Pontificia Universidad Católica de Valparaíso

Facultad de Ingeniería

Escuela de Ingeniería Informática

ESTRUCTURA Y ORGANIZACIÓN DE UNA FÁBRICA DE SOFTWARE

Autor:

Andrés Cabach Pisani

Informe final del Proyecto para optar al Título profesional de Ingeniero Civil en Informática

Profesor guía:

Broderick Crawford Labrín

Diciembre – 2006

Dedicado especialmente a mis queridos padres, por su incansable esfuerzo y apoyo.

Agradecimientos

A Exertus S.A. por el espacio otorgado para desarrollar esta memoria y a mi Profesor Guía por su constante apoyo.

Andrés Cabach Pisani

Resumen

En el desarrollo de software se estila asignar un equipo de trabajo para cada cliente/proyecto, en donde cada equipo efectúa procesos de desarrollo adecuados a éste. El problema es cómo generar una estructura y organización que se dedique al desarrollo de software especializado, sin tener que contar con grupos de trabajo en terreno variables, dependiendo de la carga de trabajo que requiera cada cliente. Para esto se propone utilizar una Fábrica de Software, a través de la formación de un equipo de trabajo constante, el cual trabajará independientemente del cliente, respondiendo a los pedidos hechos por los interlocutores empresa-clientes. En este trabajo se presenta una estructura y organización para la Fábrica de Software de Exertus, en donde se ahonda en puntos como el ciclo de vida del proyecto de software, el proceso de administración de la calidad y reutilización

de componentes, la utilización de herramientas de productividad y la definición de indicadores para el control de los productos y procesos al interior de la fábrica.

Abstract

In the software development, it's used to assign a specialized team for each client/project, where each team produces customized software development processes. The problem is how to generate a structure and organization that could dedicate to develop specialized software, without having to rely in different teams in terrain, depending of the work charge each client will need. For this purpose, it's proposed to use a Software Factory, by creating a constant team, wich will work independently from the client, responding to the requests made by the company-client speakers. In this work, a structure and organization for the Exertus Software Factory is presented, focusing in subjects like the life cycle of the software project, the quality administration and component reutilization processes, the use of productivity tools and finally, the definition of control indicators for products and processes inside the factory.

Capítulo 1
Introducción

1.1 Introducción

Las organizaciones de software experimentan presiones considerables para profesionalizar sus operaciones, se piden cada vez más procesos de desarrollo transparentes y de rápida retroalimentación para obtener una alta productividad y mejor calidad en los servicios y productos entregados. Durante más de treinta años, los ingenieros de software se han ocupado de este desafío enfatizando con ciertas innovaciones las operaciones de software. En todas partes de este período, la idea de la Fábrica de Software ha seguido surgiendo como una especie de última respuesta a estas necesidades. M. A. Cusmano (1989) proporciona una interpretación histórica del concepto de Fábrica de Software y experiencia. R. W. Bemer era probablemente el defensor más temprano, que sugiere en 1968 que en General Electric se desarrolle una Fábrica de Software para realzar la productividad del programador mediante instrumentos estandarizados, una interfaz computarizada y una base de datos con datos históricos para el control financiero y administrativo . Aproximadamente al mismo tiempo, M. D. McIlroy de AT&T propuso un concepto parecido a una fábrica, que acentúa la reutilización sistemática de código, en el desarrollo de programas nuevos .

La primera empresa en el mundo que utilizó una organización de software como una fábrica fue Hitachi en 1969, mientras la segunda Fábrica de Software fue establecida por un líder estadounidense en el campo de software a la medida, la System Development Corporation, durante 1975-1976. La explosión de la Fábrica de Software continuo, en Japón donde NEC, Toshiba, y Fujitsu lanzaron sus propios esfuerzos de la fábrica durante 1976-1977. Más recientemente, Mitsubishi y Nippon Telephone and Telegraph ha iniciado esfuerzos parecidos a una fábrica, y en 1985 en el ministerio japonés de

comercio internacional e industria, ha comenzado el proyecto de SIGMA nacional como un esfuerzo cooperativo para desarrollar una infraestructura de cual producir el software de alta calidad en gran cantidad.

Aún cuando existen diversos puntos de vista con respecto al significado de la idea de la Fábrica de Software, tal como la organización industrializada de software, la fábrica genérica de software, la fábrica de componentes basada en la experiencia y la organización de software madura, con estas expresiones no se está aludiendo al sentido físico de la producción del software como se conoce habitualmente, aunque exista alguna definición en ese sentido, sino que la Fábrica de Software se refiere al sentido de producir con rapidez y calidad a través de procesos conocidos, repetibles y gerenciables, y principalmente, mejorables continuamente, no sólo por la incorporación de técnicas y herramientas en el desarrollo del software, sino porque se mantiene constantemente el foco sobre el mejoramiento del proceso de producción y cada uno de los pasos que esto acarrea.

El término fábrica indica un compromiso a largo plazo, esfuerzos integrados (por encima de proyectos individuales) para mejorar las operaciones relativas al software. Entre los significados de la Fábrica de Software, se menciona, mejorar la efectividad del proceso, reducir la cantidad de retrabajo y reusar el ciclo de vida de los productos . De tal forma se pueden obtener mejores resultados en menor tiempo y con menos costos. Además, es una industria en la cual las actividades del desarrollo de software son predecibles, lo que significa que los costos estimados y compromisos de cronograma pueden ser satisfechos. Por lo demás su confianza, se debe a que la capacidad del proceso es conocida, y con mejoramiento continuo ya que la atención constantemente se concentra en mejorar el proceso y que el conocimiento y las habilidades para mejorar están establecidas. Todo esto debería ser logrado a través de la atención al proceso de mejoramiento de las operaciones y no a los métodos o herramientas, lo que supone la utilización de métricas y técnicas aplicadas al proceso del software y apuntaría a la satisfacción de la calidad.

1.2 Justificación del Proyecto

Exertus es una empresa, actualmente en convenio tecnológico con el Instituto Internacional para la Innovación Empresarial (3IE) y apoyado por la CORFO en la aprobación de la línea 1 y 2 de recursos para proyectos de innovación tecnológica, que asesora a sus clientes en la integración vertical de los siguientes servicios: Definición de agenda estratégica de Tecnologías de Información (TI), ejecución de proyectos de software y apoyo a la continuidad operacional a las soluciones provistas.

Exertus se encarga de guiar al cliente en obtener una mirada común, alineando la estrategia TI con su estrategia de negocio, esto es, mejoramiento continuo de procesos de negocios, definición de arquitectura de soluciones e integración de sistemas. Una vez alcanzados los acuerdos respecto a la visión de futuro que se quiere desarrollar, asesora al cliente en la materialización de la trayectoria definida, apoyándolo fuertemente en la definición y posterior ejecución de los proyectos de allí derivados.

Exertus es una empresa orientada a proveer soluciones integrales TI, que apunta a un nicho de mercado no cubierto actualmente por empresas de desarrollo, esto es, la brecha existente entre los sistemas de Planeación de Recursos de la Empresa, en inglés, Enterprise Resource Planning (ERP) y los sistemas de control automático, es decir, se orienta al desarrollo de sistemas operacionales y gestión operacional en empresas que hacen uso intensivo de las TI, en particular, a empresas del sector productivo minero, que es donde los socios poseen el conocimiento.

Exertus basa su oportunidad de negocio en la ausencia de empresas informáticas especialistas y con experiencia en control y gestión operacional en el mercado objetivo. Además, las que existen no están organizadas en forma eficiente y con las competencias técnicas adecuadas para satisfacer su demanda de servicios.

La diferenciación y en definitiva la innovación que pretende implementar este negocio es, otorgar productos de software de alta calidad y hecho a medida, en todas aquellas

empresas que hacen uso intensivo de las TI en Chile, específicamente desarrollar productos de software orientados al control y gestión operacional.

La estrategia competitiva es alta segmentación focalizada en el espacio no cubierto por los paquetes de software de clase mundial y las soluciones de control automático que las empresas poseen. Para lograr este objetivo, Exertus utilizará:

Experiencia (más de 14 años de sus socios fundadores) en proyectos de software de apoyo a la operación en el mercado objetivo: Sector industrial minero. Lo que permite una integración vertical de servicios, esto es, visión global y unificadora de la estrategia TI, tendiente a un acoplamiento con el negocio de largo plazo del cliente.

Basado en la premisa "La calidad del producto final está altamente influenciada por la calidad del proceso utilizado para desarrollarlo y mantenerlo", Exertus creará una Fábrica de Software, que en el corto plazo pondrá en proceso de certificación según el marco conceptual de la Integración del Modelo de Capacidad y Madurez, en inglés, Capability Maturity Model Integration (CMMI), formada por equipos de trabajo interrelacionados que participen desde la concepción inicial hasta la ejecución de cada una de las etapas del ciclo de vida del proyecto. Luego, los procesos de desarrollo de Exertus serán repetibles, medibles y de resultado predecible, permitiendo una acertada estimación de plazos y recursos.

Contactos a alto nivel en empresas productivas del ámbito nacional.

Acceso a una fuente de recursos humanos de la más alta calidad a nivel nacional, representada fundamentalmente por las universidades tradicionales de la quinta región, estableciendo un fuerte compromiso con el desarrollo del polo tecnológico local.

En resumen, los elementos centrales de Exertus, como negocio, son:

1. Asesoramiento a la alta gerencia en el alineamiento de la estrategia TI con la estrategia de negocio del cliente, permitiendo una integración vertical de los servicios prestados.

2. Fábrica de Software.

3. Alta segmentación en el nicho de mercado objetivo. Es decir, cubrir la brecha existente entre los sistemas de control automático y los ERP, desarrollando sistemas operacionales y de gestión operacional en la industria Minera.

4. Compromiso con el desarrollo del polo tecnológico local. Por ello, la contratación de recursos humanos, por parte de Exertus, será especialmente seleccionada de as universidades locales de la V región.

Luego, dado el tipo de trabajo a realizar y el mercado objetivo es que se requiere generar una organización especializada en el desarrollo de software capaz de cubrir las necesidades de distintos clientes, en distintas plataformas de desarrollo y, ubicadas en distintas zonas geográficas del país (en el futuro, Exertus pretende enfocarse al extranjero también) proveyendo de software especializado en el control y gestión operacional.

Comúnmente se estila asignar un equipo de trabajo para cada cliente (o proyecto), dificultando el proceso, ya que a medida que la cartera de clientes/proyectos aumenta, eventualmente, aumenta también en alguna proporción los equipos de trabajo necesarios. Cada equipo efectúa procesos de desarrollo de software adecuados al cliente, no permitiendo la creación de un proceso único y estándar para el desarrollo del software.

Luego, el problema es cómo generar una estructura y organización que se dedique al desarrollo de software especializado, sin tener que contar con grupos de trabajo en terreno variables, dependiendo de la carga de trabajo que requiera cada cliente.

Exertus propone una organización funcional a este objetivo, es decir, utilizará una Fábrica de Software, la cual debiera funcionar bajo el esquema de trabajo mostrado en la figura 1.1.

Se propone la formación de un equipo de trabajo constante, el cual trabajará independientemente del cliente, respondiendo a los pedidos hechos por los interlocutores empresa-clientes, en este caso, jefes de proyecto, que puede ser el mismo para más de un

cliente. Estos jefes de proyecto, serán la interfaz necesaria entre el cliente y el equipo de trabajo. Desde el punto de vista externo a la Fábrica de Software, se verá cada proyecto independiente uno de otro y así lo percibirá el cliente, lográndose la personalización necesaria, mediante el funcionamiento coordinado y estandarizado al interior de la fábrica.

EXERTUS S.A.

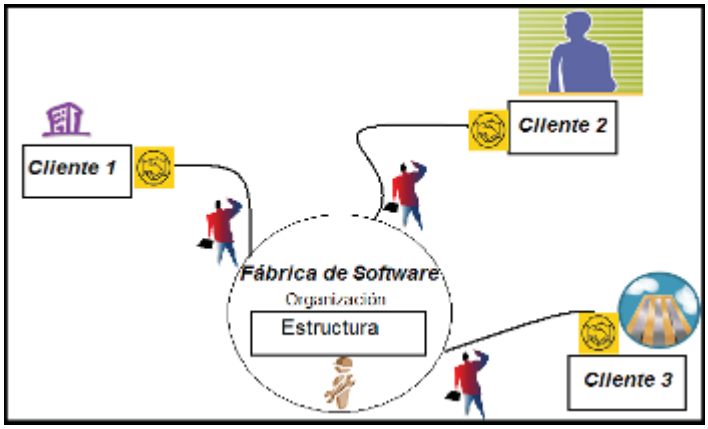


Figura 1.1: Esquema de trabajo de la Fábrica de Software

Lo que se busca es formar una estructura y organización de una Fábrica de Software que funcione con la dinámica de trabajo ya mencionada, en donde el equipo de trabajo realice el ensamblaje de componentes, nuevos o ya creados (reutilizables), para satisfacer las necesidades de soluciones TI de los clientes, con un nivel de calidad óptimo a precios competitivos dentro del mercado local e internacional.

Este trabajo se encarga de entregar la base teórica para la definición de una Fábrica de Software, a partir de la cual se generará una propuesta concreta de estructura y organización para la Fábrica de Software de Exertus dando solución a los problemas planteados.

1.3 Objetivos Generales

Construir una Estructura y Organización de Fábrica de Software.

Implementar la Estructura y Organización de la Fábrica de Software en una empresa para el desarrollo de la minería.

Para alcanzar éstos objetivos generales será necesario llevar a cabo los objetivos específicos que se presentan a continuación.

1.4 Objetivos Específicos

Efectuar un estudio teórico de Fábrica de Software.
Crear una estructura funcional de Fábrica de Software.
Definir los perfiles funcionales de Fábrica de Software.
Desarrollo de un Esquema de Fábrica de Software para la fábrica.
Diseñar ciclo de vida del producto de software de la fábrica.
Análisis de conceptos de ingeniería industrial a la producción de software.
Modelar procedimientos de operación de la Fábrica de Software.
Implementar prototipos de proyectos de la Línea de Productos.

1.5 Organización del texto

La organización de la presente memoria de título se divide en capítulos de la siguiente manera:

En el CAPÍTULO 2 se realiza un estudio teórico para conocer y entender cada una de las ventajas y características que conlleva un servicio de Fábrica de Software, además de sus requerimientos de implementación para idear un buen funcionamiento interno de la fábrica.

En el CAPÍTULO 3 se presenta una estructura funcional para la Fábrica de Software, es decir, una estructura que muestra todas las funciones y los procesos necesarios que se deben desarrollar al interior de esta. Además se definen los perfiles que deben tener los integrantes de la fábrica, junto con describir sus cargos.

En el CAPÍTULO 4 se ahonda en el ciclo de vida del proyecto de software, el cual describe el propósito, entradas, actividades, salidas y actores para cada una de las etapas que comprende el desarrollo de un producto en la Fábrica de Software.

La administración de la calidad del producto de software se detalla en el CAPÍTULO 5. En este capítulo se muestran las tareas, con sus entradas y salidas, a ejecutar en cada una de las etapas que comprende el ciclo de vida del proyecto de software.

La reutilización de componentes y el manejo de herramientas de productividad en la fábrica son especificados en el CAPÍTULO 6. En este capítulo se presentan los procesos necesarios para el manejo de una biblioteca de componentes, además de un análisis de las distintas herramientas de productividad que apoyan al proyecto de software.

En el CAPÍTULO 7 se detallan procesos críticos al interior de la fábrica que no quedan de manifiesto en los capítulos anteriores, para esto se describe su propósito, problemática y solución, acompañado de un flujo de trabajo que grafica la secuencia de pasos a seguir y los actores que en ella actúan.

En el CAPÍTULO 8 se definen las diferentes métricas orientadas a los procesos y producto para la Fábrica de Software. Estas métricas serán la base para el posterior análisis de la implementación de los prototipos.

Finalmente en el CAPÍTULO 9 se seleccionan las métricas a utilizar en los prototipos, además de efectuar el análisis del dominio de estos. Se especifica el desarrollo de los 2 prototipos implementados y la aplicación de las métricas en ellos.

Capítulo 2
Aplicando Conceptos de Ingeniería Industrial a la Producción de Software

Al remontarse a fines de los 60, se pueden encontrar algunos documentos donde ya se hacía mención al deseo de ver masificada una concepción industrializada del proceso de desarrollo del software a través de reutilización de componentes, los cuales fueran robustos y aseguraran calidad, además del uso de catálogos con componentes estandarizados para escoger aquellos necesarios y así mecanizar la labor del desarrollo del software, transformándolo en un proceso único, asegurando de esta forma la calidad del producto .

El siguiente paradigma de desarrollo industrializa el desarrollo de software, moviendo la industria hacia la madurez. Otras industrias aprendieron como personalizar y montar componentes estándar para producir productos similares pero distintos; estandarizar, integrar y automatizar procesos de producción; desarrollar herramientas genéricas y configurarlas para automatizar tareas repetitivas; influenciar las relaciones con los clientes y distribuidores reduciendo gastos y riesgos; automatizar la producción de variados productos usando la Línea de Productos y distribuir la producción a través de cadenas de suministro compuestas por distribuidores altamente especializados e interdependientes, como por ejemplo podría ser una industria de automóviles .

2.1 Economía de Reutilización

Para entender mejor la analogía entre una industria de bienes físicos y la industria del software se debe comprender la economía de reutilización. La reutilización de soluciones con subproblemas comunes en un dominio dado, puede reducir el costo total de

solucionar múltiples problemas en el dominio. La reutilización también puede reducir el plazo de desarrollo total y mejorar la calidad del producto. En otras palabras, se puede pensar en las soluciones TI, desde una perspectiva de la inversión, como el activo de producción (lenguajes, herramientas, patrones y marcos de trabajo). Se llama a este activo de producción el activo de implementación, ya que son usados directamente en la implementación del producto. El activo de implementación es acompañado por el activo de proceso que apoya su uso, como casos de prueba, documentación de usuario o de diseño. Finalmente, el activo de proceso es apoyado por herramientas y otros activos de automatización. Se puede definir un activo de producción como un artefacto de software que explícitamente tuvo la intención de proporcionar un retorno de la inversión a través de la reutilización. Para comprender las ventajas descritas sobre la reutilización, se debe adoptar un enfoque más maduro, que implica la identificación de los subproblemas comunes en un dominio dado y el desarrollo de colecciones integradas de activos de producción, que puede ser reutilizado para solucionar aquellos problemas predecibles. En este enfoque, se hablará de reutilización sistemática, que aumentará los rendimientos de los activos de producción en la inversión, esto gracias a las economías de escala y alcance.

2.2 Economía de Escala y Alcance

Las economías de escala, surgen cuando múltiples casos idénticos de un diseño son producidos en conjunto de mejor manera que individualmente, como se muestra en la figura 2.1

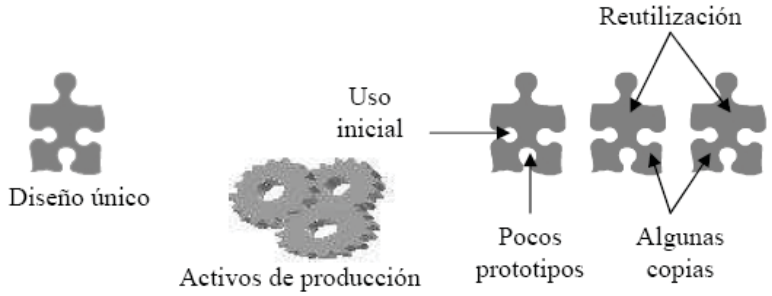


Figura 2.1: Economía de escala

El Activo de Producción, como máquinas y herramientas, se usa para producir múltiples productos idénticos. Un diseño, generalmente, es creado con casos iniciales, llamados prototipos, por un proceso con uso intensivo de recursos, llamado desarrollo y realizado por ingenieros. Muchos casos adicionales, llamados copias, generalmente son producidos por otro proceso, llamado producción, realizados por máquinas para satisfacer la demanda del mercado a un precio bajo. Se puede pensar en las economías de escala como la reducción del costo, solución del mismo problema, de la misma manera, múltiples veces, reduciendo el costo de producir cada solución.

Las economías de alcance, surgen cuando múltiples casos similares de un diseño particular (pero distintos) son producidos en conjunto, de mejor forma que individualmente, como se muestra en la figura 2.2.

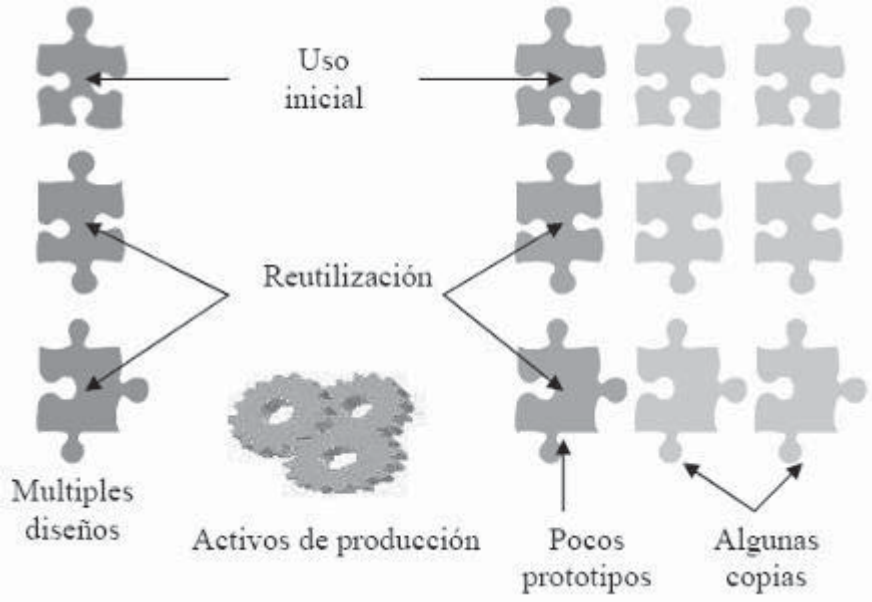


Figura 2.2: Economía de alcance

Esto se puede apreciar en la fabricación de un automóvil. Múltiples diseños de producto similares, pero distintos, a menudo son desarrollados con componentes de diseños existentes, como chasis, cuerpo, interior, etc. Los modelos a menudo son desarrollados variando rasgos como el motor y otros, en un diseño existente. En otras palabras, las

mismas prácticas, procesos, herramientas y materiales son usados para diseños y prototipos de múltiples productos similares pero distintos.

En mercados donde las copias son la masa producida (por ejemplo, programas de escritorio, procesadores de texto), el software genera las economías de escala al igual que la fabricación del automóvil. Desde luego, hay diferencias importantes entre las dos industrias, como el costo de producción y consumo de copias. En la fabricación de automóviles, como en otras industrias pesadas, la producción de copias consume recursos caros, mientras en el software se consumen medios digitales baratos. Otra diferencia entre el software y la fabricación de automóviles, es que el software, generalmente, es más caro de producir, debido a la personalización. En el software, la personalización puede variar según sea la preferencia de configuración para el programa, mientras los automóviles generalmente requieren mínima personalización .

En algunos mercados, como las empresas donde las aplicaciones de gestión son desarrolladas para obtener ventaja competitiva, las meras copias de un software no son muy utilizadas, es por esto, que en aquellos mercados, las economías de alcance son el medio primario de reutilización sistemática.

Se utiliza la industrialización de desarrollo del software para explotar las economías de alcance, sobre todo para mercados con distribución limitada. Las Fábricas de Software y cadenas de suministro son pasos claves en el camino a la industrialización.

2.3 Integración de Innovaciones Críticas

Las Fábricas de Software integran innovaciones críticas para definir un proceso altamente automatizado de desarrollo de software, que explota las economías de alcance.

Las innovaciones propuestas por la Fábrica de Software abarcan lo siguiente :

Construcción de familias de software similar. Esta actividad supone el análisis y diseño de una arquitectura común para un conjunto de sistemas y el desarrollo de un marco de trabajo que apoye esta arquitectura.

Ensamblado de componentes. La construcción de un nuevo sistema supone el uso de ensamblado y/o configuración de los componentes proporcionados por el marco de trabajo.

Desarrollo de lenguajes de modelado y herramientas específicas para el dominio. Los desarrolladores utilizan estos lenguajes para describir los requisitos específicos de un miembro de la familia de sistemas. A partir de estas especificaciones se genera automáticamente el código.

Uso de una planificación basada en requerimientos y guía activa. Todos los pasos del proyecto de desarrollo deben realizarse de acuerdo a un proceso bien definido y adaptado al dominio.

Cuando cada una de estas tecnologías está bastante madura para ser desplegada en una escala industrial, su integración produce un todo que es mayor que la suma de sus partes. Se sugiere que las Fábricas de Software apliquen estas innovaciones críticas a un patrón de cuatro partes, aparecido repetidamente en la evolución del desarrollo de software, el cual implica :

Desarrollo de marcos de trabajo para la implementación inicial de productos basados en estilos comunes arquitectónicos.

Desarrollo de herramientas basadas en lenguajes para apoyar el desarrollo de productos, adaptando, configurando y ensamblando componentes basados en marcos de trabajo.

Usar las herramientas para atraer clientes y responder rápidamente a las exigencias de cambios, construyendo software incrementalmente y manteniendo el software funcionando cuando los cambios estén hechos.

Capturar decisiones de diseño, de una forma que directamente produzca ejecutables.

Las Fábricas de Software, usan herramientas a base de lenguajes para mapear la variedad de requerimientos existentes y completar un marco de trabajo. Este mapeo efectuado por arquitectos y desarrolladores, proporciona una guía durante el desarrollo realizado por la

Fábrica de Software, la cual después es reutilizada durante el desarrollo del producto, a menudo por desarrolladores mucho menos experimentados. Herramientas basadas en lenguajes también promueven agilidad, capturando requerimientos en formas que los clientes entiendan mejor, permitiendo implementar cambios de requerimientos de forma rápida. Lamentablemente, el desarrollo de tales herramientas para automatizar el proceso de ensamblaje está actualmente más allá del alcance de la mayor parte de las organizaciones. Si esta parte del modelo puede ser hecha tan rentable como las demás, entonces se estará cerca de la realización de la visión descrita.

Se consideran las 4 innovaciones mencionadas en un comienzo, como los pilares principales para llevar a cabo una Fábrica de Software. Se puede ver que estas innovaciones no son exclusivas de la Fábrica de Software, mas bien son conceptos y prácticas de las cuales se hace mención hace ya varios años de forma separada, por ejemplo, el ensamblado de componentes es una práctica habitual para los desarrolladores que utilizan el Desarrollo de Software Basado en Componentes, en inglés, Component Based Software Development (CBSD).

Lo interesante en las Fábricas de Software, es pensar estas innovaciones como un conjunto de medidas que industrializan el desarrollo de software.

2.4 ¿Qué es una Fábrica de Software?

Una fábrica es una instalación de producción altamente organizada que produce los miembros de una línea de producción usando partes estandarizadas, instrumentos y procesos de producción.

Una Fábrica de Software es una Línea de Productos de software configurada por herramientas genéricas y procesos que usa una plantilla de una Fábrica de Software, basada en un Esquema de Fábrica de Software, para automatizar el desarrollo y el mantenimiento de un producto, adaptando, ensamblando y configurando componentes a base de marcos de trabajo .

Los elementos centrales de una Fábrica de Software son un Esquema de Fábrica de Software y una Plantilla de Fábrica de Software basada en el Esquema de Fábrica de Software. La Plantilla de Fábrica de Software configura herramientas genéricas, procesos y contenido, para formar un producto de una familia de productos dada. Un Esquema de Fábrica de Software se muestra en la figura 2.3, donde se muestra cada una de las partes de una Fábrica de Software. Se hablará de cómo construir una Fábrica de Software y como construir un producto de software usando este modelo.

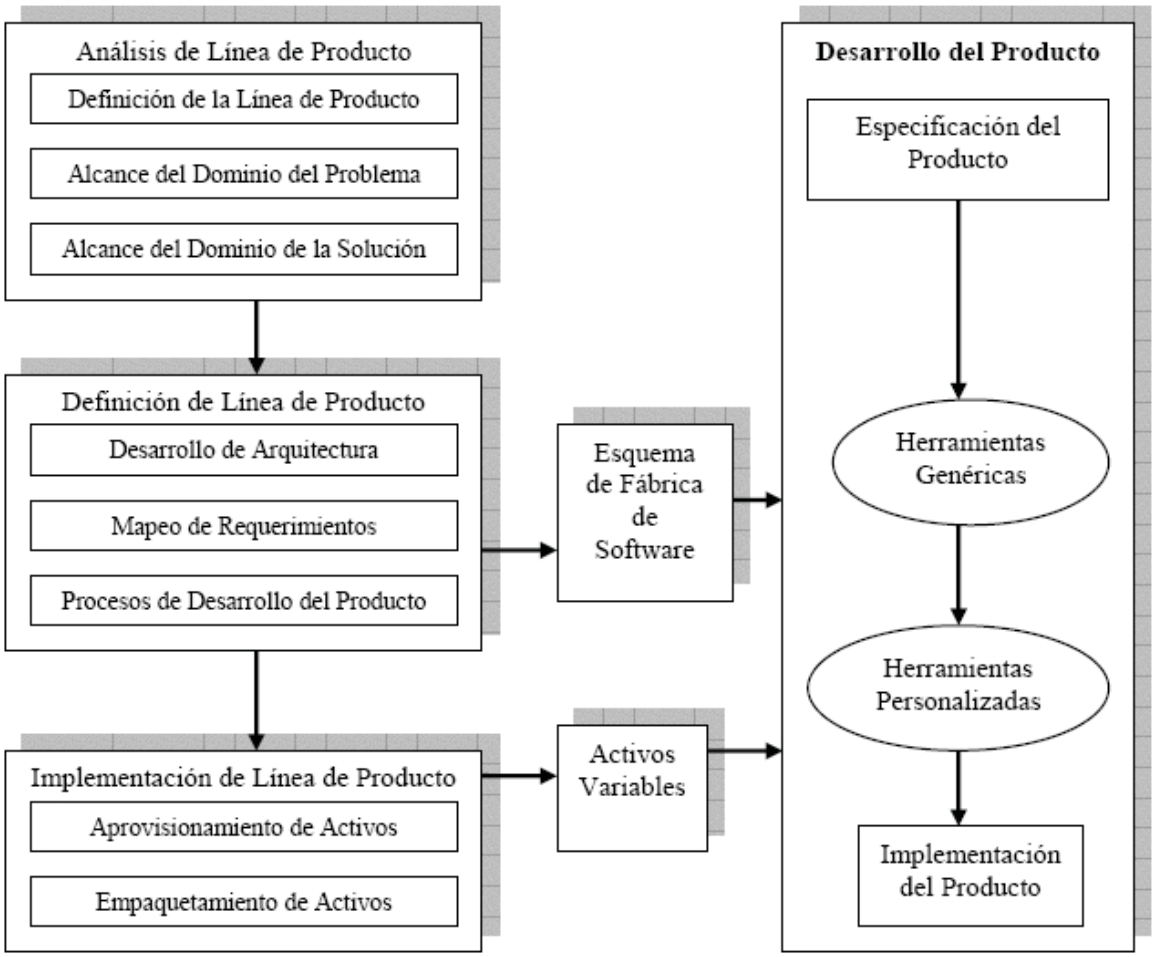


Figura 2.3: Esquema de una Fábrica de Software

2.5 Esquema de una Fábrica de Software

La necesidad de un Esquema de Fábrica de Software se hace evidente cuando se necesita un modo de clasificar y resumir Artefactos de Desarrollo, tales como: modelos, archivos de configuración, construcción de script, archivos de código fuente, archivos de Lenguaje de Consulta Estructurado, en inglés, Structured Query Language (SQL), archivos de localización, archivos de configuración para la instalación y definiciones de casos de prueba; además de un modo de definir sus relaciones, de manera de mantener la consistencia entre ellos; en otras palabras lo que se busca es de manera esquemática representar la arquitectura de la Línea de Productos.

Tanto los marcos arquitectónicos como las metodologías de modelado acostumbran ordenar las diferentes perspectivas de una arquitectura en términos de vistas, como se muestra en la tabla 2.1. La mayoría de los marcos de trabajo y estrategias reconoce entre tres y seis vistas, que son las que se incluyen en el cuadro. Una vista es el subconjunto resultante de practicar una selección o abstracción sobre una realidad, desde un aspecto determinado .

Zachman (Niveles)	TOGAF (Arquitecturas)	4+1 (Vistas)	[BRJ99] (Vistas)	POSA (Vistas)	Microsoft (Vistas)
Alcance	Negocios	Lógica	Diseño	Lógica	Lógica
Empresa	Datos	Proceso	Proceso	Proceso	Conceptual
Sistema lógico	Aplicación	Física	Implementación	Física	Física
Tecnología	Tecnología	Desarrollo	Despliegue	Desarrollo	

Representación		Casos de uso	Casos de uso		
Funcionamiento					

Tabla 2.1: Vistas en los marcos de referencia

La recomendación del Instituto de Ingenieros Eléctricos y Electrónicos, en inglés, Institute of Electrical and Electronics Engineers (IEEE) Std 1471-2000 procura establecer una base común para la descripción de arquitecturas de software e implementa, para ello tres, términos básicos, arquitectura, Vista y Punto de Vista. La arquitectura se define como la organización fundamental de un sistema, encarnada en sus componentes, las relaciones entre ellos y su entorno y los principios que gobiernan su diseño y evolución. Los elementos que resultan definitorios en la utilidad, costo y riesgo de un sistema son en ocasiones físicos y otras veces lógicos. En otros casos, son principios permanentes o patrones que generan estructuras organizacionales duraderas. Términos como Vista o Punto de Vista son también centrales. En la recomendación se los utiliza en un sentido ligeramente distinto al del uso común. Aunque reflejan el uso establecido en los estándares y en la investigación de ingeniería, el propósito del estándar es introducir un grado de formalización homogeneizando informalmente la nomenclatura. El estándar IEEE 1471 no delimita el número posible de Vistas, ya que se estima que no puede haber acuerdo en ello, pero señala lineamientos para su constitución y considera que una Vista es a un Punto de Vista como una clase es a un objeto.

La estrategia de arquitectura de Microsoft define, en relación con las conceptualizaciones más generalizadas, cuatro perspectivas llamadas también arquitecturas: negocios, aplicación, información y tecnología .

La perspectiva de negocio. Describe como un negocio trabaja. Esto incluye estrategias de negocio y proyectos, para mover la organización de su estado corriente a un estado previsto futuro. Esto incluirá lo siguiente:

- Los objetivos de alto nivel de la empresa y metas.
- Los procesos de negocio realizados por la empresa entera o una parte significativa de ella.
- Las funciones de negocio ejecutadas.
- Estructuras principales de organización.
- Las relaciones entre estos elementos.

La perspectiva de aplicación. Define la cartera de aplicaciones de la empresa y es centrada en la aplicación. Esta vista incluirá:

- Las descripciones de los servicios automatizados que apoyan los procesos de negocio.
- Las descripciones de la interacción e interdependencias (interfaces) de los sistemas de aplicación de la organización.
- Planes para desarrollar aplicaciones nuevas, la revisión de aplicaciones viejas basadas en los objetivos y metas de la fábrica.

La perspectiva de aplicación. Puede representar servicios a través de la organización, información, funcionalidad, uniendo a los usuarios de diferentes habilidades y funciones de trabajo, para alcanzar objetivos comunes de negocio.

La perspectiva de la información. Describe lo que la fábrica tiene que saber para controlar sus procesos de negocio y operaciones. Esto incluye:

- Modelos de datos estándar.
- Política de administración de datos.

Las descripciones de los patrones de producción y de consumo de información en la organización.

La perspectiva de la información también describe como los datos son unidos en el workflow, incluyendo conjuntos de datos estructurados como bases de datos y conjuntos de datos sin estructura: documentos, hojas de cálculos y las presentaciones que existen en todas partes de la organización.

La perspectiva de tecnología. Presenta el hardware y el software que apoya la organización. Esto incluye, pero no es limitado a:

Hardware de escritorio y de servidor.
Sistemas operativos.
Componentes de conectividad de red.
Impresoras.
Módems.

La perspectiva de tecnología proporciona una lógica, una descripción de la infraestructura y los componentes de sistema que son necesarios para apoyar las perspectivas de aplicación y de información. Esto define el conjunto de normas de tecnología y los servicios para desarrollar la misión del negocio.

Aunque puede haber muchas perspectivas, las perspectivas ven solo una arquitectura de la fábrica. El valor de la arquitectura de la fábrica no está en ninguna perspectiva individual, sino en las relaciones, interacciones y dependencias entre ellas.

Cada arquitectura, a su vez, se articula en tres vistas que son (1) la Vista conceptual, cercana a la semántica de negocios y a la percepción de los usuarios no técnicos; (2) la Vista lógica, que define los componentes funcionales y su relación en el interior de un sistema, en base a la cual los arquitectos construyen modelos de aplicación que representan la perspectiva lógica de la arquitectura de una aplicación; (3) la Vista física,

que es la menos abstracta y que ilustra los componentes específicos de una implementación y sus relaciones.

Un método común es usar una matriz, como se muestra en la tabla 2.2. Las columnas definen intereses, mientras las filas definen los niveles de abstracción. Cada celda define un Punto de Vista, desde el cual se puede construir algún aspecto del software.

	NEGOCIO	**INFORMACIÓN**	**APLICACIÓN**	**TECNOLOGÍA**
CONCEPTUAL	Casos de uso y escenario. Metas de negocio y objetivos.	Entidades de negocio y sus relaciones.	Procesos de negocio. Factorización de servicios.	Distribución de servicios. Calidad de la estrategia de servicio.
LÓGICA	Modelos de workflow. Definición de roles.	Esquemas de mensajes. Documentos de especificación.	Interacción de servicios. Definición de servicios. Modelo de objetos	Tipos de servidores lógicos. Servicios de mapeos.
FÍSICA	Procesos de especificación.	Esquema de base de datos. Estrategia de acceso a los datos.	Diseño detallado. Diseño dependiente de la tecnología.	Servidores físicos. Instalación de software. Disposición

				de la red.

Tabla 2.2: Matriz para clasificar Artefactos de Desarrollo

Una vez que la matriz ha sido construida, se puebla con Artefactos de Desarrollo para un producto de software específico. Desde luego, la matriz puede ser usada para construir más que un producto de software. Antes de que haya sido poblada con Artefactos de Desarrollo específicos, se definen las listas de materiales de desarrollo, requeridas para construir los miembros de una familia de producto de software. Tomando en cuenta la Línea de Productos del software, se pude ir un paso más lejos y agregar la información de cada celda que identifica al Activo de Producción, usada para construir los Artefactos de Desarrollo requeridos desde aquella perspectiva, incluyendo el lenguaje de Modelado de Dominio Específico, en inglés, Domain Specific Modeling (DSM), patrones, marcos de trabajo y herramientas. Si también se identifican los microprocesos usados para cada celda, también se pude pensar en esta matriz como un marco de trabajo para producir a los miembros de la familia de productos.

La matriz en sí misma no es una innovación, lo que es nuevo es aplicar la matriz a una familia de productos, identificando el Activo de Producción para cada celda y definiendo un mapeo entre las celdas (y al interior de las mismas), que pueden ser usadas para automatizar, total o parcialmente, transformaciones de modelos, generación de código, aplicación de patrones, construcción de pruebas de falla, diseño de interfaz de usuario, el diseño de esquema de base de datos y muchas otras tareas de desarrollo. Nótese que los Puntos de Vista no sólo definen los lenguajes usados para desarrollar los artefactos, describen también requerimientos sobre los artefactos, por lo general, expresados como restricciones o patrones.

Se puede decir que una transformación de modelos es el proceso de convertir un modelo en otro . Las transformaciones se pueden clasificar en transformaciones verticales y horizontales. Las transformaciones verticales son aquellas que se aplican a un modelo para obtener otro expresado con un nivel de abstracción diferente, mientras que las transformaciones horizontales se aplican a un modelo para obtener otro del mismo nivel de abstracción .

Un Esquema de Fábrica de Software es en realidad un grafo dirigido, cuyos nodos son Puntos de Vista y cuyos bordes, son relaciones calculables entre Puntos de Vista, llamados mapeos. Esta estructura permite relacionar nodos, que habrían sido no adyacentes en una representación de matriz. Esto también relaja la obligación artificial impuesta por una representación de matriz, que requiere que cada Punto de Vista quepa en el esquema de clasificación, creando filas y columnas. Finalmente y, lo más importante, es que permite al esquema reflejar la arquitectura del software.

Desde luego, la matriz es una simplificación útil, porque es más fácil de visualizar. Por lo tanto, se usarán ambas representaciones, de manera intercambiable. Sin embargo, el grafo es la representación más exacta, siendo la matriz simplemente una abstracción útil del grafo. Se llama al grafo un Esquema de Fábrica de Software, porque describe los artefactos que deben ser desarrollados para producir un producto de software, tal como un esquema de base de datos describe las filas que deben ser creadas para poblar la base de datos.

Desde luego, la característica esencial de un Esquema de Fábrica de Software es que proporciona una separación multidimensional, organizada en asuntos basados en varios aspectos de los artefactos, como su nivel de abstracción, posición dentro de la arquitectura, la funcionalidad o calidades operacionales. También se puede pensar en un Esquema de Fábrica de Software como una receta para construir una familia de productos de software. Claramente, los Puntos de Vista definen los ingredientes y los instrumentos usados para prepararlos.

Un aspecto importante de usar un Esquema de Fábrica de Software es la configuración. Un Esquema de Fábrica de Software básico es una receta para construir una familia de productos de software, sin embargo, para cualquier proyecto dado, sólo se preocupa de aproximadamente un producto de software. Por lo tanto, se debe configurar el Esquema de Fábrica de Software que cree una receta para construir a un miembro específico de la familia de producto. Este proceso es similar a lo que los cocineros hacen cuando ellos modifican una receta antes de utilizarla, basado en exigencias especiales, como el número de porciones requeridas, los ingredientes, instrumentos o el tiempo disponible antes de

que la comida sea servida. Ahora se puede ver que para apoyar esta clase de configuración, un Esquema de Fábrica de Software debe contener tanto partes fijas como variables. Las partes fijas son las mismas para cada miembro de la familia de productos, mientras que partes variables se cambian para acomodarlo a exigencias únicas.

2.6 Plantilla de una Fábrica de Software

La Fábrica de Software descrita hasta ahora es aún sólo papel. Si todo lo que se tiene es el Esquema de Fábrica de Software, entonces es posible describir los activos usados para construir los miembros de la familia, pero en realidad no se tienen lo activos. Para construir un miembro de la familia de productos, se debe poner en práctica el Esquema de Fábrica de Software, definiendo el lenguaje de DSM , los patrones, marcos de trabajo y herramientas, luego, empaquetarlos y ponerlos a disposición de desarrolladores de productos. En conjunto, este activo forma una Plantilla de Fábrica de Software, que incluye el código y meta datos, los que pueden ser cargados en herramientas genéricas, como un Entorno de Desarrollos Integrados, en inglés, Integrated Development Environment (IDE), para automatizar el desarrollo y el mantenimiento de miembros de familia. Tal como un Esquema de Fábrica de Software, una Plantilla de una Fábrica de Software debe ser configurada para construir a un miembro de familia específico. Mientras la configuración de un Esquema personaliza la descripción de la Fábrica de Software, configurar una Plantilla, personaliza las herramientas y otras partes del ambiente de desarrollo usados para construir un miembro específico de la familia de productos.

2.7 Construcción de una Fábrica de Software

La construcción de una Fábrica de Software implica las siguientes actividades:

- La construcción de un Esquema de Fábrica de Software que describe la fábrica. Esto es una especialización de las dos actividades de una Línea de Productos:

- El análisis de la Línea de Productos, define qué productos de la Fábrica de Software se desarrollarán. La actividad central en el análisis de la Línea de Productos define un alcance que identifica los productos de software que serán desarrollados y mantenidos usando la Fábrica de Software. El alcance no es definido enumerando descripciones de productos específicos, más bien es describiendo el dominio del problema que definen a los productos. El análisis de la Línea de Productos produce, entre otras cosas, requerimientos de la Línea de Productos, que son organizados en los Puntos de Vista que formarán parte del Esquema de Fábrica de Software.

- El diseño de Línea de Productos define cómo la Fábrica de Software desarrollará productos dentro de su alcance. La actividad central en el diseño de Línea de Productos define una arquitectura para la familia de productos de software objetivo. Esta arquitectura es similar a la arquitectura para un solo producto, pero es diseñado para apoyar la variación entre los miembros de una familia. El diseño de Línea de Productos produce varios artefactos, incluyendo un mapeo de requerimientos, el proceso de desarrollo del producto y un plan para usar herramientas automatizando las partes del proceso de desarrollo del producto. Estos artefactos también contribuyen al Esquema de Fábrica de Software: la arquitectura es organizada en Puntos de Vista, un mapa de requerimientos es expresado usando relaciones entre Puntos de Vista y el proceso de desarrollo de producto es expresado como microprocesos conectados a los Puntos de Vista.

- Construcción de una Plantilla de Fábrica de Software que instancia la Línea de Productos de software. Esto es una especialización de implementación de la Línea de Productos. La Plantilla de Fábrica de Software empaqueta el Activo de Producción para la Fábrica de Software, incluyendo el Activo de Requerimiento como lenguajes de especificación; el Activo de Implementación, como los patrones, marcos de trabajo y componentes; el Activo de Proceso, como herramientas de desarrollo; Activo de Prueba, como equipos de prueba, pruebas de unidad y pruebas de integración; y el Activo de Instalación, como configuraciones para los equipos donde el producto será instalado.

Una vez que se ha desarrollado el primer corte de la Fábrica de Software, debe mantenerse continuamente, efectuando retroalimentaciones del desarrollo de los miembros de la familia para refinar su definición e implementación. La parte más cara de desarrollar en una Fábrica de Software la componen los lenguajes y las herramientas que automatizan el desarrollo del producto. Desde luego, hay muchos otros desafíos significativos, como la realización del análisis del dominio para el Esquema de Fábrica de Software, el desarrollo de la infraestructura para apoyar la creación, la configuración y la instalación de Plantillas de Fábrica de Software.

2.8 Construcción de un producto de software

El objetivo de construir una Fábrica de Software es, rápidamente desarrollar los miembros de su familia de productos. Construir un producto usando una Fábrica de Software es un caso especial de una Línea de Productos de software.

La construcción de un producto usando una Fábrica de Software implica las siguientes actividades:

El análisis del problema determina si realmente el producto está al alcance de la Fábrica de Software. Dependiendo si es apta, se puede decidir construir parte o todo fuera de la Fábrica de Software. Se puede decidir construir un producto que no cabe dentro de la Fábrica de Software, cambiando el Esquema de Fábrica de Software y la plantilla para acomodarlo.

Las especificaciones del producto definen los requerimientos del producto, según los requerimientos de la Línea de Productos. Pueden usarse distintos mecanismos de especificaciones del producto dependiendo de los diferentes requerimientos.

El diseño del producto mapea las diferencias de requerimientos para la arquitectura de la Línea de Productos y el proceso de desarrollo del producto, produciendo una arquitectura de producto y un proceso de desarrollo de productos, personalizados.

La implementación del producto implica actividades para la familia del producto, como el desarrollo y construcción de componentes y pruebas, la ejecución de

pruebas de unidad y el ensamblaje de componentes. Se puede usar una gama de mecanismos de implementación del producto para desarrollarlo, dependiendo de las diferencias del diseño.

La instalación del producto implica la reutilización de instalaciones fallidas, la creación e instalación y configuración de los recursos requeridos, junto con los ejecutables que se han generado.

Las pruebas del producto implican la creación o reutilización de Activos de Pruebas, incluyendo casos de prueba, equipos de prueba, conjunto de datos de prueba, scripts de prueba y aplicación de herramientas de apoyo.

Capítulo 3

Estructura Funcional de Fábrica de Software

Por efectos de marketing muchas empresas desarrolladoras de software se hacen llamar Fábrica de Software, lo que no implica que allí se efectúe un desarrollo basado en el esquema descrito en la tabla 2.2, por lo tanto, se habla de servicio de Fábrica de Software para referirse al desarrollo de software bajo este esquema en particular. El servicio de Fábrica de Software puede ser brindado por cualquier empresa de TI, al igual que otros servicios como externalización o consultorías, pasando a ser un servicio más dentro de la empresa. Debido a esto, se debe entender por estructura funcional de Fábrica de Software, la estructura necesaria para implementar el servicio en una empresa, la cual no equivale a la estructura de la empresa de TI.

Con la finalidad de crear una línea única de procesos, que abarque el desarrollo del software permitiendo un mayor control de cada una de sus etapas, se divide la estructura de Fábrica de Software en diferentes áreas respectivas al desarrollo del software. A continuación se presenta en la figura 3.1 la estructura de Fábrica de Software.

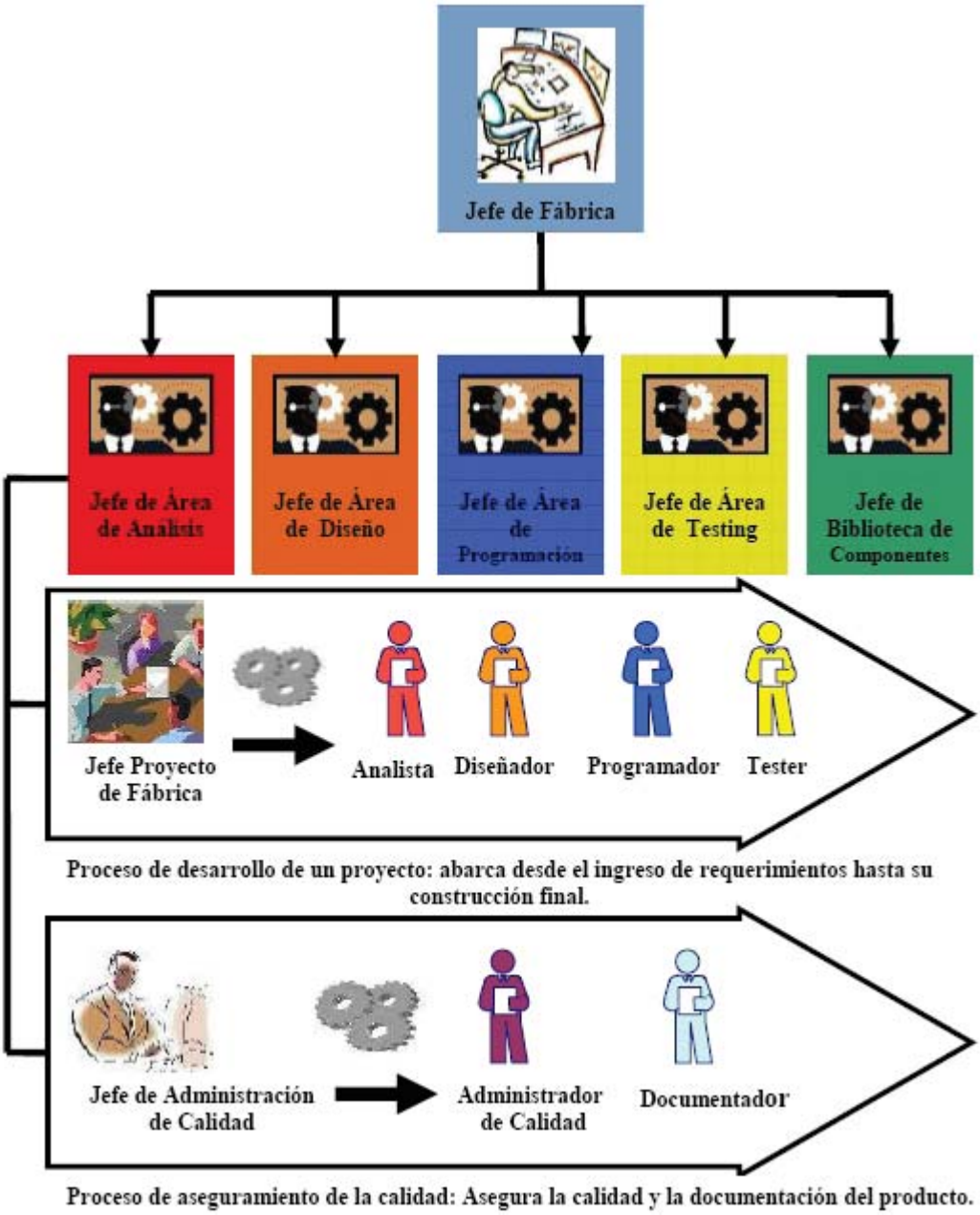


Figura 3.1: Estructura funcional de Fábrica de Software

La figura 3.1 muestra una clara jerarquía entre el jefe de fábrica y los jefes de las distintas áreas que comprende el desarrollo del software. Para entender el rol del jefe de fábrica, se puede ver la fábrica como una gran maquinaria que recibe requerimientos de un proyecto acorde a la Línea de Productos definida, para luego sacar un software como producto final. De este modo el jefe de fábrica es el responsable que esta máquina se

encuentre “a punto”, es decir, preocuparse de abastecer y asegurar el correcto funcionamiento de la Fábrica de Software, de manera de tener un espacio físico adecuado y con las herramientas adecuadas para el desarrollo del software.

Por otra parte, los jefes de las distintas áreas se encargan de cumplir con los más mínimos detalles relacionados con su actividad, para satisfacer de buena forma la etapa relacionada con su área. Se pueden distinguir cuatro áreas que están ligadas de forma secuencial, que es el caso del área de análisis, diseño, programación y testing, en donde la labor de éstas depende estrechamente del trabajo realizado por la anterior. Cada una de éstas alberga a los analistas, diseñadores, programadores y tester respectivamente. El jefe de biblioteca de componentes ejerce control sobre el administrador de componentes y el verificador y validador de componentes, los cuales se relacionan directamente con los programadores para ofrecer y recolectar componentes reutilizables para la construcción del software.

Como toda organización horizontal, no pone acento exclusivo en la segmentación de las actividades en términos de tareas que deben hacerse, si no más bien en aquellos procesos que cruzan todas estas funciones. En la figura 3.1 se puede ver reflejado con dos procesos transversales a las distintas áreas de la fábrica. Uno de estos procesos es el de desarrollo de un proyecto en particular, cuyo responsable es el jefe de proyecto de fábrica encargado de introducir los requerimientos de un proyecto en la Fábrica de Software para obtener como resultado final el sistema del cliente. Si bien los integrantes de la fábrica trabajan independiente del cliente, el jefe de proyecto de fábrica debe asegurar el avance de las actividades relacionadas con su proyecto, por lo tanto se relaciona con los integrantes de la fábrica.

Otro proceso es el de aseguramiento de la calidad que se preocupa de la calidad y documentación del producto de software desarrollado la interior de la fábrica. El encargado es el jefe de administración de calidad, quien ejerce control sobre el administrador de calidad y el documentador.

En resumen la estructura de la fábrica describe un gran equipo de trabajo, que cumple con desarrollar un producto que satisfaga los requerimientos ingresados por un jefe de proyecto de fábrica.

3.1 Estructuras Orgánicas de una Fábrica de Software

El servicio de Fábrica de Software no necesariamente puede poseer el alto equipamiento humano, y menos en sus inicios, como para cubrir cada función, descritas en las estructura funcional figura 3.1, con una persona respectivamente. Por lo tanto se puede diferenciar 3 casos que muestras la evolución de una estructura orgánica que satisface los requerimientos que implica la implementación de un servicio de Fábrica de Software.

3.1.1 Estructura Orgánica Fase 1

Esta estructura contempla un equipo humano de menos de 10 personas, como muestra en la figura 3.2, que con respecto a la estructura funcional detallada en la figura 3.1, prescinde de jefaturas de las distintas área, que cubren las etapas del desarrollo de software, debido al bajo personal que se asigna para cada una, por lo tanto es el jefe de fábrica quien controla directamente a sus integrantes (diseñador/programador, administrador de componentes y administrador de calidad), los cuales deben reportarse a él. El diseñador/programador deberá cubrir las funciones de un diseñador, programador y tester, teniendo como cliente al jefe de proyecto de fábrica/Analista, siendo este último el encargado de efectuar las labores de estrategia (explicada más adelante en el ciclo de vida del proyecto de software) y levantamiento de requisitos, adoptando además la función de análisis de los requerimientos. El administrador de componentes, además de cumplir con sus labores respectivas al cargo, debe hacerse responsable de las tareas que le compete al verificador y validador de componentes, transformándose de esta manera en el único encargado de la biblioteca de componentes. Al igual que el administrador de componentes, el administrador de calidad es el único encargado en su área, responsabilizándose de las tareas de aseguramiento de la calidad y la documentación del software.

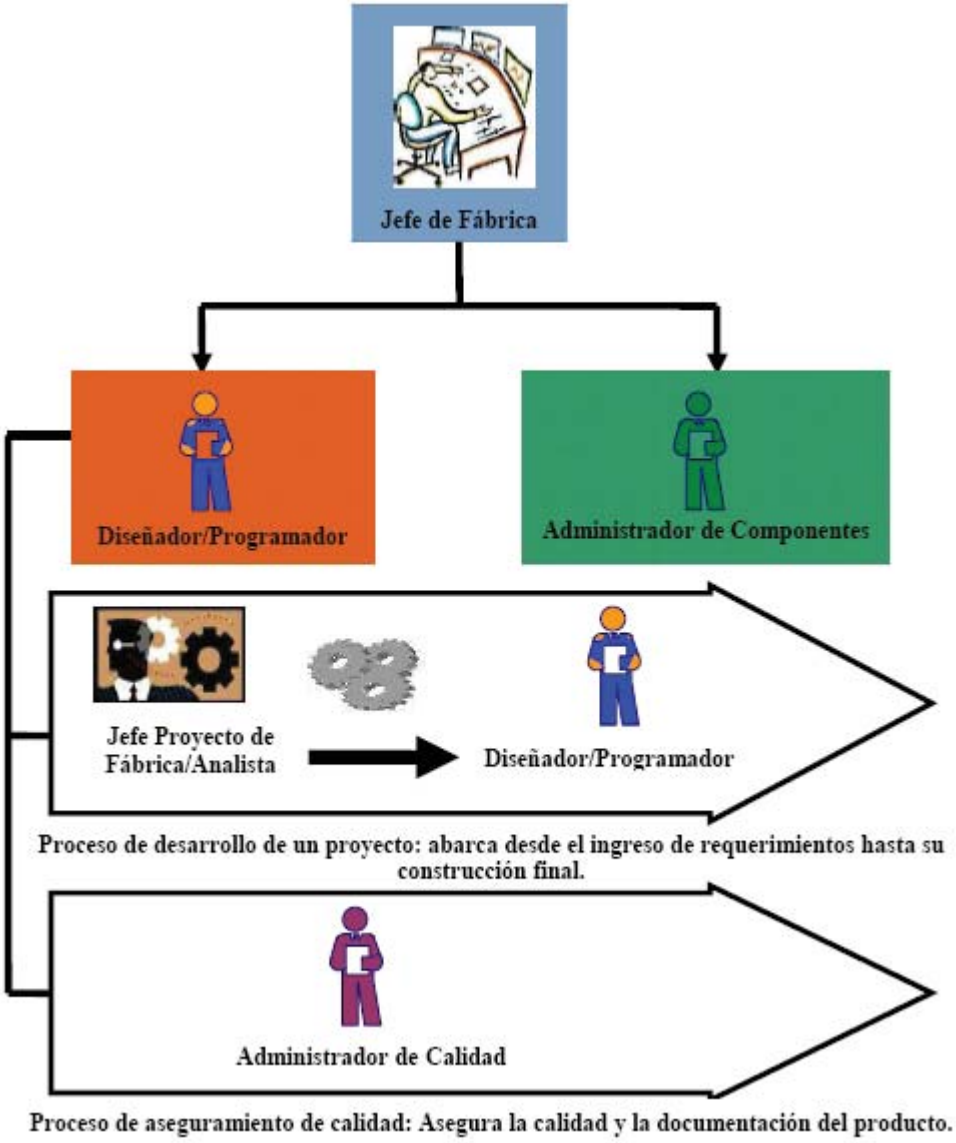


Figura 3.2: Estructura orgánica fase 1 para una Fábrica de Software

Esta estructura orgánica se detalla como muestra la tabla 3.1.

INTEGRANTES			
Jefe de fábrica	1	Diseñador/ Programador	4
Jefe de proyecto de fábrica/Analista	2	Administrador de componentes	1

Administrador de calidad	1		

Tabla 3.1: Detalle de la estructura orgánica fase 1

3.1.2 Estructura Orgánica Fase 2

Esta estructura contempla un equipo humano de entre 15 y 30 personas, como muestra la figura 3.3, una estructura orgánica de mayor personal que la anterior, en donde se puede apreciar la aparición de los cargos de analista, tester, y jefe de administración de calidad que se reportan al jefe de fábrica. El jefe de administración de calidad, es el encargado de cubrir el área de administración de calidad controlando al documentador y administrador de calidad. Las labores específicas de análisis y testing son asignadas respectivamente al analista y al tester, resaltando así una mayor dedicación de estas labores dentro de la fábrica para cumplir con su rendimiento.

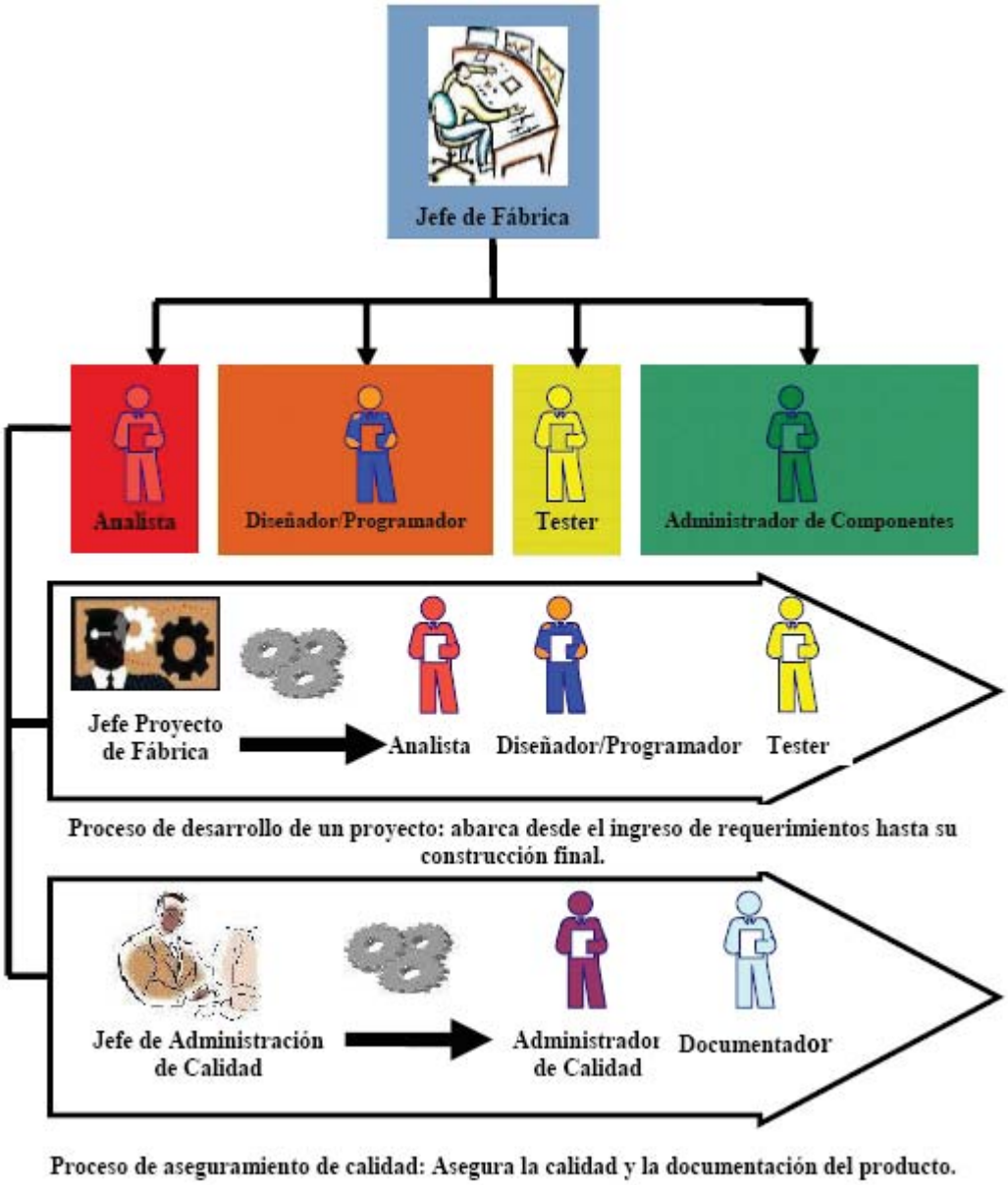


Figura 3.3: Estructura orgánica fase 2 para una Fábrica de Software

Esta estructura orgánica se detalla como muestra la tabla 3.2.

INTEGRANTES			
Jefe de fábrica	1	Analista	4
Jefe de proyecto de fábrica	2	Diseñador/Programador	8

Jefe de administración de calidad	1	Administrador de calidad	1
Administrador de componentes	1	Documentador	1
		Tester	4

Tabla 3.2: Detalle de la estructura orgánica fase 2

3.1.3 Estructura Orgánica Fase 3

Esta estructura contempla un equipo humano de más de 30 personas, la cual se traduce en una replica de la figura 3.1, pudiendo asignar a cada función, dentro de la fábrica, una respectiva persona que cumpla con las labores especificadas en el punto 3.1 y utilizando una descripción de cargos que se explicará en le punto 3.2. Esta estructura orgánica se detalla como muestra la tabla 3.2

INTEGRANTES			
Jefe de área de análisis	1	Analista	4 ó +
Jefe de área de diseño	1	Diseñador	6 ó +
Jefe de área de programación	1	Programador	8 ó +
Jefe de área de testing	1	Tester	4 ó +
Jefe de biblioteca de componentes	1	Administrador de componentes	1 ó

			+
Jefe de proyecto de fábrica	2 ó +	Verificador y validador de componentes	1 ó +
Jefe de administración de calidad	1	Administrador de calidad	1 ó +
Jefe de fábrica	1	documentador	1 ó +

Tabla 3.3: Detalle de la estructura orgánica fase 3

3.2 Perfiles Funcionales

Para comprender de mejor forma las actividades a desarrollar y características de cada integrantes de la fábrica, es que se crean los perfiles. Estos perfiles son fundamentales al momento de describir los cargos dentro de la fábrica, que serán distribuidos entre las distintas personas que compongan la Fábrica de Software. A continuación se describen las actividades, características del cargo, características personales y objetivos de cada perfil .

3.2.1 Jefe de Proyecto

Actividades:

- Realizar reuniones de coordinación con el cliente.

- Realizar reuniones de evaluación con cada rol.

- Obtener información sobre el estado el proyecto para el equipo y para el cliente

- Planificación, incluyendo estimación de tiempos, plaza, costos e hitos del proceso de desarrollo.

Características del cargo:

- *Abstracción:* Entender y comunicar aspectos no tangibles, como visión y misión al equipo de trabajo. Entender y ver el proyecto completo como unidad y las relaciones entre sus partes.

- *Concretización:* Utilizando los recursos e información disponibles, obtener conclusiones y tomar acciones específicas para manejar el proyecto.

- *Organización:* Distribuir eventos y actividades de acuerdo a los recursos y tiempos disponibles para llevar el proyecto al éxito.

- *Liderazgo:* Llevar a un equipo a lograr sus objetivos.

- *Experiencia:* Haber estado en situaciones similares en el pasado.

- *Creatividad:* Ser realista, tomando decisiones y acciones cuando el plan actual no funciona.

- *Persuasión:* Encontrar y desarrollar argumentos para mejorar y ayudar en una situación.

- Características personales:

- Saber escuchar y comunicar.

- Capacidad de tomar decisiones y realizar acciones.

- Saber trabajar bajo presión.

- **Objetivos:**

- Tener el software "a tiempo", "bajo presupuesto" y con los requisitos de calidad definidos.

- Terminar el proyecto con los recursos asignados.

- Coordinar los esfuerzos generales del proyecto, ayudando a cada uno de sus integrantes a cumplir sus objetivos particulares.

- Cumplir con éxito las diferentes fases de un proyecto, utilizando herramientas de administración.

- Cumplir con las expectativas del cliente.

3.2.2 Jefe de Fábrica

- **Actividades:**

- Verificar la calidad bajo estándares definidos por la fábrica.

- Supervisar la no entrada de trabajos que no se encuentren debidamente especificados.

- Establecer reuniones con los jefes de las distintas áreas y el jefe de proyecto de la fábrica.

- Ofrecer las herramientas necesarias para el funcionamiento correcto y eficiente de la fábrica.

- **Características del Cargo:**

- Manejar la parte técnica del servicio ofrecido por la fábrica.

- Mantener una buena comunicación con las distintas áreas de la fábrica y con el jefe de proyecto de fábrica

- Características personales:

- Saber escuchar y comunicar.

- Capacidad de tomar decisiones y realizar acciones.

- Saber trabajar bajo presión.

- **Objetivos:**

- Sincronización y coordinación de equipos de trabajo.

- Lograr que la fábrica cumpla los plazos de entrega y costos de éstos.

- Asegurarse que las especificaciones de entrada cumplen con los estándares definidos.

- Asegurarse de que los productos de salida cumplan con los estándares de calidad.

3.2.3 Jefe de Área

- **Actividades:**

- Coordinar a los funcionarios internos del área.

- Controlar las entradas y salidas del área.

- Implementar medidas de mejoras para proceso.

- Administrar los recursos tecnológicos del área

- **Características del Cargo:**

- Manejar la parte técnica del área y conocimientos de las labores de las áreas con la que se debe interactuar.

- Mantener una buena comunicación con los integrantes de su área y con el jefe de proyecto de fábrica.

- **Características personales:**

- Saber escuchar y comunicar.

- Capacidad de tomar decisiones y realizar acciones.

- Saber trabajar bajo presión.

- **Objetivos:**

- Supervisar a los funcionarios del área.

- Asegurar el cumplimiento de las tareas de la fase respectiva.

- Asegurar que las salidas cumplan con estándares y políticas del área.

- Mantener habilitada la infraestructura tecnológica del área.

3.2.4 Análisis

- **Actividades:**

Entrevistar al cliente, ayudándole a identificar sus necesidades.

- Verificar si los requisitos especificados son los correctos.

- Definir una estructura básica del sistema que incluya fuentes de información, módulos de procesamiento de información y resultados esperados.

- Determinar factores críticos de éxito.

- Transformar los requerimientos en requisitos de software.

- **Características del Cargo:**

- Capacidad de estudiar un problema de una complejidad determinada, descomponiendo el problema en subproblemas de menor complejidad.

- Conocer y manejar los métodos y tecnologías de apoyo para realizar el análisis

- Conocer las técnicas de diseño que se utilizarán en las siguientes fases.

- Conocer diferentes lenguajes de programación.

- **Características personales:**

- Debe ser una persona sociable, expresando sus ideas en forma clara en un lenguaje común con el cliente.

- Debe tener la capacidad de comunicación para escuchar y entender al cliente

- Se espera que los analistas tengan un alto grado de desarrollo de su inteligencia emocional.

- **Objetivos:**

- Determinar las necesidades esenciales y no esenciales, así como las que son de segundo nivel.

- Impedir la introducción de defectos tempranamente en la construcción del sistema.

- Construir el documento de requerimientos y requisitos de software.

- Establecer una estructura básica inicial del sistema, y sus relaciones.

- Definir la especificación de la arquitectura del sistema, en forma de un documento técnico comprensible.

3.2.5 Diseño

- **Actividades:**

- Descomposición en subsistemas.

- Definir el modelo de objetos.

- Seleccionar una técnica de administración de almacenamiento de datos.

- Interactuar con los programadores

- Asignación de subsistemas a procesadores

- Selección de estrategias de control.

- Administración de condiciones de borde.

- Generar el diseño arquitectónico y diseño detallado del sistema, basándose en los requisitos.

- Generar el documento de diseño arquitectónico de software y mantenerlo actualizado durante el proyecto.

- **Características del cargo:**

- Conocer muy bien la metodología de diseño utilizada, así como sus herramientas de apoyo.

- Conocimiento normas y estándares, del cliente y de la fábrica.

- **Características personales:**

- Habilidad para sintetizar soluciones construibles con un conjunto de restricciones.

- Generalmente son los más capacitados para realizar decisiones estratégicas debido a su experiencia previa en la construcción de sistemas similares.

- Alto nivel de abstracción.

- Habilidades para la programación.

- **Objetivos:**

- Crear una estructura interna del sistema (arquitectura) y la definición de relaciones entre subsistemas.

- Seleccionar las políticas apropiadas para nombres lógicos, espacio, unidades físicas, y acceso a datos compartidos.

- Seleccionar el método de almacenamiento apropiado para las estructuras de datos.

- Asignar procesos a unidades de procesamiento que sirva como plataforma para la ejecución de subsistemas.

3.2.6 Programación

- **Actividades:**

- Interactuar con los analistas y diseñadores.

- Manejar lenguajes de programación validos actualmente.

- Manejar herramientas de generación de código.

- Interactuar con biblioteca de componentes.

- Ensamble de componentes y crear componentes nuevos.

- Generar prototipos rápidos del sistema.

- Realizar testing unitarios.

- Hacer la documentación del código.

- **Características del cargo:**

- Se hace necesario conocer los últimos desarrollos, quien da soporte, y como pueden beneficiar al proyecto y a la organización.

- Manejar estándares de codificación.

- Los programadores deben tener experiencia en bases de datos.

- Orientado a la maquina.

- Permanentemente actualización de conocimientos.

- **Características personales:**

- Requiere conocimiento en varios ambientes de desarrollo.

- Pro activo.

- Conocer las técnicas de diseño utilizadas.

- **Objetivos:**

- Reducir la complejidad del software:

- Optimización de código.

- Utilizar mejores prácticas.

- Menor cantidad de problemas de testeo.

- Aumento de la productividad de los programadores.

- Aumento de la eficiencia en la manutención del programa.

- Aumento de la eficiencia en la modificación del programa.

- Reducir el tiempo de codificación, aumentando la productividad del programador.

- Disminuir el número de errores que ocurren durante el proceso de desarrollo.

- Disminuir el esfuerzo de corregir errores en secciones del código que se encuentran deficientes, remplazando secciones cuando se descubren técnicas más confiables, funcionales o eficientes.

- Disminuir los costos del ciclo de vida del software.

3.2.7 Testing

- **Actividades:**

- Realizar los tests, apoyado por los programadores

- Informar sobre los resultados obtenidos

- Construir un plan de testeo.

- Construir la documentación del proceso de tests.

- **Características del cargo:**

- Ser un buen programador en el lenguaje seleccionado, y tener experiencia en el desarrollo de sistemas.

- Conocer bien la metodología de diseño utilizada.

- Ser sistemático en las revisiones de código y resultados de los tests.

- **Características personales:**

- Tener una personalidad agresiva para buscar errores en el código y documentos del proyecto.

- Debe además tener una personalidad alegre, debido a que debe relacionarse con gran parte de los miembros del equipo de desarrollo.

- **Objetivos:**

- Aplicar métodos para diseñar casos de tests efectivos.

- Construir buenos casos de tests que tengan altas probabilidades de encontrar errores aún no descubiertos.

- Demostrar que las funciones del sistema parecen estar funcionando de acuerdo a sus especificaciones.

- Proveer una buena indicación de la confiabilidad del software y algunas indicaciones de la calidad del software.

3.2.8 Documentación

- **Actividades:**

- El documentador debe diseñar y construir un repositorio de información compartido, donde se almacenará la documentación de los procesos anteriores.

- Mantener el repositorio de información. Administrar las diferentes versiones.

- Especificar el formato que será usado para elaborar la documentación.

- Asegurarse que los documentos mantienen el estándar de documentación definido para el proyecto antes de incluirlos en el repositorio.

- Mantener actualizada la documentación.

- **Características del cargo:**

- Conocimiento informático.

- Debe conocer y utilizar el procesador de texto definido para el proyecto en toda su potencialidad, utilizando funcionalidades como estilos, corrector sintáctico y gramatical, control de versión, etc.

- Conocer y aplicar estándares de documentación

- **Características personales:**

- Ser ordenado, con capacidades de mantener una gran cantidad de información en forma ordenada y accesible.

- Tener creatividad para presentar la información y aptitud de expresión para escribir.

- **Objetivos:**

- Permitir el almacenamiento y recuperación de la documentación de los proyectos durante el desarrollo, manteniendo así la información al día, así como también una vez entregados los proyectos.

- Mantener la consistencia en la apariencia y estructura de los documentos.

- Asegurarse que los cambios que necesitan hacerse en el sistema serán reflejados en la documentación correspondiente.

3.2.9 Administración de Calidad

- **Actividades:**

- Revisar la fase de diseño arquitectónico

- Participar en la revisión de los requisitos del sistema.

- Revisar las políticas de control de cambios, control de errores y control de la configuración.

- Revisar la documentación.

- **Características del cargo:**

- Conocer y aplicar técnicas y estándares que aseguren la calidad de un producto de software.

- Conocer herramientas de apoyo.

- Capacitación en temas de aseguramiento de la calidad.

- **Características personales:**

- El asegurador de calidad debe ser una persona con mucha experiencia en proyectos de desarrollo de software.

- Buenas relaciones interpersonales.

- **Objetivos:**

- Asegurarse que la especificación de requisitos es una representación correcta y completa de las expectativas del cliente, y que es suficientemente clara para el equipo de desarrollo, especialmente para los diseñadores.

- Asegurarse que los diseñadores seleccionaron la metodología apropiada y que el producto final cumple con los requisitos de rendimiento, diseño y verificación.

- Asegurarse que se realizan monitoreos de errores en cada fase del desarrollo y que se respaldan las líneas.

- Asegurarse que la documentación cumple con el estándar utilizado durante el desarrollo del producto de software.

3.2.10 Administración de Componentes

- **Actividades:**

- Diseñar y construir un repositorio de componentes, donde se almacenarán los componentes reutilizables para la implementación del software.

- Mantener el repositorio de componentes.

- Especificar el formato que será usado para elaborar los componentes.

- Diseñar y mantener un sistema de búsqueda automatizada de componentes.

- **Características del cargo:**

- Conocimientos de metodologías de reutilización de componentes.

- Capacidad de modelamiento de sistemas y base de datos.

- Conocer de estándares de programación.

- **Características personales:**

- Habilidades de trabajo en equipo para comunicarse con los programadores y el verificador y validador de componentes.

- **Objetivos:**

- Permitir un almacenamiento y expansión ordenada de la biblioteca de componentes.

- Permitir la búsqueda y utilización de componentes de forma rápida y efectiva.

- Hacer valer los estándares de composición de componentes.

3.2.11 Verificación y Validación de Componentes

- **Actividades:**

- Administración de la verificación y validación de componentes.

- Planificación del proceso de verificación y validación.

- Documentar evaluación de resultados de los componentes.

- Verificación y validación los requisitos de los componentes.

- Verificación y validación del código de los componentes.

- Verificación y validación de la manutención de componentes.

- **Características del cargo:**

- Conocimiento de metodologías de reutilización de componentes.

- Manejo en los lenguajes utilizados para la programación de componentes.

- Conocimiento de los estándares impuesto por el administrador de componentes.

- **Características personales:**

- Habilidades de desarrollo de planificar pruebas para procesos de verificación y validación.

- Buena comunicación, para mantener buena relación con los programadores y administrador de componentes.

- **Objetivos:**

- Determinar que los componentes ejecutan su funcionalidad correctamente, asegurarse que no ejecuta funciones no intencionalmente definidas y proveer información sobre su calidad y confiabilidad.

- *Correctitud:* En que grado el componente está libre de fallas.

- *Consistencia:* En que grado el componente es consistente consigo mismo y con otros productos.

- *Necesidad:* En que grado lo que hay en el componente es necesario.

- *Suficiencia:* En que grado el componente es completo.

- *Rendimiento:* En que grado el componente satisface los requisitos de rendimiento.

3.3 Descripción de Cargos

A continuación se muestran las distintas tablas que describen[1] los cargos que cubren las necesidades de la Fábrica de Software para su funcionamiento. Los perfiles de los cargos se forman mediante los perfiles descritos anteriormente.

Cargo	Jefe de proyecto de fábrica.

[1] Para especificar la profesión y la experiencia se considero descripciones de cargo de CODELCO.

Profesión	Ingeniero civil informático.
Experiencia	Mínimo 5 años.
Descripción del perfil	Jefe de proyecto. Análisis.
Controla	Analista. Diseñador. Programador.
Reporta	----------
Su cliente	Cliente

Tabla 3.4: Descripción del cargo jefe de proyecto de fábrica

Cargo	Jefe de fábrica.
Profesión	Ingeniero civil informático.
Experiencia	Mínimo 5 años.
Descripción del perfil	Jefe de fábrica.
Controla	Jefe de biblioteca de componentes. Jefe de administración de calidad. Jefe de área de análisis

	Jefe de área de tester. Jefe de área de diseño. Jefe de área de programación.
Reporta	----------
Su cliente	Jefe de proyecto de fábrica.

Tabla 3.5: Descripción del cargo jefe de fábrica

Cargo	Jefe de área de análisis.
Profesión	Ingeniero informático.
Experiencia	Mínimo 3 años.
Descripción del perfil	Jefe de área. Análisis.
Controla	Analistas.
Reporta	Jefe de fábrica.
Su cliente	----------

Tabla 3.6: Descripción del cargo jefe de área de análisis

Cargo	Jefe de área de diseño.
Profesión	Ingeniero informático.
Experiencia	Mínimo 3 años.
Descripción del perfil	Jefe de área. Diseño.
Controla	Diseñadores.
Reporta	Jefe de fábrica.
Su cliente	----------

Tabla 3.7: Descripción del cargo jefe de área de diseño

Cargo	Jefe de área de programación.
Profesión	Ingeniero informático.
Experiencia	Mínimo 3 años.
Descripción del perfil	Jefe de área. Programación.
Controla	Programadores.
Reporta	Jefe de fábrica.
Su cliente	----------

Tabla 3.8: Descripción del cargo jefe de área de programación

Cargo	Jefe de área de testing.
Profesión	Ingeniero informático.
Experiencia	Mínimo 3 años.
Descripción del perfil	Jefe de área. Testing.
Controla	Tester.
Reporta	Jefe de fábrica.
Su cliente	----------

Tabla 3.9: Descripción del cargo jefe de área de testing

Cargo	Jefe de biblioteca de componentes.
Profesión	Ingeniero informático.
Experiencia	Mínimo 3 años.
Descripción del perfil	Jefe de área. Administración de componentes. Verificación y validación de componentes
Controla	Administrador de componentes.

	Verificador & validador de componentes.
Reporta	Jefe de fábrica.
Su cliente	Programador.

Tabla 3.10: Descripción del cargo jefe de biblioteca de componentes

Cargo	Jefe de administración de calidad.
Profesión	Ingeniero informático.
Experiencia	Mínimo 3 años.
Descripción del perfil	Jefe de área. Administración de calidad.
Controla	Administrador de calidad. Documentador.
Reporta	Jefe de fábrica.
Su cliente	----------

Tabla 3.11: Descripción del cargo jefe de administración de calidad

Cargo	Analista.

Profesión	Ingeniero ejecución informático.
Experiencia	Mínimo 1 año.
Descripción del perfil	Análisis.
Controla	----------
Reporta	Jefe de área de análisis.
Su cliente	Jefe de proyecto de fábrica

Tabla 3.12: Descripción del cargo analista

Cargo	Diseñador.
Profesión	Ingeniero ejecución informático.
Experiencia	Mínimo 1 año.
Descripción del perfil	Diseño.
Controla	----------
Reporta	Jefe de área de diseño.
Su cliente	Jefe de proyecto de fábrica

Tabla 3.13: Descripción del cargo diseñador

Cargo	Programador.
Profesión	Técnico informático.
Experiencia	Mínimo 1 año.
Descripción del perfil	Programación.
Controla	----------
Reporta	Jefe de área de programación.
Su cliente	Jefe de proyecto de fábrica

Tabla 3.14: Descripción del cargo programador

Cargo	Tester.
Profesión	Ingeniero ejecución informático.
Experiencia	Mínimo 1 año.
Descripción del perfil	Testing.
Controla	----------
Reporta	Jefe de área de Testing.
Su cliente	Programador.

Tabla 3.15: Descripción del cargo tester

Cargo	Administrador de componentes.
Profesión	Ingeniero ejecución informático.
Experiencia	Mínimo 1 año.
Descripción del perfil	Administración de componentes.
Controla	----------
Reporta	Jefe de biblioteca de componentes.
Su cliente	Programadores.

Tabla 3.16: Descripción del cargo administrador de componentes

Cargo	Verificador y validador de componente.
Profesión	Técnico informático.
Experiencia	Mínimo 1 año
Descripción del perfil	Verificación y validación de componentes.
Controla	----------
Reporta	Jefe de biblioteca de componentes.
Su cliente	Programadores.

Tabla 3.17: Descripción del cargo verficador y validador de componentes

Cargo	Administrador de calidad.
Profesión	Ingeniero ejecución informático.
Experiencia	Mínimo 1 año.
Descripción del perfil	Administración de calidad.
Controla	----------
Reporta	Jefe de administración de calidad.
Su cliente	Jefe de proyecto de Fábrica

Tabla 3.18: Descripción del cargo administrador de calidad

Cargo	Documentador.
Profesión	Ingeniero ejecución informático.
Experiencia	Mínimo 1 año.
Descripción del perfil	Documentación.
Controla	----------
Reporta	Jefe de administración de calidad.
Su cliente	Jefe de proyecto de fábrica

Tabla 3.19: Descripción del cargo documentador

3.4 Esquema de la Fábrica de Software

Como se señala en el estudio teórico de Fábrica de Software, la necesidad de clasificar y resumir los distintos Artefactos de Desarrollo, pertenecientes a la construcción de un miembro de la Línea de Productos, hace necesario la elaboración de un Esquema de Fábrica de Software el cual se instancia para dar origen a un producto en particular mediante la Plantilla de Fábrica de Software.

Como ya se sabe un Esquema de Fábrica de Software es básicamente una matriz que posee distintas Vistas para diferentes arquitecturas, para llevar a cabo su elaboración existen diferentes criterios, en este caso se considera el desarrollado por Microsoft, debido a que la mayor información teórica obtenida sobre las Fábricas de Software fue a través de literatura perteneciente a Microsoft, lo que facilita una mayor consistencia entre los conceptos involucrados y el desarrollo del esquema. Como se muestra en la tabla 2.2 existen 3 vistas (conceptual, lógica y física) y 4 arquitecturas (negocio, información, aplicación y tecnología).

A continuación, se muestran los contenidos de las distintas arquitecturas para las diferentes vistas en la tabla 3.20 que constituyen el esquema de Fábrica de Software a implementar.

VISTAS	ARQUITECTURAS			
	NEGOCIO	**INFORMACIÓN**	**APLICACIÓN**	**TECNOLOGÍA**
CONCEPTUAL	− Misión, visión y objetivos de la empresa. − Alianzas estratégicas.	− Información de clientes y pedidos.	− Ciclo de vida del proyecto de software. − Línea de Productos. − Políticas de administración de repositorio de componentes − Administración de calidad.	− Distribución de equipos. − Distribución de servicios y herramientas.
LÓGICA	− Descripción de cargos. − Definición de perfiles. − Organigrama.	− Modelamiento de la información. − Documentos de especificación.	− Especificación de requerimientos. − Modelo de clase o dato. − Diagramas de interacción. − Casos de pruebas	− Sistema operativo. − Herramientas de programación y modelado. − Lenguaje de base de datos.
FÍSICA	− Workflow del negocio. − Especificación de procesos del negocio.	− Esquemas de base de datos. − Políticas de administración de los datos.	− Arquitectura de la Línea de Productos. − Componentes reutilizables.	− Servidores. − Dispositivos de red. − Impresoras. − PCs.

Tabla 3.20: Esquema de la Fábrica de Software

Capítulo 4

Cadena de Valor del Ciclo de Vida del Proyecto de Software

Para representar las diferentes etapas, que surgen desde el nacimiento del proyecto hasta su elaboración e implantación, se define un modelo de proceso del desarrollo de software al interior de la Fábrica de Software. Dentro de este modelo se distinguen las etapas mostradas en la figura 4.1.

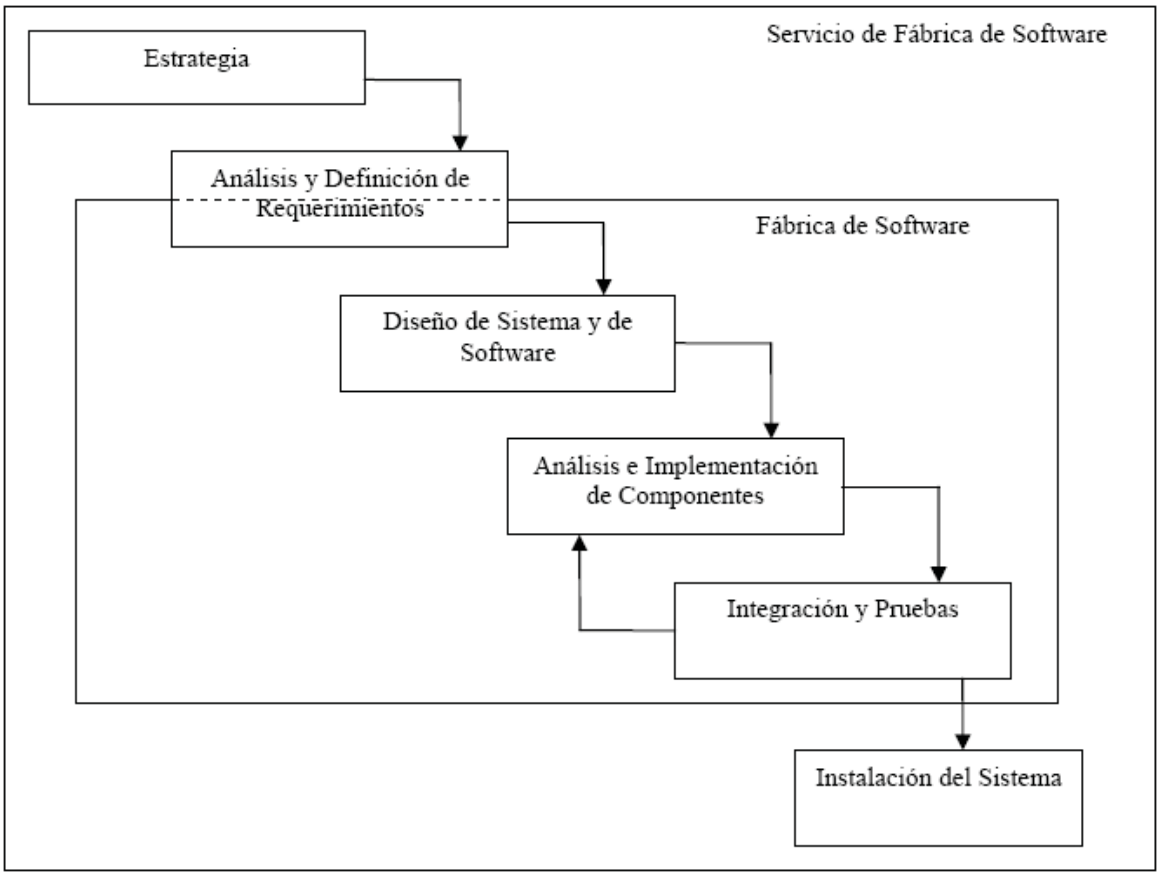


Figura 4.1: Etapas del desarrollo del proyecto de software

El modelo que muestra la figura 4.1 recoge ideas de un modelo de cascada y un modelo de desarrollo basado en la reutilización, en donde algunas de sus etapas se encuentran bajo una iteración de tipo incremental. Además este modelo propuesto se basa fuertemente en la metodología Case*Method debido a que proporciona para cada etapa una clara descripción, definición de objetivos y metas, productos de la etapa, factores críticos de éxito y la lista de tareas que conviene realizar . A continuación se muestran los puntos rescatados de algunos modelos y aquellos que provocaron el rechazo de otros.

Se rescata del modelo de cascada los siguientes puntos:

Una etapa de análisis y definición de requerimientos con necesidades claras de parte del cliente y bien comprendidas por el interlocutor cliente/fábrica. Esto es posible debido a que se lleva adelante proyectos específicos para un dominio en particular, lo que hará una fácil y mejor comprensión de los requerimientos existentes en el cliente.

Una etapa de diseño de sistema y de software.

Una etapa de instalación del sistema (Operación), en la cual se instala y pone en uso práctico.

Se recata del modelo de desarrollo basado en la reutilización el siguiente punto:

Una de las innovaciones críticas definidas en el estudio teórico de Fábrica de Software, es el desarrollo de software a través del ensamblaje de componentes reutilizables. Por lo tanto, es que se consideran etapas que abarquen procesos de análisis, desarrollo e integración de estos componentes, como lo son las etapas de análisis e implementación de componentes e integración y pruebas.

Modelos como desarrollo evolutivo y desarrollo formal de sistemas no fueron considerados por los siguientes puntos respectivamente:

El desarrollo de software al interior de la fábrica se basa en un trabajo independiente del cliente por lo tanto el contacto con este es bajo, impidiendo un

desarrollo a la par con el cliente en donde procesos como especificación, desarrollo y validación sean concurrentes.

Los requerimientos del sistema no son vistos como especificaciones matemáticas tan fácilmente.

Este modelo no muestra una etapa que haga mención explícita de la mantención del software, debido a que una eventual etapa de mantención será comprendida por una iteración total del modelo descrito. A continuación se describen las distintas etapas del modelo, independiente del tipo de metodología de desarrollo (orientado a objeto o estructurado)

4.1 Estrategia

Nombre Procedimiento: Estrategia
Tipo Procedimiento: Operación Fabrica Software

PROPOSITO
El objetivo principal de esta etapa es permitir un acercamiento entre el jefe de proyecto de fábrica y el equipo cliente, a fin de obtener la mayor información acerca de lo que el cliente espera del sistema, es decir se obtiene el concepto del sistema. Dentro de esta etapa se deben identificar los objetivos y los alcances del proyecto, existiendo una iteración de propuestas para tener algún punto de encuentro entre un equipo cliente, representante de la organización que requiere solucionar o apoyar algún proceso o conjunto de procesos mediante la incorporación de tecnologías de información y como contraparte un equipo de desarrollo que realiza una estimación preliminar de esfuerzos

en términos de personas, costos y tiempo que se requiere para el desarrollo del proyecto . Esta etapa se desarrolla fuera de la Fábrica de Software junto al cliente.

ENTRADAS

Reuniones entre el jefe de proyecto de fábrica y el equipo cliente.

Descripción preliminar de requerimientos.

Visión del sistema por parte del cliente.

Reglas del negocio.

Informe de la organización donde estará inserto el futuro sistema.

Información funcional de los sistemas y procesos que se encuentran en funcionamiento dentro de la organización.

Informe de los futuros usuarios del sistema, a fin de determinar roles y accesos.

ACTIVIDADES

Asistir al cliente en la identificación de los requerimientos del producto de software. Incluye requerimientos funcionales, seguridad, auditoria, respaldo, integración sistemas y requerimientos de capacitación.

Determinar los beneficios cuantitativos y cualitativos.

Formalizar los requerimientos identificados y presentarlos al cliente para su aprobación.

Confeccionar plan de desarrollo.

Definir entrevistas a realizar.

Estimar costos, tiempo y plazo.

Generar carpeta del proyecto.

SALIDAS

Identificación del sistema, se debe hacer una reseña acerca del sistema para un mejor orden y entendimiento de todas las partes que participaran en el proyecto de la manera siguiente:

- Nombre

- Sigla.

- Aceptante proyecto.

- Encargado cuenta.

- Jefe de proyecto.

- Fecha documento.

- Fecha actualización del documento.

Descripción global y funcional del sistema:

- En este punto se hace una explicación del sistema con respecto al contexto en donde se enmarca.

- En que consiste a grandes rasgos, sus funciones futuras, ubicación y descripción de su organización.

- Se debe realizar el diagrama de contexto del sistema.

Objetivos y alcances del sistema: Se explicaran tanto los objetivos generales como los objetivos específicos que se quieren lograr orientado a un enfoque de cliente. Referente a los alcances se debe señalar las funciones propias del negocio

y sus posibles impactos en la organización.

Definición de procesos:

- Modelo de Procesos o sistemas actualmente en funcionamiento.

- Descripción de Procesos. Indicar naturaleza (manual o automático).

Definición de requerimientos:

- Administrativos, funcionales, seguridad y auditoria.

- Respaldo e información histórica.

- Disponibilidad, rendimiento y rapidez.

- Interfaces, integración de sistemas y capacitación.

Usuarios del sistema: Definición de roles, creación de cuentas y definición de tipos de accesos.

Supuestos, efectos esperados y restricciones

Condiciones de satisfacción

Glosario de términos: Se recomienda crear términos representativos y respetar los términos en todo el sistema.

ACTORES

Jefe de proyecto de fábrica.

Jefe de fábrica.

Cliente

4.2 Análisis y Definición de Requerimientos

Nombre Procedimiento: Análisis y Definición de Requerimientos
Tipo Procedimiento: Operación Fabrica Software

PROPOSITO
En esta etapa el interlocutor cliente/fábrica, con ayuda de analistas si es necesario, se encarga de obtener los requerimientos del cliente para formularlos de manera clara y completa a la fábrica, en donde son validados y verificados por analistas al interior de

ella. El objetivo principal es realizar un acercamiento formal y profundo a determinar qué es exactamente lo que el cliente y su organización requiere del sistema, es decir, se obtienen los requerimientos funcionales y no funcionales detallados del sistema y las reglas del negocio . En la figura 4.1 se muestra una parte dentro de la fábrica y otra fuera, debido a que no tiene una división clara con respecto al espacio de su desarrollo, más bien es una extensión de la fábrica hacia el cliente de manera transparente para ambos, pudiéndose efectuar labores de análisis en el cliente mismo si el producto lo amerita.

ENTRADAS

Definición de requerimientos preliminar.

Informe de usuarios de sistema.

Informe de supuestos, efectos esperados y restricciones.

Diagrama de sistemas actualmente en funcionamiento.

ACTIVIDADES

Fuera de la fábrica:

- Generar un programa de trabajo con reuniones periódicas.

- Construir minutas de resultado de reuniones las que posteriormente serán discutidas y validadas con el objetivo de determinar si se están interpretando correctamente los requerimientos.

- Confirmar formalización del sistema.

- Confirmar los requerimientos del sistema.

- Verificar que el alcance, restricciones, objetivos del sistema y requerimientos de información son aprobados por el cliente.

- Reconfeccionar plan de desarrollo.

- Definir plan de acción para compromisos de adquisición, equipamiento, configuración, conectividad y soporte.

- Estrategia inicial de transición.

Dentro de la fábrica:

- Generar diagramas.

- Definir usuarios concurrentes al sistema.

- Analizar interfaces o integración con otros sistemas.

- Definir prioridades de implementación.

- Generar diagrama de interacción (para escenarios que lo requieran).

- Generar estado de avance del proyecto.

SALIDAS

Requerimientos de información:

- Requerimientos de información funcional.

- Requerimientos de auditoría/control.

- Requerimientos de respaldo/recuperación.

- Expectativas de desempeño.

- Requerimientos distribuidos.

- Información histórica.

- Expansión prevista del sistema.

Especificación de requerimientos funcionales y no funcionales, del usuario y del sistema.

Análisis de interfaces o integraciones:

- Relación con sistemas divisionales.

- Descripción de interfaces.

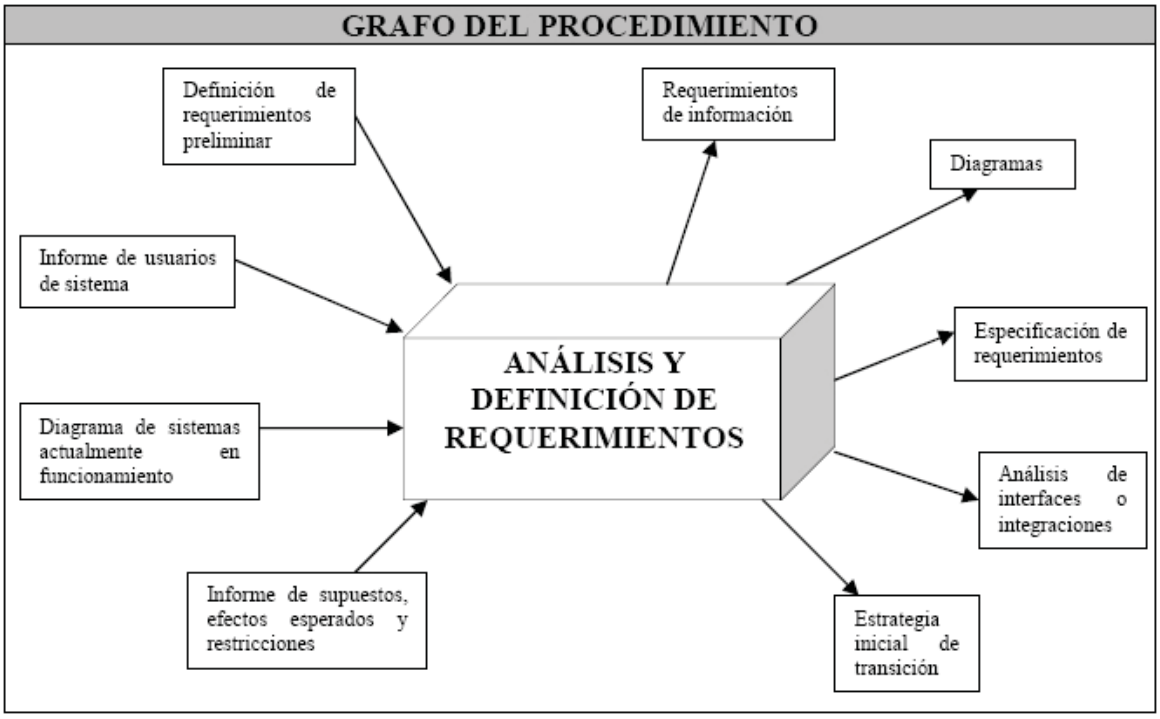
Estrategia inicial de transición: Equipamiento computacional, uso de componentes reusables, manejo de excepciones, calidad, seguridad, documentación de vistas de usuario y descripción del software utilizado.

ACTORES

Jefe de proyecto de fábrica

Jefe de área de análisis.

Analista.



4.3 Diseño de Sistema y de Software

Nombre Procedimiento: Diseño de Sistema y de Software
Tipo Procedimiento: Operación Fabrica Software

PROPOSITO
La Fábrica de Software abarca completamente en su interior esta etapa, donde se recibe las salidas de la etapa anterior que son la base para el trabajo que se efectúa. La etapa de diseño se basa en los procesos típicos del diseño de un proyecto de software, los cuales se entrelazan para complementarse, dando origen a una etapa de diseño

recursiva con el fin de refinarse.

ENTRADAS

Requerimientos de información.

Especificación de requerimientos.

Diagramas.

Manual de análisis.

ACTIVIDADES

Diseñar la arquitectura del sistema, considerando el conocimiento mas detallado del dominio utilizando una arquitectura genérica (especifica para el dominio).

Diseñar la integración o interfaz con otros sistemas.

Diseño de modelo de datos.

Generar diagramas de interacción

Especificar las interfaces de los objetos.

Asignar nombres y siglas de identificación a los elementos de sistema.

Documentar decisiones de diseño.

Diseñar lógicamente la red de comunicaciones y configuraciones.

Diseñar los procesos de auditoría, control, respaldos.

Diseñar plan de prueba de sistema.

Sostener reuniones periódicas con el jefe de proyecto de fábrica y otros (jefe de área u otros integrantes).

Gestionar la asignación de roles: Clientes del sistema externos e internos.

Definir de acuerdo a roles la modalidad de acceso a tablas.

Mantener control de resultados de aseguramiento de calidad.

Participar en revisiones de control de calidad.

Mantener control de cambios de las aplicaciones.

SALIDAS

Arquitectura del software, mediante un modelo de clases de diseño.

Especificaciones de la arquitectura del sistema.

Integración con otros sistemas: Definir links a otras bases y diseñar procesos.

Diseño de Diagramas y modelos.

Decisiones de diseño documentadas

Diseño de la interfaz de usuario:

- Definición de elementos de pantalla a utilizar.

- Diagrama de módulos del menú principal.

- Diseño procedimientos administrativos.

Definición de tablas: Código, nombre, descripción, columnas, descripción columnas, tipo, claves primarias y claves foráneas.

Dimencionamiento detallado:

- Definir requerimientos de producción del sistema, incluyendo espacios y tiempos de respuesta.

- Ratificar cantidad de accesos concurrentes.

- Diseño lógico red de comunicaciones y configuraciones:

- Definición del alcance.

- Cobertura geográfica y administrativa.

- Tipo de arquitectura (cliente/servidor, host/slave, mixto).

Definición de elementos requeridos:

- Hardware: Plataforma departamental, central, estaciones clientes.

- Software : Sistema operativo, bases de datos, comunicaciones.

- Estaciones clientes (capacidad requerida).

- Procedimientos de respaldos y control de la red.

- Plan de capacidades de la red.

Diseño de auditoría y control:

- Diseñar los mecanismos, niveles de seguridad y acceso.

- Roles y uso de recursos.

- Control de acceso por usuario o grupos de usuarios y por módulos.

- Procedimientos para administrar los privilegios usuarios/rol.

- Consideraciones de auditoría (custodia de información, disposiciones legales).

- Manual de procedimientos administrativos y de usuario.

Diseño de respaldo y recuperación:

- Tipo de respaldo incluyendo su periodicidad y frecuencia.

- Duración de los respaldos.

- Procedimientos para respaldos y recuperación.

- Definir requerimientos de hardware para operación de los mecanismos de respaldos.

Plan de pruebas del sistema:

- Preparar datos de prueba (conjunto de prueba, responsable, programa de carga).

- Definir y diseñar el plan de prueba.

- Definir recursos de hardware y de software.

Plan de instalación del sistema:

- Preparar plan de entrega, criterios de aceptación o condiciones de satisfacción.

- Definir plan de adiestramiento.

- Definir plan de carga de datos.

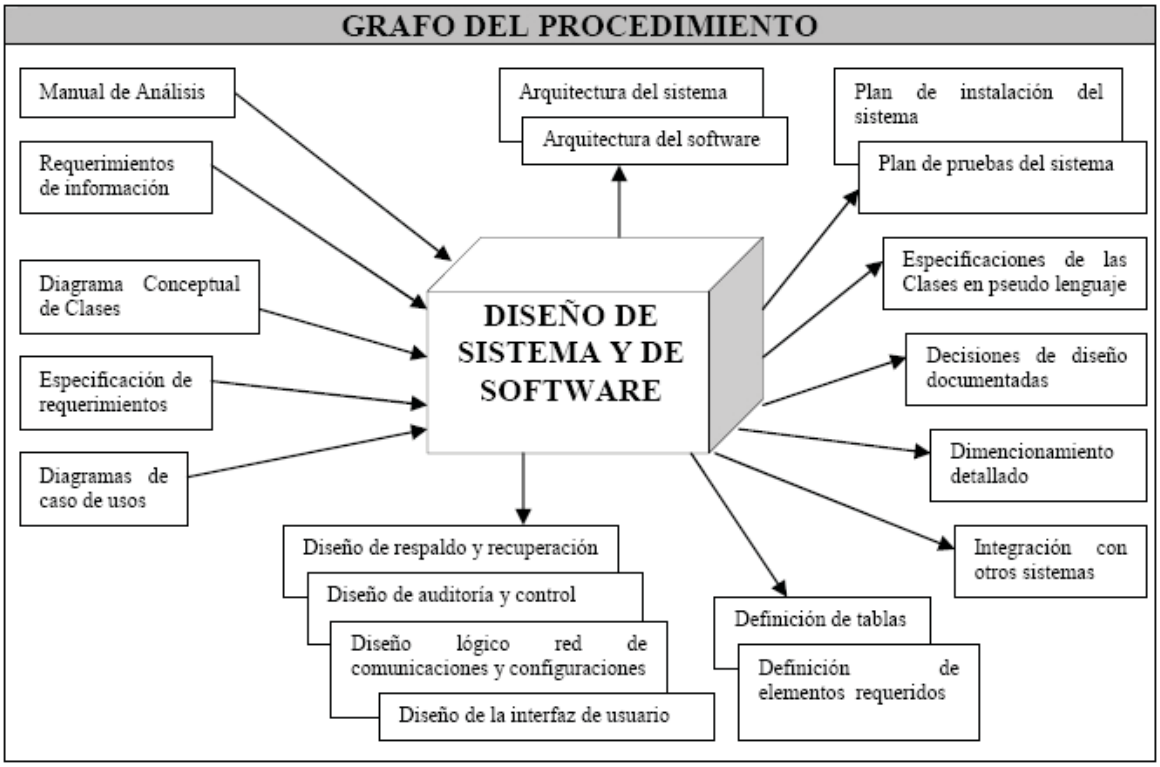
- Definir en forma detallada los requerimientos operacionales (incluye disponibilidad del sistema y horarios).

Definir plan de instalación (incluye compromisos de entrega de equipos, su reparación y factores que afectan la transición).

ACTORES

Jefe de área de diseño.

Diseñador.



4.4 Análisis e Implementación de Componentes

Nombre Procedimiento: Análisis e Implementación de Componentes
Tipo Procedimiento: Operación Fabrica Software

PROPOSITO
En esta etapa se analiza la documentación de la etapa de Diseño del Sistema y del Software, para dividir el software en incrementos a implementar. Se analizan aquellos componentes que se poseen para reutilizar y se desarrollan los restantes. Una vez finalizada la etapa de análisis e implementación de componentes necesarios para un

determinado incremento, son entregados a la siguiente etapa, en donde pueden ser devueltos producto de fallas en su comportamiento.

Es en esta etapa cuando intervienen los integrantes del área de la biblioteca de componentes, para recopilar componentes potencialmente reutilizables, junto con verificarlos, validarlos y generar su respectiva documentación. Cuando los componentes potencialmente reutilizables son detectados para formar parte del repositorio, se efectúan labores (detalladas en el capítulo 6) que se encuentran exentas de plazos propios del proyecto, debido a que esta actividad es independiente del desarrollo del producto en particular. A continuación se detallan las actividades a las que se somete cada incremento generado en esta etapa.

ENTRADAS

- Plan de pruebas (unitarias)
- Arquitectura del sistema y del software.
- Especificaciones de la arquitectura.
- Diseño de diagrama y modelo.
- Diseño de la interfaz de usuario.

ACTIVIDADES

- Detectar los componentes necesarios para su implementación.
- Analizar componentes existentes.
- Desarrollar componentes necesarios para el incremento.
- Establecer calendarios detallados de trabajo por componentes y programador.

Revisar que la programación sea compatible y consistente con los estándares.

Efectuar pruebas a los componentes nuevos.

SALIDAS

Pruebas unitarias documentadas.

Especificaciones de componente.

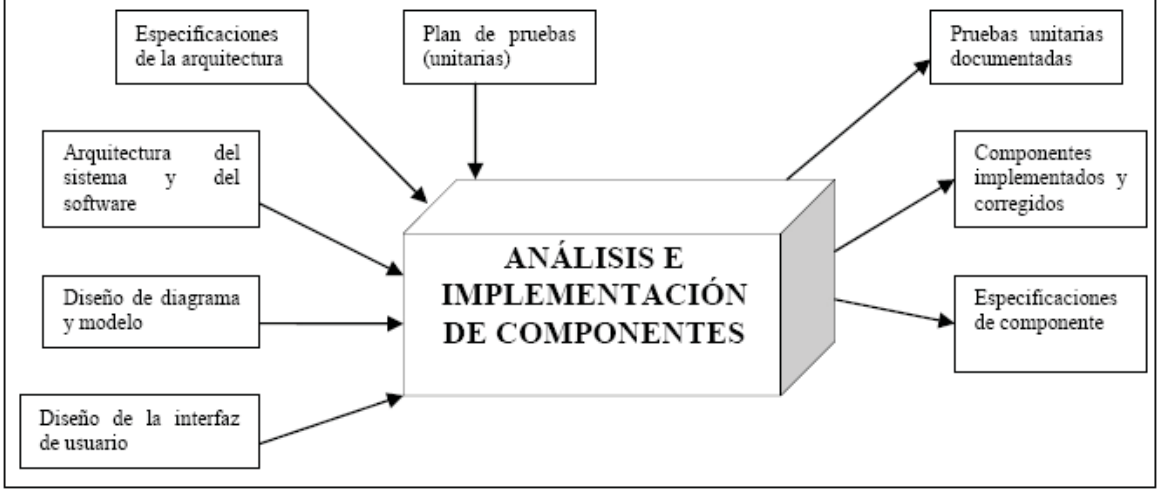
Componentes implementados y corregidos.

ACTORES

Jefe de área de programación.

Programadores.

GRAFO DEL PROCEDIMIENTO



4.5 Integración y Pruebas

Nombre Procedimiento: Integración y Pruebas
Tipo Procedimiento: Operación Fabrica Software

PROPOSITO
Esta etapa recibe los componentes requeridos por el incremento, desde la etapa de Análisis e Implementación de Componentes. Una vez con los componentes listos, se efectúa la integración de éstos para satisfacer en conjunto las características y funcionalidades del incremento correspondiente a la iteración. Junto con integrar los componentes se efectúan las pruebas de integración a los incrementos . Otra labor importante dentro de esta etapa es la elaboración de un manual de usuario, que permita el claro entendimiento de las funcionalidades del sistema a medida que éstas se implementan, dando origen a un documento formal del software que se entrega al cliente junto al software. Una vez completada la etapa se pasa a la siguiente iteración, entre la etapa de Análisis e Implementación de Componentes e Integración y Pruebas, para formar el siguiente incremento que sumado al anterior van completando las especificaciones del software. Finalmente cuando se integra el último incremento del software, se da por finalizada su implementación, siendo entregado a la etapa de instalación del sistema.

ENTRADAS

Plan de prueba de sistema

Especificaciones de los componentes del incremento.

Componentes implementados y corregidos.

ACTIVIDADES

Integrar componentes.

Definir plan de prueba de incremento.

Ejecutar la prueba de la integración.

Revisar resultados de la prueba.

Revisar que la programación sea compatible y consistente con los estándares.

Obtener de todos los participantes la aprobación de la planificación para el fin de etapa, plazos para la aceptación, carga de datos, instalación y término.

SALIDAS

Construcción de incrementos

Construcción del sistema (cuándo se completan todos los incrementos).

Manuales técnicos: Sistema, operación y usuario

Prueba del sistema e incrementos:

- Preparar ambiente de prueba: Datos, comunicaciones, sistema operativo, usuarios, roles, permisos y librerías.

- Ejecutar las pruebas definidas.

- Funcionalidad.

- Prueba de límites o excepciones.

- Ejecutar pruebas de enlace de sistema: Usabilidad, integración, desempeño, integridad, operabilidad.

- Documentar los resultados de las pruebas (esperados/obtenidos).

- Corregir errores.

- Sintonizar bese de datos, resolver problemas de comunicación.

- Obtener arquitectura red revisada.

- Obtener Indicaciones de desempeño.

- Controlar versiones de programas y su estado de revisión.

- Listas de errores (sobresalientes).

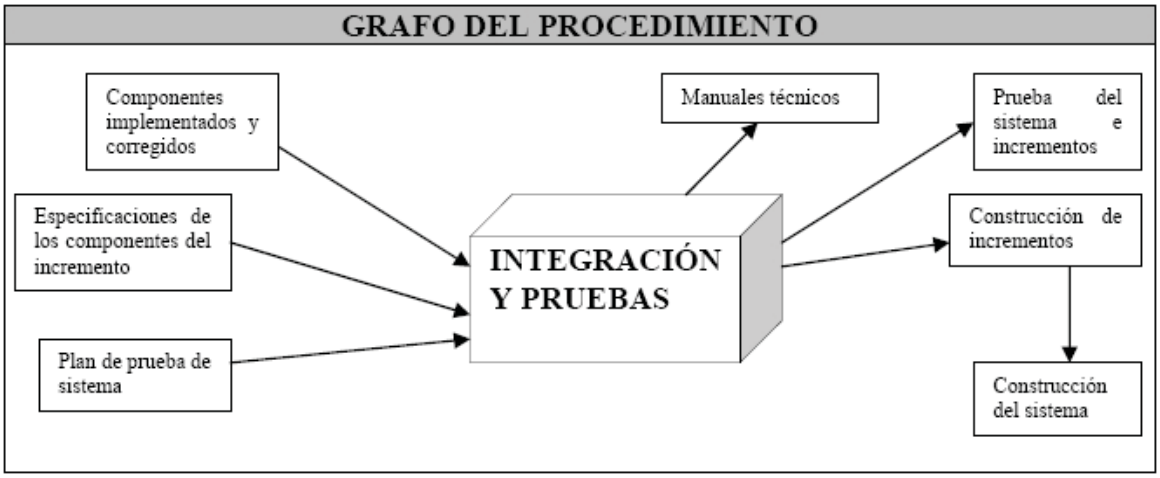
ACTORES

Jefe de área de programación.

Jefe de área de testing

Programadores.

Tester



4.6 Instalación del Sistema

Nombre Procedimiento: Instalación del Sistema
Tipo Procedimiento: Operación Fabrica Software

PROPOSITO
El objetivo principal de esta etapa es poner en marcha el sistema, llevando bitácoras de fallas y correcciones al sistema tanto en la arquitectura como en la aplicación . La instalación del sistema es una etapa que se implementa fuera de la Fábrica de Software, pero es parte de su servicio. En esta etapa se efectúa el cierre del proyecto, mediante la entrega e instalación del software al cliente .

ENTRADAS

Manuales técnicos: operación. Sistema y usuario.

Informe de usuarios del sistema y de pruebas del sistema

Datos de entrada al sistema

Soporte técnico (hardware y software)

ACTIVIDADES

Entrenamiento y capacitación según roles.

Carga de datos en el sistema.

Prueba general de aceptación del sistema.

Instalación configuración computacional.

Puesta en marcha.

Medición del servicio.

Cambios requeridos.

Mejoras.

Bitácora de fallas operativas.

Proveer servicio operacional.

SALIDAS

Entrenamiento y capacitación:

- Clientes entrenados según roles definidos.

- Base de datos de adiestramiento y apuntes.

Prueba general de aceptación del sistema:

- Registro de fallas y correcciones.

- Recopilación de opiniones de usuarios.

- Requerimientos de cambios.

- Obtener firma de aceptación clientes y control de calidad.

Carga de datos en el sistema: Datos iniciales, datos convertidos y entrada de datos al sistema.

Instalación configuración computacional:

- Hardware operativo (verificación de instalaciones y revisiones) y sistema operativo.

- Aprobación de instalaciones, sistema operativo.

- Requerimientos de insumos y consumos futuros (disquetes, cartujos, papel y discos).

Puesta en marcha:

- Aprobación del sistema.

- Inició y duración de la puesta en marcha.

Bitácora de fallas y correcciones de la arquitectura del sistema: Tipo de falla, descripción falla, nombre detector, solución, fecha y ejecutado por.

Documentación: Mantención del manual de sistema, operación y usuario incluyendo incorporaciones, modificaciones o eliminaciones.

ACTORES

Jefe de proyecto de fábrica.

Jefe de fábrica

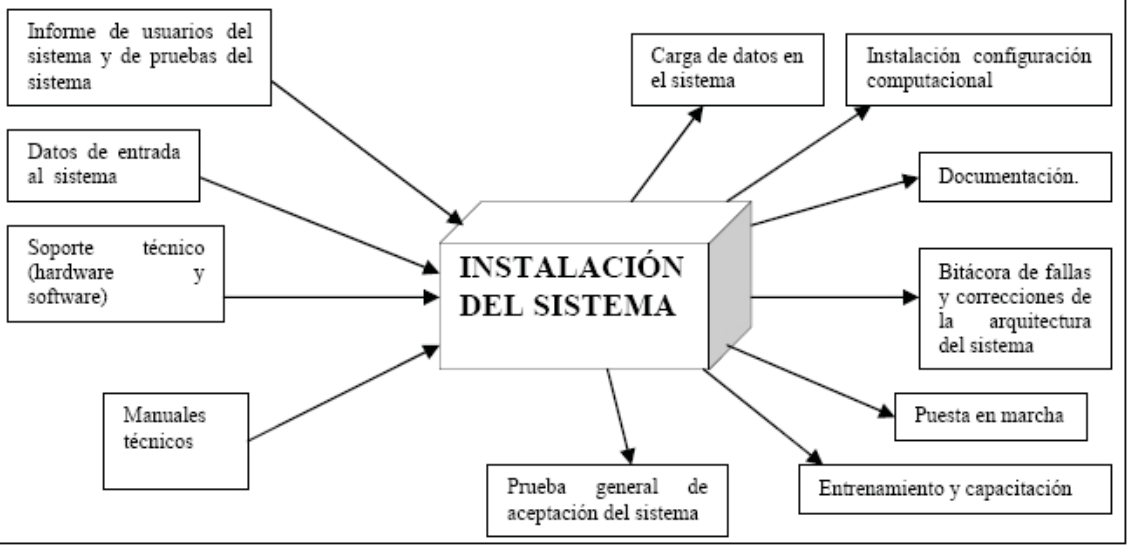
Jefe de área de programación

Jefe de área de testing

Testers

Programadores.

GRAFO DEL PROCEDIMIENTO



Capítulo 5
Administración de la Calidad

La calidad se define como “el grado en que un conjunto de características inherentes cumple con unos requisitos”. El aseguramiento de la calidad pretende dar confianza en que el producto reúne las características necesarias para satisfacer todos los requisitos del sistema de información. Por tanto, para asegurar la calidad de los productos resultantes el equipo de calidad deberá realizar un conjunto de actividades que servirán para :

Reducir, eliminar y lo más importante, prevenir las deficiencias de calidad de los productos a obtener.

Alcanzar una razonable confianza en que las prestaciones y servicios esperados por el cliente o el usuario queden satisfechas.

Para conseguir estos objetivos, es necesario desarrollar un plan de aseguramiento de calidad específico que se aplicará durante la planificación del proyecto de acuerdo a la estrategia de desarrollo adoptada en la gestión del proyecto. En el plan de aseguramiento de calidad se reflejan las actividades de calidad a realizar (normales o extraordinarias), los estándares a aplicar, los productos a revisar, los procedimientos a seguir en la obtención de los distintos productos y la normativa para informar de los defectos detectados a sus responsables y realizar el seguimiento de los mismos hasta su corrección. El grupo de aseguramiento de calidad participa en la revisión de los productos seleccionados para determinar si son conformes o no a los procedimientos, normas o criterios especificados, siendo totalmente independiente del equipo de desarrollo. Las actividades a realizar por el grupo de aseguramiento de calidad vienen dadas por el plan. Sus funciones están dirigidas a :

Identificar las posibles desviaciones en los estándares aplicados, así como en los requisitos y procedimientos especificados. Comprobar que se han llevado a cabo las medidas preventivas o correctoras necesarias.

Las revisiones son una de las actividades más importantes del aseguramiento de la calidad, debido a que permiten eliminar defectos lo más pronto posible, cuando son menos costosos de corregir. Además existen procedimientos extraordinarios, como las auditorías, aplicables en desarrollos singulares y en el transcurso de las cuales se revisarán tanto las actividades de desarrollo como las propias de aseguramiento de calidad. La detección anticipada de errores evita el que se propaguen a los restantes procesos de desarrollo, reduciendo substancialmente el esfuerzo invertido en los mismos.

Métrica 3 es una metodología de planificación, desarrollo y mantenimiento de sistemas de información. Puede ser utilizada libremente con la única restricción de citar la fuente de su propiedad intelectual, el Ministerio de Administraciones Públicas de España. Este Ministerio, desde el Consejo Superior de Informática, ofrece así un instrumento para la sistematización de las actividades que dan soporte al ciclo de vida del software en el desarrollo de sistemas de información. Métrica 3, proporciona cuatro interfaces que definen actividades orientadas a la mejora y perfeccionamiento de los procesos principales para garantizar la consecución del objetivo del desarrollo, dentro de las cuales esta el aseguramiento de la calidad, la que cubre los proceso de estudio de viabilidad del sistema, análisis del sistema de información, diseño del sistema de información, construcción del sistema de información, implantación y aceptación del sistema y mantenimiento de sistemas de información .

A continuación se describen utilizando las siguientes tablas, las actividades de aseguramiento de la calidad, a cumplir en el desarrollo del producto dentro de la Fábrica de Software basado en la metodología Métrica 3 , considerando las etapas del Ciclo de Vida del Proyecto de Software (CVPS) definidas en el capítulo 4.

5.1 Estrategia

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Constitución del equipo de aseguramiento de calidad	Recursos disponibles (externo)	Equipo de aseguramiento de calidad Plan de acción	Sesiones de trabajo Planificación	Grupo de aseguramiento de la calidad Jefe de proyecto de fábrica
Determinación de los sectores del sistemas de información objeto de aseguramiento de calidad	Equipo de aseguramiento de calidad Propuesta de sistema de información	Sectores del sistemas de información objeto de aseguramiento de calidad	Sesiones de trabajo	Grupo de aseguramiento de la calidad Jefe de proyecto de fábrica
Identificación de las propiedades de calidad	Sectores del sistemas de información objeto de aseguramiento de calidad	Propiedades de Calidad del sistemas de información objeto de	Sesiones de trabajo	Grupo de aseguramiento de la calidad Jefe de proyecto de fábrica

		aseguramiento de calidad		

Tabla 5.1: Identificación de las propiedades de calidad para el sistema

Constitución del equipo de aseguramiento de calidad. Se constituye el equipo de trabajo inicial necesario para determinar y valorar la conveniencia de establecer un plan de aseguramiento de calidad para el proyecto y se fija un plan de acción.

Determinación de los sectores del sistema de información objeto de aseguramiento de calidad. Para el sistema de información se determina qué sectores van a estar afectados por un plan de aseguramiento de calidad. Para ello se pueden considerar varios criterios relativos a las características que reúna el sistema, como pueden ser, un período de vida largo, sujeto a cambios constantes, equipo de trabajo heterogéneo en el desarrollo del sistema, operación continua, alta disponibilidad, fuerte interacción con otros sistemas y portabilidad.

Identificación de las propiedades de calidad. Una vez identificados los sectores del sistema de información para aplicar el plan asegurador de calidad, se definen para cada uno de ellos aquellas propiedades que permitan evaluar la calidad en cuanto a las características de operación, facilidad de mantenimiento y adaptabilidad a nuevos entornos. Algunas de estas propiedades pueden ser, por ejemplo, la facilidad de uso, eficiencia, seguridad, portabilidad, integridad y fiabilidad.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Necesidad del plan de aseguramiento de calidad	Estudio de riesgo Plan de trabajo del sistema	Necesidad de un plan de aseguramiento	Sesiones de trabajo	Grupo de aseguramiento

para el sistema	Propiedades de calidad del sistemas de información objeto de aseguramiento de calidad	de calidad		de la calidad
Alcance del plan de aseguramiento de calidad	Sistema de calidad existente en la organización (externo) Necesidad de un plan de aseguramiento de calidad	Plan de aseguramiento de calidad de cada sector	Sesiones de trabajo	Grupo de aseguramiento de la calidad
Impacto en el coste del sistema	Plan de aseguramiento de calidad Estudio de inversión	Coste del plan de aseguramiento de calidad	Análisis coste/beneficio	Grupo de aseguramiento de la calidad Jefe de proyecto de fábrica

Tabla 5.2: Establecimiento del plan de aseguramiento de calidad

Necesidad del plan de aseguramiento de calidad para el sistema. Se determina la necesidad de incluir un plan de aseguramiento de calidad en el sistema propuesto

teniendo en cuenta el análisis de riesgos y el enfoque del plan de trabajo, junto con las propiedades de calidad establecidas, para el sistema de información objeto de aseguramiento de calidad, en la tarea Identificación de las Propiedades de Calidad

Alcance del plan de aseguramiento de calidad. Si en la organización existe un sistema de calidad, se realiza una valoración de las directrices generales establecidas en el mismo, con el fin de proceder a su adaptación al plan de aseguramiento de calidad específico del sistema de información implicado, con el que se deben cubrir las propiedades de calidad identificadas anteriormente. En algunos casos puede ser necesario reflejar en el plan de aseguramiento de calidad ciertas normas que difieran en mayor o menor medida de las establecidas en el sistema de calidad. El contenido del plan de aseguramiento de calidad se fijará de acuerdo a los estándares del sistema de calidad, si existen, y en cualquier caso se incluirán aspectos tales como:

Propósito y alcance del plan en términos de propiedades de calidad.

Objetivos.

Actividades y tareas relacionadas con el aseguramiento de calidad a realizar a lo largo del desarrollo del software y responsabilidades.

Productos mínimos exigibles de ingeniería del software.

Estándares, prácticas y normas aplicables durante el desarrollo del software.

Tipos de revisiones, verificaciones y validaciones que se van a llevar a cabo, así como los responsables de su realización.

Criterios para la aceptación o rechazo de cada producto resultante de un proceso.

Procedimientos para implementar acciones correctoras o preventivas y realizar su seguimiento, identificando responsables.

Métodos para la salvaguarda y mantenimiento de la documentación obtenida en las actividades de aseguramiento de calidad.

Impacto en el coste del sistema. Una vez identificada la necesidad de un plan de aseguramiento de calidad y definido su alcance, se establece el coste adicional asociado al sistema de información propuesto, con el fin de aportar esta información al coste total del sistema y en consecuencia determinar su viabilidad económica. Este coste aporta la información necesaria para poder valorar globalmente la solución y determinar su viabilidad en el caso de que sea necesario un plan de aseguramiento de calidad.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
Ajuste del plan de aseguramiento de calidad	Plan de aseguramiento de calidad	Plan de aseguramiento de calidad con ajustes	Sesiones de trabajo	Grupo de aseguramiento de la calidad Jefe de proyecto de fábrica
Aprobación del Plan de aseguramiento de calidad	Plan de aseguramiento de calidad con ajustes	Registro de aprobación / rechazo		Grupo de aseguramiento de la calidad

Tabla 5.3: Adecuación del plan de aseguramiento de calidad a la solución

Ajuste del plan de aseguramiento de calidad. Una vez ponderadas las dificultades económicas, técnicas, etc. que surjan como consecuencia de incluir un plan de aseguramiento de calidad en la solución, puede ser necesario modificar el enfoque de algunas de las propiedades de calidad y el modo en que se les da respuesta, recogiendo estas variaciones en el plan de aseguramiento de calidad.

Aprobación del plan de aseguramiento de calidad. Se registra la aprobación del plan de aseguramiento de calidad asociado al sistema de información que conforman la solución. En caso de rechazo se incluirá el motivo.

5.2 Análisis y Definición de Requerimientos

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
Definición del plan de aseguramiento de calidad para el sistema de información	Plan de aseguramiento de calidad aprobado	Plan de aseguramiento de calidad: Aspectos generales	Sesiones de trabajo	Grupo de aseguramiento de la calidad Jefe de proyecto de fábrica

Tabla 5.4: Especificación inicial del plan de aseguramiento de calidad

Definición del plan de aseguramiento de calidad para el sistema de información. Se especifican de forma clara y detallada:

Estándares y normas a aplicar durante el proceso de desarrollo.

Procedimientos para informar, hacer seguimiento y resolver errores, identificando los responsables de llevarlos a cabo.

Mecanismos de modificación de productos estableciendo cómo se van a gestionar dichas modificaciones y el modo de comunicarlo a los implicados.

Modo de valorar las propiedades de calidad.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
Contenido del plan de aseguramiento de calidad para el sistema de información	Plan de aseguramiento de calidad: aspectos generales	Plan de aseguramiento de calidad: detallado Dossier de aseguramiento de calidad: índice	Sesiones de trabajo	Grupo de aseguramiento de la calidad Jefe de proyecto de fábrica

Tabla 5.5: Especificación detallada del plan de aseguramiento calidad

Contenido del plan de aseguramiento de calidad para el sistema de información. Una vez conocido el alcance del sistema de información objeto del análisis, se completa el plan de aseguramiento de calidad definido anteriormente, incluyendo:

Actividades y tareas a realizar en cuanto al aseguramiento de calidad y su emplazamiento a lo largo del proceso de desarrollo. Se valora, incluyéndo en su caso, la realización de auditorías.

Descripción de cada uno de los productos obtenidos en el proceso de desarrollo, como por ejemplo planes de pruebas, catálogo de requisitos, etc.

Revisiones a realizar, su propósito y criterios que se deben seguir en la revisión.

Calendario para la ejecución de estas actividades, incluyendo los recursos humanos y materiales necesarios para llevarlo a cabo.

Una vez definido el plan de aseguramiento de calidad, toda la información resultante de las actividades llevadas a cabo por el grupo de aseguramiento de calidad se incluirá en el dossier de calidad que formará parte del producto software.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
Revisión del catálogo de requisitos	Validación de requisitos Plan de aseguramiento de calidad Dossier de aseguramiento de calidad	Dossier de aseguramiento de calidad: Revisión de requisitos	Revisión técnica	Grupo de aseguramiento de la calidad
Revisión de la consistencia entre productos	Validación de requisitos Modelos del análisis revisados Plan de aseguramiento de calidad: Detallado Dossier de aseguramiento de calidad: Revisión de requisitos	Dossier de aseguramiento de calidad: Revisión de la consistencia entre productos	Revisión técnica	Grupo de aseguramiento de la calidad

Tabla 5.6: Revisión del análisis de consistencia

Revisión del catálogo de requisitos. Se valida que los requisitos se han especificado de una forma estructurada de acuerdo a los criterios establecidos en el plan de aseguramiento de calidad y que su contenido es preciso y completo. Asimismo, se comprueba que los requisitos del sistema de información son consistentes y que el equipo de desarrollo asume que puede satisfacerlos. En el caso de detectarse alguna deficiencia se remitirán las no conformidades a su responsable con el fin de que modifique o añada la información correspondiente, de acuerdo a los estándares establecidos.

Revisión de la consistencia entre productos. Se comprueba que todos los productos obtenidos se ajustan a las normas y estándares establecidos en el plan de aseguramiento de calidad y que responden a los requisitos especificados. Se revisa que se ha realizado la verificación y validación de los productos resultantes del análisis, así como la trazabilidad de requisitos.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
Registro de la aprobación del análisis del sistema de información	Aprobación del análisis del sistema de información	Dossier de aseguramiento de calidad: Registro de la aprobación del análisis del sistema de información		Grupo de aseguramiento de la calidad

Tabla 5.7: Registro de la aprobación del análisis del sistema

Registro de la aprobación del análisis del sistema de información. Se registra en el dossier de aseguramiento de calidad, la aprobación o rechazo de los productos resultantes

del análisis del sistema de información, teniendo en cuenta los criterios establecidos en el plan de aseguramiento de calidad en cuanto al tipo de productos a entregar, contenido, y normativa aplicada. En caso de rechazo se registran las no conformidades.

5.3 Diseño de Sistema y de Software

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión de la consistencia entre productos del diseño	Modelos del diseño verificados Plan de aseguramiento de calidad: Detallada Dossier de aseguramiento de calidad: Registro de la aprobación del análisis del sistema de información	Dossier de aseguramiento de Calidad: Revisión de la arquitectura del sistema	Revisión técnica	Grupo de aseguramiento de la calidad
Registro de la aceptación de la arquitectura	Aceptación técnica del diseño Dossier de aseguramiento de calidad: Revisión	Dossier de aseguramiento de Calidad:		Grupo de aseguramiento de la calidad

del sistema	de la arquitectura del sistema	Registro de la aceptación de la arquitectura del sistema		

Tabla 5.8: Revisión de la verificación de la arquitectura del sistema

Revisión de la consistencia entre productos del diseño. Se comprueba que todos los productos resultantes del diseño se ajustan a las normas y estándares establecidos en el plan de aseguramiento de calidad y se revisa que se ha realizado la verificación y validación de los mismos. Se comprueba que el diseño de la arquitectura del sistema responde a los requisitos especificados en el análisis.

Registro de la aceptación de la arquitectura del sistema. Se comprueba que los responsables de operación están de acuerdo con el diseño de la arquitectura del sistema, teniendo en cuenta el entorno tecnológico en el que va a estar operativo. Se registra la aprobación o rechazo de los productos resultantes teniendo en cuenta los criterios establecidos en el plan de aseguramiento de calidad. En caso de rechazo se registran las no conformidades.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión del diseño de las pruebas unitarias, de	Plan de Pruebas: especificación técnica de niveles de prueba Plan de aseguramiento de	Dossier de aseguramiento de calidad: Revisión del diseño de las pruebas	Revisión técnica	Grupo de aseguramiento de la calidad

integración y del sistema	calidad: Detallado Dossier de aseguramiento de calidad: Registro de la aceptación de la arquitectura del sistema			
Revisión del plan de pruebas	Plan de pruebas o planificación de las pruebas Plan de aseguramiento de calidad: Detallado Dossier de aseguramiento de calidad: Revisión del diseño de las pruebas	Dossier de aseguramiento de calidad: Revisión del plan de pruebas	Revisión técnica	Grupo de aseguramiento de la calidad

Tabla 5.9: Revisión de la especificación técnica del plan de pruebas

Revisión del diseño de las pruebas unitarias, de integración y del sistema. Según los criterios de revisión establecidos en el plan de aseguramiento de calidad, se comprueba que el diseño de las pruebas unitarias, de integración y del sistema, cumplen dichos criterios, en cuanto a la especificación de verificaciones, casos de prueba asociados a cada verificación, registro de resultados de las pruebas, informes de incidencias en la ejecución, etc. Con respecto a la especificación de los casos de prueba, se comprobará que se han tenido en cuenta las propiedades de calidad establecidas anteriormente. Asimismo, en función de las características del sistema de información, se determina

según los criterios de selección establecidos en el plan de aseguramiento de calidad sobre qué verificaciones y casos de prueba, unitarias, de integración y del sistema, se va a llevar a cabo la revisión.

Revisión del plan de pruebas. Se comprueba que en el plan de pruebas se han detallado tanto las pruebas de implantación como las de aceptación. Las verificaciones asociadas a las pruebas de implantación deben contemplar los requisitos no funcionales relacionados con las propiedades de calidad. En las pruebas de aceptación, se revisará que las verificaciones y casos de prueba propuestos van dirigidos a la comprobación de los criterios de aceptación establecidos por el usuario. Se revisa la planificación de las pruebas con el fin de establecer el propio plan de acción del grupo de aseguramiento de calidad.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión de los requisitos de documentación de usuario	Especificación de requisitos de Documentación de usuario Dossier de aseguramiento de calidad: Revisión del plan de pruebas	Dossier de aseguramiento de calidad: Revisión de los requisitos de documentación de usuario	Revisión técnica	Grupo de aseguramiento de la calidad
Revisión de los	Especificación de requisitos de	Dossier de	Revisión	Grupo de

requisitos de implantación	implantación Dossier de aseguramiento de calidad: Revisión de los requisitos de documentación de usuario	aseguramiento de calidad: Revisión de los requisitos de implantación	técnica	aseguramiento de la calidad

Tabla 5.10: Revisión de los requerimientos de implantación

Revisión de los requisitos de documentación de usuario. Se comprueba que se han identificado los requisitos necesarios relativos a la documentación que se va a entregar a los usuarios y a operación (tipos de documentos y estructura, formato en que se desarrollarán, estándares a seguir, soporte, número de copias a editar, etc.).

Revisión de los requisitos de implantación. Se comprueba que se han identificado y detallado los requisitos necesarios para la implantación del sistema relacionados con la instalación, formación e infraestructura.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Registro de la aprobación del diseño del sistema	Aprobación del diseño del sistema de información Dossier de	Dossier de aseguramiento de calidad: Registro de la aprobación del diseño del		Grupo de aseguramiento de la calidad

de información	aseguramiento de calidad: Revisión de los requisitos de implantación	sistema de información		

Tabla 5.11: Registro de la aprobación del diseño del sistema de información

Registro de la aprobación del diseño del sistema de información. Se registra la aprobación o rechazo de los productos resultantes del diseño del sistema de información, teniendo en cuenta los criterios establecidos en el plan de aseguramiento de calidad en cuanto al tipo de productos a entregar, contenido y normativa aplicada. En caso de rechazo se registrarán las no conformidades.

5.4 Análisis e Implementación de Componentes

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión de estándares	Componentes de software Dossier de aseguramiento de calidad: Registro de la aprobación del diseño del sistema de información	Dossier de aseguramiento de calidad: Revisión del código de componentes y procedimientos	Revisión técnica	Grupo de aseguramiento de la calidad

Tabla 5.12: Revisión del código de componentes

Revisión de normas de construcción. Se comprueba que se ha generado el código de los componentes y de los procedimientos de operación y seguridad de acuerdo con los criterios de nomenclatura y normativa vigentes en la organización y especificados en el proceso diseño del sistema de información. Si se considera oportuno se puede llevar a cabo una revisión más detallada del código por alguna propiedad de calidad relativa a la modularidad o autodefinición

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión de la realización de las Pruebas unitarias	Resultado de las pruebas unitarias Dossier de aseguramiento de calidad: Revisión del código de componentes y procedimientos	Dossier de aseguramiento de calidad: Revisión de la realización de las pruebas unitarias	Revisión técnica	Grupo de aseguramiento de la calidad

Tabla 5.13: Revisión de las pruebas unitarias

Revisión de la realización de las pruebas unitarias. Se comprueba la realización de las pruebas unitarias. Se lleva a cabo la revisión de las verificaciones y casos de prueba que se hayan determinado aplicando los criterios de selección y de acuerdo al modo que se recogió en los criterios de revisión del plan de aseguramiento de calidad. Para todo esto, se tendrá en cuenta la normativa establecida para la gestión de los resultados de dichas pruebas.

5.5 Integración y Pruebas

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión de la realización de las pruebas de integración	Evaluación del resultado de las pruebas de integración Dossier de aseguramiento de calidad: Revisión de la realización de las pruebas unitarias	Dossier de aseguramiento de calidad: Revisión de la realización de las pruebas de integración	Revisión técnica	Grupo de aseguramiento de la calidad
Revisión de la realización de las pruebas del sistema	Evaluación del resultado de las pruebas del sistema Dossier de aseguramiento de calidad: Revisión de la realización de las pruebas de integración	Dossier de aseguramiento de calidad: Revisión de la realización de las pruebas del sistema	Revisión técnica	Grupo de aseguramiento de la Calidad

Tabla 5.14: Revisión de las pruebas de integración del sistema

Revisión de la realización de las pruebas de integración. Se comprueba la realización de las pruebas de integración. Se lleva a cabo la revisión de las verificaciones y los casos

de prueba que se hayan determinado aplicando los criterios de selección y de acuerdo al modo que se recogió en los criterios de revisión del plan de aseguramiento de calidad. Para todo esto, se tendrá en cuenta la normativa establecida para la gestión de los resultados de dichas pruebas.

Revisión de la realización de las pruebas del sistema. Se comprueba la realización de las pruebas del sistema. Se lleva a cabo la revisión de las verificaciones y los casos de prueba que se hayan determinado de acuerdo al modo que se recogió en los criterios de revisión del plan de aseguramiento de calidad. Para todo esto, se tendrá en cuenta la normativa establecida para la gestión de los resultados de dichas pruebas. En el caso de existir casos de prueba adicionales, incorporados como consecuencia de las medidas correctoras tomadas para solventar los errores detectados, el grupo de aseguramiento de calidad revisará que se han resuelto de forma correcta. Igualmente se revisarán las incidencias no resueltas con el fin de validar hasta que punto se ven comprometidas las propiedades de calidad establecidas inicialmente.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión de los manuales de usuario	Manuales de usuario Dossier de aseguramiento de calidad: Revisión de la realización de las pruebas del sistema	Dossier de aseguramiento de calidad: Revisión de los manuales de usuario	Revisión técnica	Grupo de aseguramiento de la calidad

Tabla 5.15: Revisión de los manuales de usuario

Revisión de los manuales de usuario. Se comprueba que los manuales de operación y de usuario se han descrito de forma clara y concisa y se ajustan a los criterios y normativa establecidos.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión de la formación a usuarios finales	Especificación de la formación a usuarios finales Dossier de aseguramiento de calidad: Revisión de los manuales de usuario	Dossier de aseguramiento de calidad: Revisión de la formación a usuarios finales	Revisión técnica	Grupo de aseguramiento de la calidad

Tabla 5.16: Revisión de la formación a usuarios finales

Revisión de la formación a usuarios finales. Se revisa que se han definido los esquemas de formación a los usuarios finales del sistema de información y que se han identificado los distintos perfiles de usuario en función de sus capacidades, habilidades, experiencia y responsabilidades, así como los recursos necesarios para llevarlo a cabo.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES

Registro de la aprobación del sistema de información	Dossier de aseguramiento de calidad: Revisión de la formación a usuarios finales Presentación y aprobación del sistema de Información	Dossier de aseguramiento de calidad: Registro de la aprobación del sistema de información		Grupo de aseguramiento de la calidad

Tabla 5.17: Registro de la aprobación del sistema de información

Registro de la aprobación del sistema de información. Se registra la aprobación o rechazo del sistema de información construido teniendo en cuenta los criterios establecidos en el plan de aseguramiento de calidad en cuanto al tipo de productos a entregar, contenido y normativa aplicada. En caso de rechazo se registrarán las no conformidades.

5.6 Instalación del Sistema

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión del plan de implantación del sistema	Plan de implantación Equipo de implantación Dossier de	Dossier de aseguramiento de calidad: Revisión del plan de	Revisión técnica	Grupo de aseguramiento de la calidad

	aseguramiento de calidad: Registro de la aprobación del sistema de información	implantación		

Tabla 5.18: Revisión del plan de implementación del sistema

Revisión del plan de implantación del sistema. Se revisa que se ha elaborado un plan de implantación de acuerdo a la estrategia de implantación establecida en el proceso de diseño de sistema y de software, conforme a los requisitos de implantación establecidos. Asimismo, se comprueba que se ha establecido un plan de trabajo para la implantación que permita determinar las actividades a realizar por el grupo de aseguramiento de calidad durante el proceso de implantación.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Revisión de la realización de las pruebas de implantación del sistema	Evaluación del resultado de las pruebas de implantación Dossier de aseguramiento de calidad: Revisión del plan de	Dossier de aseguramiento de calidad: Revisión de las pruebas de implantación	Revisión técnica	Grupo de aseguramiento de la calidad

	implantación			
Registro de la aprobación de las pruebas de implantación del sistema	Dossier de aseguramiento de calidad: Revisión de las pruebas de implantación	Dossier de aseguramiento de calidad: Registro de la aprobación de las pruebas de implantación por operación		Grupo de aseguramiento de la calidad

Tabla 5.19: Revisión de las pruebas de implantación del sistema

Revisión de la realización de las pruebas de implantación del sistema. Se comprueba la realización de las pruebas de implantación. Se lleva a cabo la revisión de las verificaciones y casos de prueba que se hayan determinado para el sistema de información implicado en la implantación del sistema, tal y como se especificó en los criterios de revisión de los respectivos planes de aseguramiento de calidad. Para todo esto, se tendrá en cuenta la normativa establecida para la documentación de los resultados de dichas pruebas. En el caso de existir casos de prueba adicionales, incorporados como consecuencia de las medidas correctoras tomadas para solventar los errores detectados, el grupo de aseguramiento de calidad revisará que se han resuelto de forma correcta. Igualmente se revisarán las incidencias no resueltas con el fin de valorar hasta que punto se ven comprometidas las propiedades de calidad establecidas inicialmente.

Registro de la aprobación de las pruebas de implantación del sistema: se registra la aprobación o rechazo de las pruebas de implantación por parte del responsable de operación.

TAREA	PRODUCTOS DE	PRODUCTOS	TÉCNICAS Y	PARTICIPANTES

	ENTRADA	DE SALIDA	PRACTICAS	
Revisión de la realización de las pruebas de aceptación del sistema	Evaluación del resultado de las pruebas de aceptación Dossier de aseguramiento de calidad: Registro de la aprobación de las pruebas de implantación por operación	Dossier de aseguramiento de calidad: Revisión de la realización de las pruebas de aceptación	Revisión técnica	Grupo de aseguramiento de la calidad
Registro de la aprobación de las pruebas de aceptación del sistema	Dossier de aseguramiento de calidad: Revisión de la realización de las pruebas de aceptación	Dossier de aseguramiento de calidad: Registro de la aprobación de las pruebas de aceptación por el usuario		Grupo de aseguramiento de la calidad

Tabla 5.20: Revisión de las pruebas de aceptación del sistema

Revisión de la realización de las pruebas de aceptación del sistema. Se comprueba la realización de las pruebas de aceptación. Se lleva a cabo la revisión de las verificaciones y casos de prueba que se hayan determinado para cada sistema de información implicado en la implantación del sistema, de acuerdo al modo que se recogió en los criterios de revisión de los respectivos planes de aseguramiento de calidad. Para todo esto, se tendrá en cuenta la normativa establecida para la documentación de los resultados de dichas

pruebas. En el caso de existir casos de prueba adicionales, incorporados como consecuencia de las medidas correctoras tomadas para solventar los errores detectados, el grupo de aseguramiento de calidad revisará que se han resuelto de forma correcta. Igualmente se revisarán las incidencias no resueltas con el fin de valorar hasta que punto se ven comprometidas las propiedades de calidad establecidas inicialmente.

Registro de la aprobación de las pruebas de aceptación del sistema. Se registra la aprobación o rechazo de las pruebas de aceptación por parte del responsable de usuarios finales.

TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Registro de la aprobación de la implantación del sistema	Aprobación del sistema	Dossier de aseguramiento de calidad: Registro de la aprobación de la implantación del sistema	Revisión técnica	Grupo de aseguramiento de la calidad

Tabla 5.21: Registro de la aprobación de la implementación del sistema

Registro de la aprobación de la implantación del sistema. Se registra la aprobación de la implantación del sistema y se comprueba que el dossier de aseguramiento de calidad forma parte del producto software.

Capítulo 6
Reutilización de Componentes y Herramientas de Productividad

La reutilización de componentes de software implica directamente un incremento en la calidad de los sistemas desarrollados mediante la composición de tales componentes. Por analogía con otras áreas podemos afirmar que el uso de "piezas" (componentes) en repetidas ocasiones (reutilización) ofrece garantías de que la "pieza" carece de errores o "fallos". Cuanto más se reutiliza un componente menor es la probabilidad de encontrar errores en él. Así pues, 3 de las grandes ventajas que ofrece la reutilización de software son:

Aumento de la productividad.
Incremento en la calidad.
Disminución del tiempo de entrega.

6.1 Repositorio de Componentes de Software Reutilizables

El término componente reutilizable de software será usado en el capítulo 6 para referirse a código, es decir, procedimientos, funciones y hojas de estilo. Un requisito previo e indispensable para la reutilización de los componentes es la existencia de un repositorio (biblioteca de componentes software reutilizables) y de un sistema automático de gestión de dicha biblioteca.

Los componentes almacenados en el repositorio deben tener una representación estándar y estar bien documentados, siendo el sistema gestor de la biblioteca el encargado de organizar, proteger y gestionar dichos componentes. Es necesario por tanto un mecanismo

de búsqueda para localizar los componentes de interés, siendo también imprescindible un mecanismo de análisis para entender los componentes, y otro para medir su calidad. Debe disponerse además de las capacidades necesarias para identificar las relaciones entre diferentes componentes.

Las propiedades que debe reunir la biblioteca de componentes de software reutilizable son las siguientes:

Un sistema de gestión de bases de datos extenso, para almacenar y recuperar componentes, facilitando así el acceso, la búsqueda, el versionado, el control y la seguridad.
Operaciones que permitan al usuario crear, editar, ver y componer componentes.
Un esquema organizativo que actúe como una ayuda navegacional a través de la biblioteca.
Un formalismo de representación de componentes estándares.
Facilidad para crecer.

La reutilización de componentes de software comprende dos actividades principales y complementarias:

1. El desarrollo-para (desarrollar componentes reutilizables)

2. El desarrollo-con (reutilizar componentes existentes)

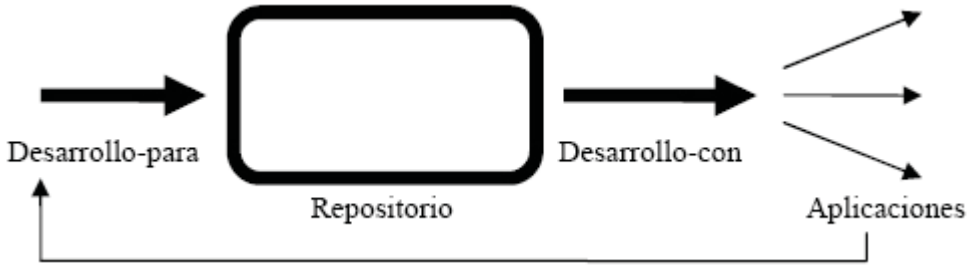


Figura 6.1: Ciclo de vida de la reutilización

El desarrollo-para reutilizar, se trata de construir componentes software reutilizables, que puedan ser usados en el desarrollo de aplicaciones similares en un contexto diferente (reutilización horizontal) o aplicaciones diferentes en el mismo contexto (reutilización vertical).

El desarrollo-con, comprende las actividades tradicionales de implementación de aplicaciones software, pero incluye además la búsqueda, selección y adaptación de componentes que se encuentran en un repositorio, para su inclusión en el sistema a desarrollar.

Actualmente, el desarrollo-con recibe el nombre de desarrollo sobre la base de componentes. A pesar de los diferentes roles, el desarrollo-para y el desarrollo-con son complementarios, como muestra la figura 6.1. El desarrollo-para es una actividad evolutiva que utiliza la experiencia acumulada durante el desarrollo-con, mientras este último utiliza los componentes desarrollados por el primero. Las actividades en cada uno de los desarrollos pueden verse en la figura 6.2 y la figura 6.3.

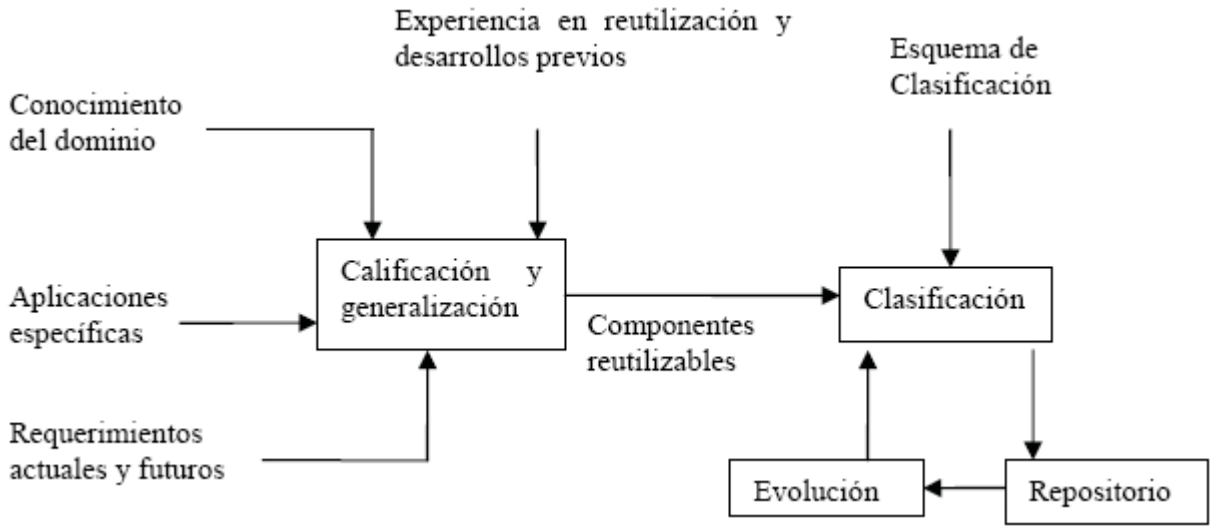


Figura 6.2: Actividades en el desarrollo-para

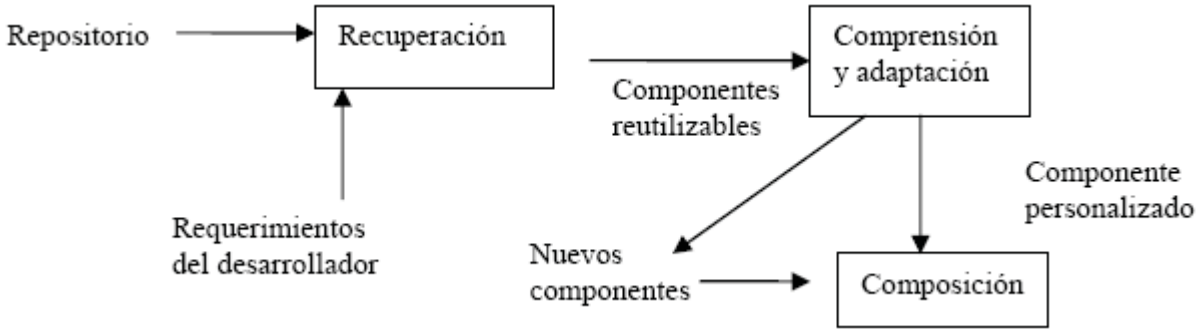


Figura 6.3: Actividades en el desarrollo-con

6.2 Clasificación y Recuperación de Componentes Reutilizables

El proceso de clasificación cataloga un componente reutilizable de software en un repositorio de componentes mediante una traducción directa del componente a una estructura de representación interna, por la cual el componente podrá ser recuperado en el futuro. En los mecanismos tradicionales de clasificación para la reutilización de componentes de software, la representación interna consiste en un conjunto de términos que describen al componente, de acuerdo a las categorías preestablecidas en el esquema de clasificación o en el lenguaje de representación. Para la asignación de términos (indización) puede utilizarse un lenguaje libre o un vocabulario controlado (específico del dominio). El vocabulario controlado establece el conjunto de términos que pueden utilizarse en el proceso de indización y su significado, algo así como un glosario especializado.

La recuperación de potenciales componentes para su reutilización, consiste en la búsqueda y selección de componentes presentes en el repositorio, que cumplen con las características impuestas por el usuario (restricciones). La búsqueda puede ser hecha mediante la visualización de una jerarquía de componentes en la biblioteca, lo que se denomino hojear; o siguiendo un conjunto de enlaces de hipertexto (o hipermedia), o por recuperación lineal. La recuperación lineal es la actividad típica de los sistemas de recuperación de información, en los que el usuario describe las características del objeto requerido mediante una consulta, y el sistema devuelve como resultado una lista de objetos que cumplen total o parcialmente con tales características. La estrecha relación

entre clasificación y recuperación, mediante sus actividades comunes, queda de manifiesto en la figura 6.4.

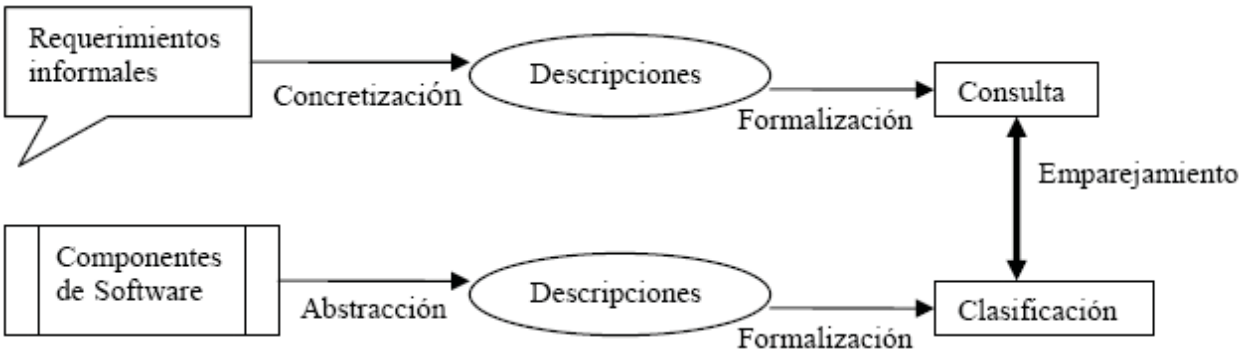


Figura 6.4: Recuperación y clasificación

Los mecanismos de recuperación en los sistemas de reutilización también pueden incluir métricas para asegurar el potencial de reutilización, y la facilidad de adaptación del componente, e información tal como:

Frecuencia de reutilización.
Accesos sin reutilización.
Accesos y uso sin modificación.
Uso con adaptaciones.
Calidad de la documentación.
Versión, lenguaje del componente.
Complejidad del componente (tamaño, número de entradas/salidas, etc.).

La teoría de recuperación de la información parece ser el acercamiento adecuado a los principales problemas de la clasificación y recuperación de componentes de software. A diferencia de lo que ocurre en los Sistemas Gestores de Bases de Datos (SGBD), los cuales se basan en la coincidencia total, en la recuperación de información o componentes de software es generalmente imposible formular la consulta precisa, y la información recuperada puede incluir ítems que coinciden total o parcialmente con los criterios de la consulta. Los sistemas de recuperación de la información permiten la "coincidencia

parcial", lo cual es uno de los requerimientos básicos de cualquier sistema de recuperación, donde la posibilidad de encontrar un componente que coincida exactamente es muy baja. El propósito de un Sistema de Recuperación de Información (SRI) es el de satisfacer gran variedad de necesidades de información. Una necesidad de información es formulada mediante una consulta al SRI, el cual responderá con una lista de ítems en la base de datos. La figura 6.5 muestra las principales funciones de un SRI.

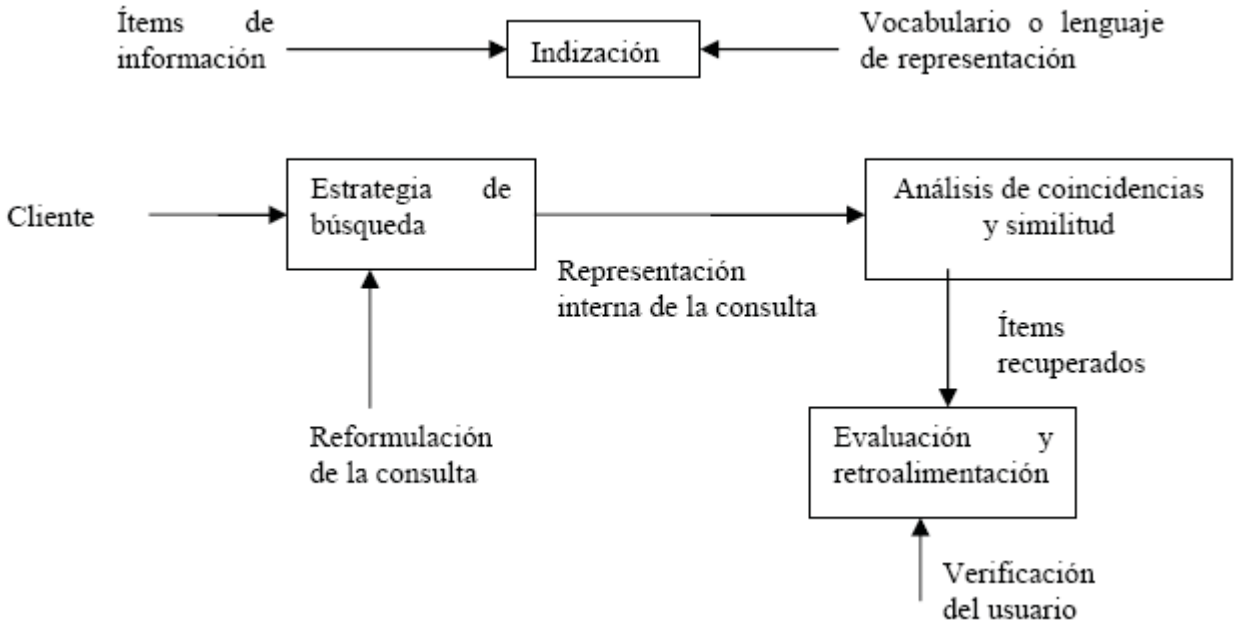


Figura 6.5: Principales funciones de un SRI

6.3 Herramientas de Productividad

Dentro de la Fábrica de Software, es importante destacar las herramientas que apoyan al desarrollo del producto de software en sus diferentes etapas y procesos, para lo cual se considera la tecnología de Ingeniería de Software Asistida por Ordenador, en inglés, Computer Aided Software Engineering (CASE) que supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información y se plantean los siguientes objetivos:

Automatizar: El desarrollo del software, la documentación, la generación del código, el chequeo de errores y la gestión del proyecto.

Permitir: La reutilización del software, la portabilidad del software y la estandarización de la documentación.

La siguiente clasificación es la más habitual basada en las fases del ciclo de desarrollo que cubren:

Upper CASE, herramientas que ayudan en las fases de planificación, análisis de requisitos y estrategia del desarrollo, usando diagramas de Lenguaje Unificado de Modelado, en inglés, Unified Modeling Language (UML).

Middle CASE, herramientas para automatizar tareas en el análisis y diseño de la aplicación.

Lower CASE, herramientas que semiautomatizan la generación de código, crean programas de detección de errores, soportan la depuración de programas y pruebas. Además automatizan la documentación completa de la aplicación.

A continuación se detallan las herramientas CASE que son importantes para el funcionamiento de la Fábrica de Software.

Herramientas de planificación de proyectos. Las herramientas de esta categoría se concentran en dos áreas primordiales:

Estimación de esfuerzos de proyecto y de costes de software. Calculan el esfuerzo estimado, la duración del proyecto y el número recomendado de personas.

Planificación de proyectos. Capacitan al administrador para definir todas las áreas del proyecto (la estructura de desglose de tareas), para crear una red de tareas (normalmente empleando una entrada gráfica), para representar las interdependencias entre tareas y para modelar la cantidad de paralelismo que sea posible para ese proyecto.

Herramientas de análisis y diseño. Permiten al desarrollador crear un modelo del sistema que se va a construir y también la evaluación de la validez y consistencia de este

modelo. Proporcionan un grado de confianza en la representación del análisis y ayudan a eliminar errores con anticipación:

- *Herramientas de análisis y diseño (Modelamiento):* Los modelos contienen una representación de los datos, de la función y del comportamiento (en el nivel de análisis), así como caracterizaciones del diseño de datos, arquitectura, procedimientos e interfaz. Al efectuar una comprobación de la consistencia y validez del modelo, las herramientas de análisis y diseño proporcionan un cierto grado de visión en lo tocante a la representación del análisis, y ayudan a eliminar errores antes de que se propaguen al diseño, o lo que es peor, a la propia implementación.

- *Herramientas para el diseño y desarrollo de interfaces:* Las herramientas de desarrollo y diseño de interfaz son en realidad un conjunto de primitivas de componente de programas tales como menús, botones, estructuras de ventanas, iconos, mecanismos de desplazamiento, controladores de dispositivos, etc., Sin embargo, estos conjuntos de herramientas se están viendo sustituidos por herramientas de generación de prototipos de interfaz que permiten una rápida creación en pantalla de sofisticadas interfaces de usuario, que se ajustan al estándar de interfaz que se haya adoptado para el software.

Herramientas de programación. Se engloban aquí los compiladores, los editores y los depuradores de los lenguajes de programación convencionales, además de los entornos de programación orientados a objetos, los lenguajes de cuarta generación, los entornos de programación gráfica, los generadores de aplicaciones y los lenguajes de consulta de bases de datos.

Herramientas Generadoras de código. A partir de las especificaciones del diseño se puede generar código tanto para los programas como los esquemas de bases de datos (sentencias de definición en SQL) convenientes. Actualmente, las herramientas CASE ofrecen interfaces con diversos lenguajes de cuarta generación para la construcción de sistemas de manera rápida.

Herramientas de prueba. Las herramientas de Pruebas de Software Asistida por Ordenador, en inglés Computer Aided Software Testing (CAST), y es un área bastante reciente dentro de la tecnología CASE. Algunas funcionalidades que suelen tener este tipo de herramientas son:

Gestión de pruebas.
Definir requisitos y objetivos de prueba.
Diseñar pruebas.
Construir entornos de ejecución de pruebas.
Ejecutar pruebas.

Herramientas de mantenimiento. La categoría de herramientas de mantenimiento se puede subdividir en:

Herramientas de ingeniería inversa: Toman el código fuente como entrada y generan modelos de diseño y análisis estructurado, listas de utilización y otra información con el diseño.
Herramientas interactivas de reingeniería de sistema: Se utilizan para modificar sistemas de base de datos.

Herramientas de soporte. Se engloban en esta categoría las herramientas que recogen las actividades aplicables en todo el proceso de desarrollo, como las que se relacionan a continuación:

Herramientas de documentación: Las herramientas de producción de documentos y autoedición prestan su apoyo a casi todos los aspectos de la ingeniería del software, que va desde la descripción textual de un pseudocódigo hasta diagramas más o menos complejos.
Herramientas de control de calidad: La mayor parte de las herramientas CASE que afirman que tiene como principal interés el control de calidad son en realidad herramientas métricas que hace una auditoria del código fuente para determinar si

es justa o no a ciertos estándares del lenguaje. Otras herramientas extraen métricas técnicas como base para medir la calidad del software que se esta construyendo.

Herramientas de bases de datos: El software de gestión de bases de datos sirve como fundamentos para establecer una base de datos. Dado el énfasis acerca de los objetos de configuración, las herramientas de gestión de bases de datos para pueden evolucionar a partir de los sistemas de gestión de bases de datos relacionales para transformarse en sistemas de gestión de bases de datos orientadas a objetos.

En la tabla 6.1 se muestra la relación existente entre las herramientas CASE y las etapas del CVPS. Además, se destaca la relación de las herramientas CASE con los diferentes actores de la Fábrica de Software a lo largo del desarrollo de un producto de software.

	Estrategia	Análisis y Definición de Requerimientos	Diseño de Sistema y de Software	Análisis e Implementación de Componentes	Integración y Pruebas	Instalación del Sistema
Herramientas de planificación de proyectos	Jefe de proyecto de fábrica					Jefe de proyecto de fábrica
Herramientas de análisis y diseño		Jefe de área de análisis Analistas	Jefe de área de diseño Diseñadores			

Herramientas para el diseño y desarrollo de interfaces			Jefe de área de diseño Diseñadores			
Herramientas de programación				Jefe de área de programación Programadores Verificador y validador de componentes	Jefes de las áreas de programación y testing Programadores y testers	
Herramientas Generadoras de código				Jefe de área de Programación Programadores	Jefes de las áreas de programación y testing Programadores y testers	
Herramientas de prueba				Verificador y validador de componentes	Jefes de las áreas de programación y testing	

					Programadores y testers	
Herramientas interactivas de reingeniería de sistema		Jefe de área de análisis Analistas	Jefe de área de diseño Diseñadores			
Herramientas de ingeniería inversa				Jefe de área de programación Programadores		
Herramientas de documentación	Documentador	Documentador	Documentador	Documentador	Documentador	Documentador
Herramientas de control de calidad	Jefe de administración de calidad Administrador de calidad	Jefe de administración de calidad Administrador de calidad	Jefe de administración de calidad Administrador de calidad	Jefe de administración de calidad Administrador de calidad	Jefe de administración de calidad Administrador de calidad	Jefe de administración de calidad Administrador de calidad

Herramientas de bases de datos				Jefe de área de programación Programadores		

Tabla 6.1: Relación entre las herramientas CASE y las etapas del CVPS

Para la implantación de herramientas CASE en una empresa es necesario considerar la siguiente estrategia:

Identificar la magnitud de problemas a resolver en la fábrica.

Identificar el nivel estratégico que deben tener los sistemas.

Evaluar los recursos de hardware y software disponibles en la fábrica y el medio.

Evaluar el nivel del personal.

Efectuar un estudio de costo-beneficio definiendo metas a lograr.

Elegir las herramientas apropiadas para la fábrica.

Establecer un programa de capacitación de personal de sistemas y usuarios.

- Elegir una aplicación que reúna la mayor parte de los siguientes requisitos:

- Gran impacto de resultados.

- Disponibilidad de recursos.

- Mínimo nivel de riesgos.

- Máxima colaboración de usuarios.

- Tamaño reducido de solución.

Se establecerán interfases de compatibilidad de los nuevos sistemas que deben convivir con los sistemas anteriores.

En la práctica los limites entre las diferentes herramientas, existentes en el mercado, son difusos. Las herramientas se venden como productos individuales pero proporcionan ayuda a diferentes actividades, por lo tanto no es fácil ubicar un producto utilizando una clasificación de herramientas CASE en particular .

A continuación, se muestran algunos de los productos CASE que se encuentran hoy para satisfacer las funcionalidades de algunas de las herramientas descritas anteriormente, priorizando por la utilización de herramientas CASE de código abierto como una alternativa de bajo costo para la automatización e integración de tareas dentro de la Fábrica de Software.

- Plone[2]: Es un Sistema de Gestión de Contenido, en inglés, Content Management System (CMS), basado en Zope y programado en Python. En un desarrollo basado en código abierto. Puede utilizarse como servidor intranet o extranet, un sistema de publicación de documentos y una herramienta de trabajo en grupo para colaborar entre entidades distantes.

- Open Workbench[3]: Es una herramienta de planificación de proyectos, al estilo de Microsoft Project. Se ejecuta en entornos Windows y es completamente libre. Esta herramienta soporta un amplio rango de funciones como la asignación de tareas hacia delante y hacia atrás, análisis de hitos, y tiempo estimado para la terminación de la solución.

[2] http://plone.org/about/team/

[3] http://www.openworkbench.org/

- ArgoUML[4]: Es una herramienta de código abierto para el análisis y el diseño de los sistemas de software orientados al objeto. Se trata de facilitar la comunicación entre desarrolladores, clientes, analistas y demás personas que intervienen en un proyecto utilizando un lenguaje gráfico denominado UML. En el caso de esta utilidad de modelado, ha sido construida en Java. Entre sus funciones podemos mencionar la posibilidad de intercambiar diseños con otras herramientas (como por ejemplo Rational Rose), diagramas de todo tipo, soporte para bases de datos, generación de código e incluso agentes que nos aconsejarán cómo mejorar nuestros diseños. Los diagramas pueden ser exportados a todo tipo de formatos gráficos para su inclusión en la documentación del proyecto.

- Nvu[5]: De forma similar al Dreamweaver, Nvu es un editor de código abierto que genera código de Lenguaje de Marcas de Transferencia de Hipertexto, en inglés, HyperText Markup Language (HTML) muy limpio. Nvu tiene la capacidad de servir como cliente web, es decir permite navegar por la web como cualquier otro buscador como Mozilla, Opera o Internet Explorer. De ahí que se le desarrollaron una serie de características tales como administración de contraseñas, gestor de descargas, etc. Nvu posee un arsenal de herramientas muy útiles como el validador de HTML, gestor de Hojas de Estilo en Cascada, en inglés, Cascade Style Sheet (CSS), etc. Además tiene incorporado un gestor de publicación de sitios, así como un limpiador de código.

- DBDesigner[6]: Es un software libre orientado al diseño de bases de datos para sistemas. Integra el diseño, modelado, creación y mantenimiento en un solo ambiente.

---

[4] http://argouml.tigris.org/

[5] http://nvu.com/

[6] http://fabforce.net/dbdesigner4/

DBDesigner 4 esta desarrollado y optimizado para bases de datos MySQL, aunque también brinda soporte para bases de datos Oracle, SQL-Server y conexiones vía ODBC. Esta escrito para los sistemas Linux con soporte para KDE/Gnome y Microsoft Windows 98, 2000/XP.

- MonoUML[7]: Es una herramienta de tipo CASE de código abierta la cual permite modelar diagramas UML y a través de ellos generar código fuente de manera rápida. Además MonoUML es capaz de hacer ingeniería inversa a través de aplicaciones ya escritas. MonoUML esta soportado por tecnologías abiertas como lo son la plataforma de desarrollo mono basado en los estándares de .NET y los estándares de UML 2.0, gracias a esto

- Scite[8]: Es un editor de textos con soporte para muchísimos lenguajes (C, Java, Phyton, etc.). Es multiplataforma y con licencia para libre uso, copia, modificación y distribución

- JUnit[9]: Es una herramienta que permite desarrollar nuestros casos y colecciones de prueba fácilmente, heredando de sus propias clases base y utilizando los mecanismos que nos proporciona. Además, ofrece una serie de interfaces gráficos para visualizar estas pruebas, ejecutarlas, ver sus resultados, seleccionar aquellas que queremos ejecutar, etc. A estas colecciones de clases, junto con sus herramientas se las conoce como "marcos de pruebas", ya que gracias a ellas, se tiene toda la infraestructura

[7] http://www.monouml.org

[8] http://www.scintilla.org/

[9] http://www.junit.org/

necesaria para desarrollar pruebas unitarias de forma rápida, cómoda, extensible y fiable. Todos los marcos de trabajo heredados de JUnit han recibido la denominación xUnit, con la que se indica que se trata de una migración, y se siguen las normas que marcó JUnit.

- phpMyAdmin[10]: Es una poderosa herramienta de código abierto para la programación de aplicaciones de PHP Preprocesador de Hipertexto, en inglés, PHP Hypertext Pre-processor (PHP) con MySql y Apache. La instalación es sumamente sencilla, solamente se debe copiar dentro de la sección de publicación del servidor web, para poder acceder a ella y comenzar a trabajar. Una de sus grandes ventajas es que es multiplataforma, es decir que puede trabajar en múltiples sistemas operativos.

- Doxygen[11]: Es una herramienta de código abierto para generar documentación a partir del código fuente, similar a javadoc, pero con soporte para mas lenguajes de programación (C++, C, C#, Java, IDL y PHP)

- RMS[12]: Herramienta de código abierto de análisis de calidad y métrica de código fuente. RMS proporciona un método estándar para el análisis de código fuente C, ANSI C++ y Java en diferentes sistemas operativos.

---

[10] http://www.phpmyadmin.net/

[11] http://www.doxygen.org/

[12] http://mSquaredTechnologies.com/

Capítulo 7

Procedimientos de Operación de la Fábrica de Software

Al interior de la fábrica se ejecutan diferentes procesos para llevar acabo la construcción de un producto de software, entre los cuales están aquellos que cubren las distintas etapas del CVPS descrito en el capítulo 4, acompañados de un proceso de aseguramiento de la calidad detallado en el capítulo 5. Sin embargo existen procesos críticos al interior de la fábrica que no quedan de manifiesto anteriormente, para lo cual se describen de forma detallada en este capítulo.

7.1 Ingreso de Proyecto a la Fábrica de Software

Nombre Procedimiento: Ingreso de proyecto a la Fábrica de Software.
Tipo Procedimiento: Particular a la Fábrica de Software.

PROPÓSITO

PROPÓSITO

Definir las actuaciones de los distintos involucrados al momento de responder al ingreso de un proyecto.

Generar una secuencia de pasos a seguir para realizar el ingreso de un proyecto al interior de la fábrica.

Definir los responsables de cada paso de la secuencia.

Asegurar que todo proyecto ingrese validado a la Fábrica de Software en términos técnicos y de planificación (costos y tiempos).

PROBLEMA

Enviar proyectos sin considerar la opinión del jefe de fábrica.

Enviar proyectos fuera de estándar definido por la Fábrica de Software.

Enviar proyectos incompletos en alguno de sus aspectos técnicos o comerciales.

Realizar proyectos que por falta de profundidad en su estudio sean el inicio de una situación de no conformidad.

SOLUCIÓN

Se genera un flujo de trabajo que incluye aspectos técnicos y comerciales para un ingreso consistente del proyecto en la Fábrica de Software:

Aspectos técnicos específicos los que son estudiados y proporcionados por el jefe de proyecto de fábrica.

Aspectos comerciales definidos por el jefe de proyecto de fábrica y validados por el jefe de fábrica, respecto de formato de presentación, estándares, tecnología y

disponibilidad.

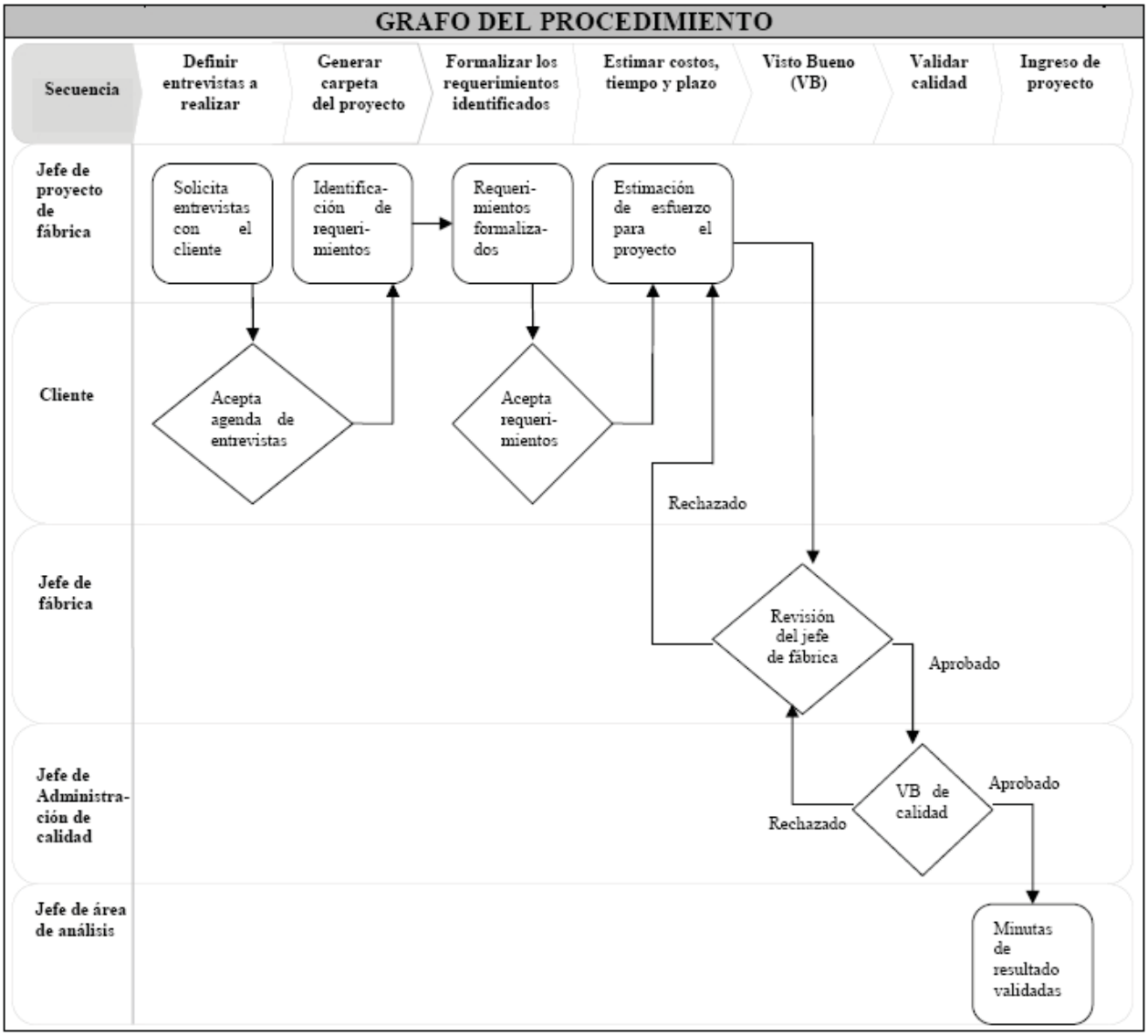
Visto bueno por parte del jefe de fábrica.

Ingreso consistente del proyecto a la Fábrica de Software, validado por el área de análisis.

N°	SECUENCIA	ACTUACIÓN	RESPONSABLE
1	Definir entrevistas a realizar	El jefe de proyecto de define una agenda de entrevistas junto con el cliente, por algún medio valido, correo, fax, teléfono, para realizar las entrevistas necesarias que den origen a la especificación de las necesidades del cliente. Este servicio puede ser un desarrollo o una mantención.	Jefe proyecto de fábrica
2	Generar carpeta del proyecto	Se efectúan las diversas entrevistas, en la modalidad acordada, con el equipo del cliente. De esta forma se genera la carpeta del proyecto, en donde se identifican los requerimientos, objetivos, alcance, beneficios cuantitativos y cualitativos del proyecto.	Jefe proyecto de fábrica
3	Formalizar los requerimientos identificados	Se formalizan las necesidades del proyecto para ser presentados al cliente en espera de su aprobación o ajustes.	Jefe proyecto de fábrica
4	Estimar costos,	Se efectúa una estimación de esfuerzos, con	Jefe proyecto de

	tiempo y plazo	la finalidad de estimar costos tiempo y de esta manera definir un plazo claro de término del proyecto. Se genera una planificación del proyecto con sus respectivos hitos.	fábrica
5	Visto bueno	El jefe de proyecto envía la carpeta del proyecto, aprobada por el cliente, al jefe de fábrica. El jefe de fábrica se encarga de verificar que la formulación del proyecto cumpla con los estándares preestablecidos por la Fábrica de Software, además de validar que la Fábrica de Software se encuentre en condiciones técnicas óptimas de asumir el proyecto. En caso de aprobación es entregado al proceso de aseguramiento de calidad, en caso contrario se entregan las observaciones al jefe de proyecto de fábrica.	Jefe de fábrica
6	Valida calidad	El administrador de calidad asignado, valida la entrada del proyecto a la fábrica, siguiendo los pasos indicados por su área frente a la administración de calidad en esta etapa del proyecto. En caso de aprobación pasa a su ingreso definitivo a la Fábrica de Software, en caso contrario son indicada las observaciones la	Jefe de administración de calidad

		jefe de fábrica.	
7	Ingreso de proyecto	El jefe de área de análisis hace la recepción del proyecto, que se encuentra validado tanto técnicamente como en calidad. El jefe de área de análisis programa reuniones con el jefe de proyecto de fábrica y se generan las minutas de resultado de reuniones con el objetivo de determinar si se están interpretando correctamente los requerimientos.	Jefe de área de análisis



7.2 Análisis y Definición de Requerimientos

Nombre Procedimiento: Análisis y definición de requerimientos.
Tipo Procedimiento: Particular a la Fábrica de Software.

PROPÓSITO

Definir las actuaciones de los distintos involucrados al momento de responder en el análisis y definición de requerimientos.

Generar una secuencia de pasos a seguir para realizar el análisis y definición de requerimientos de un proyecto al interior de la fábrica.

Definir los responsables de cada paso de la secuencia.

Asegurar que la etapa de análisis y definición de requerimientos de un producto pase validada a la siguiente etapa.

PROBLEMA

Utilizar un plan de acción sin aprobación del jefe de proyecto de fábrica.

Comenzar con el diseño del sistema y del software sin previa revisión de calidad del material de generado en la etapa de análisis y definición de requerimientos.

Comenzar con el diseño del sistema y del software sin la autorización del jefe de área de análisis.

Inserción de errores a nivel de análisis.

SOLUCIÓN

Se genera un flujo de trabajo que incluye aspectos técnicos para la etapa de Análisis y Definición de Requerimientos del producto de software al interior de la fábrica:

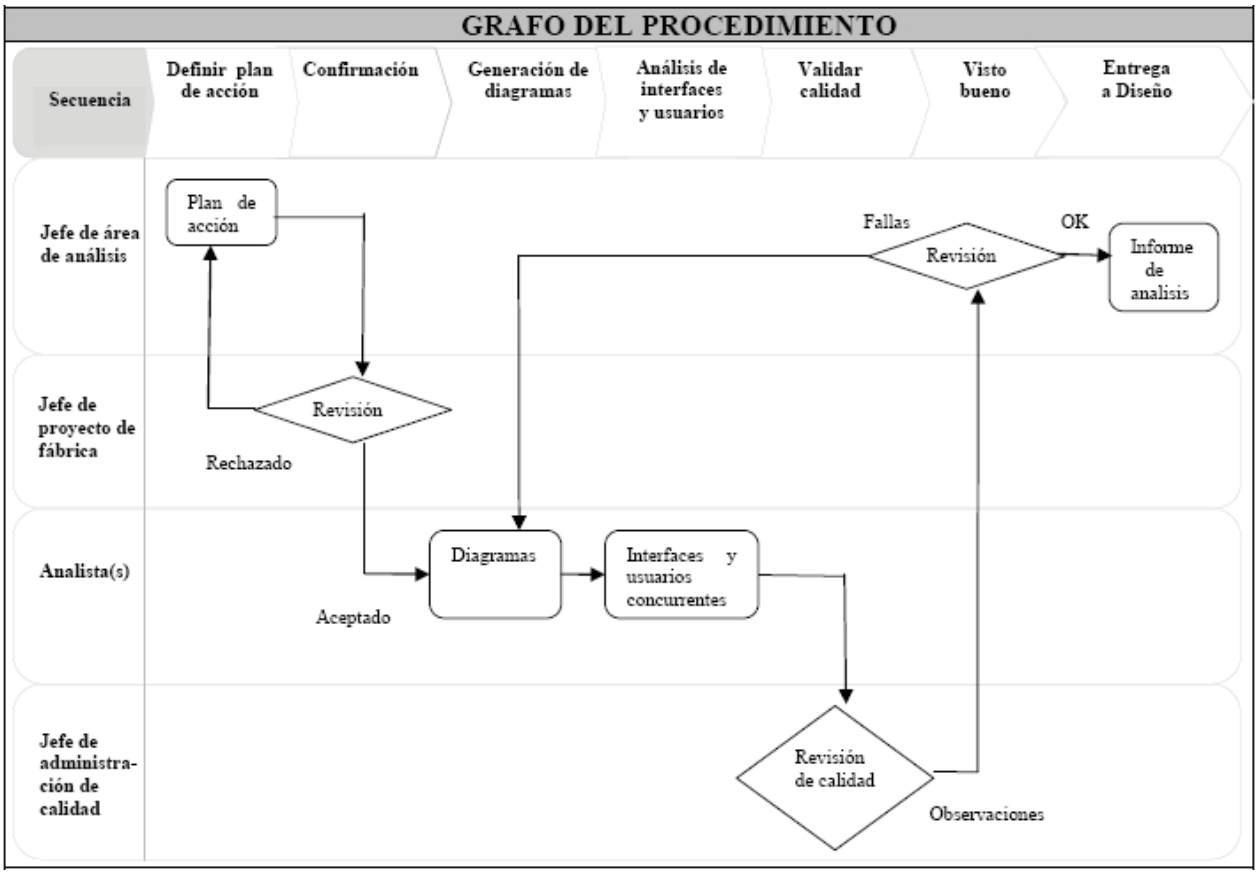
Confirmación del plan de acción por parte del jefe de proyecto de fábrica

Visto bueno por parte del área de administración de calidad.

Aprobación del jefe de área de análisis

N°	SECUENCIA	ACTUACIÓN	RESPONSABLE
1	Definir de plan de acción	El jefe de área de análisis define los compromisos necesarios de equipamientos, configuraciones, conectividad y soporte, los cuales deben ser aprobados por el jefe de proyecto de fábrica.	Jefe de área de análisis
2	Confirmación	Jefe de proyecto de fábrica revisa el plan de acción. En caso de aprobarlo queda definida la estrategia inicial de transición del sistema, en caso contrario es devuelto al jefe de área de análisis para su revisión.	Jefe proyecto de fábrica
3	Generación de diagramas	Se generan los diagramas del sistemas (según metodología: estructurada o orientada a objeto) y aquellos de interacción para los escenarios que sea necesario.	Analista(s)
4	Análisis de interfaces y usuarios	Se analiza las interfaces necesarias e integración con otros sistemas, además de aquellos usuarios que son concurrentes al sistema.	Analista(s)
5	Validar calidad	El administrador de calidad asignado, valida el material generado en este procedimiento, siguiendo los pasos indicados por su área frente a la administración de calidad en esta	Jefe de administración de calidad

		etapa del proyecto.	
6	Visto bueno	El jefe de área de análisis revisa observaciones efectuadas por el proceso de aseguramiento de la calidad. En caso de existir problemas es devuelto a los analistas, en caso contrario da su aprobación y posterior entrega al área de diseño.	Jefe de área de análisis
7	Entrega a Diseño	El jefe de área de análisis entrega al área de diseño la información de análisis del proyecto, estipulada en el documento de análisis.	Jefe de área de análisis



7.3 Diseño de Sistema y de Software

Nombre Procedimiento: Diseño de sistema y de software.
Tipo Procedimiento: Particular a la Fábrica de Software.

PROPÓSITO

Definir las actuaciones de los distintos involucrados al momento de hacer el diseño del sistema y del software.

Generar una secuencia de pasos a seguir para realizar el diseño del sistema y del software de un proyecto al interior de la fábrica.

Definir los responsables de cada paso de la secuencia.

Asegurar que la etapa de diseño del sistema y del software de un producto pase validada a la siguiente etapa.

PROBLEMA

Comenzar con la implementación del producto sin previa revisión de calidad del material generado en la etapa de diseño del sistema y del software.

Envió de la arquitectura del sistema a implementar sin la autorización del jefe de área de diseño.

Inserción de errores a nivel de diseño.

SOLUCIÓN

Se genera un flujo de trabajo que incluye aspectos técnicos para la etapa de Análisis y Definición de Requerimientos del producto de software al interior de la fábrica:

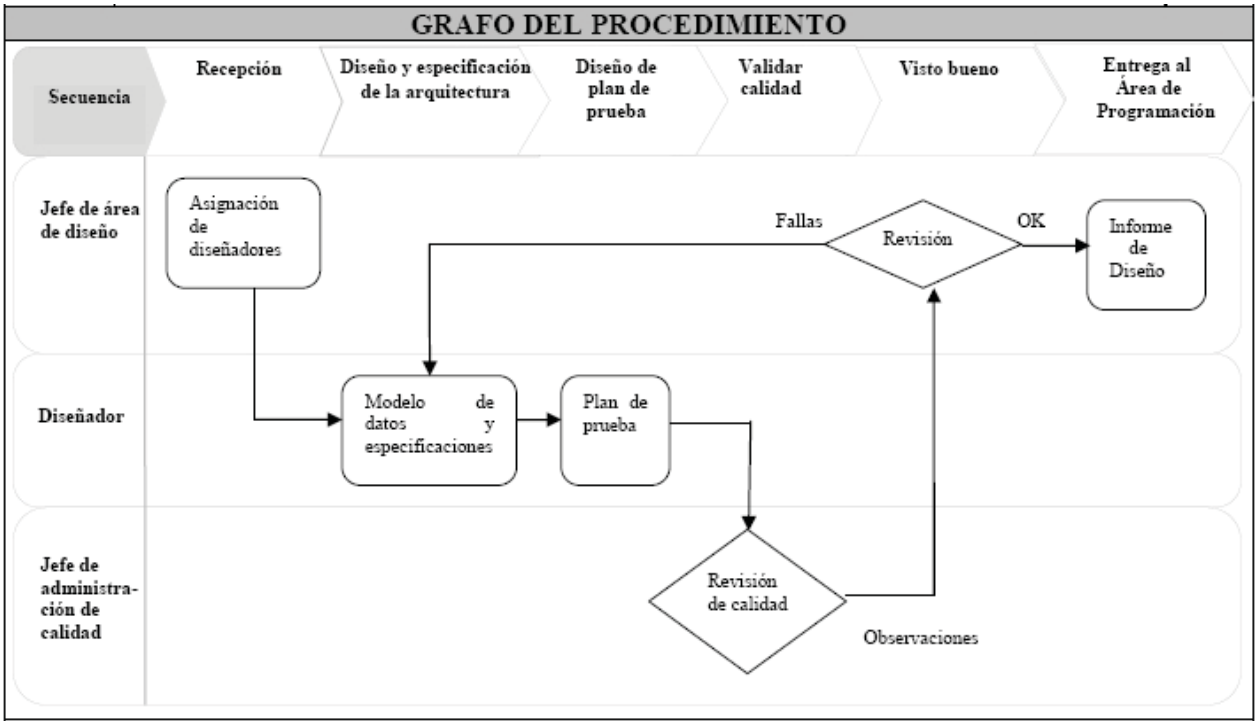
Recepción de requerimiento por parte del jefe de área de diseño.

Visto bueno por parte del área de administración de calidad.

Aprobación del jefe de área de diseño

N°	SECUENCIA	ACTUACIÓN	RESPONSABLE
1	Recepción	El jefe de área de diseño recibe la información de análisis del proyecto proveniente del área de análisis, asignando las labores a los diseñadores.	Jefe de área de diseño
2	Diseño y especificación de la arquitectura	Diseño del modelo de datos y especificación de interfaces, nombres y siglas del sistema. Diseño lógico de la red de comunicaciones y configuraciones Definición de acceso de usuarios	Diseñador
3	Plan de prueba	Diseño del plan de prueba inicial para el sistema.	Diseñador
4	Validar calidad	El administrador de calidad asignado, valida el material generado en este procedimiento, siguiendo los pasos indicados por su área frente a la administración de calidad en esta etapa del proyecto.	Jefe de administración de calidad

5	Visto bueno	El jefe de área de diseño revisa observaciones efectuadas por el proceso de aseguramiento de la calidad. En caso de existir problemas es devuelto a los diseñadores, en caso contrario da su aprobación y posterior entrega al área de programación.	Jefe de área de diseño
6	Entrega al Área de Programación	El jefe de área de diseño entrega al área de programación la información de diseño del proyecto, estipulada en el documento de diseño.	Jefe de área de diseño



7.4 Manejo de Incrementos del Software

Nombre Procedimiento: Manejo de incrementos del software.
Tipo Procedimiento: Particular a la Fábrica de Software.

PROPÓSITO
Definir las actuaciones de los distintos involucrados al momento de dividir en incrementos la implementación de un proyecto. Generar una secuencia de pasos a seguir para el manejo de los incrementos de software en la implementación de este al interior de la fábrica. Definir los responsables de cada paso de la secuencia. Asegurar que todo incremento de software sea validado en términos técnicos antes de continuar con el siguiente.

PROBLEMA
Generara una división de incrementos inadecuada sin considerar la opinión del jefe de área de programación. Enviar la solicitud de construir el siguiente incremento sin ser validado el anterior. Generara incrementos incompletos en sus funciones.

SOLUCIÓN
Se genera un flujo de trabajo que incluye aspectos técnicos para el manejo de

incrementos al momento de implementar le producto de software al interior de la fábrica:

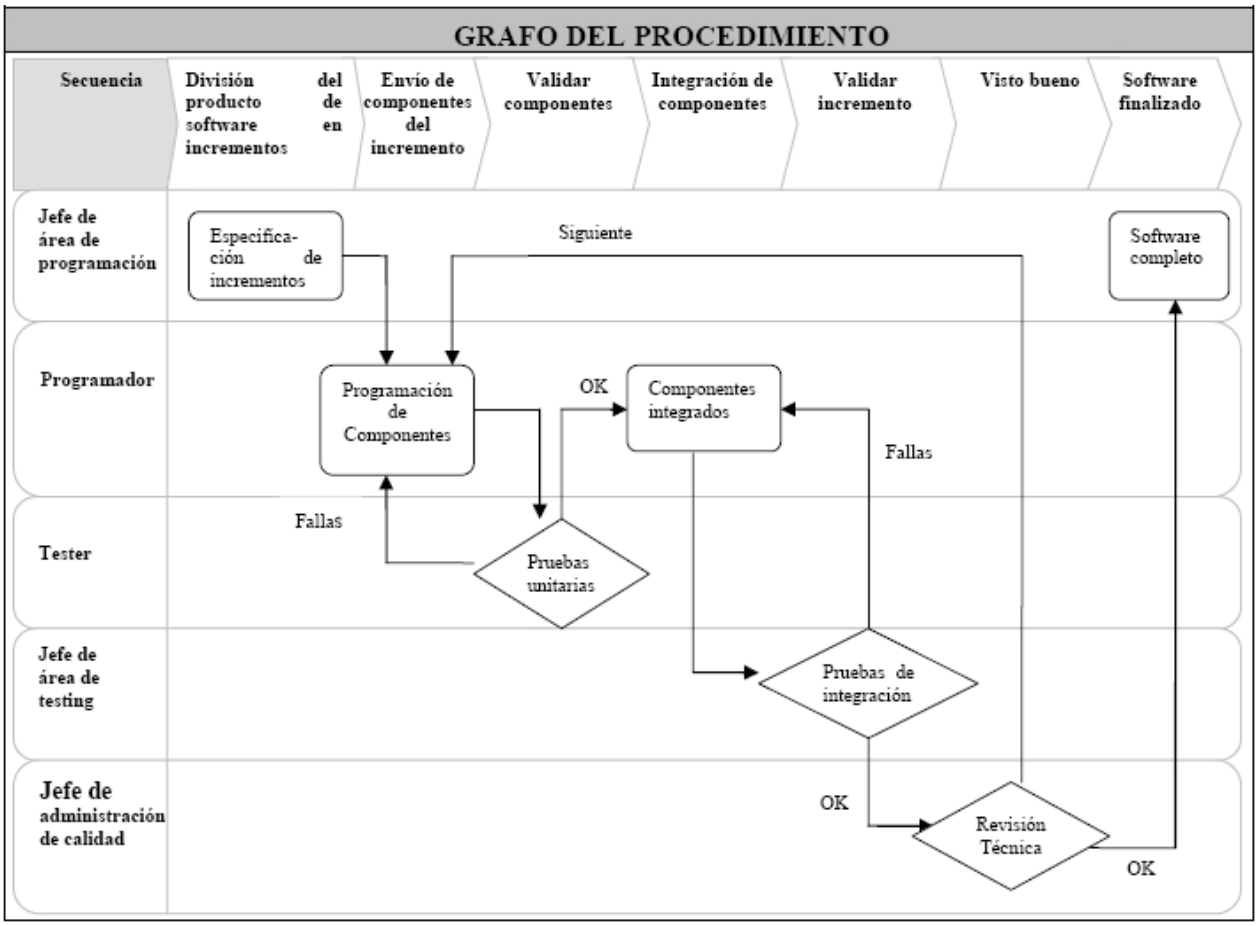
Aspectos técnicos de los componentes e incrementos que son validados por el área de testing.

Visto bueno por parte del área de administración de calidad.

División consistente de incrementos para su posterior implementación.

N°	SECUENCIA	ACTUACIÓN	RESPONSABLE
1	División del producto de software en incremento	Una vez recibida la información del diseño del producto de software, el área de programación se encarga de dividir su implementación en incrementos, dividiendo según las funcionalidades y procedimientos con que debe cumplir el producto.	Jefe área de programación
2	Envió de componentes del incremento	Los programadores reciben la orden de los componentes que se deben programar para la elaboración del respectivo incremento. Una vez programados aquellos componentes nuevos son enviados a someterse a pruebas unitarias.	Programador
3	Validar componentes	El tester encargado recibe aquellos componentes nuevos(los componentes reutilizados vienen validados por defecto) y efectúa las pruebas unitaria necesarias para validar los componentes por separados que	Tester

		serán enviado a su fase de integración.	
4	Integración de componentes	Los programadores integran los componentes recibidos para lograr de este modo la funcionalidad total del incremento.	Programador
5	Validar incremento	Se recibe los componentes integrados, lo que da paso a efectuar las pruebas de integración, para dar origen a un incremento validado	Jefe área de testing
6	Visto bueno	El administrador de calidad recibe las pruebas efectuadas a los incremento para asegurar que el proceso se efectuó correctamente.	Jefe de administración de calidad
7	Software finalizado	Después de haber efectuado todas las iteraciones necesarias para implementar los incrementos, el jefe del área de programación certifica que a finalizado la construcción del software, implementando el ultimo incremento sumado a los anteriores.	Jefe área de programación



7.5 Ingreso de Componentes Reutilizables a la Biblioteca

Nombre Procedimiento: Ingreso de componentes reutilizables a la biblioteca.
Tipo Procedimiento: Particular a la Fábrica de Software.

PROPÓSITO
Definir las actuaciones de los distintos involucrados al momento de proponer un

componente para ser ingresado en la biblioteca de componentes.

Generar una secuencia de pasos a seguir para realizar el ingreso de un componente reutilizable en la biblioteca.

Definir los responsables de cada paso de la secuencia.

Asegurar que todo componente reutilizable que ingrese a la biblioteca sea validado en términos técnicos por los encargados de la biblioteca de componente.

PROBLEMA

Ingresar componentes a la biblioteca que no sena validados.

No mantener una adecuada clasificación de los componentes propuestos.

Ingreso de componentes a la biblioteca que no sean verificados.

Ingreso de componentes que no cumplen con los estándares requeridos.

SOLUCIÓN

Se genera un flujo de trabajo que incluye aspectos técnicos para el ingreso consistente de un componentes de software a la biblioteca de componentes:

Aspectos funcionales proporcionados por los Programadores.

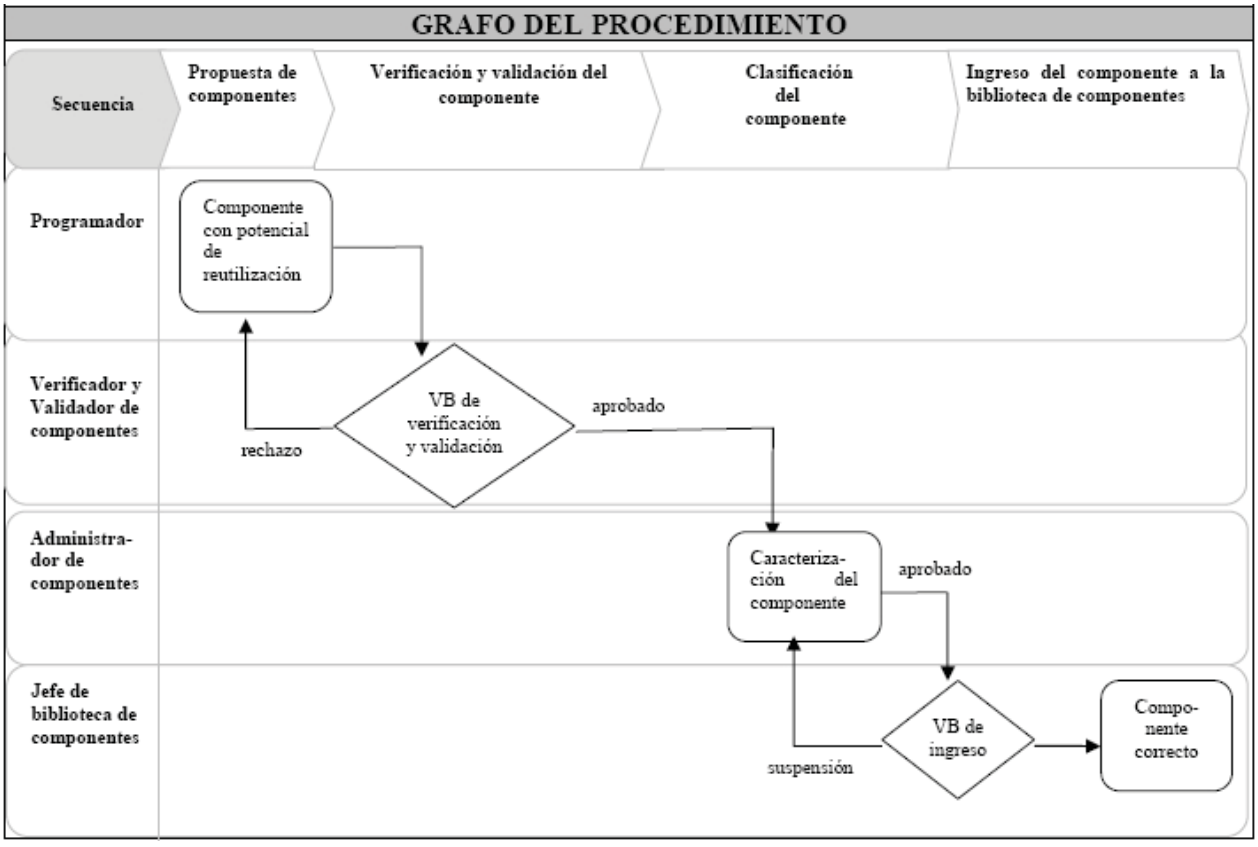
Aspectos técnicos validados y verificados por el verificador y validador de componentes.

Clasificaciones idóneas del componente por parte del administrador de componentes.

Ingreso consistente de un componente con potencial de reutilización a la biblioteca d e componentes, abalado por el jefe de biblioteca.

N°	SECUENCIA	ACTUACIÓN	RESPONSABLE
1	Propuesta de componente	El programador considera que el componente elaborado en su área posee un determinado potencial de reutilización (componente que cumple previamente con estándares de calidad), por lo tanto comunica a la biblioteca de componentes de esto por los medios existentes al interior de la Fábrica de Software.	Programador
2	Validación y verificación del componente	El verificador y validador de componente recibe la notificación del área de programación y procede a efectuar las respectivas pruebas a los componentes que acrediten una correcta validación y verificación del componente. En caso de aprobación se notifica al administrador de componente de la existencia de un componente valido para ser ingresado al sistema de gestión de componentes reutilizables, en caso contrario de rechazo de la propuesta efectuada por el programador, se le envía una notificación del rechazo.	Verificador y validador de componentes
4	Clasificación del	El administrador de componentes efectúa el procedimiento de caracterización del componente para dar con una clasificación	Administrador de componentes

	componentes	optima para su posterior registro en el sistema de gestión de componentes reutilizables	
5	Ingreso del componente a la biblioteca de componentes	Se recibe la notificación de un componente que se encuentra validado, verificado y clasificado. El jefe de biblioteca de componente revisa la información y en caso de aprobación ingresa el componente, sino comunica la administrador de la suspensión del componente.	Jefe de biblioteca de componentes



7.6 Reutilización de Componentes de la Biblioteca

Nombre Procedimiento: Reutilización de componentes de la biblioteca.
Tipo Procedimiento: Particular a la Fábrica de Software.

PROPÓSITO
Definir las actuaciones de los distintos involucrados al momento de responder a la solicitud de un componente para su reutilización. Generar una secuencia de pasos a seguir para realizar la reutilización de un componente de la biblioteca, en la construcción del producto al interior de la Fábrica de Software. Definir los responsables de cada paso de la secuencia. Asegurar que los componentes existentes en la biblioteca sean aprovechados correctamente para su reutilización en términos técnicos.

PROBLEMA
Solicitar componentes sin tener bien definido lo que se busca de parte del programador. Evitar recuperar de la biblioteca, componentes incompatibles en el aspecto técnico con lo requerido. Evitar niveles bajos de reutilización producto de la falta de una secuencia clara de

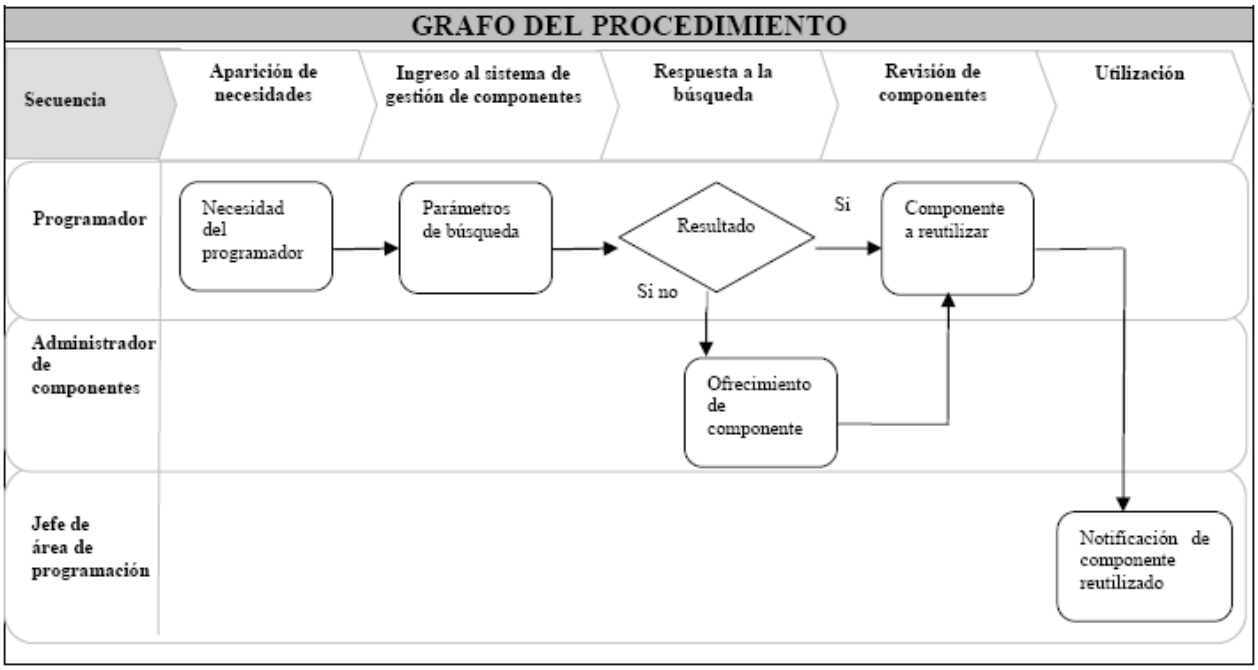
cómo hacer efectiva la reutilización.

SOLUCIÓN
Se genera un flujo de trabajo que incluye aspectos técnicos para la reutilización de un componente en la construcción de un determinado producto al interior de la Fábrica de Software: Aspectos técnicos específicos del componente que s pretende reutilizar, el cual es estudiado por el programador. Visto bueno por parte del jefe de área de programación. Reutilización consistente en la implementación del proyecto al interior de la Fábrica de Software, mediante el sistema de gestión de componentes que conforman la biblioteca

N°	SECUENCIA	ACTUACIÓN	RESPONSABLE
1	Aparición de necesidades	El programador tiene conocimiento de las necesidades del proyecto, por lo tanto desea chequear si se encuentran disponibles para su reutilización.	Programador
2	Ingreso al sistema de gestión de componentes	El programador mediante su cuenta de usuario del sistema de gestión de componentes ingresa a este. Una vez dentro ingresa los parámetros necesarios para la búsqueda del componente adecuado a sus necesidades y restricciones técnicas.	Programador

3	Respuesta a la búsqueda	Si bien la respuesta del sistema esta automatizada, es el administrador de componentes el responsable que la información devuelta por el sistema sea acorde a los estándares establecidos, permitiendo de este modo al programador una fácil evaluación sobre el resultado a su consulta, comparando necesidades versus resultados. En el caso de no tener una respuesta satisfactoria, por parte del programador, se puede comunicar directamente con el administrador de componentes (mediante fax, mail, teléfono, etc.) o intentar una nueva consulta al sistema.	Administrador de componentes
4	Revisión del componente	El programador recibe el componente, mediante algún medio extraíble o vía internet, luego procede a revisar la funcionalidad efectiva del componente en el procedimiento de implementación del software.	Programador
5	Utilización	Cuando se efectúa la utilización del componente por parte del programador en algún incremento del software, debe comunicar al jefe de área de programación aquellos componentes utilizados. De esta forma el jefe de área de programación se da por entendido de los componentes	Jefe de área de programación

		reutilizados en al proceso.	



7.7 Manejo de Incidentes Críticos

Nombre Procedimiento: Manejo de incidentes críticos.
Tipo Procedimiento: Particular a la Fábrica de Software.

PROPÓSITO
Definir las actuaciones de los distintos involucrados de forma general al momento de detectar la necesidad de un cambio o problema. Generar una secuencia de pasos a seguir para atender una situación de cambio o

problema en un proyecto al interior de la fábrica.

Definir los responsables de cada paso de la secuencia.

Asegurar que toda irregularidad al interior de la fábrica sea atendida de manera correcta.

PROBLEMA

Resolver incidentes críticos sin considerar la opinión del jefe de fábrica.

Resolver incidentes críticos sin la previa opinión del cliente en caso de que sea necesario.

Resolver incidentes críticos sin agendar reuniones entre el equipo interno de la Fábrica de Software y el jefe de proyecto de fábrica.

SOLUCIÓN

Se genera un flujo de trabajo para la notificación de un incidente critico durante el proceso de desarrollo de un producto de software al interior de la Fábrica de Software:

Aspectos técnicos que son estudiados y proporcionados por el administrador de calidad.

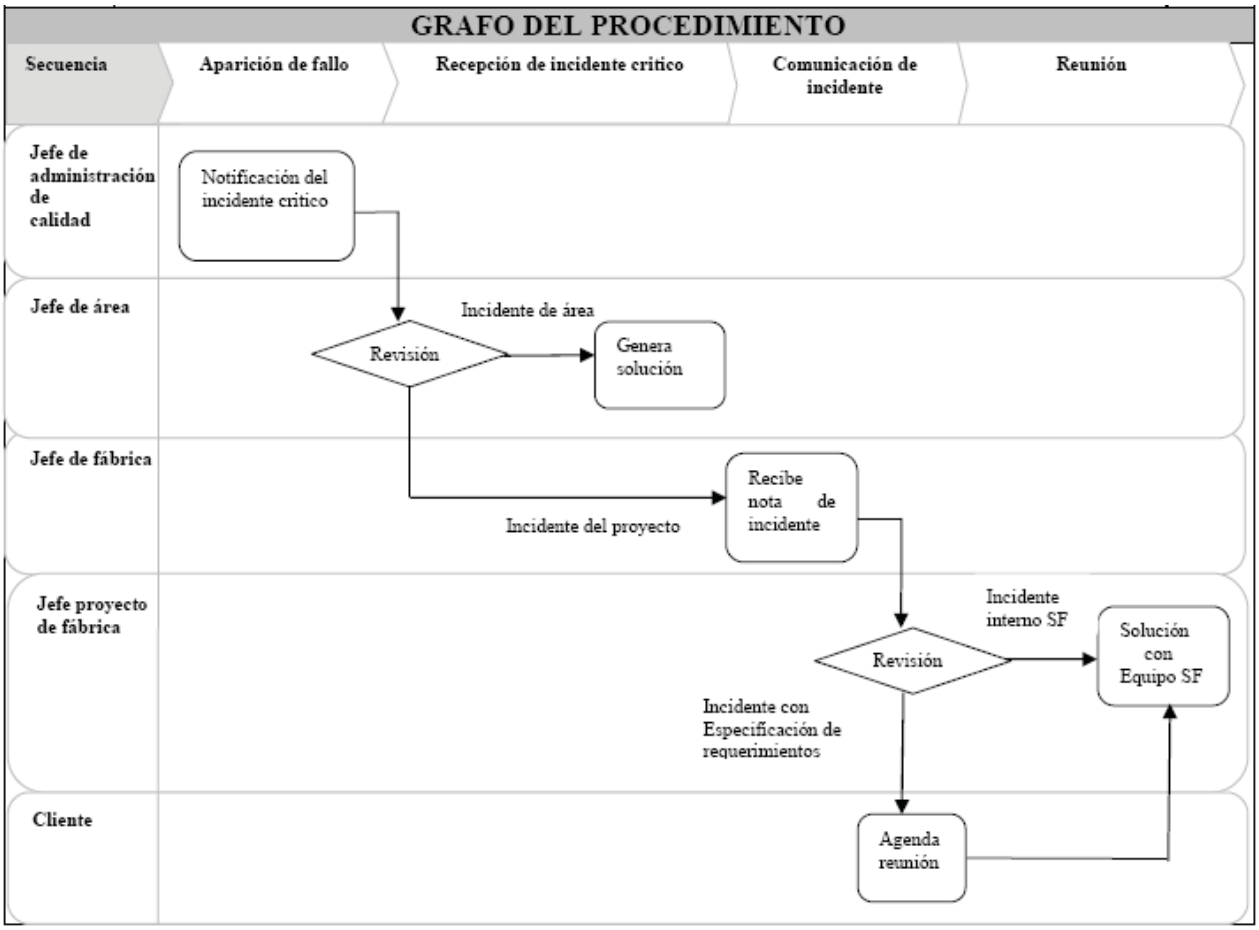
Aspectos comerciales definidos por el jefe de proyecto de fábrica y validados por el jefe de fábrica, que son reformulados mediante un contacto con el cliente.

Desarrollo consistente de un proyecto de software al momento de un problema, validado por el jefe de área según sea la etapa en la que se encuentre la anomalía.

N°	SECUENCIA	ACTUACIÓN	RESPONSABLE

1	Aparición del fallo	Durante el la administración de la calidad a lo largo del desarrollo del producto de software, el administrador de calidad detecta la necesidad de un cambio o la aparición de un problema en el control de las etapas del CVPS, para lo cual le comunica al jefe del área respectivo sobre el hecho.	Jefe de administración de calidad
2	Recepción de incidente critico	El jefe del área correspondiente recibe la notificación de parte del administrador de calidad y procede a ver si el problema es propio del área o del proyecto. En caso de ser del área, el jefe de área corrige el error con los integrantes de su área, de no ser así se notifica al jefe de fábrica lo sucedido.	Jefe de área
3	Comunicación de incidente	Recibe el jefe de fábrica la notificación de parte de un jefe de área sobre un cambio o problema en el proceso de desarrollo del software (etapa relativa al jefe de área). Se comunica con le jefe de proyecto de fábrica para dar a conocer la situación.	Jefe de fábrica
4	Reunión	El jefe de proyecto de fábrica recibe la solicitud del jefe de fábrica y solicita una reunión con los integrantes del área correspondiente a donde fue detectado el cambio o problema, para la cual se toma una decisión en conjunto. Si el jefe de proyecto de fábrica lo estima	Jefe proyecto de fábrica

		necesario puede consultar al cliente.	



7.8 Instalación del Sistema

Nombre Procedimiento: Instalación del sistema.
Tipo Procedimiento: Particular a la Fábrica de Software.

PROPÓSITO
Definir las actuaciones de los distintos involucrados al momento de la instalación del sistema. Generar una secuencia de pasos a seguir para realizar la instalación del sistema. Definir los responsables de cada paso de la secuencia. Asegurar que la etapa de instalación del producto sea validada y satisfactoria para el cliente.

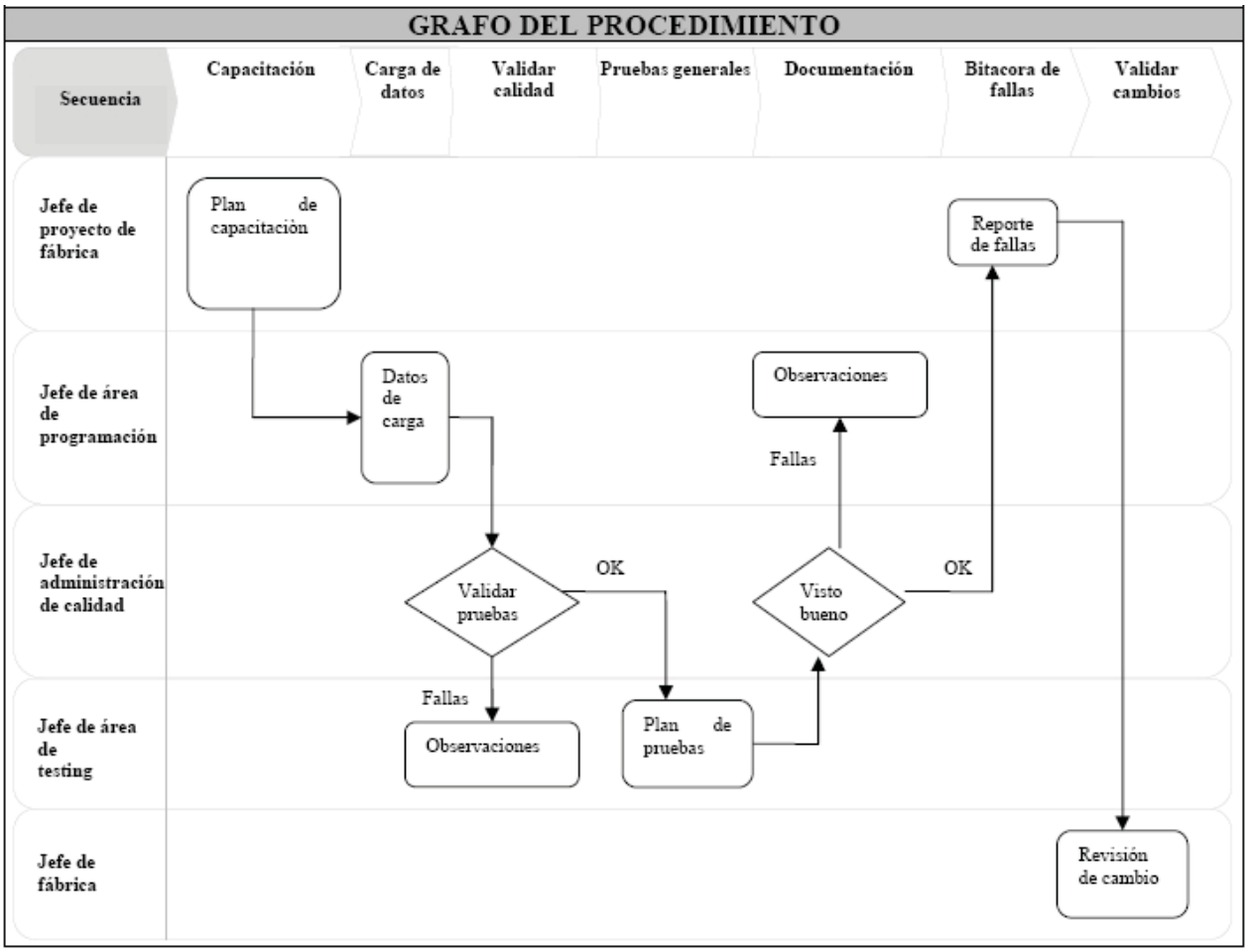
PROBLEMA
Comenzar con el uso del sistema sin una correcto entrenamiento de los diferentes usuarios. Iniciar la puesta en marcha sin el visto bueno del cliente. Desorden en la administración de fallas en la instalación. Darle fin al desarrollo del producto.

SOLUCIÓN
Se genera un flujo de trabajo que incluye aspectos técnicos para la etapa de Instalación del Sistema del CVPS: Capacitación del sistema y previa carga de datos para su uso. Validación de las pruebas en la instalación y sus resultados. Control de las fallas posteriores por parte del jefe de proyecto de fábrica Evaluación por parte del jefe de fábrica en los cambios al sistema que se soliciten

a la Fábrica de Software

N°	SECUENCIA	ACTUACIÓN	RESPONSABLE
1	Capacitación	El jefe de proyecto de fábrica coordina el entrenamiento a los usuarios según sus roles, además de disponer de una base de datos de prueba para la capacitación.	Jefe de proyecto de fábrica
2	Carga de datos	El jefe de área de programación se encargan de efectuar la carga de datos necesarios para la puesta en marcha del sistema (datos iniciales, datos convertidos y entrada de datos al sistema).	Jefe de área de programación
3	Validar calidad	Se revisa el plan de implementación del sistema y las pruebas a realizar. En caso de aprobación se procede a efectuar las pruebas, en caso contrario es envían las observaciones al área de testing.	Jefe de administración de calidad
4	Pruebas generales	El jefe de área de testing gestiona los casos de pruebas para el sistema una vez instalado en el cliente.	Jefe de área de testing
5	Documentación	Se revisan los registros y realización de pruebas de implantación y aceptación del sistema. En caso de aprobación del cliente se da inicio al apuesta en marcha del sistema y entrega oficial de la documentación del sistema, en caso contrario se hacen las	Jefe de área de administración de calidad

		modificaciones necesarias en la configuración del sistema.	
6	Bitácora de fallas	El jefe de proyecto de fábrica se hace cargo de la bitácora de fallas y correcciones a la arquitectura que sean detectadas en el uso del sistema (tipo de falla, descripción, nombre detector, solución, fecha y ejecutado por.)	Jefe de proyecto de fábrica
7	Validar cambios	Se hace la evaluación de los cambios que se soliciten después de la puesta en marcha del sistema para ver que este dentro de lo estipulado como modificación y no mantención. El jefe de fábrica asigna la modificación según corresponda el área	Jefe de fábrica


GRAFO DEL PROCEDIMIENTO
Secuencia
Capacitación
Carga de datos
Validar calidad
Pruebas generales
Documentación
Bitacora de fallas
Validar cambios
Jefe de proyecto de fábrica
Jefe de área de programación
Jefe de administración de calidad
Jefe de área de testing
Jefe de fábrica
Plan de capacitación
Datos de carga
Validar pruebas
OK
Fallas
Observaciones
Plan de pruebas
Visto bueno
OK
Fallas
Observaciones
Reporte de fallas
Revisión de cambio

Capítulo 8

Métricas de la Fábrica de Software

Las métricas van a servir para utilizarlas en un proyecto de software para ayudar en la estimación, control de calidad, evaluación de la productividad y control de proyectos. Existen cuatro razones principales para medir: caracterizar, evaluar predecir y mejorar. Para llevar acabo esto es que se establecen indicadores, los cuales proporcionan una visión profunda del proceso del software, del proyecto de software o del producto en si, para esto se definen indicadores.

Las métricas puedes ser orientada al tamaño, es decir cantidad de Líneas De Código (LDC), como también las métricas pueden ser orientadas a las funciones utilizando el cálculo de Puntos de Función (PF).

Para efectuar el cálculo de las LDC debe considerarse lo siguiente:

Debe contabilizarse cada línea nueva o modificada.
Las líneas para la instrumentación de código (por ejemplo para las pruebas) no deben incluirse en el tamaño total, salvo que tengan un carácter definitivo.
Las líneas de código de programas de prueba tan solo se contabilizan si se desarrollan con el nivel de calidad exigido al entregar el producto.
No se consideran los comentarios.
No se contabiliza el pseudocódigo.

Las ventajas de utilizar LDC:

Fácil de calcular.

Existen muchos modelos de estimación basados en LDC.

Existen muchas medidas de LDC

Los inconvenientes de utilizar LDC:

Dependientes de los lenguajes de programación.

Perjudican a los programas cortos, pero bien diseñados.

Difícil uso en estimación debido al nivel de detalle.

Para efectuar el cálculo de los PF debe considerarse la tabla 8.1:

	CUENTA		SIMPLE	MEDIO	COMPLEJO		
NÚMERO DE ENTRADAS DE USUARIO		X	3	4	6	=	
NÚMERO DE SALIDAS DE USUARIO		X	4	5	7	=	
NÚMERO DE PETICIONES DE USUARIO		X	3	4	6	=	
NÚMERO DE ARCHIVO		X	7	10	15	=	
NÚMERO DE INTERFACES EXTERNAS		X	5	7	10	=	
CUENTA TOTAL	→						

Tabla 8.1: Tabla de cálculo de puntos de función

Los parámetros de medición son:

Entradas de usuario: entradas de usuario que proporcionan diferentes datos orientados a la aplicación.

Salidas de usuario: salidas que proporcionan al usuario información orientada a la aplicación (por ejemplo informes, pantallas, mensajes de error, etc.).

Peticiones de usuario: entradas interactivas que producen la generación de alguna respuesta del software inmediata en forma de de salida interactiva.

Archivos: se cuenta cada archivo maestro lógico, i.e., cada grupo de datos que puede ser una parte de una gran base de datos o sistema de archivos.

Interfaces externas: interfaces legibles por la máquina que se utilizan para transmitir información a otro sistema (por ejemplo cinta, red, etc.).

Se calcula en base a la siguiente expresión:

$$\mathbf{PF = cuenta\_total*(0{,}65+0{,}01*} \sum_{i=1...14} Fi \mathbf{)}$$

(8.1)

Donde los valores de ajuste complejidad (Fi) se calculan respondiendo a las siguientes preguntas en una escala desde 0 (no importante o aplicable) hasta 5 (absolutamente esencial):

1. ¿Requiere el sistema copias de seguridad y de recuperación fiables?

2. ¿Se requiere comunicación de datos?

3. ¿Existen funciones de procesamiento distribuido?

4. ¿Es crítico el rendimiento?

5. ¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?

6. ¿Requiere el sistema entrada de datos interactiva?

7. ¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?

8. ¿Se actualizan los archivos maestros de forma interactiva?

9. ¿Son complejas las entradas, las salidas, los archivos o las peticiones?

10. ¿Es complejo el procesamiento interno?

11. ¿Se ha diseñado el código para ser reutilizable?

12. ¿Están incluidas en el diseño la conversión e instalación?

13. ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?

14. ¿Se ha diseñado la aplicación para facilitar los cambios y ser fácilmente utilizada por el usuario?

Las ventajas de utilizar PF:

Independientes del lenguaje de programación.
Permiten hacer estimaciones más fácilmente.
Los inconvenientes de usar PF:
Basadas en cálculos subjetivos.
Parámetros y factores no evidentes.
No tienen un significado físico directo.

8.1 Indicadores del Proyecto

Indicadores de productividad. Se obtiene considerando el total de LDC o PF y el total de meses utilizados para la generación de estas.

IP_1= LDC/Meses

(8.2)

IP_2= PF/Meses

(8.3)

Otra forma es considerando las personas que trabajaron en el desarrollo de la líneas de código o puntos de función.

IP_3= LDC/Personas

(8.4)

IP_4= PF/Personas

(8.5)

Indicador de esfuerzo. Un concepto que hace referencia la productividad que va tener la fabrica es el esfuerzo necesario para la construcción de sus productos, considerando los meses y personas necesarias para elaborara una determinada cantidad de LDC o PF.

IE_1= LDC/(Personas*Meses)

(8.6)

IE_2= PF/(Personas*Meses)

(8.7)

Indicador de coste. Basado en los gastos fijos de la fábrica mensualmente para mantener el desarrollo por LDC o PF para estimar cuanto me cuesta hacer una parte de mi producto.

IC_1= Gastos mensuales/LDC

(8.8)

IC_1= Gastos mensuales/PF

(8.9)

Indicador de tamaño y tiempo. Considerando datos de proyectos de la fábrica anteriores o basándose en datos aproximados, se puede determinar la cantidad de LDC por PF (ver tabla 8.2) con lo cual se estima el tamaño del producto contabilizando sus PF y multiplicándolo por las respectivas LDC/PF.

IT_1= (LDC/PF)*PF

(8.10)

LENGUAJES DE PROGRAMCIÓN	LDC/PF
Java	58
SQL	12
SmallTalk	22
Visual Basic	32
Ada95	53
C++	64
Pascal	90
FORTRAN	106
COBOL	106
C	128
Ensamblador	320

Tabla 8.2: Relación de LDC por PF según el lenguaje de programación

Luego, con el tamaño estimado y la productividad mensual de la fábrica calculada por la ecuación (8.6) se puede estimar el tiempo de construcción del producto.

IT_2= (LDC/PF)*PF*(LDC/(Personas*Meses)) (8.11)

8.2 Indicadores de los Procesos

Para efectuar una medición de los procesos propuestos para la fábrica, se elaboran indicadores para los siguientes procesos:

Proceso de aseguramiento de la calidad. La Eficacia de la Eliminación de Defectos (EED) es una medida de la habilidad de filtrar errores con las actividades de aseguramiento de calidad, al aplicarse a todas las actividades del marco de trabajo del desarrollo del producto.

EED = E/(E+D)
(8.12)

Donde E es número de errores encontrados antes de la entrega, D el número de defectos y el objetivo es que EED sea igual a 1.

Nótese que si E es muy grande, EED estará próxima a 1, cuanto más errores encontremos antes de la entrega, mejor funcionarán las técnicas de garantía de calidad

La EED también puede utilizarse para medir la habilidad de un equipo para encontrar errores antes de pasar a la siguiente etapa.

EEDi = Ei/(Ei+Ei+1)
(8.13)

Donde Ei son los errores detectados en la actividad i del CVPS y Ei+1 son los errores detectados en la actividad i+1 del CVPS que no se detectaron y provienen de la actividad i. El objetivo es que EEDi sea igual a 1.

Otro indicador para le proceso de aseguramiento de la calidad es el rendimiento de sus inspecciones para cada una de las etapas del CVPS.

R = Errores encontrados*100/(Errores encontrados+EN)
(8.14)

Donde EN son los escapes netos y R el rendimiento.

Se considera escapes netos a todos los defectos que se insertaron antes o durante una etapa determinada, pero que no fueron encontrados en esa etapa, sino en otra etapa posterior. Además se elabora una tabla con los siguientes datos:

Insertados: Aquellos errores que han sido insertados en una etapa.
Eliminados: Aquellos errores que han sido eliminados.
Acumulativo insertados: Es la suma de los errores insertados hasta una etapa determinada.
Acumulativos eliminados: Es la suma de los errores eliminados hasta una etapa determinada.

El rendimiento total del proceso se calcula como:

RTP = (AEEPT*100)/(AEEPT+EPT) (8.15)

Donde RTP es el rendimiento total del proceso, AEEPT es el acumulado de errores eliminados previo al testing y EPT son los escapes previo al testing.

Proceso de construcción. Al revisar las etapas de Análisis e Implementación de componentes e Integración y Pruebas, se puede ver como un solo proceso de construcción del producto, para el cual se pueden generar los siguientes indicadores:

Eficiencia de la construcción = (RR/RPR)*100 (8.16)

Eficacia de la construcción = (RE/RR)*100 (8.17)

Efectividad = (Eficiencia de la construcción*Eficacia de la construcción)/100 (8.18)

Eficiencia en el tiempo = (RTTP/RT)*100 (8.19)

Donde RR son el número de requisitos realizados, RPR número de requisitos por realizar, RTTP número de requisitos terminados en le tiempo planeado y RT es el número de requisitos totales.

Para determinar que tan terminado va el software se mide la complesión de la construcción.

$Complesión_1$ = CI/CAI*100 (8.20)

$Complesión_2$ = (CI*DE)/(CAI*DE) (8.21)

Fiabilidad de la construcción = (CC/CI)*100 (8.22)

Donde CI es el número de componentes implementados, CAI número de componentes a implementar, CC número de componentes correctos y DE son los días estimados.

Proceso de reutilización de componente. Para revisar los procesos de recuperación de componentes y su eventual reutilización en cantidad y en tiempo es que se elaboran estos indicadores:

Eficiencia de la reutilización = CR/CAI (8.23)

Eficiencia en el tiempo = (CTTP/CT)*100 (8.24)

Donde CR es número de componentes reutilizados, CTTP número e componentes terminados en el tiempo planeado y CT son el número de componentes totales.

8.3 Indicadores del Producto

Un punto importante para la Fábrica de Software es la expectativa de calidad del producto, más aun cuando la interacción con el cliente es baja debido a la estructura definida, en donde el gran contacto con el cliente es en el momento que el jefe de proyecto genera los requerimiento y son validados para la entrada a la fábrica, la cual mediante sus procesos internos genera la salida del producto. Por lo tanto surge la pregunta ¿cumple realmente, el producto entregado por la Fábrica de Software bajo los procesos definidos en capítulos anteriores (CVPS, administración de calidad y reutilización de componentes), con las exigencias efectuadas por el cliente en su inicio?

Al responder la pregunta anterior, de cierta forma se puede verificar si la fábrica asegura realmente la calidad del producto utilizando las siguientes métricas:

Métrica de calidad 1. Una métrica que es posible emplear para valorar cual es la calidad de los productos finales es la de la cantidad de defectos cada miles de líneas de código.

Métrica de calidad 2. Como una forma de medir la complejidad del sistema se utiliza la cantidad de clases (considerando un desarrollo de tipo orientado a objeto) que este posee. Por este motivo se considera conveniente utilizar dicho valor y definir la métrica de defectos por clases.

Para evaluar los defectos se definen dos tipos, aquellos que influyen en una funcionalidad en particular del prototipo (clase 1) y otros que ocasionan problemas en más de una funcionalidad (clase 2), siendo estos últimos multiplicados por el numero de funcionalidades a las que afectan. Para especificar estas métricas se definen las siguientes ecuaciones:

$$\mathbf{DKLC} = \frac{D + \sum_{i=1}^{n} Di}{KLC}$$

(8.25)

$$\mathbf{DC} = \frac{D + \sum_{i=1}^{n} Di}{C}$$

(8.26)

Donde KLC es la cantidad de 1000 líneas de código, C cantidad de clases del sistema. D cantidad de defectos de clase 1, n cantidad de defectos de clase 2, Di numero de funcionalidades afectadas por un defecto i de clase 2, DKLC cantidad de defectos cada 1000 líneas de código y DC cantidad de defectos por clases.

Otro punto importante al interior de la Fábrica de Software es la reutilización de componentes, gestionados por la biblioteca de componentes a través de los procesos especificados en el capítulo 6. La reutilización de componentes es importante a destacar en la fábrica, debido a que esta se especializa en un domino en particular (sistemas de control y gestión de operaciones en la minería), permitiendo un mejor manejo y conocimiento de los elementos que componen al producto final. Esto trae beneficios propios de un desarrollo con reutilización como es la mejora en la productividad evitando implementar y validar lo que ya se hizo en desarrollos anteriores.

Si bien la reutilización es un proceso que se puede llevara acabo desde el análisis hasta la construcción de un producto, para el diseño de Fábrica de Software de Exertus se rescata el proceso de reutilización únicamente en la etapa de implementación, o sea los componentes reutilizables serán básicamente códigos.

Métrica de reutilización: Para medir los porcentajes de reutilización de un producto se propone el siguiente indicador.

$$\mathbf{PR} = \frac{LR}{LDC} * \mathbf{100}$$

(8.27)

Donde PR es el porcentaje de reutilización y LR la cantidad total de líneas de código reutilizadas.

Capítulo 9

Análisis de Métricas para los Prototipos de Software

9.1 Selección de Métricas

Del conjunto de indicadores establecido en el capítulo anterior, se utilizará solo un subconjunto de ellos para efectos de análisis en los prototipos implementados.

Las mediciones que se efectuaran a los procesos durante la implementación de los prototipos son los siguientes:

Proceso de aseguramiento de la calidad: Rendimiento de las inspecciones total del proceso y por cada etapa.

Proceso de construcción: Eficiencia de la construcción, eficacia de la construcción y efectividad.

Proceso de reutilización de componente: Eficiencia de la reutilización.

Las mediciones que se efectuaran a los prototipos como producto son los siguientes:

Calidad: se aplicaran las métricas de calidad 1 y 2 (errores por KLC y clases) a prototipos de la línea de producto de Exertus al momento de testear el producto, luego se comparan los datos obtenidos de estas métricas en versiones iniciales de software elaborados bajo un desarrollo tradicional anteriormente en Exertus versus la aplicación de las métricas a los prototipos de software implementados bajo el esquema de desarrollo de fábrica definido.

Reutilización: Lo que se busca con la aplicación de la métrica de reutilización es clarificar los beneficios que trae la reutilización. Se implementan prototipos propios de la línea de productos de Exertus, que permitan poner en práctica los procesos de

rescate de componentes reutilizable y su posterior reutilización, analizando los distintos porcentajes de reutilización que cada prototipo posea y de que manera afecta al producto. Para esto se realiza la comparación tasa de error y porcentaje de reutilización, utilizando las métricas calidad y reutilización definidas previamente.

9.2 Análisis del Dominio

Los sistemas miembros de esta línea de producto son sistemas de gestión y control de operación enfocado en el dominio de la minería. Estos productos buscan recolectar y almacenar datos relacionados con elementos propios de las operaciones de la minería como por ejemplo el control y gestión de recursos materiales (camiones, neumáticos, maquinarias, etc.) y procesos propios de la actividad minera (procesos[13] de chancado de rocas, formación de la pila, sistemas de riego, etc.). Estos sistemas se encargan de recolectar los datos necesarios para el control, mediante los distintos actores involucrados, siendo almacenados en la base de datos del sistema para poder consultar a ella y tomar las decisiones que determinan acciones dentro de la línea de productividad del mineral. El sistema será implementado como una aplicación web, ya que por la descripción efectuada anteriormente se puede deducir la existencia de clientes en distintos sectores que acceden mediante interfaces web (servidor web) a un sistema en el cual ingresan datos y efectúan consultas a un servidor de base de datos para la posterior toma de decisiones.

Una vez clara la línea de producto se efectúa un desarrollo previo de componentes a utilizar por los futuros productos y de esta forma iniciar el funcionamiento de la biblioteca de componente junto con su sistema gestor de componentes (clasificación de componentes para su ingreso). A continuación se detallan algunos componentes detectados que serán de utilidad en la implementación de los prototipos a considerar en este proyecto.

---

[13] Para ver más detalles sobre los procesos de extracción del mineral dirigirse a la página: http://www.codelco.cl/educa/divisiones/norte/info/procesos3.html

9.2.1 Clases

- **Usuario.class:**

- Descripción: Clase que maneja el registro y acceso de usuarios al sistema.

- Atributos: int id; String nombre, contraseña y tipo;

- Métodos: create(), load(), verificarUsuario(String usuario, String clave), setid(int i), setnombre(String nombre), setcontrasena(String), settipo(String), getid(), getnombre(), getcontrasena() y gettipo().

- **Proveedor.class:**

- Descripción: Clase que maneja la información de los proveedores de un elemento a controlar y gestionar en el sistema.

- Atributos: int id, telefono; String nombre, direccion y mail;

- Métodos: create(), load(), setid(int i), setnombre(String n), settelefono(int t), setdireccion(String d), setmail(String m), getid(), getnombre(), gettelefono(), getdireccion() y getmail().

- **Motivo.class:**

- Descripción: Clase que maneja la parametrización de los motivos (reparaciones, cambios detenciones, etc.) de un elemento a control y gestionar en el sistema.

- Atributos: int id; String nombre;

- Métodos: create(), load(), setid(int i), setnombre(String n), getid() y getnombre().

- **Estado.class:**

- Descripción: Clase que maneja la parametrización de los estados de un elemento a controlar y gestionar en el sistema.

- Atributos: int id; String nombre;

- Métodos: create(), load(), setid(int i), setnombre(String n), getid() y getnombre().

- **Conectar.java:**

- Descripción: Clase que maneja la conexión con la base de datos y las sentencias SQL.

- Atributos: Connection conn; Statement stmt;.

- Métodos: conectar(), desconectar(), ejecutar(String sql) y select(String sql).

- **Registro.java:**

- Descripción: Clase que maneja operaciones de registros.

- Métodos: obtenerSgteId(String tabla), campoRepetido(String tabla,int col,String campo), obtenerCantidadReg(String tabla), obtenerCampoString(String tabla,String campo,String PK), obtenerCampoInt(String tabla,String campo,String PK) y obtener<<*nombre_tabla*>>()[14].

- **ValNum.java:**

- Descripción: Clase que maneja operaciones de validaciones de números.

- Métodos: obtenerEntero(String entero).

- **Fecha.java:**

- Descripción: Clase que maneja operaciones relacionadas con la fecha.

- Métodos: obtenerFechaActual() y sumarFecha(String fecha,int meses).

---

[14] Obtiene todos los registros de la tabla especificada en <<*nombre_tabla*>> devolviendo una arreglo de registros, útil para efectuar los reportes.

9.2.2 Componentes JavaScript

Lafecha.js: Muestra en pantalla la fecha actual, nombre del día y numero, nombre del mes y año

Menu.js: Permite crear menús de botones gráficos.

9.2.3 Hojas de Estilo

EstiloInicio.css: Especifica las interfazes gráficas correspondiente al inicio del sistema (colores de fondo, colores de letra, tamaño de letra, especio entre letras, descripción de tablas, descripción de formularios, textbox, botones, posiciones, etc.).

EstiloAdmin.css: Especifica las interfazes gráficas correspondiente al administrador del sistema (colores de fondo, colores de letra, tamaño de letra, especio entre letras, descripción de tablas, descripción de formularios, textbox, botones, posiciones, etc.).

EstiloIngresador.css: Especifica las interfazes gráficas correspondiente al usuario ingresador del sistema (colores de fondo, colores de letra, tamaño de letra, especio entre letras, descripción de tablas, descripción de formularios, textbox, botones, posiciones, etc.).

EstiloConsultor.css: Especifica las interfazes gráficas correspondiente al usuario consultor del sistema (colores de fondo, colores de letra, tamaño de letra, especio entre letras, descripción de tablas, descripción de formularios, textbox, botones, posiciones, etc.).

9.2.4 Componentes Gráficos

index.htm: Pagina web HTML de ingreso al sistema.

downIndex.jsp: Pagina web en java para ingresar al sistema.

topIndex.jsp: Pagina web en java para ingresar al sistema.

Botones.png: Botones parametrizables según su texto e imagen.

Banner.png: Imagen del sector superior de todas las interfazes gráficas parametrizable según el texto e imágenes.

Alerta.jpg: Imagen que aparece junto a cada mensaje de error del sistema.

Lineatitulo.jpg: Imagen de subrayado de títulos de interfazes graficas del sistema.

9.2.5 Plantilla de Prototipos

Como fue señalado en el punto 3.4 se crea un Esquema de Fábrica de Software el cual resume y clasifica los distintos artefactos de desarrollo pertenecientes a la construcción de un miembro de la Línea de Productos, que es instanciado al momento de construir un producto, originando la Plantilla de Fábrica de Software.

Para esta plantilla, que se muestra en la tabla 9.1 se destaca la arquitectura de la aplicación y de tecnología con sus tres Vistas (conceptual, lógica y física).

		ARQUITECTURAS	
		APLICACIÓN	**TECNOLOGÍA[15]**
VISTAS	**CONCEPTUAL**	• CVPS basado en Case*Method[16] • Línea de Productos de Exertus • Especificaciones del manejo del repositorio de componente[17] • Aseguramiento de la calidad basado en Métrica 3[18]	
	LÓGICA	• Especificación de requerimientos funcionales y no funcionales. • Diagramas de caso de uso. • Diagramas de secuencia. • Diagramas de colaboración. • Diagrama de clase de diseño.	• Eclipse SDK 3.1 • Dreamweaver MX 2004 • Java ™ 2 Runtime Enviroment, Estándar Edition 1.5 • Racional Rose Enterprise Edition • MySQL Server 5.0 • Fireworks MX 2004 • Windows XP.
	FÍSICA	• Clases. • Componentes JavaScript. • Componentes gráficos. • Hojas de estilo.	

Tabla 9.1: Plantilla de Fábrica de Software para los prototipos a implementar

9.3 Sistema de Control de Neumáticos (SCN)

SCN es un sistema que permite el control de los neumáticos utilizados por las distintas maquinas al interior de las empresas mineras, facilitando el monitoreo, registro y optimización del recurso. A continuación se detalla el progreso de la construcción de este prototipo, respecto a las etapas de construcción del producto y a las tareas de administración de calidad ya definidas.

9.3.1 Estrategia

- **Requerimientos funcionales:**

- Administrador:

1. El sistema debe permitir registrar usuarios para el sistema
2. El sistema debe permitir parametrizarlo con los nombre de proveedores, tipos de equipos, motivos de cambios, motivos de reparaciones y estado del neumático.
3. El sistema debe permitir registrar el ingreso de la compra de neumáticos.

- Usuarios ingresador:

1. El sistema debe permitir el registro de datos respecto a chequeos (estado en el que se encuentra) y posibles reparaciones del neumático.
2. Debe permitir registrar las bajas de neumáticos y su motivo.
3. Debe permitir el registro de la asignación de neumáticos a determinadas maquinas.

- Consultores:

1. Debe permitir conocer los datos del proveedor de un determinado neumático.
2. Debe permitir conocer los datos del equipo que usa un determinado neumático.
3. Entregar un informe de las bajas mensuales de neumáticos.
4. El sistema debe permitir saber el tiempo de vida restante para el neumático.
5. El sistema debe permitir ver la información respecto a los chequeos efectuados y

posibles reparaciones (último chequeo, próximo chequeo y motivos de reparaciones o de cambio) de un neumático.

- **Requerimientos no funcionales:**

1. La forma visible en que se presenten los datos a los usuarios del sistema debe ser clara, precisa y fácil de usar, esto se hace extensible al uso general de todas y cada una de las partes del sistema.

2. El sistema debe ser seguro, asegurando que ingresen solo aquellos usuarios que tengan los permisos para hacerlo, accediendo a información que le corresponda según el usuario.

3. El sistema debe encontrarse en un servidor y ser de tipo aplicación web permitiendo su acceso remoto a este.

4. El sistema debe permitir un control eficiente de los neumáticos que se utilizan en las maquinas utilizadas en la mina.

- **Información funcional:**

- A todo neumático se le consideran 3 años de vida.

- Al ser recibidos los lotes de neumáticos por parte del proveedor, hacen su ingreso en bodega quedando registrado en el sistema, el cual establece por defecto 6 meses después de la fecha de ingreso como su fecha de próximo chequeo al neumático.

- Cuando se efectúa una reparación a un neumático, se debe posteriormente chequearlo.

- **Administración de calidad:**

- Identificación de las propiedades de calidad para el sistema: Los sectores de mayor prioridad a controlar en el prototipo serán los tres módulos funcionales, administración, ingreso de información y consulta, basándose en la eficiencia de estos.

9.3.2 Análisis y Definición de Requerimientos

- Diagramas de casos de uso:

Forma grafica de alto nivel: SCN.

Figura 9.1: Forma grafica de alto nivel del SCN

Casos de uso: Sistema de administración.

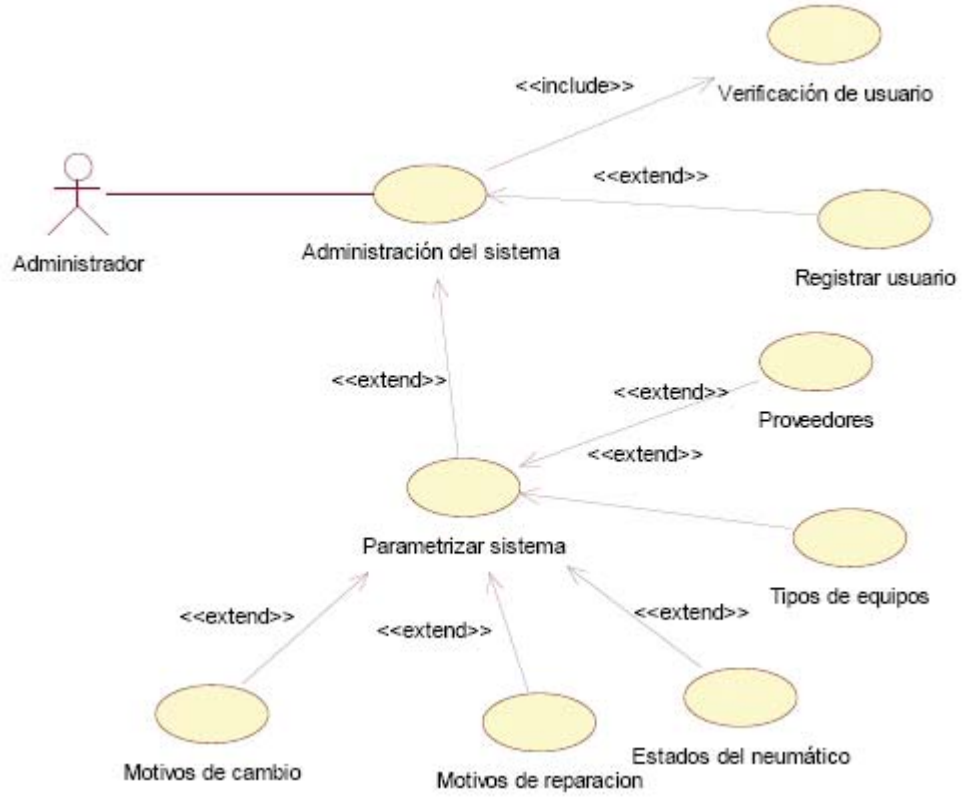


Figura 9.2: Casos de uso de sistema de administración

Casos de uso: Sistema de registro y control.

Figura 9.3: Casos de uso de sistema de registro y control (administrador)

Casos de uso: Sistema de registro y control.

Figura 9.4: Casos de uso de sistema de registro y control (consultor)

Casos de uso: Sistema de registro y control.

Figura 9.5: Casos de uso de sistema de registro y control (usuario ingresador)

- **Diagramas de secuencia y colaboración:**

Caso de uso: Sistema de administración.
Escenario: El administrador ingresa el nombre de un equipo satisfactoriamente.

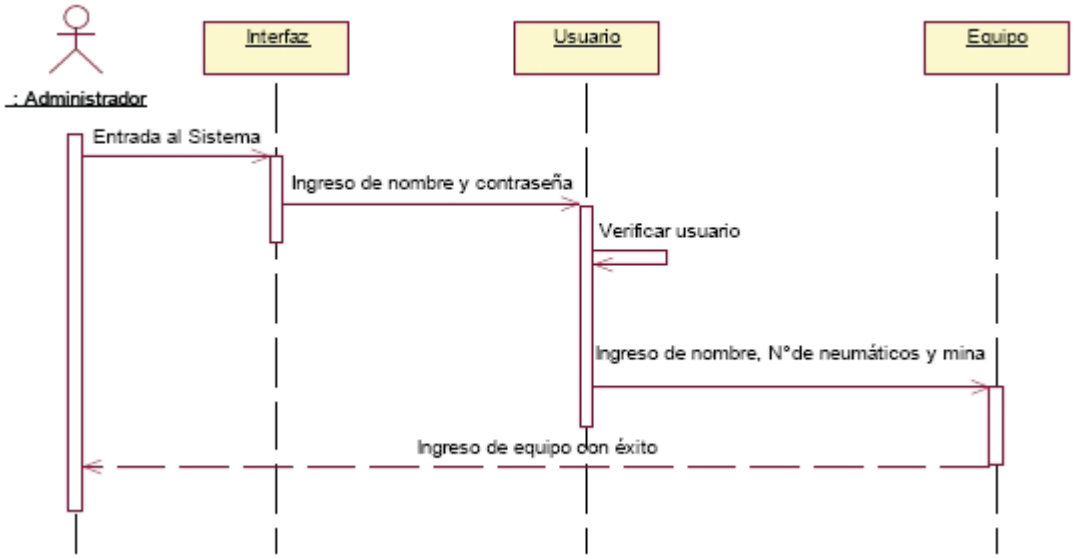


Figura 9.6: El administrador ingresa el nombre de un equipo satisfactoriamente (diagrama de secuencia)

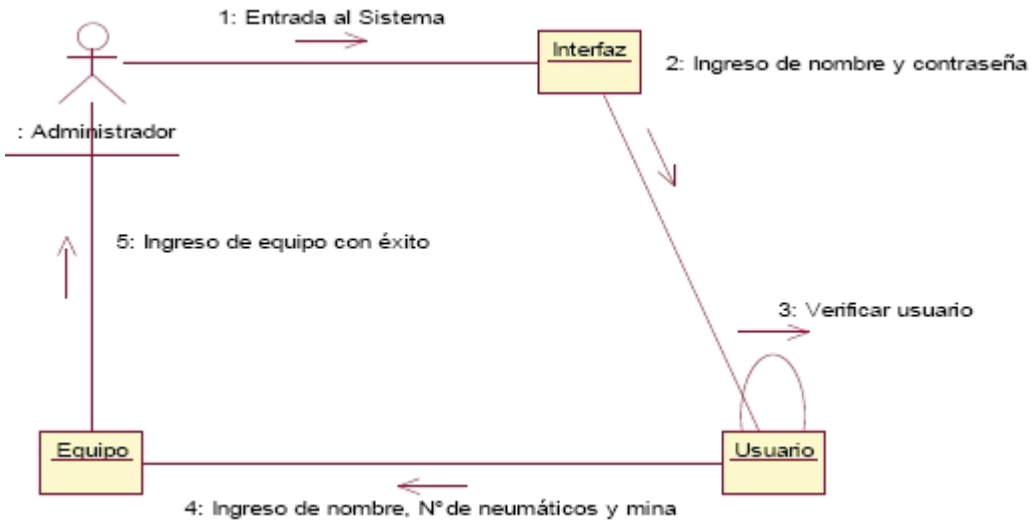


Figura 9.7: El administrador ingresa el nombre de un equipo satisfactoriamente (diagrama de colaboración)

Caso de uso: Sistema de registro y control.
Escenario: El administrador ingresa un lote de neumáticos satisfactoriamente.

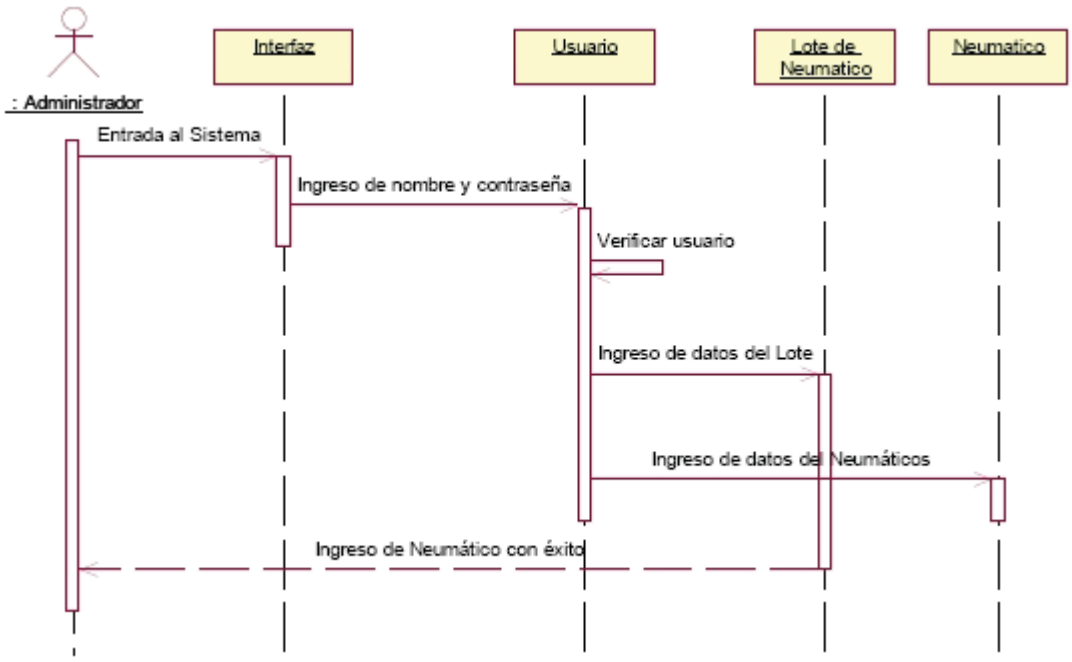


Figura 9.8: El administrador ingresa un lote de neumáticos satisfactoriamente (diagrama de secuencia)

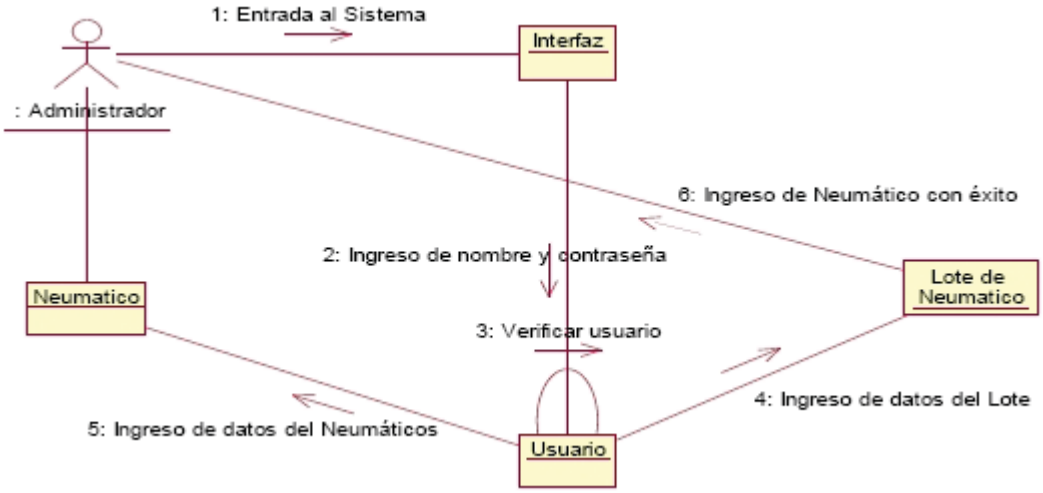


Figura 9.9: El administrador ingresa un lote de neumáticos satisfactoriamente (diagrama de colaboración)

Caso de uso: Sistema de registro y control.
Escenario: El usuario ingresador registra un chequeo efectuado a un neumático satisfactoriamente.

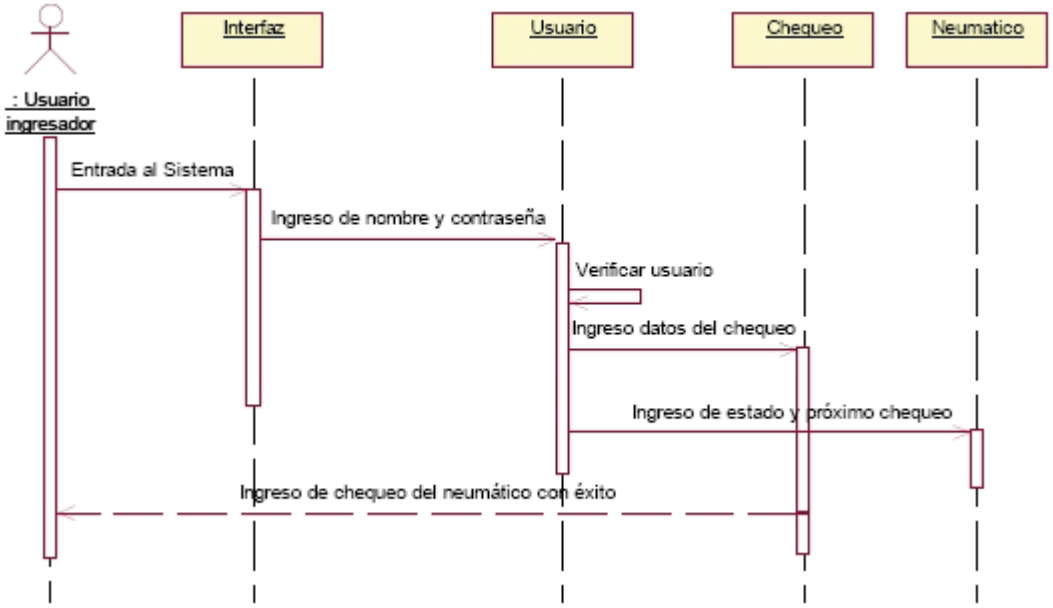


Figura 9.10: El usuario ingresador registra un chequeo efectuado a un neumático satisfactoriamente (diagrama de secuencia)

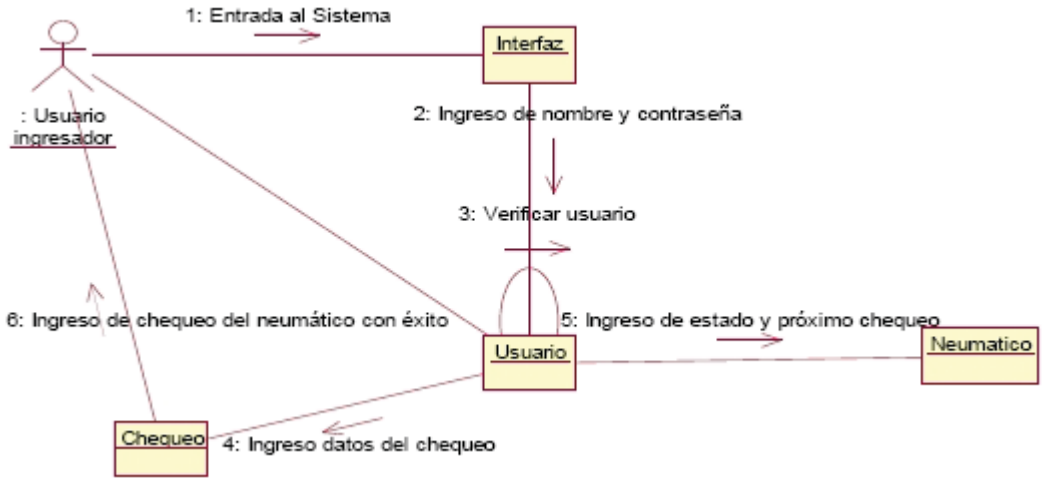


Figura 9.11: El usuario ingresador registra un chequeo efectuado a un neumático satisfactoriamente (diagrama de colaboración)

Caso de uso: Sistema de registro y control.
Escenario: El consultor busca los datos del proveedor de un neumático satisfactoriamente.

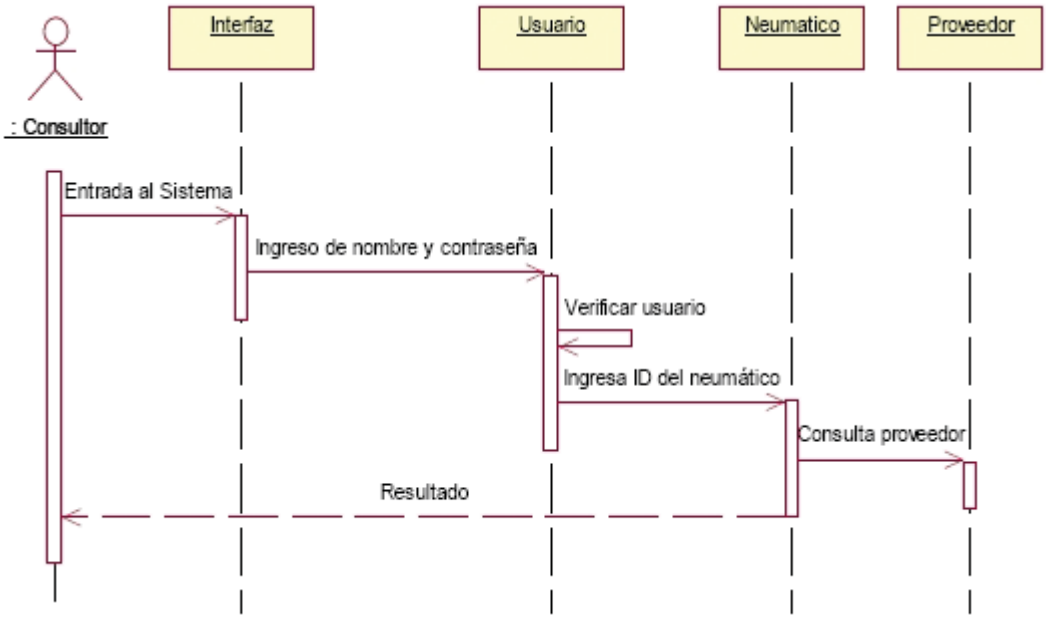


Figura 9.12: El consultor busca los datos del proveedor de un neumático satisfactoriamente (diagrama de secuencia)

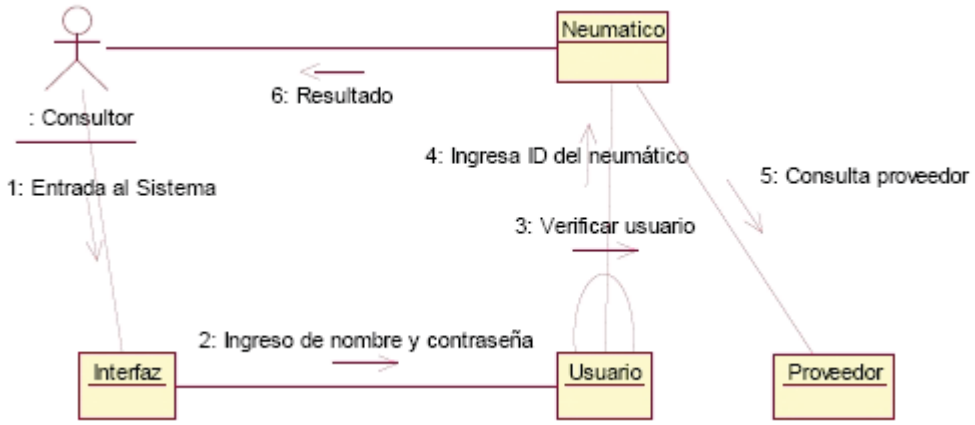


Figura 9.13: El consultor busca los datos del proveedor de un neumático satisfactoriamente (diagrama de colaboración)

- **Administración de calidad:**

Revisión del análisis de consistencia: La revisión comparativa de los requisitos generados en la etapa de estrategia con respecto a los casos de uso, muestra la ausencia de la funcionalidad de asignación de neumáticos a determinadas maquinas, generando un nuevo *extend* para el caso de uso *Sistema de registro y control* con respecto al actor *Usuario ingresador*.

Figura 9.14: Casos de uso de sistema de registro y control (corregido)

9.3.3 Diseño de Sistema y de Software

- **Diagrama de clase de diseño:**

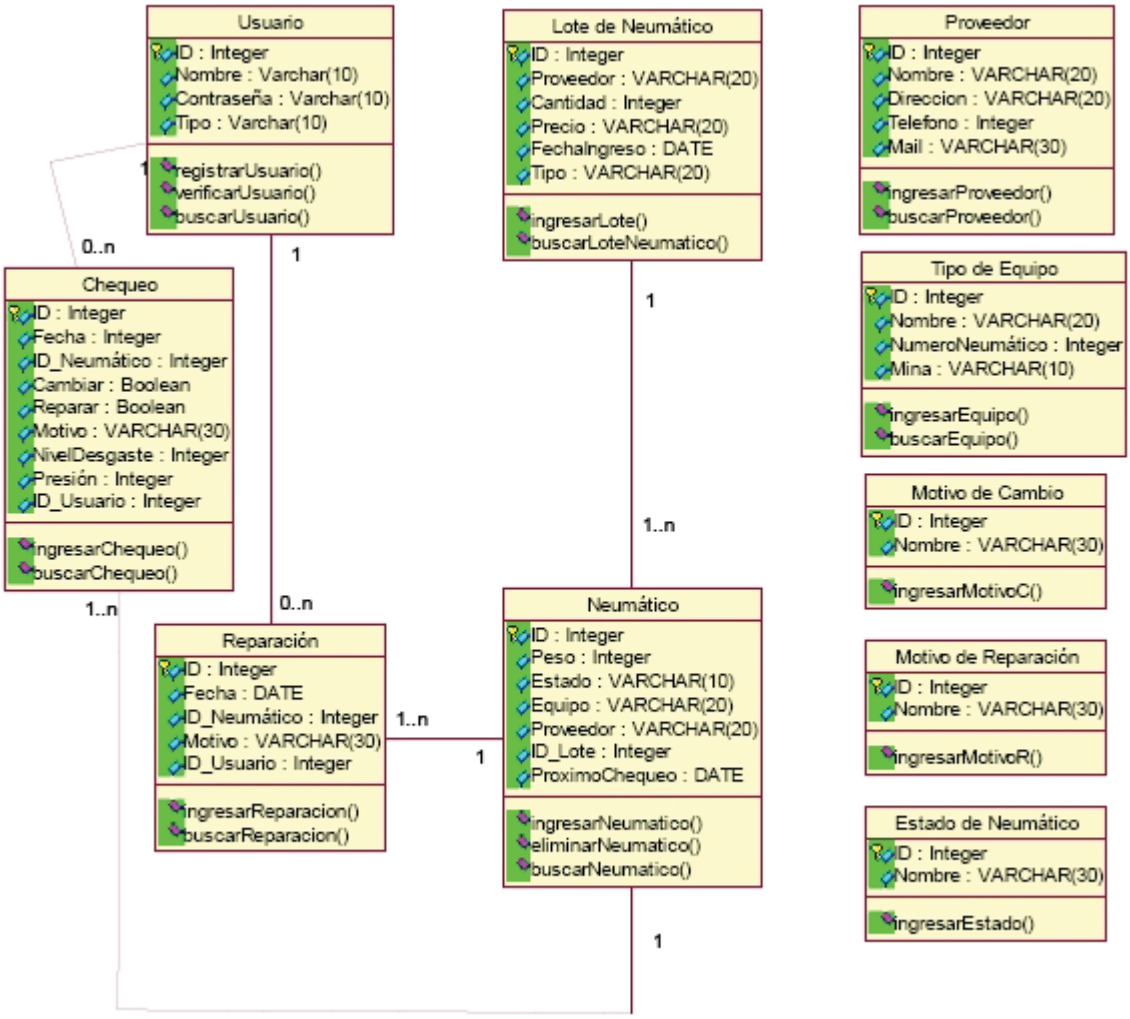


Figura 9.15: Diagrama de clase de diseño de SCN

- **Pruebas de integración:**

- Acceder al sistema con los tres tipos de usuarios y usuarios incorrectos

- Ingresar proveedor para los siguientes casos: Sin nombre - Sin mail - Sin teléfono - Teléfono con caracteres - Todos los campos llenos correctamente.

- Ingresar equipo para los siguientes casos: Todos los campos llenos correctamente - Sin nombre - Sin mina - Sin número de neumáticos - Numero de neumático con caracteres.

- Ingresar motivo de cambio y reparación para los siguientes casos: Sin nombre - Con nombre

- Ingresar lote de neumático para los siguientes casos: Sin cantidad de neumáticos - Sin peso de cada neumático - Sin precio del lote - Sin tipo de neumático - Cantidad de neumático con caracteres - Peso con caracteres - Precio con caracteres - Todos los campos llenos correctamente.

- Ingresar estado para los siguientes casos: Sin nombre - Con nombre.

- Ingresar usuario para los siguientes casos: Sin nombre - Sin contraseña - Sin confirmación de contraseña - Contraseña con confirmación de contraseña diferentes - Todos los campos llenos correctamente.

- Ingresar un chequeo para los siguientes casos: Sin seleccionar neumático - Sin nivel de desgaste - Sin ingresar la presión - Seleccionando más de un neumático - Seleccionando un neumático - Llenando los campos de manera correctamente.

- Ingresar una reparación para los siguientes casos: Sin seleccionar neumático - Seleccionando más de un neumático - Seleccionando un neumático con llenando los campos de manera correctamente.

- Asignar neumáticos a un equipo para los siguientes casos: Sin seleccionar neumático - Seleccionando más de un neumático - Seleccionando un neumático con llenando los campos de manera correctamente.

- Ejecutar dar de baja para los siguientes casos: Seleccionando uno o más neumáticos - Sin seleccionar un neumático.

- Consultar por el origen de un neumático

- Consultar por los chequeos de un neumático

- Consultar por las reparaciones de un neumático

- Consultar por las bajas de neumáticos en un mes

- Consultar por la ubicación de un neumático

- **Administración de calidad:**

- Revisión de la verificación de la arquitectura del sistema:
1. Relación de las tablas *Chequeo* y *Reparacion* con la tabla *Usuario* innecesaria para el alcance del prototipo, no es necesario llevar un control de los chequeos o reparaciones efectuadas según el usuario, por lo tanto el atributo *ID_Usuario* como clave foránea se elimina de las respectivas tablas junto con su relación.
2. El atributo precio en la tabla *Lote de Neumatico* debe ser de tipo *Integer* para eventuales operaciones matemáticas a las cuales pueda estar expuesto.
3. El atributo *Fecha* en la tabla de *Chequeo* de ser de tipo *DATE*
4. Nombre de la tabla Tipo de Equipo poco apropiado.

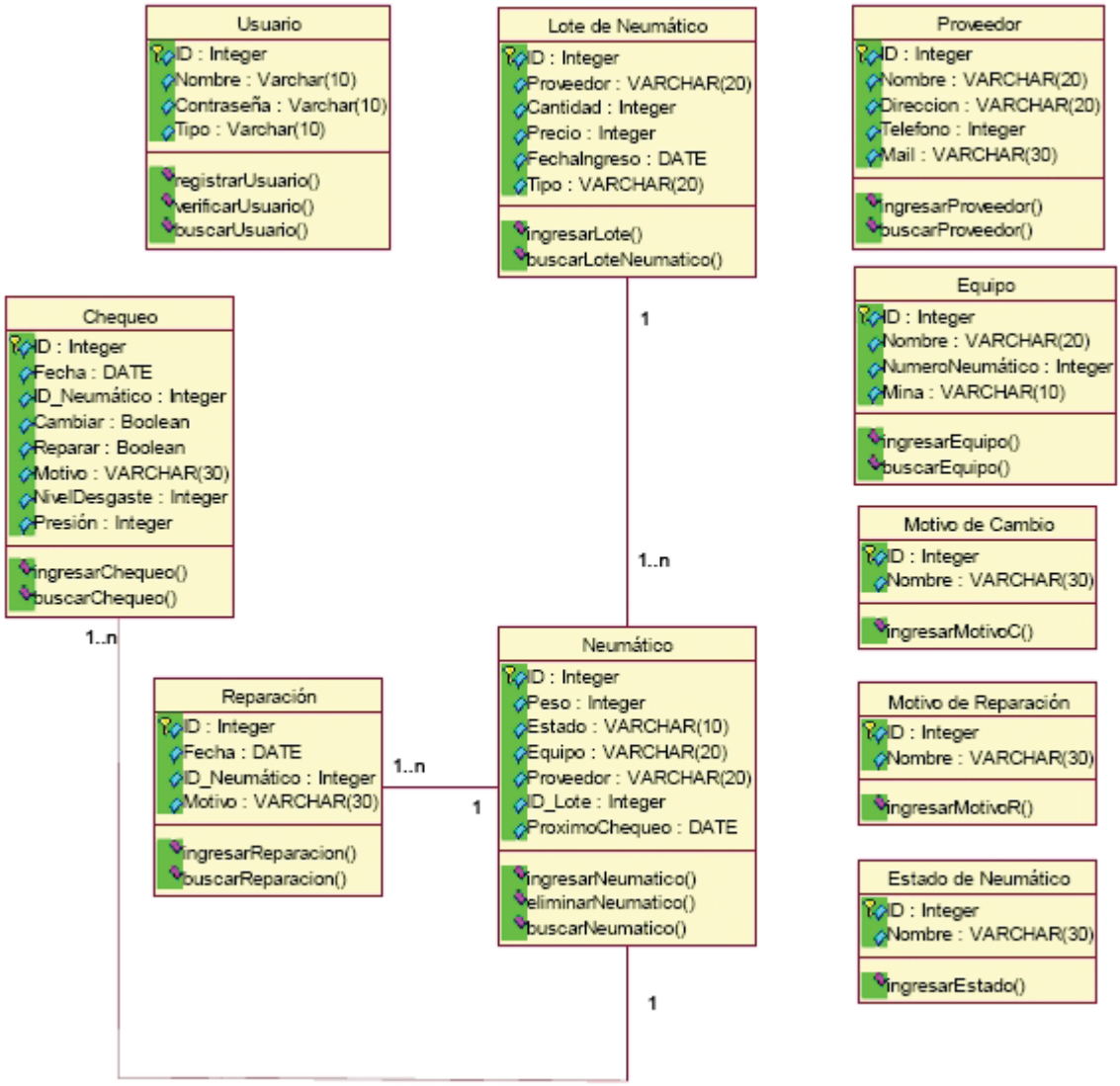


Figura 9.16: Diagrama de Clase de Diseño (corregido)

9.3.4 Análisis e Implementación de Componentes

- **Definición de incremento:**

- Incremento 1: Cumple con todos aquellos requerimientos necesarios para el usuario Administrador, junto con las tarea de ingreso al sistema.

- Incremento 2: Cumple con todos aquellos requerimientos necesarios para el usuario Ingresador.

- Incremento 3: Cumple con todos aquellos requerimientos necesarios para el usuario Consultor.

- **Iteración 1:**

- Componentes reutilizados: Usuario.class, Proveedor.class, Motivo.class (Cambio y Reparación), Estado.class, Conectar.java, Registro.java, ValNum.java, Fecha.java, Lafecha.js, Menu.js, EstiloInicio.css, EstiloAdmin.css, index.htm, downIndex.jsp, topIndex.jsp, Botones.png, Banner.png, Alerta.jpg y Lineatitulo.jpg.

- Componentes desarrollados: Equipo.java, Neumatico.java, LoteNeumatico.java, leftAdmin.htm, rightAdmin.htm, administrador.htm, IngMotivo.jsp, IngEstado.jsp, IngProveedor.jsp, IngEquipo.jsp, IngUsuario.jsp y IngNeumatico.jsp.

- Componentes con potencial de reutilización: Neumatico.java, LoteNeumatico.java, IngProveedor.jsp, IngMotivo.jsp, IngEstado.jsp, IngUsuario.jsp, leftAdmin.htm, rightAdmin.htm y administrador.htm.

- **Iteración 2:**

- Componentes reutilizados: EstiloIngresador.css.

- Componentes desarrollados: Chequeo.java, Reparacion.java, ingresador.htm, rightIngre.htm, leftIngre.htm, IngChequeo.jsp, IngAsign.jsp, IngBaja.jsp y IngReparacion.jsp.

- Componentes con potencial de reutilización: Chequeo.java, Reparacion.java, rightIngre.htm, leftIngre.htm y ingresador.htm

- **Iteración 3:**

- Componentes reutilizados: EstiloConsultor.css.

- Componentes desarrollados: NeumaticoBaja.java, ConBaja.jsp, ConChequeo.jsp, ConProveedor.jsp, ConReparación.jsp, ConUbicacion.jsp, leftConsu.htm, rightConsu.htm y consultor.htm.

- Componentes con potencial de reutilización: NeumaticoBaja.java, leftConsu.htm, rightConsu.htm y consultor.htm.

9.3.5 Integración y Prueba

- **Errores detectados en la iteración 1:**

1. No esta parametrizado el tipo de usuario, por lo tanto se pueden ingresar usuarios que no tengan acceso a nada.

2. Una vez que se cierra una sesión, el siguiente usuario puede acceder a funcionalidades de otros usuarios haciendo "atrás" en el navegador.

3. El tipo de neumático no se encuentra parametrizado, permitiendo el ingreso de cualquier dato como "tipo de neumático"

- **Errores detectados en la iteración 2:**

1. Los chequeos cuyo resultado es reparar pueden registrarse con motivos de cambio y las reparaciones pueden quedar registradas con motivos de cambio (Di=2).

2. La opción dar de baja para los usuario ingresador es redundante, debido que al ingresar un chequeo con resolución "cambiar neumático" el neumático queda eliminado (no puede darse de baja sin un previo chequeo).

3. El ingreso de reparaciones a un neumático solo debe permitirse a aquellos neumáticos que cuyo estado sea "reparar"

4. La asignación de un neumático aun equipo debe permitirse solo aquellos neumáticos cuyo estado no sea el de "reparar".

5. El usuario ingresador desconoce el último chequeo del neumático al ingresar un chequeo.

- **Errores detectados en la iteración 3:**

1. Las bajas mensuales no se muestran, debido a que son eliminados del sistema aquellos neumáticos eliminados y no se efectúa un almacenamiento a modo de histórico.

2. La consulta ubicación del neumático tira error si el neumático no ha sido asignado a algún equipo.

9.3.6 Mediciones de Procesos y Producto

Proceso de aseguramiento de la calidad: Se calcula el rendimiento de las inspecciones para cada etapa como muestra la tabla 9.2.

	Insertados	Eliminados	Acumulativos eliminados	Acumulativos insertados	Escapes netos	Rendimiento
Estrategia	2	0	0	2	2	
Inspección de estrategia	0	0	0	2	2	0%
Análisis y definición de requerimientos	2	0	0	4	4	
Inspección de análisis	0	1	1	4	3	25%
Diseño de sistema y de software	6	0	1	10	9	
Inspección de diseño	0	4	5	10	5	44,44%

Análisis e implementación de componentes	0	0	5	10	5	
Inspección de análisis e implementación de componentes	0	0	5	10	5	0%
Integración y pruebas	6	0	5	16	11	
Inspección de integración y pruebas	0	11	16	16	0	100%

Tabla 9.2: Rendimiento de las inspecciones para cada etapa del SCN

Rendimiento total del proceso = (5*100)/(5+11)= 31,25%

Proceso de construcción: Se calcula la eficiencia de la construcción, eficacia de la construcción y efectividad, para esto se analizan los requisitos del prototipo como muestra la tabla 9.3. Para efectuar el análisis de este proceso se considera la construcción del prototipo sin los efectos de corrección por las respectivas pruebas efectuadas para cada incremento.

REQUISITOS	POR REALIZAR	REALIZADOS	REALIZADOS CORRECTOS
ADMINISTRADOR	3	3	2
USUARIO INGRESADOR	3	2	0
CONSULTORES	5	4	3

NO FUNCIONALES	4	3	3
TOTAL	15	12	8

Tabla 9.3: Análisis de requisitos del SCN

Eficiencia de la construcción = (12/15)*100 = 80%.

Eficacia de la construcción = (8/12)*100 = 66,66%.

Efectividad = (80*66,66)/100 = 53.32%.

Proceso de reutilización de componente: Se calcula la eficiencia de la reutilización, para esto se analiza la reutilización de los componentes, como muestra la tabla 9.4, para las diferentes iteraciones.

	ITERACIÓN 1	**ITERACIÓN 2**	**ITERACIÓN 3**
Componentes a implementar	32	10	10
Componentes reutilizados	20	1	1
Eficiencia por iteración	62,5%	10%	10%
Eficiencia total			42,3%

Tabla 9.4: Eficiencia de la reutilización por iteración del SCN

Para llevar acabo la aplicación de las métricas al producto se debe efectuar la suma de las líneas de código y de las clases generadas previos a las pruebas del prototipo, como muestra la tabla 9.5 (suma de líneas de código).

ARCHIVOS FUENTES	LDC	ARCHIVOS FUENTES	LDC	ARCHIVOS FUENTES	LDC
ConBaja.jsp	53	topIndex.jsp	20	consultor.htm	12
ConChequeo.jsp	113	Chequeo.java	93	index.htm	11
ConProveedor.jsp	95	Conectar.java	50	ingresador.htm	11
ConReparación.jsp	87	Equipo.java	57	leftAdmin.htm	20
ConUbicacion.jsp	85	Estado.java	39	leftConsu.htm	23
downIndex.jsp	111	Fecha.java	36	leftIngre.htm	18
IngAsign.jsp	125	LoteNeumatico.java	75	rightAdmin.htm	23
IngBaja.jsp	85	MotivoCambio.java	39	rightConsu.htm	23
IngChequeo.jsp	211	MotivoReparacion.java	39	rightIngre.htm	23
IngEquipo.jsp	103	Neumatico.java	96	EstiloAdmin.css	89
IngEstado.jsp	73	Proveedor.java	66	EstiloConsultor.css	99
IngMotivo.jsp	108	Registro.java	139	EstiloIngresador.css	89
IngNeumatico.jsp	152	Reparacion.java	57	EstiloInicio.css	56
IngProveedor.jsp	126	Usuario.java	75	Lafecha.js	31

IngReparacion.jsp	122	ValNum.java	17	Menu.js	86
IngUsuario.jsp	127	administrador.htm	11	Líneas de código SQL	11
Total de LDC = 3310					

Tabla 9.5: Numero de líneas de código por archivo fuente del SCN

- Métrica de calidad 1:

DKCL = 11/3,310= 3.32

- Métrica de calidad 2:

Numero total de clases es 31, DC= 11/31=0,35

- Métrica de reutilización: Para la aplicación de la métrica de reutilización se calcula el total de LCD reutilizadas como muestra la tabla 9.6.

ARCHIVOS FUENTES	LDC	ARCHIVOS FUENTES	LDC	ARCHIVOS FUENTES	LDC
Conectar.java	50	Registro.java	139	EstiloInicio.css	56
Estado.java	39	Usuario.java	75	downIndex.jsp	111
Fecha.java	36	ValNum.java	17	topIndex.jsp	20
MotivoCambio.java	39	EstiloAdmin.css	89	Menu.js	86
MotivoReparacion.java	39	EstiloConsultor.css	99	Lafecha.js	31
Proveedor.java	66	EstiloIngresador.c	89	index.htm	11

		ss			
Total de LDC reutilizadas = 1092					

Tabla 9.6: Numero de líneas de código reutilizadas en el SCN

PR = (1092/3310)*100=32,99%

9.4 Sistema de Control de Máquinas y Equipos (SCME)

SCME es un sistema que permite el control de las maquinas y equipos utilizadas por las distintas minas al interior de las empresas mineras, facilitando el monitoreo, registro y optimización del recurso. A continuación se detalla el progreso de la construcción de este prototipo, respecto a las etapas de construcción del producto ya definido y políticas de administración de calidad.

9.4.1 Estrategia

- **Requerimientos funcionales:**

- Administrador:

1. El sistema debe permitir registrar usuarios para el sistema.
2. El sistema debe permitir parametrizarlo con los nombre de proveedores, equipos y maquinas, piezas, motivos y estados.
3. El sistema debe permitir registrar el ingreso de nuevas maquinarias y equipos.

- Usuarios ingresador:

1. El sistema debe permitir el registro de datos respecto a chequeos, estado en el que se encuentra, niveles de agua, niveles de aceite y engrase.
2. Debe permitir el control de cambio de piezas de los equipos o maquinas.
3. Debe permitir registrar las detenciones o fallas de los equipos o maquinas.
4. Debe permitir el registro de la asignación y uso de un equipo o maquina.

- Consultores:

1. Debe permitir detectar las detenciones no resueltas.

2. Debe permitir conocer los chequeos históricos de un equipo o maquina.
3. Debe permitir conocer los cambios de pieza históricos de un equipo o maquina.
4. Entregar un informe de los operadores que han utilizado una determinada maquina.
5. El sistema debe permitir saber el próximo chequeo de un equipo o maquina.

- **Requerimientos no funcionales:**

1. La forma visible en que se presenten los datos a los usuarios del sistema debe ser clara, precisa y fácil de usar, esto se hace extensible al uso general de todas y cada una de las partes del sistema.

2. El sistema debe ser seguro, asegurando que ingresen solo aquellos usuarios que tengan los permisos para hacerlo, accediendo a información que el corresponda según el usuario.

3. El sistema debe encontrarse en un servidor y ser de tipo aplicación web permitiendo su acceso remoto a este.

4. El sistema debe permitir un control eficiente de las maquinas y equipos que se utilizan en las minas.

- **Información funcional:**

- La parametrización de los estados se efectúa considerando los posibles estados en los que se encuentran las maquinas o equipos al momento de ingresar a la minera posterior a su compra.

- Los chequeos de maquinas o equipo se deben efectuar después de cada cambio de pieza o detención de la maquina o equipo, o en caso que el equipo de manutención lo estime conveniente.

- La máquina o equipo no debe exceder la cantidad de horas de uso establecidas como máximas previo a un chequeo.

- **Administración de calidad:**

- Identificación de las propiedades de calidad para el sistema: Los sectores de mayor prioridad a controlar en el prototipo serán los tres módulos funcionales, administración, ingreso de información y consulta, basándose en la eficiencia de estos.

9.4.2 Análisis y Definición de Requerimientos

- **Diagramas de casos de uso:**

Forma grafica de alto nivel: SCME.

Figura 9.17: Forma grafica de alto nivel del SCME

Caso de uso: Sistema de administración.

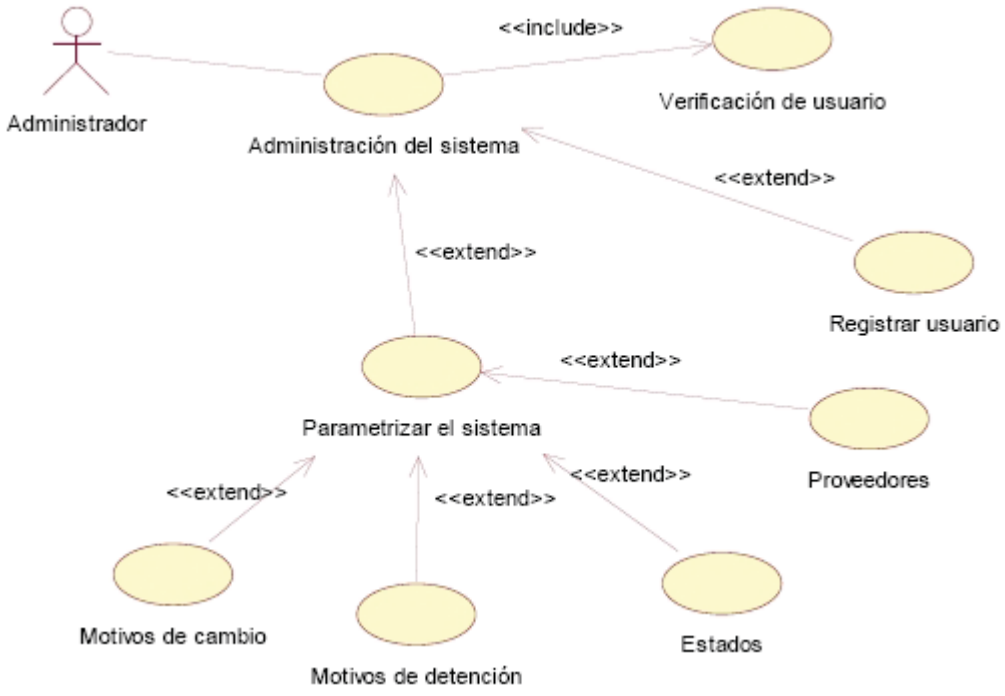


Figura 9.18: Caso de uso de sistema de administración

Caso de uso: Sistema de registro y control.

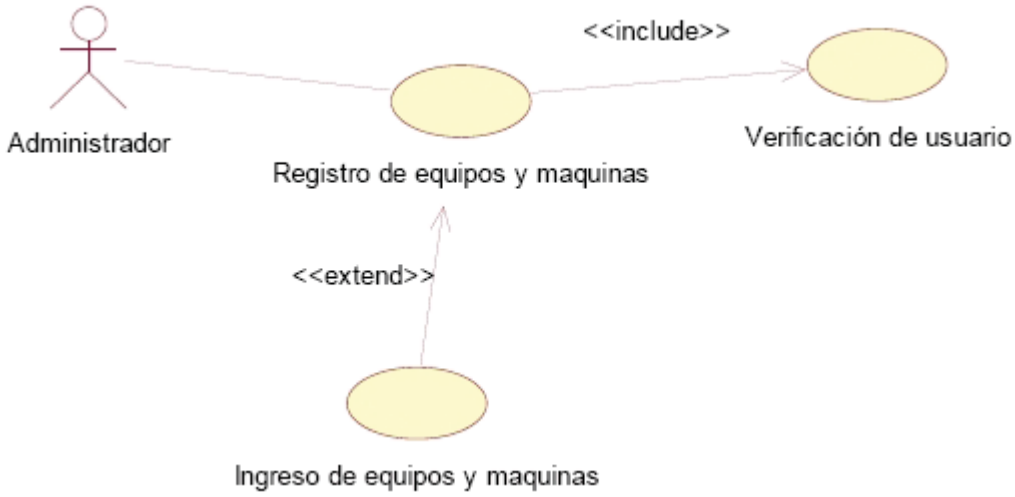


Figura 9.19: Caso de uso de sistema de registro y control (administrador)

Caso de uso: Sistema de registro y control.

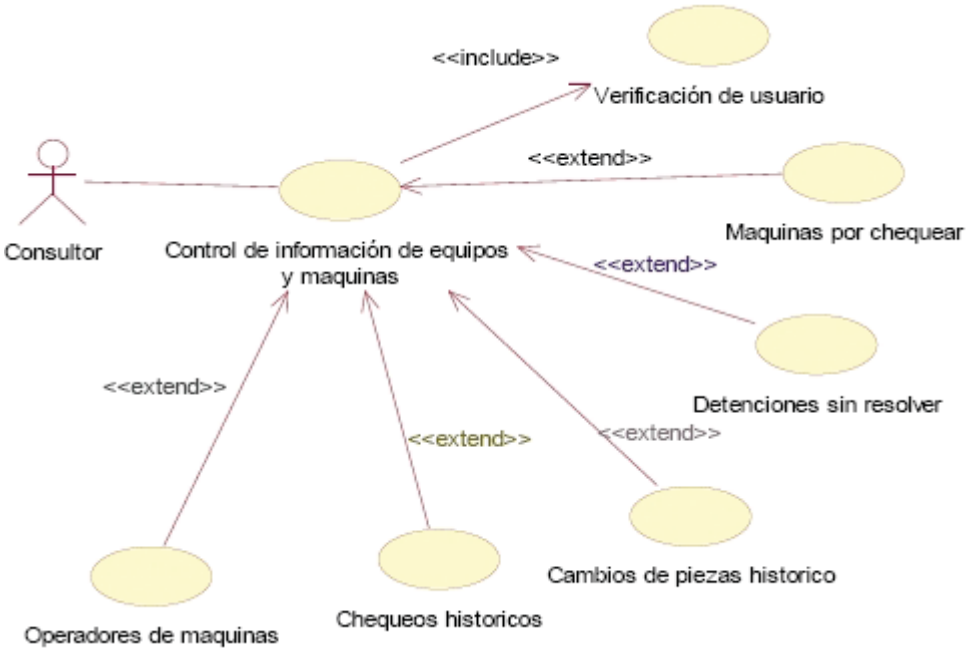


Figura 9.20: Caso de uso de sistema de registro y control (consultor)

Caso de uso: Sistema de registro y control.

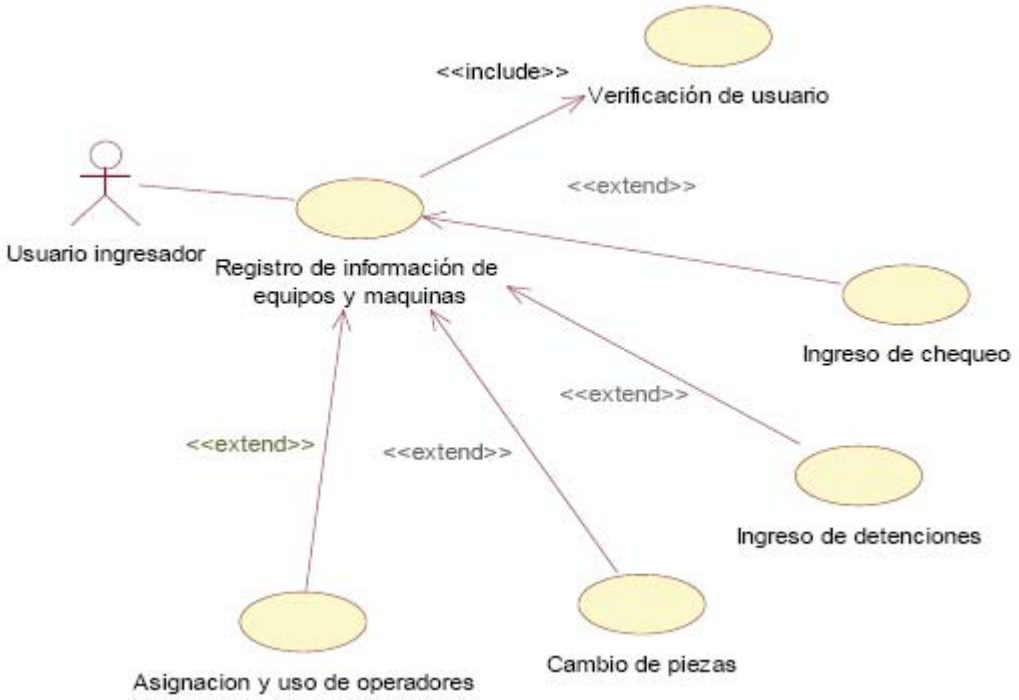


Figura 9.21: Caso de uso de sistema de registro y control (usuario ingresador)

- **Diagramas de secuencia y colaboración:**

Caso de uso: Sistema de administración.
Escenario: El administrador parametriza al sistema con un nombre de motivo de detención de una máquina o equipos de manera exitosa.

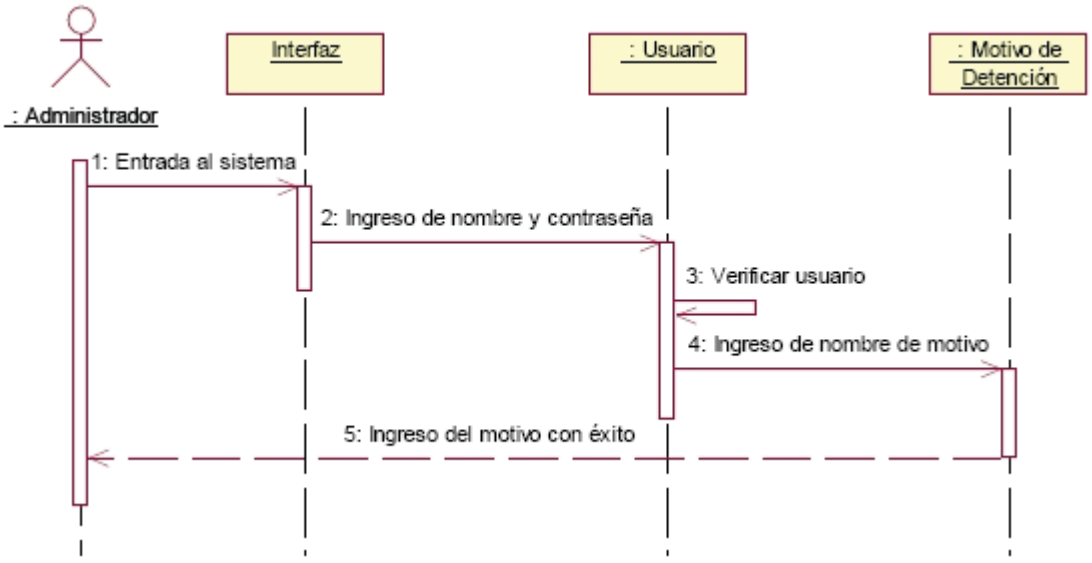


Figura 9.22: El administrador parametriza al sistema con un nombre de motivo de detención de una máquina o equipos de manera exitosa (diagrama de secuencia)

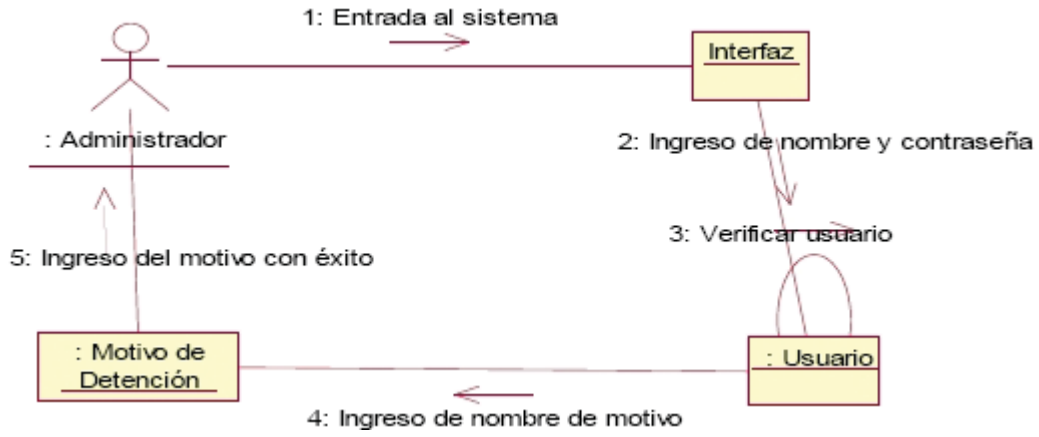


Figura 9.23: El administrador parametriza al sistema con un nombre de motivo de detención de una máquina o equipos de manera exitosa (diagrama de colaboración)

Caso de uso: Sistema de registro y control.

Escenario: El administrador ingresa la compra de una máquina o un equipo con éxito.

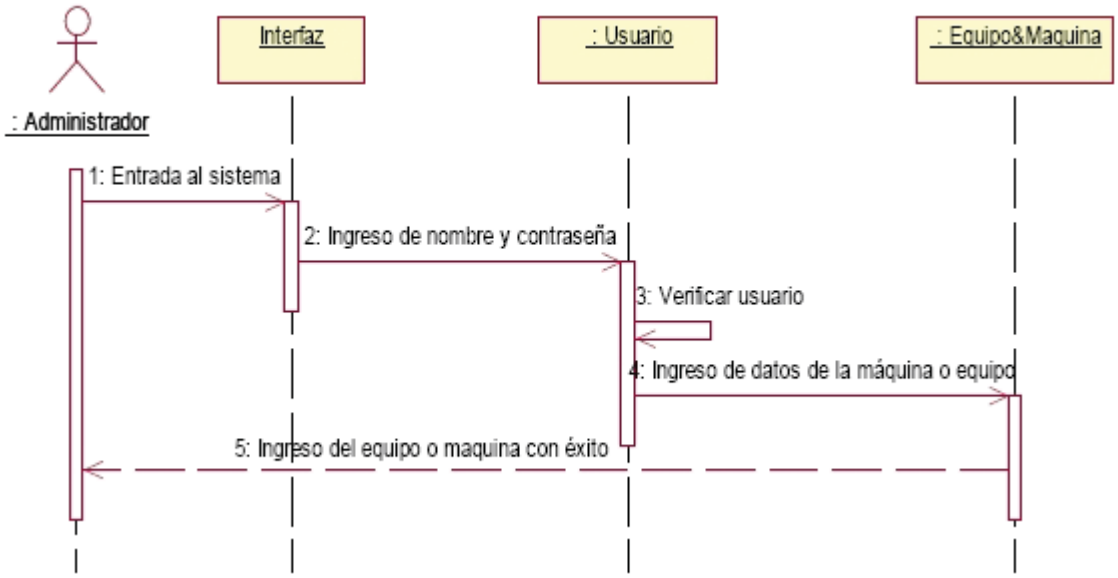


Figura 9.24: El administrador ingresa la compra de una máquina o un equipo con éxito (diagrama de secuencia)

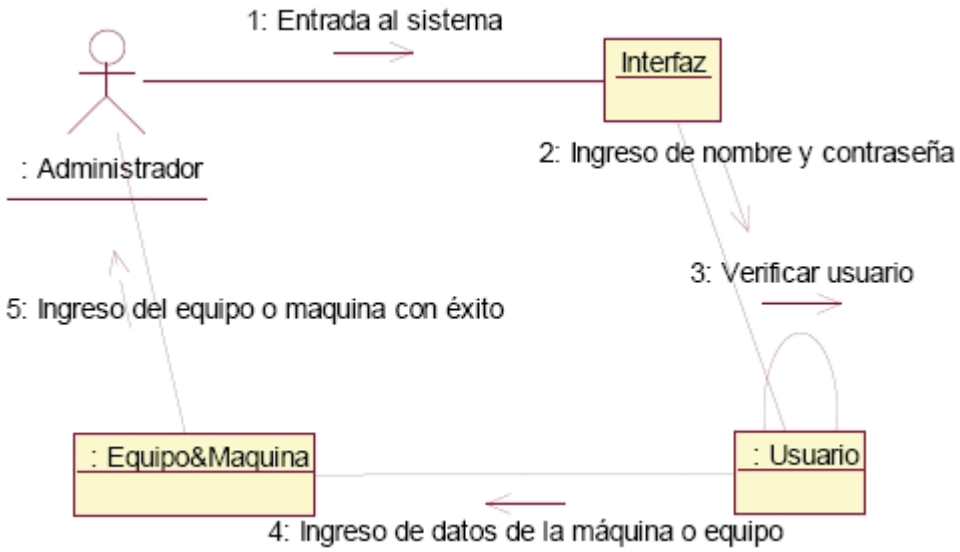


Figura 9.25: El administrador ingresa la compra de una máquina o un equipo con éxito (diagrama de colaboración)

Caso de uso: Sistema de registro y control.
Escenario: El usuario ingresador registra el cambio de pieza en una máquina o equipo de manera exitosa.

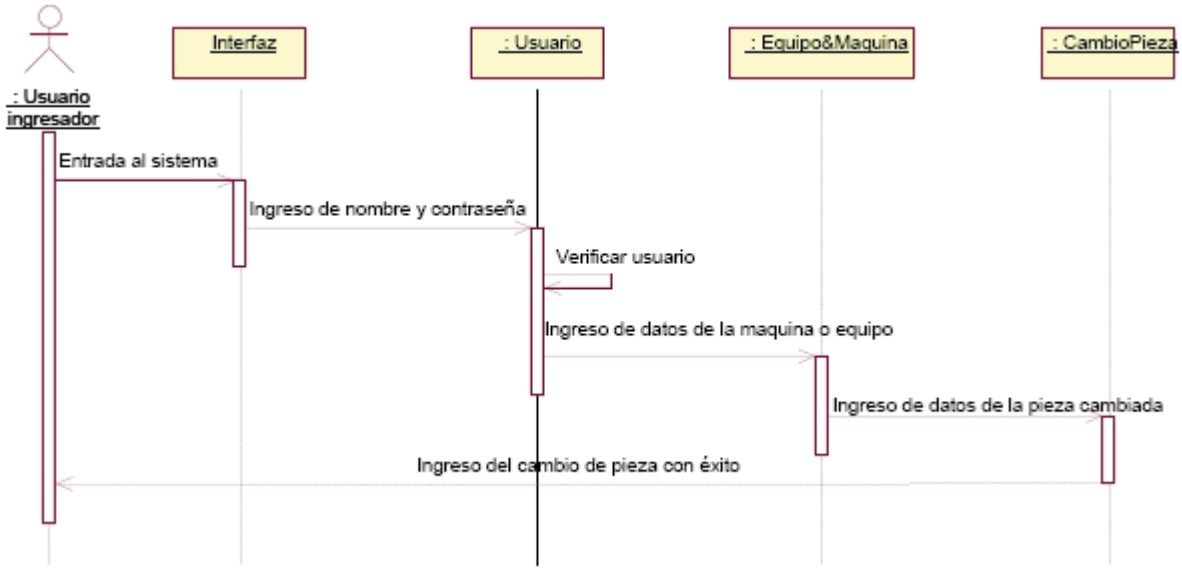


Figura 9.26: El usuario ingresador registra el cambio de pieza en una máquina o equipo de manera exitosa (diagrama de secuencia)

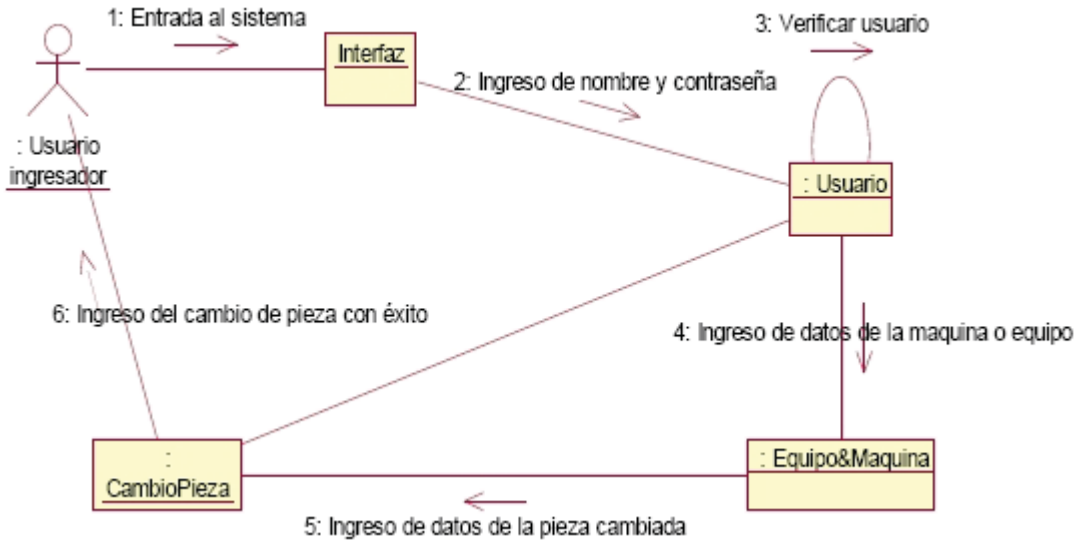
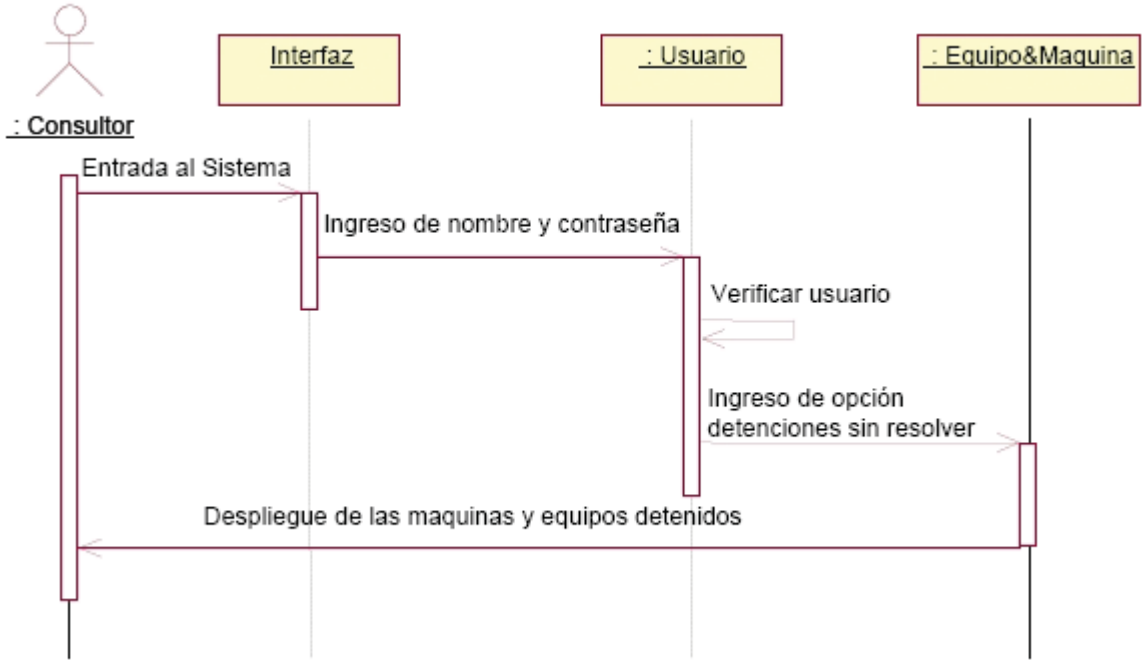


Figura 9.27: El usuario ingresador registra el cambio de pieza en una máquina o equipo de manera exitosa (diagrama de colaboración)

Caso de uso: Sistema de registro y control.
Escenario: El consultor busca las máquinas y equipos que se encuentran detenidos satisfactoriamente.

Figura 9.28: El consultor busca las máquinas y equipos que se encuentran detenidos satisfactoriamente (diagrama de secuencia)



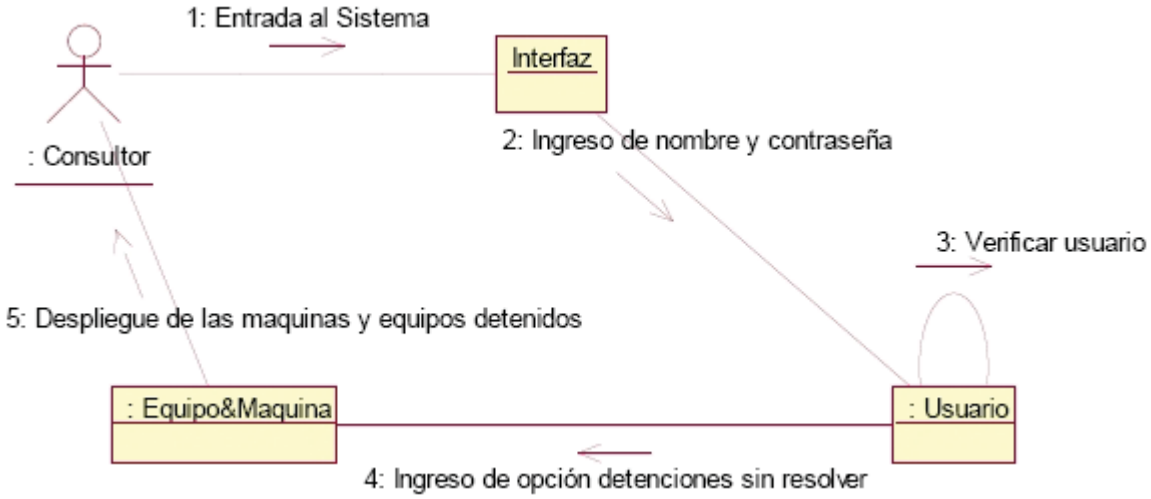


Figura 9.29: El consultor busca las máquinas y equipos que se encuentran detenidos satisfactoriamente (diagrama de colaboración)

- **Administración de calidad:**

- Revisión del análisis de consistencia: La revisión comparativa de los requisitos generados en la etapa de estrategia con respecto a los casos de uso, muestra la ausencia de

la funcionalidad de parametrizar los nombres de las piezas disponibles para el sistema, generando un nuevo *extend* para el caso de uso sistema de administración con respecto al actor administrador.

Figura 9.30: Casos de uso de sistema de administración

9.4.3 Diseño de Sistema y de Software

- **Diagrama de clase de diseño:**

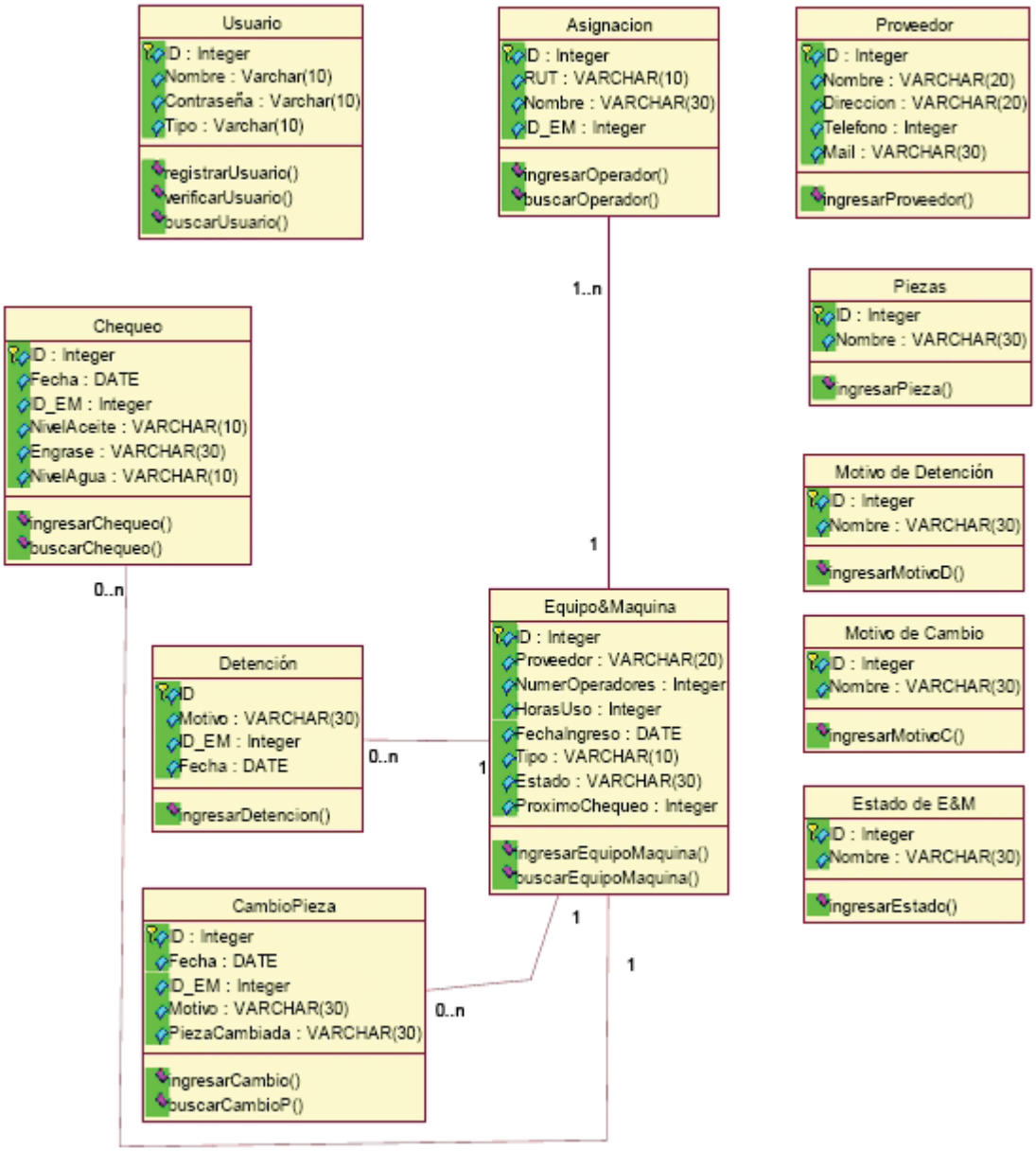


Figura 9.31: Diagrama de clase de diseño de SCME

- **Pruebas de integración:**

- Acceder al sistema con los tres tipos de usuarios y usuarios incorrectos.

- Ingresar proveedor para los siguientes casos: Sin nombre - Sin mail - Sin teléfono - Teléfono con caracteres - Todos los campos llenos correctamente.

- Ingresar pieza para los siguientes casos: Sin nombre – Con nombre.

- Ingresar motivo de cambio de pieza y detención para los siguientes casos: Sin nombre - Con nombre.

- Ingresar equipos o maquinas para los siguientes casos: Sin cantidad de horas - Sin nombre - Todos los campos llenos correctamente.

- Ingresar estado para los siguientes casos: Sin nombre - Con nombre.

- Ingresar usuario para los siguientes casos: Sin nombre- Sin contraseña - Sin confirmación de contraseña - Contraseña y confirmación de contraseña diferentes - Todos los campos llenos correctamente.

- Ingresar un chequeo para los siguientes casos: Sin seleccionar una máquina o equipo - Seleccionando más de un equipo o máquina - Sin engrase – Son nivel de aceite – Sin nivel de agua - Seleccionando una máquina o equipo y llenando los campos de manera correctamente.

- Ingresar una detención para los siguientes casos: Sin seleccionar una máquina o equipo - Seleccionando más de un equipo o máquina - Seleccionando una máquina o equipo y llenando los campos de manera correctamente.

- Ingresar un cambio de pieza para los siguientes casos: Sin seleccionar una máquina o equipo - Seleccionando más de un equipo o máquina - Seleccionando una máquina o equipo y llenando los campos de manera correctamente.

- Asignar neumáticos a un equipo para los siguientes casos: Sin seleccionar una máquina o equipo - Seleccionando más de un equipo o máquina - Sin nombre - Sin RUT - Sin horas de uso - Seleccionando una máquina o equipo y llenando los campos de manera correctamente.

- Consultar por detenciones sin resolver.

- Consultar por máquinas o equipos con chequeos pendientes.

- Consultar por los chequeos históricos de una máquina o equipo para los siguientes casos: Sin seleccionar una máquina o equipo - Seleccionando más de un equipo o máquina - Seleccionando una máquina o equipo.

- Consultar por los operadores que han utilizado una máquina o equipo para los siguientes casos: Sin seleccionar una máquina o equipo - Seleccionando más de un equipo o máquina - Seleccionando una máquina o equipo.

- Consultar por las piezas cambiadas de una máquina o equipo para los siguientes casos: Sin seleccionar una máquina o equipo - Seleccionando más de un equipo o máquina - Seleccionando una máquina o equipo.

- **Administración de calidad:**

1. El campo *NumerOperadores* de la tabla *EquipoMaquina* no es necesario para cumplir con los requerimientos.

2. La tabla *Pieza* debe tener un campo *Tipo* que permita distinguir si la pieza pertenece aun equipo o una maquina.

3. En la tabla *EquipoMaquina* es necesaria la existencia de un campo *Nombre* de tipo *VARCHAR(30)*, que permita guardar el nombre de la maquina al momento de ser ingresada al sistema.

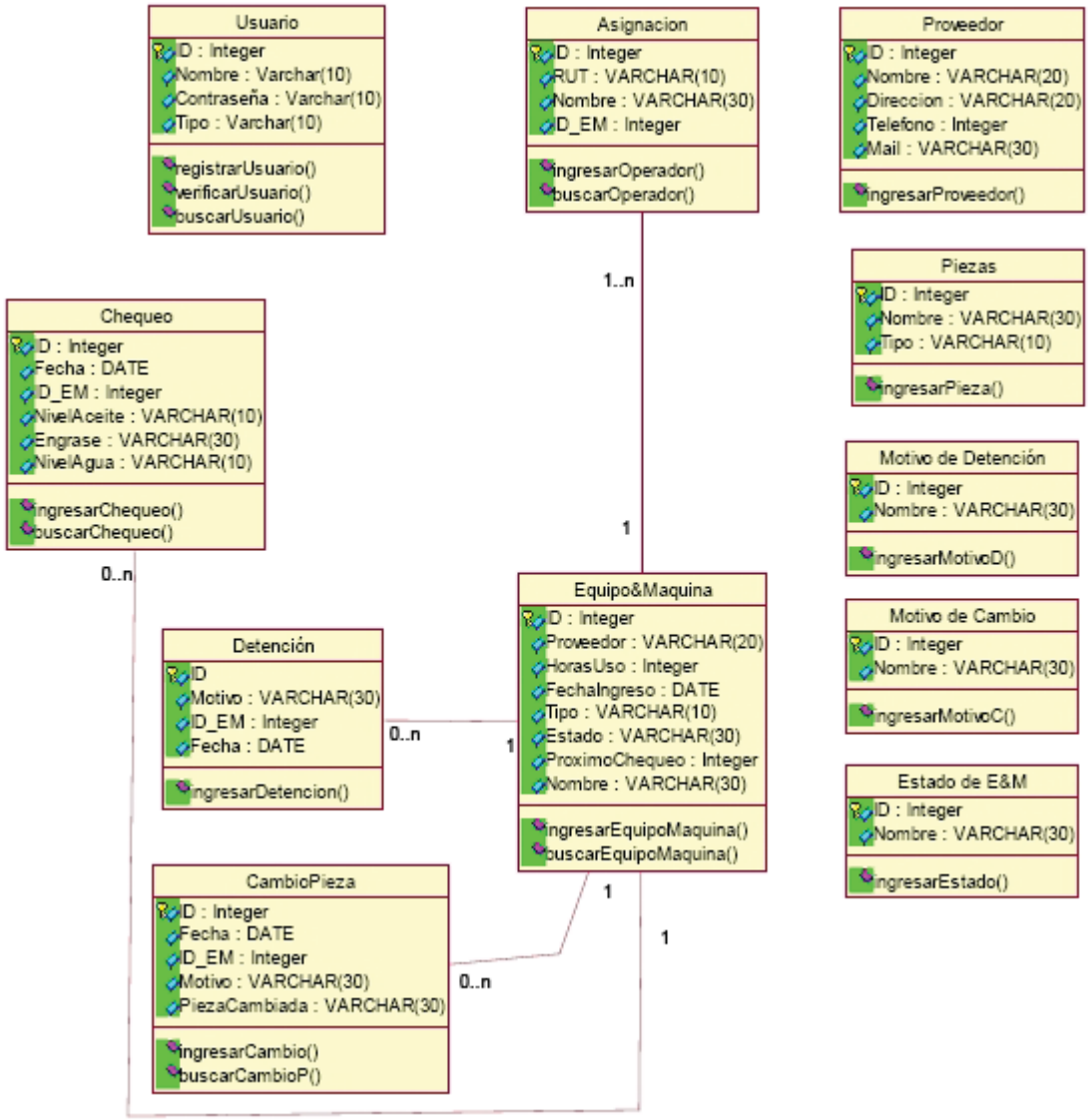


Figura 9.32: Diagrama de clase de diseño (corregido)

9.4.4 Análisis e Implementación de Componentes

- **Definición de incremento:**

- Incremento 1: Cumple con todos aquellos requerimientos necesarios para el usuario Administrador, junto con las tarea de ingreso al sistema.

- Incremento 2: Cumple con todos aquellos requerimientos necesarios para el usuario Ingresador.

- Incremento 3: Cumple con todos aquellos requerimientos necesarios para el usuario Consultor.

- **Iteración 1:**

- Componentes reutilizados: Usuario.class, Proveedor.class, Motivo.class (Cambio y Detención), Estado.class, Conectar.java, Registro.java, ValNum.java, Fecha.java, Lafecha.js, Menu.js, EstiloInicio.css, EstiloAdmin.css, index.htm, downIndex.jsp, topIndex.jsp, IngProveedor.jsp, IngMotivo.jsp, IngEstado.jsp, IngUsuario.jsp, leftAdmin.htm, rightAdmin.htm, administrador.htm, Botones.png, Banner.png, Alerta.jpg y Lineatitulo.jpg.

- Componentes desarrollados: EquipoMaquina.java, Pieza.java, IngPieza.jsp y IngEyM.jsp.

- Componentes con potencial de reutilización: EquipoMaquina.java, Pieza.java, IngPieza.jsp y IngEyM.jsp.

- **Iteración 2**

- Componentes reutilizados: EstiloIngresador.css, rightIngre.htm, leftIngre.htm y ingresador.htm

- Componentes desarrollados: Asignacion.java, Chequeo.java, CambioPieza.java, IngAsign.jsp, IngCambioP.jsp, IngChequeo y IngDetencion.jsp.

- Componentes con potencial de reutilización: Asignacion.java y IngAsign.jsp.

- **Iteración 3**

- Componentes reutilizados: EstiloConsultor.css, leftConsu.htm, rightConsu.htm y consultor.htm.

- Componentes desarrollados: ConChequeo.jsp, ConOperador.jsp, ConCheqHist.jsp, ConDetencion.jsp y ConPieza.jsp.

- Integración y Prueba

- **Errores detectados en la iteración 1:**

1. Una vez que se cierra una sesión, el siguiente usuario puede acceder a funcionalidades de otros usuarios haciendo "atrás" en el navegador.

- **Errores detectados en la iteración 2:**

1. Las máquinas pueden quedar con cambios de pieza de un equipo y los equipos pueden quedar con cambios de pieza de una máquina (Di=2).

2. No queda registrada la fecha en que es asignada la máquina o equipo a un operador.

3. Se puede efectuar una asignación de maquina o equipo que esta detenido a un operador

4. Se puede ingresar la detención de un equipo que ya se encuentra detenido por fallas.

- **Errores detectados en la iteración 3:**

1. Al consultar por las maquinas o equipos utilizadas por un operador, aparecen registradas maquinas que se encontraban detenidas.

2. Al consultar por los cambios de pieza históricos de un equipo o máquina puede no ser correcto debido al error 2

9.4.5 Mediciones de Producto y Procesos

- Proceso de aseguramiento de la calidad: Se calcula el rendimiento de las inspecciones para cada etapa como muestra la tabla 9.7.

	Insertados	Eliminados	Acumulativos eliminados	Acumulativos insertados	Escapes netos	Rendimiento

Estrategia	0	0	0	0	0	
Inspección de estrategia	0	0	0	0	0	
Análisis y definición de requerimientos	1	0	0	1	1	
Inspección de análisis	0	1	1	1	0	100%
Diseño de sistema y de software	5	0	1	6	5	
Inspección de diseño	0	3	4	6	2	60%
Análisis e implementación de componentes	0	0	4	6	2	
Inspección de análisis e implementación de componentes	0	0	4	6	2	0%
Integración y pruebas	6	0	4	12	8	
Inspección de integración y pruebas	0	8	12	12	0	100%

Tabla 9.7: Rendimiento de las inspecciones para cada etapa del SCME

Rendimiento total del proceso = (4*100)/(4+8)= 33,33%

- Proceso de construcción: Se calcula la eficiencia de la construcción, eficacia de la construcción y efectividad, para esto se analizan los requisitos del prototipo como muestra la tabla 9.8. Para efectuar el análisis de este proceso se considera la construcción del prototipo sin los efectos de corrección por las respectivas pruebas efectuadas para cada incremento.

REQUISITOS	POR REALIZAR	REALIZADOS	REALIZADOS CORRECTOS
ADMINISTRADOR	3	3	3
USUARIO INGRESADOR	4	4	2
CONSULTORES	5	5	3
NO FUNCIONALES	4	3	3
TOTAL	16	15	11

Tabla 9.8: Análisis de requisitos del SCME

Eficiencia de la construcción = (15/16)*100 =93,75%.

Eficacia de la construcción = (11/15)*100 = 73,33%.

Efectividad = (93,75*73,33)/100 = 68.74%.

- Proceso de reutilización de componente: Se calcula la eficiencia de la reutilización, para esto se analiza la reutilización de los componentes, como muestra la tabla 9.9, para las diferentes iteraciones.

	ITERACIÓN 1	ITERACIÓN 2	ITERACIÓN 3
Componentes a implementar	31	11	9
Componentes reutilizados	27	4	4
Eficiencia por iteración	87,09%	36,36%	44,44%
Eficiencia total			68,62%

Tabla 9.9: Eficiencia de la reutilización por iteración del SCME

Para llevar acabo la aplicación de las métricas al producto se debe efectuar la suma de las líneas de código y de las clases generadas previos a las pruebas del prototipo, como muestra la tabla 9.10 (suma de líneas de código).

ARCHIVOS FUENTES	LDC	ARCHIVOS FUENTES	LDC	ARCHIVOS FUENTES	LDC
ConCheqHist.jsp	126	Asignacion.java	57	consultor.htm	12
ConChequeo.jsp	51	CambioPieza.java	66	index.htm	11
ConDetencion.jsp	53	Chequeo.java	75	ingresador.htm	11
ConOperador.jsp	124	Conectar.java	50	leftAdmin.htm	20
ConPieza.jsp	124	Detencion.java	57	leftConsu.htm	23
downIndex.jsp	111	EquipoMaquina.jav	100	leftIngre.htm	18

		a			
IngAsign.jsp	154	Estado.java	39	rightAdmin.htm	23
IngCambioP.jsp	148	Fecha.java	36	rightConsu.htm	23
IngChequeo.jsp	170	MotivoCambio.java	39	rightIngre.htm	23
IngEyM.jsp	135	MotivoDetencion.java	39	EstiloAdmin.css	89
IngEstado.jsp	73	Pieza.java	46	EstiloConsultor.css	99
IngDetencion.jsp	131	Proveedor.java	66	EstiloIngresador.css	89
IngMotivo.jsp	108	Registro.java	160	EstiloInicio.css	56
IngPieza.jsp	84	Usuario.java	75	Menu.js	86
IngProveedor.jsp	123	ValNum.java	17	Lafecha.js	31
IngUsuario.jsp	125	administrador.htm	11	Líneas de código SQL	11
topIndex.jsp	20	Total de LDC = 3418			

Tabla 9.10: Numero de líneas de código por archivo fuente del SCME

- Métrica de calidad 1:

DKCL = 8/3,418= 2,34

- Métrica de calidad 2:

Numero total de clases = 32, DC= 8/32= 0,25

- Métrica de reutilización: Para la aplicación de la métrica de reutilización se calcula el total de LCD reutilizadas como muestra la tabla 9.11.

ARCHIVOS FUENTES	LDC	ARCHIVOS FUENTES	LDC	ARCHIVOS FUENTES	LDC
administrador.htm	11	Estado.java	39	IngMotivo.jsp	108
consultor.htm	12	Fecha.java	36	IngProveedor.jsp	123
index.htm	11	MotivoCambio.java	39	IngUsuario.jsp	125
ingresador.htm	11	MotivoDetencion.java	39	topIndex.jsp	20
leftAdmin.htm	20	Proveedor.java	66	EstiloAdmin.css	89
leftConsu.htm	23	Registro.java	160	EstiloConsultor.css	99
leftIngre.htm	18	Usuario.java	75	EstiloIngresador.css	89
rightAdmin.htm	23	ValNum.java	17	EstiloInicio.css	56
rightConsu.htm	23	downIndex.jsp	111	Menu.js	86
rightIngre.htm	23	IngEstado.jsp	73	Lafecha.js	31
Conectar.java	50	Total de LDC reutilizadas = 1706			

Tabla 9.11: Numero de líneas de código reutilizadas en el SCME

PR = (1706/3418)*100=49,91%

9.5 Análisis de Resultados

Resumiendo los resultados, de las métricas aplicadas a los procesos empleados para el desarrollo de ambos prototipos, se obtiene la tabla 9.12.

	Eficiencia de la construcción	**Eficacia de la construcción**	**Efectividad de la construcción**	**Eficiencia de la reutilización**	**Rendimiento total del proceso (SQA)**
Prototipo de SCN	80%	66,66%	53.32%	42,3%	31,25%
Prototipo de SCME	93,75%	73,33%	68.74%	68,62%	33,33%

Tabla 9.12: Resultados de la aplicación de las métricas en los procesos

Al analizar las cifras se ve claramente un aumento en todos los indicadores al momento de efectuar un segundo prototipo. El aumento de componentes disponibles a reutilizar, producto de su captura de construcciones previas de prototipos pertenecientes a la Línea de Productos de Exertus, es lo que genera un inmediato aumento en la eficiencia de la reutilización al disponer de más componentes para reutilizar e implementar a su vez las funcionalidades del sistema.

Otro factor que aumenta junto a la eficiencia de la reutilización es la efectividad de la construcción, debido a que cada vez son más los requerimientos que se satisfacen mediante componentes prefabricados para el dominio, asegurando de mejor forma la realización correcta del requisito.

Las tareas de inspección de errores después de cada etapa, contempladas en la administración de calidad, permiten eliminar futuros defectos con anterioridad, para esto

es que se mide el rendimiento del proceso previo a una etapa de prueba del producto. La densidad de errores insertados en cada etapa disminuye en el segundo prototipo con respecto al primero, pero sigue siendo la etapa de integración de componentes aquella donde se inserta la mayor cantidad de errores con respecto a otras etapas del desarrollo del prototipo. Por lo tanto el aumento de componentes reutilizados al momento de efectuar la integración de componentes, propios de un incremento, evita la introducción de errores en esta etapa lo que se traduce en un alza en el rendimiento total del proceso de aseguramiento de calidad.

Los resultados, de las métricas aplicadas al producto se comparan con datos obtenidos mediante la aplicación de las mismas métricas a desarrollos efectuados en Exertus bajo un esquema de desarrollo tradicional (utilizando lenguajes OO), como se muestra en la tabla 9.13.

	Proyecto 1 de desarrollo tradicional	**Proyecto 2 de desarrollo tradicional**	**Prototipo de SCN**	**Prototipo de SCME**
Errores/Clases	0,21	0,82	0,35	0,25
Errores/ KLC	3,1	8,2	3,32	2,34

Tabla 9.13: Resultados de la aplicación de las métricas al producto

Los porcentajes de reutilización en el prototipo de SCN es de un 32,99% en comparación a un 49,91%, este aumento en la cantidad de líneas de códigos reutilizadas tiene una directa relación con la disminución de errores por clases y la disminución de errores por cada mil líneas de código. Además se puede apreciar la paulatina mejora con respecto a los datos obtenido bajo un desarrollo tradicional, principalmente en los errores/KLC.

Conclusiones

Una de las ideas principales de la Fábrica de Software es la industrialización del desarrollo de software, en donde se pretende elaborar una línea única de procesos bien definidos que cubran las distintas etapas del desarrollo de software, permitiendo un mejor y mayor control de cada una de estas etapas asegurando de este modo la calidad del producto y el mejoramiento de los procesos. Además la Fábrica de Software estimula un alto nivel de reutilización con el fin de hacer del desarrollo de software un trabajo más productivo, especializándose en un dominio en particular.

La Fábrica de Software aspira a tener procesos controlables y mejorables en el tiempo, por lo tanto es vital cada vez más el modelamiento de sus procesos internos, junto con la declaración de indicadores para los procesos que permita llevar un control real de cada uno de estos, permitiendo un mejoramiento mediante una retroalimentación gracias a la información entregada por los indicadores. Por lo tanto será necesario efectuar un mayor modelamiento y declaración de más indicadores en trabajos futuros, para de esta forma abarcar más variables que puedan ayudar a la industrialización del desarrollo de software al interior de la fábrica.

Después de las mediciones y comparaciones efectuadas a los prototipos implementados queda al descubierto las mejoras obtenidas en otros procesos y el producto mismo gracias a un aumento en la eficiencia de la reutilización permitiendo la salida de un producto de calidad. Sin embargo se pudo apreciar en el desarrollo de los prototipos la alta potencialidad de reutilización existente en tarea de análisis, diseño y pruebas, en donde diagramas de caso de uso, diagrama de clases o planes de prueba podrían ser vistos como componentes a reutilizar. Si bien este proyecto contempla una reutilización a nivel de código e interfaces gráficas en una etapa de programación, no queda descartada la posibilidad de implementar a futuro una reutilización transversal al desarrollo que abarque distintas áreas y no solo la de programación.

La alta potencialidad de reutilización mostrada entre un proyecto y otro, se debe a una de las principales características de la Fábrica de Software que es su especialización en un dominio mediante una Línea de Productos. Si bien los porcentajes de reutilización de un proyecto a otro tienden a aumentar, jamás logrará un 100% de reutilización y tampoco es el objetivo a perseguir por la Fábrica de Software, ya que en ese caso se encontraría en una situación de comoditización del producto y no de un desarrollo de software a la medida.

La elaboración de un producto, al interior de la Fábrica de Software, mediante el ensamble de componentes, en gran parte reutilizados, trae también la ventaja de ofrecer una mantención estable del producto, debido a que gran parte del software se encuentra construido por componentes validados, verificados y documentados, permitiendo una rápida identificación del problema o modificaciones en el producto. Esto no deja de ser importante al considerar que el servicio post venta es uno de los más complicados en el área de desarrollo de software, transformando de esta forma a la fábrica en un esquema de desarrollo de software que ofrece alta competitividad en el mercado.

Con el desarrollo de esta memoria se puede apreciar que el esquema propuesto por la Fábrica de Software se escapa a una metodología detallada de pasos a seguir, para transformarse en una forma de ver el desarrollo de software, en donde se aplican conceptos industriales conocidos y manejados en otras áreas, los que deben ser estudiados acorde al contexto en donde se implemente e integrados para obtener los beneficios de la fábrica. Por lo tanto se hace innecesaria la reinvención de muchos de estos puntos, como la administración de la calidad o el ciclo de vida del proyecto de software, lo importante es adecuarlos e integrarlos con el fin común de la industrialización del desarrollo del software.

Esta memoria aporta a la empresa Exertus el estudio práctico necesario para la implementación del la Fábrica de Software y el desarrollo de varios de los puntos clave para constituirla, dejando la base para la profundización de otros. Además de un análisis previo de los beneficios y dificultades que puede generar este esquema de desarrollo de software en la construcción de un producto, permitiendo a la empresa tener un

conocimiento de cómo administrar estos beneficios y de que manera poder generar las mejoras durante su implementación. Es así como de esta experiencia se elaboro el articulo "Fabrica de Software: Hacia la Industrialización del Desarrollo de Software en la Minería" el cual fue seleccionado para el XIX Encuentro Chileno de Computación y presentado en Iquique, permitiendo mostrar la experiencia de Exertus a nivel nacional.

Bajo un punto de vista personal y profesional, esta memoria fue un aporte significativo a mi formación como ingeniero civil informático, debido a su inmediata conexión con el mundo laboral y profesional, producido por las constantes reuniones generadas con los representantes de Exertus para el desarrollo de cada uno de los punto que abarcó este proyecto de innovación tecnológica, en las dependencias de la 3IE permitiendo un roce con otras empresas del área.

Referencias

Aaen Ivan ; Bøttcher Peter y Mathiassen Lars. "The Software Factory: Contributions and Illusions". En: Information Systems Research Seminar (20ª: 1997: Escandinavia): *Proceedings for Twentieth Information Systems Research Seminar,* Oslo: UiO, 1997, vol. 17, p. 407-433.

Sanchez Alejandro ; Montejano Gérman ; Peralta Mario y Riesco Daniel. *Métricas de Calidad y un Modelo Costo – Beneficio Ajustados a un Caso Real de la Industria del Software* [En línea]. San Luis: Departamento de Informática. Universidad Nacional de San Luis, 2001. [Consulta: 8 sep. 2006]. Disponible a: <http://www-2.dc.uba.ar/materias/isoft2/cacic2001.pdf>

Anguita Patricio. *Fábrica de Software* [En línea]*: Creando software de calidad.* 10 oct. 2005 [Consulta: 15 ene. 2006] <http://www.fabricadesoftware.cl/viewtopic.php?t=546>

Barker Richard. *Case*Method: Tasks and Deliverables,* England: AddisonWesley, 1990, Caps 3-9.

Barros Justo José Luís. *Técnicas para la clasificación/recuperación de componentes software reutilizables y su impacto en la calidad* [En línea]. Vigo: 1998. [Consulta: 6 ago. 2006] Disponible a: <http://trevinca.ei.uvigo.es/~jbarros/publicaciones/quatic98.pdf>

Bemer R. W. "The Economics of Program Production" En: A. J. H. Morrell (Ed.). *Position papers for Panel Discussion*, Amsterdam: Information Processing 68, 1969.

De Amescua Antonio ; De Miguel Adoración ; Lloréns Juan y Velasco Manuel. "Definición de una metodología orientada a la reutilización de software: ROM". En: *SIMO Jornada sobre Tecnología de Objetos (2ª: 1996: Madrid)* Madrid: Departamento de Informática de la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, 1996.

Del Juncal Huerta Jorge Luis ; Landeros Gómez Ruth Priscila. *Herramientas Case* [En línea]. [Consulta 2 ago. 2006]. Disponible a: <http://www.monografias.com/trabajos14/herramicase/herramicase.shtml>

Flebes Estrada Aylin; Perez Estevez Isabel. *Medir el proceso de control de configuración, ¿una utopía para la Industria Nacional de Software?* [En línea]. Instituto Superior Politécnico "José Antonio Echeverría", Habana, Cuba. Ago. 2003 [Consulta: 6 nov. 2006] Disponible a: <http://www.inf.udec.cl/revista/ediciones/edicion9/febles.pdf>

Fuller Padilla David. "Roles en el desarrollo de software". En: *Apuntes de taller de ingeniería de Software,* 2003, Cap 4, p.1-49.

Genero Bocco Marcela. *HERRAMIENTAS CASE* [En línea]. [Consulta 3 ago. 2006]. Disponible a: <http://alarcos.inf-cr.uclm.es/doc/aplicabbdd/Herramientas%20CASE.pdf>

Greenfield Jack ; Short Keith. "Software Factories". En: *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*, Indianapolis: John Wiley and Sons, 2004, Cap. 5, p.155-188.

Instituto Nacional de Estadísticas e Informática. *Herramientas Case.* Lima: Talleres de la Oficina de Impresiones de la Oficina Técnica de Difusión Estadística y Tecnológica Informática del INEI, 1999, Caps. 4 y 8.

Kelly Steven. *Improving Developer Productivity With Domain* [En línea] developer.* The Independent Magazine for Software Developers, 3 jul. 2005. [Consulta: 6 mar. 2006]. Disponible a: <http://www.developerdotstar.com/mag/articles/domain_modeling_language.html>

Lima Marcos ; Majluf Nicolás. "El Diseño Organizacional: La estructura sigue a la estrategia". El Mercurio de Santiago, Cuerpo B, 2 de abr. 2006, Cuarta clase.

Menéndez-Barzanallana Asensio Rafael. *Principales herramientas Case del mercado y su uso* [En línea]. Murcia: Departamento Informática y Sistemas - Universidad de Murcia, 17 jun. 2006. [Consulta 1 ago. 2006]. Disponible a: <http://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html>

Ministerio de Administración Pública de España. "Aseguramiento de la Calidad" [En línea] En: *Métrica. Versión 3: Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información*. [Consulta: 14 ago. 2006]. Disponible a: <http://www.csi.map.es/csi/metrica3/>

Muñoz Javier ; Pelechano Vicente. *MDA vs Factorías de Software*. Valencia: Dept. de Sistemas Informáticos y Computadores. Universidad Politécnica de Valencia, 2005.

Platt Michael. *Microsoft Architecture Overview: Executive summary*. Microsoft Corporation, jul. 2002. [Consulta: 11 mar. 2006]. Disponible a: <http://www.enterprise-architecture.info/Images/Documents/Microsoft%20Architecture%20Overview.pdf>

Pressman Roger S. "Proceso de software y métricas de proyectos". En: *Ingeniería del Software: Un enfoque práctico*, 5ª edición Madrid: McGRAW-HILL, 2002, Cap 4, p. 53-75.

Reina Quintero Antonia Maria ; Torres Valderrama Jesús. "Implicaciones de transformaciones oblicuas en el desarrollo de un framework generador de aplicaciones orientadas a aspectos" En: Estevez A., Pelechano V., Vallecillo A (Ed.). Actas del II Taller sobre Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (DSDM 2005). España: 13 sept. 2005, p. 67-73.

Reynoso Carlos Billy. "Introducción a la Arquitectura de Software" [En línea]. En: *Arquitectura de Software.* España: MSDN, mar. 2004. [Consulta: 21 abr. 2006]. Disponible a: < http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp>

Sommerville Ian. *Ingeniería de software*, 6ª ed. Mexico: Pearson Educación, 2002, (Caps. 3, 7,10 ,12,14).

Wikipedia [En línea]: *La enciclopedia libre.* 3 ago. 2006. [Consulta: 14 ago. 2006]. Disponible a: <http://es.wikipedia.org/wiki/METRICA>

APÉNDICE

APÉNDICE A – GLOSARIO

Glosario

3IE: Instituto Internacional para la Innovación Empresarial.

Artefacto de Desarrollo: Es aquel elemento necesario para el desarrollo del software: documentos, modelos, archivos, códigos, etc.

Activo de Implementación: Ver Activo de Producción.

Activo de Instalación: Es un Activo de Producción**,** por ejemplo configuraciones para los computadores en los cuales los miembros de familia serán instalados.

Activo de Proceso: Acompaña al Activo de Implementación, apoyando su uso (casos de prueba), documentación de usuario, de diseño o herramientas de desarrollo.

Activo de Producción: Es un artefacto de software que explícitamente tuvo la intención de proporcionar un retorno de la inversión a través de la reutilización, por ejemplo, lenguajes, herramientas, patrones y marcos de trabajo. Se llama al Activo de Producción el Activo de Implementación, ya que son usados directamente en la implementación del producto.

Activo de Prueba: Es un Activo de Producción**;** equipos prueba, pruebas de unidad y suites de prueba de integración.

Activo de Requerimiento: Es un Activo de Producción, por ejemplo lenguajes de especificación.

CAST: Computer Aided Software Testing. Pruebas de Software Asistida por Ordenador.

CASE: Computer Aided Software Engineering. Ingeniería de Software Asistida por Ordenador.

CBSD: Component Based Software Development.
Desarrollo de Software Basado en Componentes.

CMMI: Capability Maturity Model Integration.
Integración del Modelo de Capacidad y Madurez.

CMS: Content Management System.
Sistema de Gestión de Contenido.

CSS: Cascading Style Sheets.
Hojas de Estilo en Cascada.

DSM: Domain Specific Modeling.
Modelado de Dominio Específico.

ERP: Enterprise Resource Planning.
Planeación de Recursos de la Empresa.

Esquema de Fábrica de Software: Permite clasificar y resumir artefactos de desarrollo, a través de una Matriz en donde las columnas definen intereses, mientras las filas definen los niveles de abstracción. Cada celda define una perspectiva o Punto de Vista, del cual se puede construir algún aspecto del software. También se puede representar en forma de grafo, permitiendo ver de mejor forma la arquitectura del software.

Fábrica de Software: Es una Línea de Productos de software configurada por herramientas genéricas, procesos, que usa una Plantilla de una Fábrica de Software, basada en un Esquema de Fábrica de Software, para automatizar el desarrollo y el mantenimiento de un producto, adaptando, ensamblando y configurando componentes a base de marcos de trabajo.

HTML: HyperText Markup Language.
Lenguaje de Marcas de Transferencia de Hipertexto.

IDE: Integrated Development Environment.
Entorno de Desarrollos Integrados.

IEEE: Institute of Electrical and Electronics Engineers.
Instituto de Ingenieros Eléctricos y Electrónicos.

Línea de Productos: Una Línea de Productos es un grupo de productos relacionados entre sí que se ofrecen a la venta. Al contrario que la agrupación de productos en la que varios productos se combinan en uno, la creación de Líneas de Productos implica el ofrecer varios productos relacionados entre sí pero de forma individual. Una línea puede comprender productos de varios tamaños, tipos, precios, etc.

PHP: PHP Hypertext Preprocessor.
PHP Preprocesador de Hipertexto.

Plantilla de Fábrica de Software: Empaqueta Activos de Producción como: Activos de Requerimientos, Activos de Implementación, Activos de Proceso, Activos de Prueba y Activos de Instalación, para formar un producto de una familia de productos dada.

Punto de Vista: Cada celda de un Esquema de Fábrica de Software corresponde a un Punto de Vista o perspectiva. En el se describen los artefactos necesarios para desarrollar un aspecto del software.

Script: Es un guión o conjunto de instrucciones. Permiten automatizar tareas creando pequeñas utilidades. Son interpretadas por un intérprete y usualmente son archivos de texto.

SQL: Structured Query Language.
Lenguaje de Consulta Estructurado.

UML: Unified Modeling Language.
Lenguaje Unificado de Modelado.

Vista: Es el subconjunto resultante de practicar una selección o abstracción sobre una realidad, desde un aspecto determinado.

Workflow: Es la definición de los flujos de trabajo, de los procesos internos que se realizan en una empresa y la incorporación de éstos en los sistemas de información corporativos.

APÉNDICE B – DOCUMENTOS A FLUIR

}