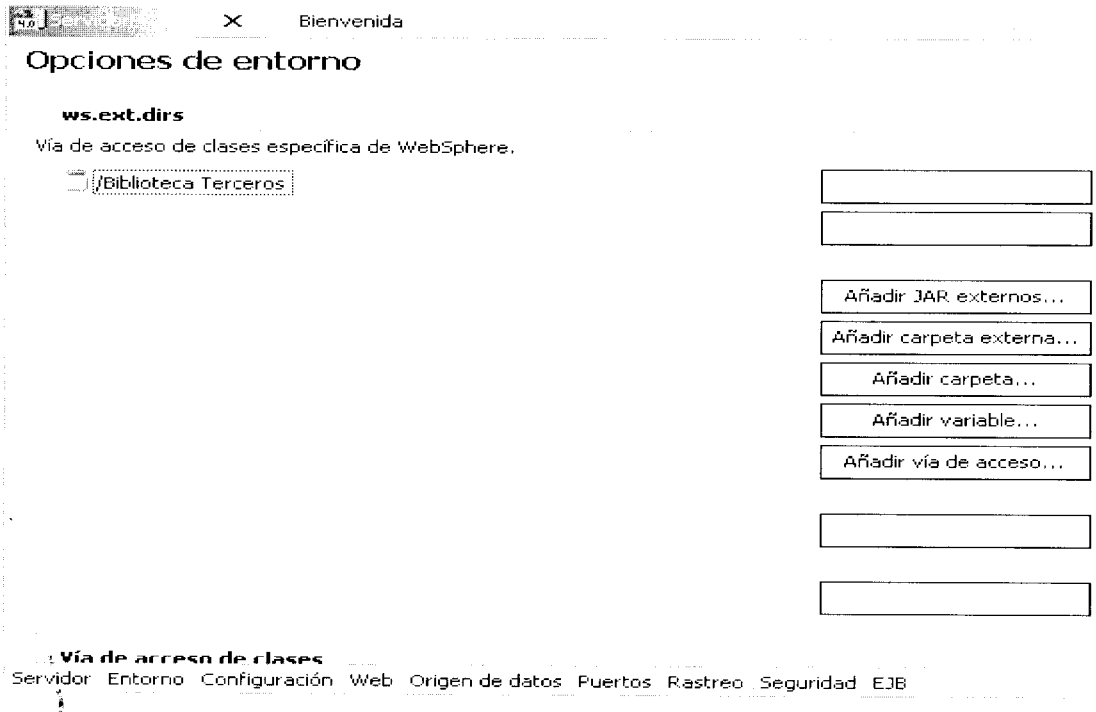
La ventana de servidor en su parte inferior tiene una serie de pestañas mediante las cuales se configura el comportamiento del mismo.

1. En la pestaña "Servidor" se debe deshabilitar el checkbox descrito por: "Habilitar la sustitución de método en caliente en la modalidad de depuración". Tal y como se muestra en la siguiente Figura.



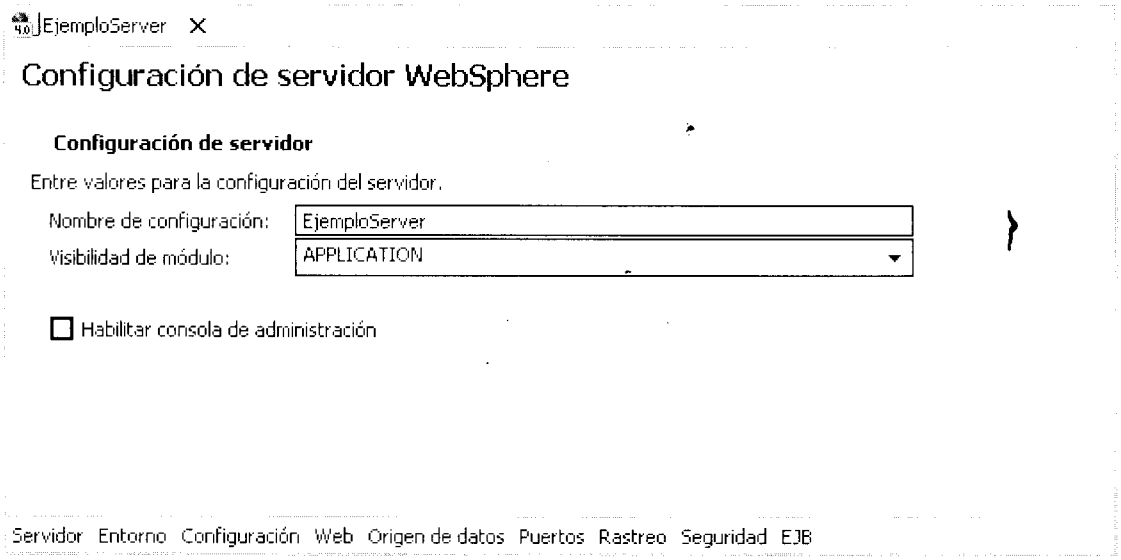
Consola servidor WebSphere en su perspectiva Servidor.

2. En la pestaña "Entorno" se deben añadir los recursos externos al proyecto, quedando la perspectiva de la siguiente manera. En este caso se añadió el JAR Biblioteca Terceros.



Perspectiva "Entorno" del la consola del servidor.

3. En la pestaña "Configuración" en el desplegable: "Visibilidad de módulo", se debe colocar el valor APPLICATION. Tal como se observa en la figura a continuación:

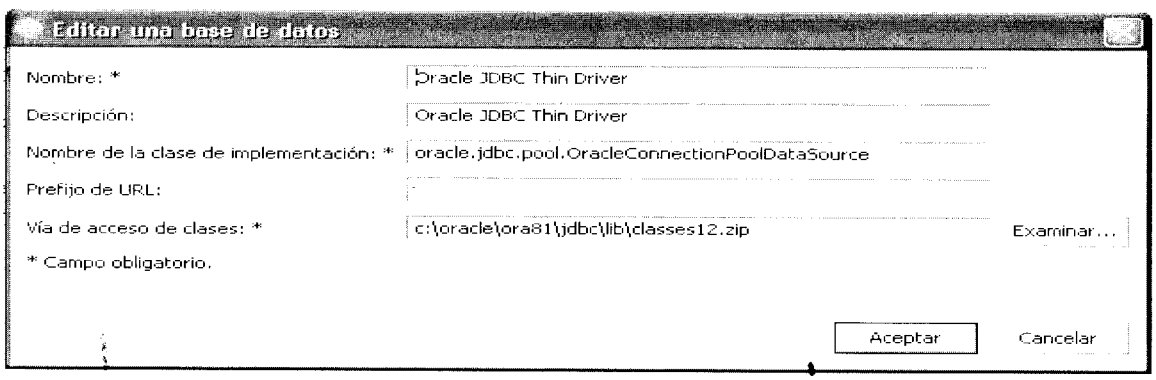


Consola servidor WebSphere en su perspectiva configuración.

4. En la pestaña "Origen de Datos" se debe comenzar por la primera tabla: "Lista controladores de JDBC". Presionar el botton añadir y entrar los datos

- Nombre: Oracle JDBC Thin Driver
- Descripción: Oracle JDBC Thin Driver
- Nombre de la implementación: oracle.jdbc.pool.OracleConnectionPoolDataSource
- Vía de acceso a clases: c:\oracle\ora81\jdbc\lib\classes12.zip (Este tiene que apuntar al classes12.zip, como se supone que esté instalado el cliente de oracle, debería estar en esa dirección o en una muy parecida).

El cuadro de datos se muestra a través de la siguiente figura

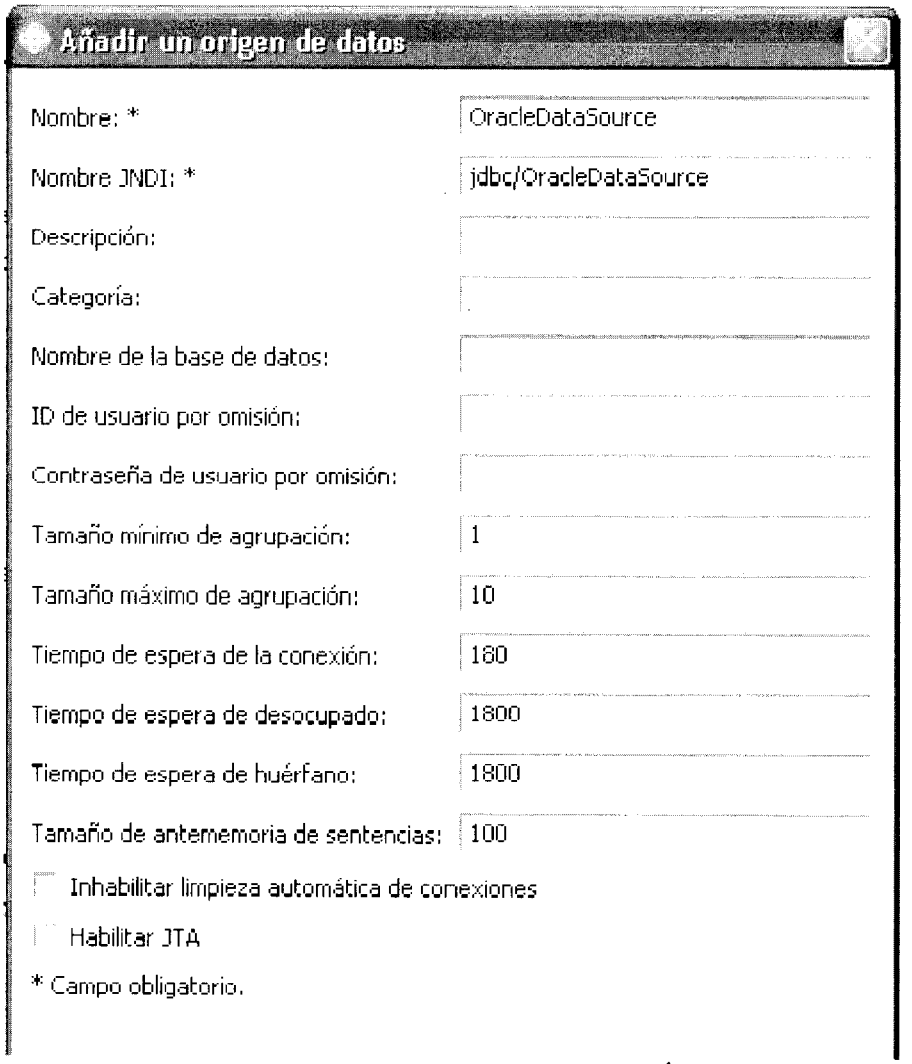


Especificación de driver de conexión a la Base de Datos.

Luego para el driver seleccionado se debe configurar el origen de datos correspondiente. Se presiona añadir a la tabla que presenta la descripción: "Origen de datos definidos en el controlador".

- Nombre: OracleDataSource.
- Nombre JNDI: jdbc/OracleDataSourc.
- Los demás campos los dejáis por defecto.

La tabla debe quedar como se muestra a continuación la Figura

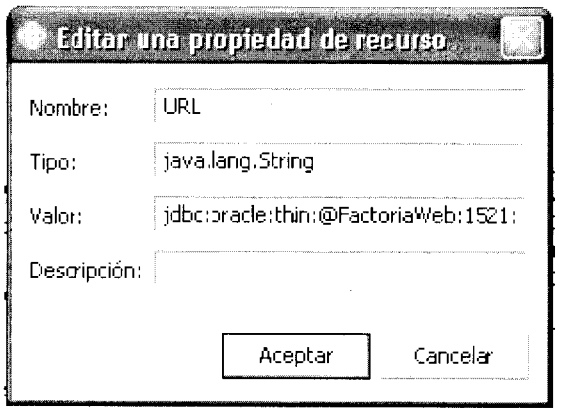


Configuración del origen de datos.

Por último se deben colocar los valores a las propiedades al datasource, para esto hay que añadir los siguientes campos:

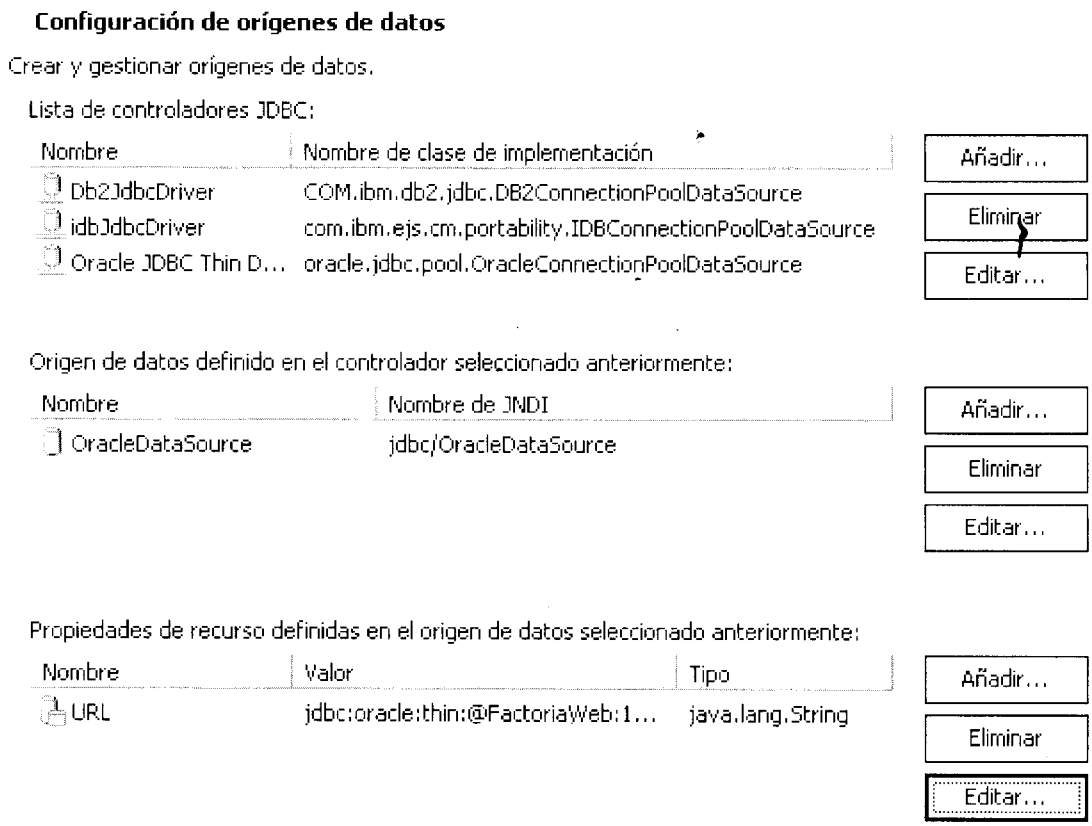
- Nombre: URL
- Tipo: java.lang.String
- Valor:
jdbc:oracle:thin:@IP_SERVIDOR:PUERTO_PARA_CONECTAR:BASE_DE_DATOS
Ejemplo: jdbc:oracle:thin:@10.33.3.17:1521: BDFacMacth o con el nombre del estación de trabajo jdbc:oracle:thin:@FactoriaWeb:1521:BDFacMacth

La siguiente Figura muestra como debe quedar la tabla de propiedades.



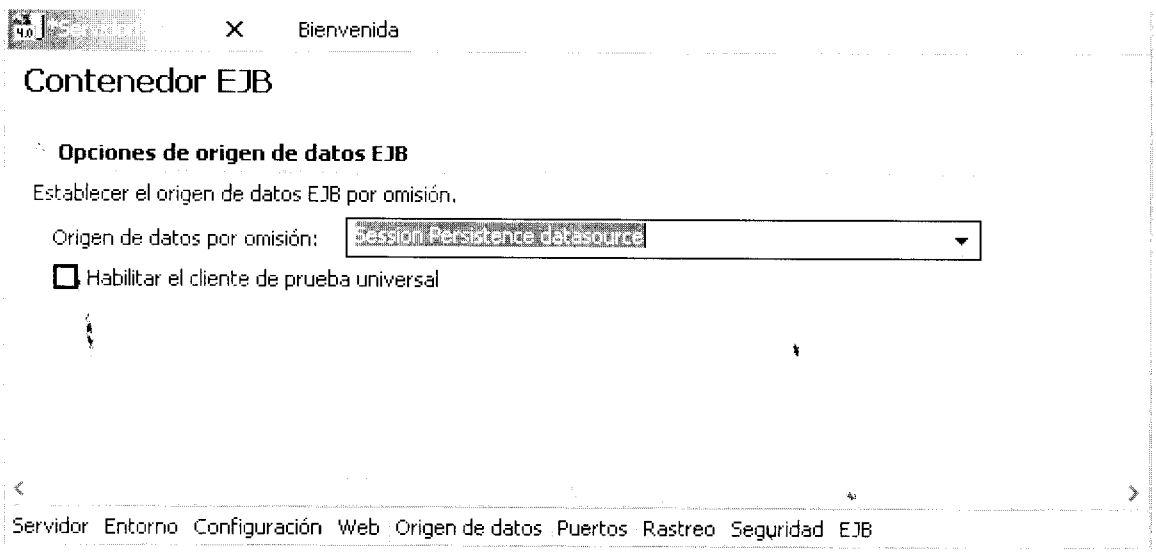
Cuadro de propiedades.

Finalmente la configuración del origen de datos debe quedar como se muestra a continuación en la Figura.



Configuración del Origen de Datos en WebSphere.

5. Pestaña EJB, se debe colocar en el origen de datos por omisión: Session Persistence datasource.



Perspectiva EJB de la consola servidor.

El resto de las opciones se dejan por defecto, y con esto quedaría configurado el entorno de desarrollo.

6.- Hay que añadir las opciones de login, esto se hace en la ruta donde esté instalado el WSAD, por ejemplo:

D:\Archivos de programa\Ibm\WebSphere Studio\runtimes\aes_v4\java\jre\lib\security

Allí han de estar estos dos archivos, con este contenido.

login.cfg

```
HW{
com.carrefour.c4j.security.auth.login.CarrefourEJBLoginModule required debug=true;
};
PA{
com.carrefour.c4j.security.auth.login.CarrefourEJBLoginModule required debug=true;
};
EP{
com.carrefour.c4j.security.auth.login.CarrefourEJBLoginModule required debug=true;
};
PF{
com.carrefour.c4j.security.auth.login.CarrefourEJBLoginModule required debug=true;
};
DP{
com.carrefour.c4j.security.auth.login.CarrefourEJBLoginModule required debug=true;
};
TS{
com.carrefour.c4j.security.auth.login.CarrefourEJBLoginModule required debug=true;
};
FT{
com.carrefour.c4j.security.auth.login.CarrefourEJBLoginModule required debug=true;
};
```

java.security

```
#
# This is the "master security properties file".
#
# In this file, various security properties are set for use by
# java.security classes. This is where users can statically register
# Cryptography Package Providers ("providers" for short). The term
# "provider" refers to a package or set of packages that supply a
# concrete implementation of a subset of the cryptography aspects of
# the Java Security API. A provider may, for example, implement one or
# more digital signature algorithms or message digest algorithms.
#
# Each provider must implement a subclass of the Provider class.
# To register a provider in this master security properties file,
# specify the Provider subclass name and priority in the format
#
#   security.provider.<n>=<className>
#
# This declares a provider, and specifies its preference
# order n. The preference order is the order in which providers are
# searched for requested algorithms (when no specific provider is
# requested). The order is 1-based; 1 is the most preferred, followed
# by 2, and so on.
#
# <className> must specify the subclass of the Provider class whose
# constructor sets the values of various properties that are required
# for the Java Security API to look up the algorithms or other
# facilities implemented by the provider.
#
# There must be at least one provider specification in java.security.
# There is a default provider that comes standard with the JDK. It
# is called the "SUN" provider, and its Provider subclass
# named Sun appears in the sun.security.provider package. Thus, the
# "SUN" provider is registered via the following:
```

```
#
#    security.provider.1=sun.security.provider.Sun
#
# (The number 1 is used for the default provider.)
#
# Note: Statically registered Provider subclasses are instantiated
# when the system is initialized. Providers can be dynamically
# registered instead by calls to either the addProvider or
# insertProviderAt method in the Security class.
#
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.jsse.JSSEProvider
#
# Class to instantiate as the system Policy. This is the name of the class
# that will be used as the Policy object.
#
policy.provider=sun.security.provider.PolicyFile
# The default is to have a single system-wide policy file,
# and a policy file in the user's home directory.
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${java.home}/lib/security/java.pol
policy.url.3=file:${user.home}/.java.policy
# whether or not we expand properties in the policy file
# if this is set to false, properties (${...}) will not be expanded in policy
# files.
policy.expandProperties=true
# whether or not we allow an extra policy to be passed on the command line
# with -Djava.security.policy=somefile. Comment out this line to disable
# this feature.
policy.allowSystemProperty=true
# whether or not we look into the IdentityScope for trusted Identities
```

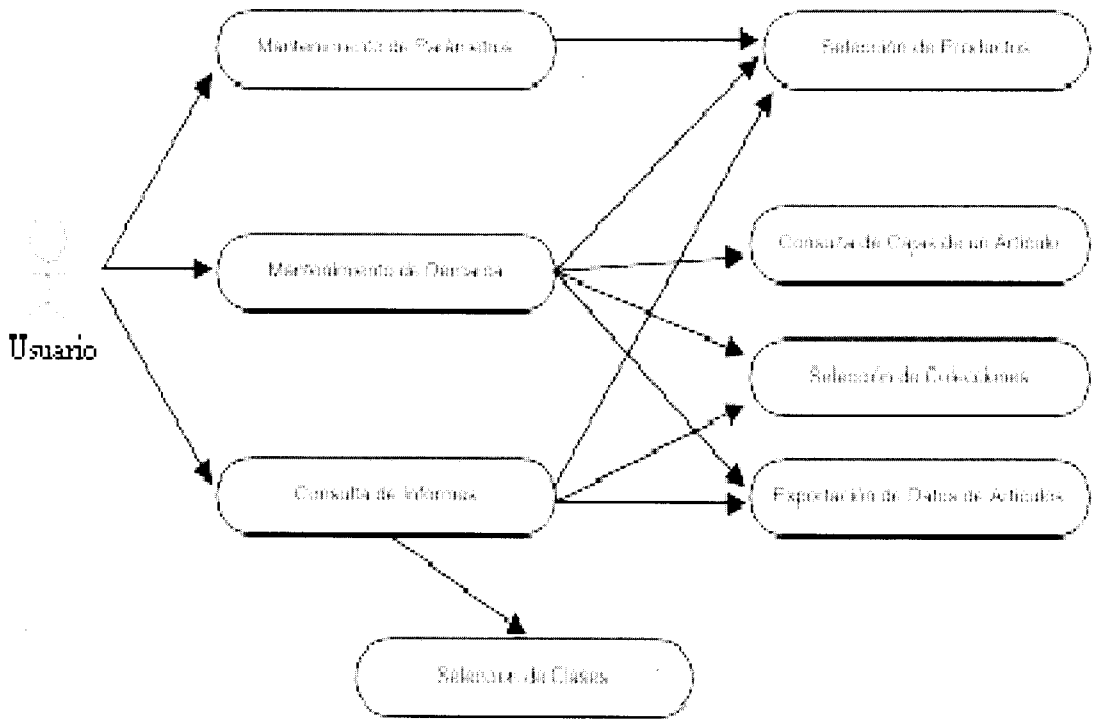
```
# when encountering a 1.1 signed JAR file. If the identity is found
# and is trusted, we grant it AllPermission.
policy.ignoreIdentityScope=false
#
# Default keystore type.
#
keystore.type=jks
#
# Class to instantiate as the system scope:
#
system.scope=sun.security.provider.IdentityDatabase
#
# List of comma-separated packages that start with or equal this string
# will cause a security exception to be thrown when
# passed to checkPackageAccess unless the
# corresponding RuntimePermission ("accessClassInPackage."+package) has
# been granted.
package.access=sun.
#
# List of comma-separated packages that start with or equal this string
# will cause a security exception to be thrown when
# passed to checkPackageDefinition unless the
# corresponding RuntimePermission ("defineClassInPackage."+package) has
# been granted.
#
# by default, no packages are restricted for definition, and none of
# the class loaders supplied with the JDK call checkPackageDefinition.
#
#package.definition=
#indicación de fichero de configuración para C4J
login.config.url.1=file:${java.home}/lib/security/login.cfg
```

Este fichero probablemente existirá, y la única línea importante será la última, que le indica al C4J de donde coger el archivo de configuración.

Solo quedaría agregar el proyecto al servidor, arrancarlo y ejecutarlo para levantar las aplicaciones Carrefour. Con este paso se concluye la plantilla a seguir por todo el equipo de implementación de la factoría, con el fin de lograr la configuración del framework, y puesta en marcha de las aplicaciones. Algún cambio que se quiera realizar sobre la estructura se le debe consultar al responsable del equipo de implementación.

Anexo13: Recepción de especificaciones en el análisis funcional.

Caso de uso en la recepción de las especificaciones



Expansión del caso de uso "Exportación de datos de Artículos" en la recepción de especificaciones.

FICHA DESCRIPTIVA	
ID	CAUSO - 09
Nombre	Exportar de Datos de Artículos
Precondiciones	El usuario se encuentra en la ***pantalla de informes o en la de mantenimiento de demarca*** y se muestran registros de artículos en las mismas.
Poscondiciones	Se genera un fcihero excel que puede guardar en base a los estándares establecidos
Actor principal	Toos

Descripción:
Permite al usuario exportar la información que se muestra por pantalla.

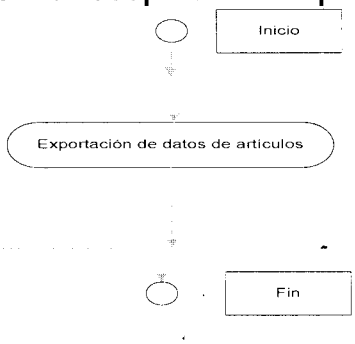
Camino Habitual	***Camino Alternativo***

El caso de uso Exportación de Datos de Artículos se activa en el momento en el que se muestran datos por pantalla.

Permite exportar los datos de las tablas a un fichero Excel con la siguiente información:

- En la parte superior de la hoja del fichero, se mostrarán los criterios de selección escogidos por el usuario.
- Debajo de los criterios de selección, aparecerán todas las columnas con los registros resultantes que en ese momento se podían visualizar por la pantalla que invocó este servicio.

Diagrama de actividad en la recepción de especificaciones.



Anexo 14: Especificación de diseño de páginas.

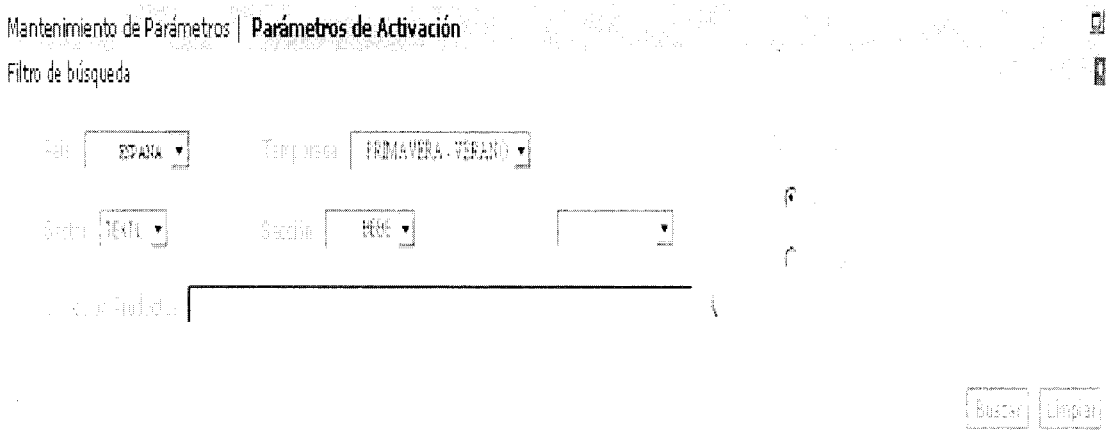
Inicialmente se deben especificar donde se ubicará la página JSP, dentro de la maqueta, especificar su nombre y dar una breve descripción de la misma. Luego de eso se debe especificar la internas gráfico y la tabal de descripción de componentes.

Por ejemplo:

Nombre de la página: Criterios de selección

Los criterios de selección se mostrarán en la parte superior del marco central tal y como se muestra en la siguiente figura:

Ejemplo de Interfaz Gráfica



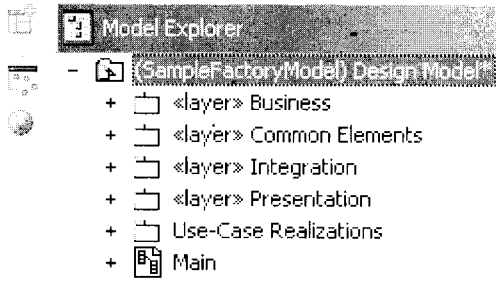
En la siguiente tabla se detallan los componentes de la interfaz gráfica en cuanto a: Objeto de Ventana (Nombre), tipo de objeto siguiendo las especificaciones de C4J y breve descripción del mismo.

Descripción de los Componentes de Interfaz.

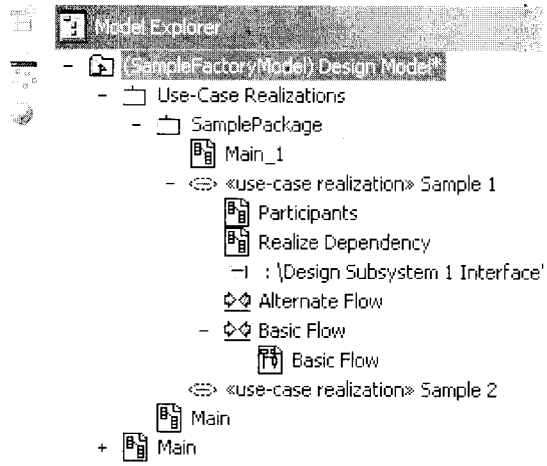
Objeto de Ventana	*Tipo de Objeto*	*Descripción/Acción*
País	Lista Desplegable	Tiene el mismo funcionamiento que el definido en el módulo anterior.
Temporada	Lista Desplegable	El usuario debe elegir una de las cuatro posibilidades: - Primavera – Verano - Otoño – Invierno - Anual Por defecto, estará a blanco
Sector	Lista Desplegable	Tiene el mismo funcionamiento que el definido en el módulo anterior.
Sección	Lista Desplegable	Se cargarán automáticamente en esta lista desplegable todas las secciones correspondientes al sector seleccionado. Por defecto, estará a blanco.
Clase	Lista Desplegable	Puede adquirir cualquiera de los siguientes valores: - Básico - Tendencia - Permanente Por defecto estará a blanco
Selección Productos	Campo informativo + Botón Búsqueda	Se detallará con posterioridad.
Buscar	Botón	Se habilita cuando se introduzcan todos los criterios de selección (son todos excepto la selección de producto)
Limpiar	Botón	Permite eliminar la selección de criterios efectuada y la información correspondiente a la misma. De este modo, todos los criterios de selección vuelven a sus valores por defecto.
Ocultar	Botón	Tiene el mismo funcionamiento que el definido en el módulo anterior.

Desplegar	Botón	Tiene el mismo funcionamiento que el definido en el módulo anterior.

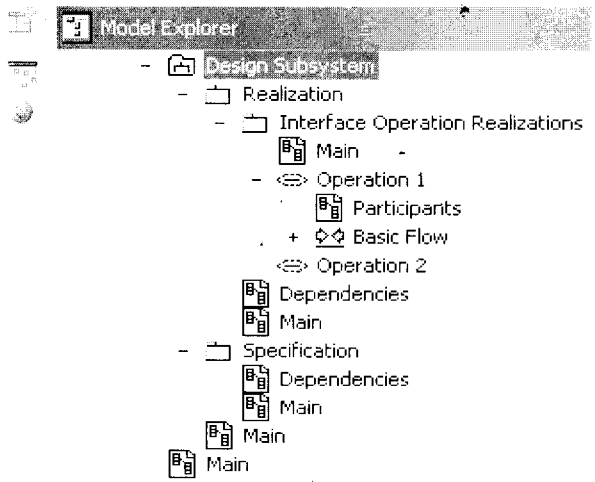
Anexo 15: Estructura del modelo de diseño



Anexo 16: Realización de los casos de Uso



Anexo 17: Organización de un Subsistema de diseño



Anexo 18: Tablas Descriptiva del Flujo de Trabajo de Diseño

Orden Lógico	Actividad	Entrada	Role	Entrega	Herramienta
1	Creación del modelo de diseño.	-Análisis técnico y funcional. - Reporte de componentes del repositorio. Suministrado por: Jefe de línea de producción y gestor de repositorio.	Arquitecto	-Modelo de diseño. -Descripción y organización de los subsistemas. -CSR a emplear. Entregado a: Diseñador de casos de uso.	-Rational XDE para Java. -WSAD.
2	Especificación y Realización, de los casos de uso.	-Modelo de diseño. -Descripción y organización de los subsistemas y CSR a emplear. Suministrado por: Arquitecto	Diseñador de casos de uso.	-Subsistemas especificados y realizados: conceptos, diagramas de clases y de interacción. Entregado a: Diseñador de componentes.	-Rational XDE para Java. -WSAD.

3	Planificación y distribución de diseño	-Subsistemas especificados y realizados. Suministrado por: Diseñador de casos de uso.	Diseñador de componentes general.	-Registro de planificación y distribución del diseño. -Casos de uso por subsistema y capas en las que interactúan. Entregado a: Diseñadores por capas de arquitectura.	-Rational XDE para Java. -WSAD.
5	Diseño de la capa de presentación	Registro de planificación y distribución del diseño. Suministrado por: Diseñador de componentes	Diseñador de la capa de presentación	-Diagramas de clases detallados sobre clases de lógica presentación. -Diagramas de secuencia: representación de las decisiones de páginas a presentar. - Especificación de los XML de conexión con los servicios de presentación. Entregado a: Diseñador de componentes	-Rational XDE para Java. -WSAD.
6	Diseño de la capa de Lógica de Negocio.	Registro de planificación y distribución del diseño. Suministrado por: Diseñador de componentes.	Diseñador de capa de Lógica de Negocio.	- Diagramas de clases detallados sobre clases de lógica de negocio. -Diagramas de secuencias que representan el flujo de los objetos relacionados, con la toma de decisiones de la aplicación. Entregado a: Diseñador de componentes	-Rational XDE para Java. -WSAD.
7	Diseño de la capa de Acceso a datos.	Registro de planificación y distribución del diseño. Suministrado por: Diseñador de componentes.	Diseñador de capa de acceso a datos.	- Diagramas de clases detallados sobre clases de acceso a datos -Diagramas de secuencias que representan el flujo de los objetos relacionados con la abstracción del	-Rational XDE para Java. -WSAD.

				acceso a los datos. Entregado a: Diseñador de componentes	
8	Flujo de sucesos de diseño.	-Modelo de diseño completo. Suministrado por: diseñadores de capas específicas.	Diseñador de componentes	-Documento que describe y detalla la funcionalidad y la forma del(o los) subsistemas. Entregado a : Diseñador de casos de uso	-Rational XDE para Java. -WSAD.
9	Aprobación del modelo de diseño como un todo.	-Flujo de suceso de diseño. -Modelo de diseño Suministrado por: Diseñador de componentes.	Diseñador de casos de uso y Arquitecto.	-Modelo de diseño. -El flujo de sucesos de diseño. Entrega a: Jefe de Implementación y Arquitecto (jugando su role en la implementación).	-Rational XDE para Java. -WSAD.

Anexo 19: Tablas Descriptiva del Flujo de Trabajo de Implementación

Orden Lógico	Actividad	Entrada	Role	Entregable	Herramientas
1.-	Implementación de la arquitectura.	-Arquitectura del modelo de diseño. -Flujo de sucesos de diseño. -Base de componentes de configuración y de reutilización del framework Suministrado por: Jefe de diseño	Arquitecto	-Estructura de directorios dentro del proyecto. -Configuración del framework a punto para trabajar Entregado a: Jefe de programación	-Rational XDE -WSAD integrado con PDM y Checkclipse
1.1	Agregar componentes reutilizables en caso necesario	-Documento de Solicitud de componente al repositorio. Suministrador por: Arquitecto	Gestor de Repositorio	-Componente reutilizable Entregado a: Arquitecto	-Rational XDE -WSAD integrado con PDM y Checkclipse
2.-	Planificar la implementación.	-Arquitectura completa-Especificación de las Interfaces Gráficas-Modelo de diseño de componentes. -Modelo de diseño de la base de datos. Suministrado por: Arquitecto	Jefe de programación	-Registro de planificación y distribución de la implementación Entregado a: programadores por capa.	-Rational XDE -WSAD integrado con PDM y Checkclipse
3.-	Implementar Base de Datos	-Modelo lógico de la Base de Datos. Suministrado por: Jefe de programación.	Programador de bases de datos	-Base de datos implementadas. Entregado a: Jefe de programación y responsable de calidad.	- Oracle. -Erwin Studio o Power Designed -WSAD integrado con PDM y Checkclipse
3.1	Revisión de funcionalidad y control	-Base de datos implementadas y XMLs de	Jefe de	-Base de datos implementadas y	-Lista de

	de la calidad de la bases de datos.	integración, Suministrado por: Programador de bases de datos	programación y responsable de calidad.	XMLs de integración, probados y con la calidad requerida. Entregado a: Programador de la capa de la capa de acceso a datos	comprobación -WSAD integrado con PDM y Checkclipse -Cuaderno de registro de defectos.
4.	Implementar capa de acceso a datos	-Base de datos. -XMLs de integración. -Diseño segmentado de componentes de acceso a datos. Suministrado por: Programador de bases de datos	Programador de capa de Acceso a datos	-Capa de acceso a datos implementada. Entregado a: Jefe de programación y responsable de calidad.	-Racional XDE -WSAD integrado con PDM y Checkclipse -Oracle
4.1	Revisión de funcionalidad y control de la calidad de la capa de acceso a datos.	-Capa de acceso a datos implementada. Suministrado por: Programador de capa de Acceso a datos.	Jefe de programación y responsable de calidad.	-Capa de acceso a datos implementada, probados y con la calidad requerida. Entregado a : Programador de la capa lógica de negocio.	-Lista de comprobación -WSAD integrado con PDM y Checkclipse -Cuaderno de registro de defectos.
5	Implementación de la capa de lógica de negocio.	-Diseño segmentado de los componentes de lógica de negocio. -Capa de acceso a datos implementada, probada y con la calidad requerida. Suministrado por: Jefe de programación y	Programador de la capa lógica de negocio.	-Implementación de la lógica de negocio y conexión con la capa acceso a datos. Entregado a: Jefe de programación y el responsable de calidad.	-Rational XDE. -WSAD integrado con PDM y Checkclipse

		responsable de calidad.			
5.1	Revisión de funcionalidad y control de la calidad de la capa de lógica de negocio.	-Implementación de la lógica de negocio y conexión con la capa acceso a datos. Suministrado por: Programador de la capa lógica de negocio.	Jefe de programación y el responsable de calidad	-Implementación de la lógica de negocio y conexión con la capa acceso a datos, probada y con la calidad requerida. Entregado a: *en este* *caso no se efectúa entrega.*	-Lista de comprobación -WSAD integrado con PDM y Checkclipse -Cuaderno de registro de defectos.
6	Implementación de lógica de presentación	-Diseño segmentado de los componentes de lógica de presentación. Suministrado por: Jefe de programación.	Programador De la capa de lógica de presentación	-Implementación de los componentes de lógica de presentación. Entregado a: Programador de Interfaz gráfica.	-Rational XDE. -WSAD integrado con PDM y Checkclipse
7	Implementación de presentación.	-Especificación de las Interfaces Gráficas. -Especificación sobre XMLs a mapear. - Implementación de los componentes de lógica de presentación. Suministrado por: Jefe de programación.	Programador de interfaz gráfica.	-Implementación de la capa de presentación. Entregado a: Jefe de programación y el responsable de calidad.	-Rational XDE. -WSAD integrado con PDM y Checkclipse.
7.1	Revisión de funcionalidad y control de la calidad de la capa de presentación.	-Implementación de la capa de presentación. Suministrado por: Programador de interfaz gráfica.	Jefe de Programación y el responsable de calidad.	-Implementación de la capa de presentación, probada y con la calidad requerida. Entregado a: *en este* *caso no se efectúa*	-Rational XDE. -WSAD integrado con PDM y Checkclipse -Lista de comprobación

				entrega.	
8	Integración final de todas las capas	-Capa de lógica de negocio conectada con el resto de las capas inferiores. -Capa de presentación Suministrado por: Programador de interfaz gráfica y de lógica de negocio.	Jefe de Programación (pueden participar otros programadores)	-Aplicación integrada. Entregado a: Jefe de línea de desarrollo y responsable de calidad.	-Rational XDE. -WSAD integrado con PDM y Checkclipse
8.1	Revisión de funcionalidad y control de la calidad.	-Aplicación integrada. Suministrador por: Jefe de Programación	Jefe de línea de desarrollo, Jefe de programación y responsable de calidad.	-Aplicación integrada, probada y con la funcionalidad requerida. Entregada a: Jefe de fase de Pruebas	-Rational XDE. -WSAD integrado con PDM y Checkclipse -Lista de comprobación

Anexo 20: Tablas Descriptiva del Flujo de Trabajo de Pruebas.

Orden L.	Actividad	Entrada	Role	Entrega
1	-Planificar prueba	- Análisis funcional- Análisis técnico - Modelo de diseño – Resultado de la implementación	-Jefe de prueba	- Plan de prueba

2	-Diseñar prueba	- Análisis funcional - Análisis técnico - Modelo de diseño - Resultado de la implementación - Plan de prueba	-Jefe de prueba	- Casos de prueba - Procedimientos de prueba
3	-Ejecutar prueba de integración	- Caso de prueba - Procedimiento de prueba - Resultado de la implementación	-Probador	- Defecto
4	-Reportar defecto	- Defecto	- Jefe de prueba	- Reporte de defecto
5	-Evaluar prueba	- Plan de prueba - Modelo de prueba - Defectos	-Jefe de prueba	- Evaluación de prueba

Anexo 21: Tabla de Documentación de los Componentes.

Esta posee una cabecera en la que se especifican aspectos generales de cada componente a almacenar. Dentro de los que se encuentra:

-Código. El número de proyecto + número de componente + número de versión. Se podría agregar el número del equipo que lo implementó, si fuesen varios.

-Nombre: Nombre del componente.

-Palabras Clave: Se incluyen palabras mediante las cuales se pudiesen referenciar el componente en el momento de hacer búsquedas.

Código: _____

Nombre: _____

Palabras Clave: _____

Documentación de los Componentes

			Descripciones
Dtos de creación del componente	Fecha		
	Proyecto Solicitante		
	Tiempo de Desarrollo		
Datos sobre el acceso al componente	Todos		
	Acceso restringido		
	Sin acceso		
Disponibilidad:	Si		
	No		
Funcionalidad:			
Utilización:			

Glosario de Términos y Siglas

- ✓ **Actor**: Alguien o algo, fuera del sistema o negocio que interactúa con el sistema o negocio.
- ✓ **Artefactos**: Una parte de la información que (1) es producida, modificada, o usada por un proceso, (2) define un área de responsabilidad, y (3) está sujeta al control de versión. Un artefacto puede ser un modelo, un elemento del modelo, o un documento. Un documento puede adjuntar otros documentos. Una parte de la información que es usada o producida por un proceso de desarrollo.
- ✓ **API**: Del inglés (Application Programming Interface - Interfaz de Programación de Aplicaciones) consiste en proporcionar un conjunto de funciones de uso general.
- ✓ **Applet**: Programa informático basado en el lenguaje de programación Java. No se usa muy a menudo, pero puede ser muy útil, ya que permite crear muchos tipos de efecto incrustando código Java en el código HTML de las páginas web.
- ✓ **Casos de uso**: Los casos de uso constituyen una forma de representación visual de las funcionalidades del sistema.
- ✓ **CMM**: El Modelo de Capacidad y Madurez o CMM (Capability Maturity Model), es un método de definir y gestionar los procesos a realizar por una organización. Fue desarrollado inicialmente para los procesos relativos al software por la Universidad Carnegie-Mellon para el SEI (Software Engineering Institute).
- ✓ **COCOMO**: Constructive Cost Model, es un modelo que permite realizar estimaciones y planificaciones de proyectos de sistemas informáticos.
- ✓ **Freeware**: Es un software de computadora que se distribuye sin cargo. A veces se incluye el código fuente, pero no es lo usual
- ✓ **Diagrama conceptual:** Se considera como un análisis de actividades y consiste en la solución de negocios para el usuario y se expresa con los casos de uso.
- ✓ **Diseño detallado:** traduce el diseño lógico en una solución implementable y costo-efectiva o económica.
- ✓ **DVD**: Digital Versatile Disc es un formato de almacenamiento multimedia en disco óptico que puede ser usado para guardar datos, incluyendo películas con alta calidad de video y sonido.
- ✓ **EAR**: (en inglés Enterprise Archive-Archivo Empresarial), contiene la información relaciona da con un proyecto empresarial.

- ✓ **Especificación lógica:** traduce los escenarios de uso creados en el diseño conceptual en un conjunto de objetos de negocio y sus servicios. Se convierte en parte en la especificación funcional que se usa en el diseño físico. Es independiente de la tecnología. El diseño lógico refina, organiza y detalla la solución de negocios y define formalmente las reglas y políticas específicas de negocios.
- ✓ **Evento sistémico**: es un evento de alto nivel generado por un actor externo. Se asocia a operaciones del sistema: las que se emiten en respuesta a los eventos del sistema. Por ejemplo, cuando un cajero que usa una terminal de punto de venta oprime el botón "terminar venta", está generando un evento sistémico que indica que "la venta ha terminado". Del mismo modo, cuando alguien que usa un editor de texto pulsa el botón "revisar ortografía", está produciendo un evento del sistema.
- ✓ **Flujo Grama**: Diagrama de Flujo
- ✓ **FTP**: Es uno de los diversos protocolos de la red Internet, concretamente significa *File Transfer Protocol* (Protocolo de Transferencia de Archivos) y es el ideal para transferir grandes bloques de datos por la red.
- ✓ **Herramientas CASE**: (del inglés **C**omputer **A**ided **S**oftware **E**ngineering, que viene a significar Ingeniería de Software asistida por ordenador). Son instrumentos o sistemas automatizados que brindan soporte a las actividades de producción de software.
- ✓ **HTML**: "Hyper Text Marckup Language", lenguaje de marcas, con que se programan las páginas Web. Brinda facilidades para mostrar imágenes, textos hipervínculos, tablas, etc., y es interpretado por los navegadores Web.
- ✓ **IDEF-0: (**en Inglés Integration Definition for Function Modeling - Definición de la integración para la modelización de las funciones). Consiste en una serie de normas que definen la metodología para la representación de funciones modelizadas.
- ✓ **IEEE**: corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.
- ✓ **Intranet**: Es una adaptación de las mismas tecnologías que existen en Internet, para que sean utilizadas dentro de la red internan de una empresa u organización de forma tal que sus miembros puedan intercambiar información de todo tipo, utilizando el Web como interfaz común.
- ✓ **ISO 9001**: La norma ISO 9001, es un método de trabajo, con el fin de mejorar la calidad y satisfacción de cara al consumidor. Esta dirigido a mejorar los aspectos organizativos de una empresa.

- ✓ **ISP**: (en inglés ***Internet Service Provider*** - Proveedor de Servicios de Internet), Un ISP ofrece a los usuarios un servicio a través del cual estos pueden acceder a Internet.
- ✓ **JAR:** (en inglés Java Archive- Archivo Java), contiene la información relaciona da con un proyecto Java, empaqueta las clases java de un proyecto.
- ✓ **Java Scripts:** Es un lenguaje interpretado orientado a la programación Web, se ejecuta en la capa cliente de dichas aplicaciones. Permite crear dinamismo en las páginas Web.
- ✓ **JSP**: Java Server Pages (JSP) es la tecnología para generar páginas web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java.
- ✓ **MVC :** Es un patrón de diseño que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres capaz, de forma que las modificaciones en una pueden ser hechas con un mínimo impacto en las otras.
- ✓ **Lenguaje no Procedural**: Lenguaje de manejo de datos, operan sobre conjuntos de registros. En los lenguajes no procedurales se especifica qué datos deben obtenerse sin decir cómo hacerlo. El lenguaje no procedural más utilizado es el SQL (Structured Query Language).
- ✓ **Protocolo:** Se le llama protocolo de red o protocolo de comunicación al conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red.
- ✓ **Plugin:** Es un programa que interactúa con otro programa para aportarle una función o utilidad específica, este programa adicional es ejecutado por la aplicación principal. Se usan para añadir nuevas funcionalidades sin afectar las ya existentes.
- ✓ **Plataforma:** Base, elemento de apoyo
- ✓ **Servlets de Java:** Los servlets son objetos que corren dentro del contexto de un servidor de aplicaciones y extienden su funcionalidad.
- ✓ **SOA:** La Arquitectura Orientada a Servicios (en inglés Service-oriented architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.
- ✓ **SQL:** El Lenguaje de Consulta Estructurado (**S**tructured **Q**uery **L**anguage) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.
- ✓ **SUN:** Sun Microsystems, empresa informática fabricante de semiconductores y software. Es la compañía que creó y mantiene el lenguaje Java en sus diferentes ediciones.

- ✓ **UML**: "Unified Modeling Language" Lenguaje gráfico que brinda un vocabulario y reglas para especificar, construir, visualizar y documentar los artefactos de un sistema utilizando el enfoque orientado a objetos.
- ✓ **WAR**: (en inglés Web Archive- Archivo Web), contiene la información relaciona da con un proyecto Web.
- ✓ **XML:** del inglés (eXtensible Markup Language- Lenguaje de Marcado Ampliable o Extensible). Una de las principales funciones consiste en separar la estructura del contenido en paginas HTML, permitiendo el desarrollo de vocabularios modulares, compatibles con cierta unidad y simplicidad del lenguaje.
- ✓ **XSL**: (siglas de Extensible Stylesheet Language - "lenguaje extensible de hojas de estilo") es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio específico.