

**REUSO DE SOFTWARE:
FACTORES ORGANIZACIONALES Y EL PROGRAMA DE REUSO**



T E S I S

**MAESTRIA EN ADMINISTRACION DE SISTEMAS
DE INFORMACION**

***Instituto Tecnológico y de Estudios
Superiores de Monterrey***

**POR:
GERARDO RODRIGUEZ ROJANO**

ABRIL DE 1997

**REUSO DE SOFTWARE:
FACTORES ORGANIZACIONALES Y EL PROGRAMA DE REUSO.**



TESIS

**MAESTRÍA EN ADMINISTRACIÓN DE SISTEMAS
DE INFORMACIÓN**

**INSTITUTO TECNOLÓGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY**

**POR
GERARDO RODRÍGUEZ ROJANO**

ABRIL DE 1997

**REUSO DE SOFTWARE:
FACTORES ORGANIZACIONALES Y EL PROGRAMA DE REUSO.**

POR

GERARDO RODRÍGUEZ ROJANO

TESIS

**PRESENTADA A LA
DIVISIÓN DE GRADUADOS E INVESTIGACIÓN**

**ESTE TRABAJO ES REQUISITO PARCIAL
PARA OBTENER EL TÍTULO DE:
MAESTRO EN ADMINISTRACIÓN
DE SISTEMAS DE INFORMACIÓN**

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

ABRIL DE 1997

Agradecimientos.

Al director de la Facultad de Informática de la Universidad Autónoma de Querétaro, el Ing. Luis Fernando Saavedra, por su amistad y el gran apoyo que recibí de su parte para cursar mis estudios de posgrado.

Agradecimientos.

Al asesor de la tesis, el MC. Alberto Lamadrid A., por su interés y paciencia para desarrollar este trabajo.

A los señores sinodales de la tesis:

MC. Mauricio Padilla G. y MC. Humberto Cárдинas A. por compartir sus experiencias conmigo y haber emitido sugerencias para la elaboración de la tesis.

A SINERGY SD., por su amistad y comprensión para poder conocer su empresa.

A la Dirección de Informática Educativa del Instituto Nacional de Capacitación Fiscal, por brindarme su confianza y apoyo para el desarrollo de la investigación.

Al programa SUPERA, por su valiosa y oportuna ayuda.

Dedicatorias.

A mis padres: Nachito y Consuelito, por motivarme siempre para seguir estudiando y porque siempre creyeron en mí.

A Claudia, por su cariño y comprensión que siempre me brindó.

RESUMEN.

Con el uso incrementado de la computación para apoyar en las actividades de una persona hasta apoyar en los procesos de una organización, se tiene la necesidad de producir software cada vez más confiable y en tiempo y costos establecidos. Dicho software es cada vez más complejo y se adapta bien a las necesidades exigentes de los usuarios. Esta complejidad radica en aplicaciones con interfaces amigables, manejo e intercambio de múltiples informaciones de diferentes tipos como texto, hipertextos, gráficos, imágenes, hojas de cálculo; así como el acceso a múltiples bases de datos, correos electrónicos, sistemas de búsquedas de información en redes de computadoras y en aplicaciones muy grandes como sistemas de información que integran procesos de varias áreas de negocios de una organización. En fin, la complejidad del software actual es evidente y el poder de cómputo del hardware está aumentando.

Además de esta complejidad, el medio ambiente está cambiando, crecen las necesidades de los usuarios, lo que implica crear y modificar software más rápido, con calidad y a un menor costo. Para poder lograrlo, se debe adoptar un proceso de desarrollo de software basado en el reuso del mismo software, esto es, reusar componentes de software producidos en proyectos anteriores, y así incrementar la productividad y mejorar la calidad en los productos de software. Lo anterior atiende la afirmación de que la mejor manera de hacer software es reutilizando el software ya probado en otros proyectos para acelerar el desarrollo de aplicaciones y lograr mejor calidad.

Para introducir el reuso en los procesos de desarrollo de software se requiere de un cambio en los procesos de construcción de software. El desarrollo de software tiene que pasar de ser un proceso puramente artesanal a ser un proceso de producción sistemático, es decir, que sea un proceso de producción más industrializado.

Para ayudar a introducir el reuso de software en los procesos de desarrollo de aplicaciones computacionales se llevó a cabo este trabajo. Durante la investigación, se encontró que aspectos organizacionales no técnicos son impedimentos principales para el éxito de dicha introducción del reuso en los procesos de desarrollo. El diseño de procesos de trabajo de reuso de software y de su integración a una organización de negocios es crítica para vencer muchos impedimentos que afectan el éxito del reuso de software.

Por lo anterior, la investigación se centró en determinar las características de una organización de desarrollo de software que está diseñada para cambiar y así poder adaptarse al medio ambiente cambiante, proponer modelos que describen estructuras organizacionales de soporte para integrar procesos de reuso y establecer los lineamientos para implementar un programa de reuso.

Como resultado de la investigación se determinó la necesidad de definir una organización de desarrollo de software que es flexible, de tal forma que puede adaptarse al medio ambiente cambiante y que además aprende de estos cambios. Dicha organización tiene las características de estar diseñada para estar basada en sus procesos principales, muestra un comportamiento adaptativo, soporta alineación a través de sus entidades funcionales, se enfoca en el cliente final, se comparte conocimiento entre los miembros de la organización para generar un aprendizaje organizacional y aprende a reusar experiencias. Estas son las características mínimas que debiera poseer una organización de desarrollo de software que intenta implementar un programa de reuso. Para apoyar la implementación de un programa de reuso, se describió un modelo que presenta una estructura organizacional de soporte y tres modelos más que son derivados de esta estructura. Los cuatro modelos describen los elementos organizacionales básicos y sus interrelaciones. Por último, se plantean las consideraciones previas antes de planear e implementar un programa de reuso y se propone un programa.

INDICE.

CAPÍTULO 1. INTRODUCCIÓN.

1.1 Antecedentes.....	1
1.2 Justificación.....	2
1.3 Definición del objetivo de la tesis.....	4
1.4 Valor Agregado.....	4
1.5 Metodología.....	4
1.6 Organización del Contenido.....	5

CAPÍTULO 2. INGENIERÍA DE SOFTWARE Y REUSO DE SOFTWARE.

2.1 ¿Qué es Ingeniería de Software?.....	7
2.1.1 El ciclo de vida del desarrollo de software en modelo de cascada (ciclo de vida clásico).....	8
2.1.2 Visión genérica de la ingeniería del software de acuerdo con Roger S. Pressman.....	9
2.2 ¿Qué es un proyecto de desarrollo de software exitoso?.....	11
2.3 ¿Qué hace un ingeniero de software?.....	11
2.4 ¿Cuáles son las características de un proyecto de Ingeniería de Software?.....	12
2.5 Interés por la Calidad.....	12
2.6 Productividad.....	14
2.7 El Reuso de Software.....	
2.7.1 Definición de reuso de software.....	14
2.7.2 Componentes de software.....	15
2.7.2.1 Componentes ejecutables por la máquina.....	16
2.7.2.2 Componentes no ejecutables por la máquina.....	17
2.7.3 Componentes ejecutables como un mecanismo de refuerzo según Roger S. Pressman.....	17
2.7.4 Ventajas de reusar código fuente considerando la visión de Bindu R. Rao.....	18
2.7.5 Bibliotecas de software reusable.....	18
2.7.6 Consideraciones sobre el Reuso.....	19
2.8 Nuevas Estructuras Organizacionales.....	20
2.9 Nuevos Modelos de Procesos.....	20

CAPÍTULO 3. CARACTERÍSTICAS DE UNA ORGANIZACIÓN FLEXIBLE DE SOFTWARE.

3.1 El diseño de la organización y sus problemas.....	22
3.2 Características de una organización flexible de desarrollo de software.	23
3.2.1 La organización debe estar diseñada para estar basada en sus procesos principales.	23
3.2.2 Mostrar comportamiento adaptativo.....	23
3.2.3 Soportar alineación a través de sus entidades funcionales.	24
3.2.4 Un enfoque en el cliente final.....	24
3.2.5 Aprendizaje organizacional.	25
3.2.6 Reusar experiencias.....	25

CAPÍTULO 4. FACTORES ORGANIZACIONALES Y EL REUSO DE SOFTWARE.

4.1 Introducción.....	26
4.2 El modelo general.	
4.2.1 Introducción.....	27
4.2.2 El director del departamento de sistemas.	28
4.2.3 La Biblioteca de reuso central.....	30
4.2.4 Biblioteca secundaria de reuso.	30
4.2.5 Biblioteca de productos terminados.	31
4.2.6 Administrador de Bibliotecas de software.	31
4.2.7 Los equipos de desarrollo de software.....	32
4.2.8 Líderes de proyecto.....	33
4.2.9 El Comité de Evaluación de Componentes.....	34
4.3 Variaciones en el modelo general.	
4.3.1 Introducción.....	35
4.3.2 Director de Productores y Consumidores.	
4.3.2.1 Introducción.....	36
4.3.2.2 El director del departamento de sistemas.	37
4.3.2.3 El equipo de productores.....	38
4.3.2.4 El líder del Equipo de Productores.	39
4.3.2.5 El equipo de consumidores (equipos de proyectos).	40
4.3.2.6 El Líder de un equipo de consumidores.	41
4.3.2.7 Comité de evaluación de componentes.....	41
4.3.2.8 Otros puntos importantes a considerar.....	41
4.3.3 Director de productores y consumidor para empresas pequeñas.....	42
4.3.4 Director de equipos de desarrollo de software para empresas pequeñas.	43
4.4 Incentivos para promover reuso de software.	
4.4.1 Introducción.....	44

4.4.2 Incentivos para el reusador.	44
4.4.3 Incentivos para el contribuidor.	45

CAPÍTULO 5. INSTALANDO UN PROGRAMA DE REUSO.

5.1 Consideraciones previas al programa.	46
5.2 El Programa de Reuso.	48
5.3 El Programa propuesto.....	49
5.4 Consideraciones para el seguimiento de un programa de reuso.	50

CAPÍTULO 6. MAPEO DE LOS MODELOS EN CASOS REALES DE REUSO DE SOFTWARE.

6.1 Introducción	51
6.2 Instituto Nacional de Capacitación Fiscal (INCAFI).	51
6.3 SINERGY.	55

CAPÍTULO 7. CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS.

7.1 Conclusiones y Recomendaciones.....	60
7.2 Trabajos futuros.....	61

REFERENCIAS.	62
-------------------	----

CAPITULO 1.

INTRODUCCIÓN.

1.1 Antecedentes.

Los dirigentes de las organizaciones demandan sistemas de información cada vez más confiables, es decir, que su realización se lleve a cabo de forma correcta conforme a unos estándares de calidad, y por otra parte, que su desarrollo se realice en el tiempo y costos establecidos.

La situación real en los centros de proceso de datos dista mucho de los deseos de los ejecutivos, en cuanto a la calidad de los sistemas que producen, así como a los tiempos y costos realmente implicados. Todo ello es debido fundamentalmente a la falta de empleo de metodologías, herramientas adecuadas y a los problemas que existen con relación a las estructuras organizacionales. Además, el empleo de nuevas tecnologías requiere de adoptar el uso de nuevas metodologías que implican la necesidad de generar un cambio organizacional, cambio que se hará lo más pronto posible de acuerdo a la capacidad de aprendizaje de la organización [MARCOS, 96].

Así pues, en la actualidad se plantea la necesidad de utilizar metodologías que comprendan todas las etapas del ciclo de vida de sistemas de información que vayan desde análisis de requerimientos hasta la operación y mantenimiento, pasando por las especificaciones, diseño, codificación y pruebas, todo ello ligado al uso de herramientas computacionales avanzadas que permitan automatizar el proceso de desarrollo de sistemas informáticos [McCLURE,92]. Además, a la creación de estructuras humanas de soporte que ayuden a implantar nuevos procesos para el desarrollo de software, donde se busque producir aplicaciones que cumplan con las expectativas de las organizaciones [HUBERT,90],[FAFCHAMPS,94].

Después de varias décadas de discusión, los problemas del software están todavía presentes, los cuales están caracterizados por altos costos y a la falta de desarrollo y mantenimiento de aplicaciones solicitadas en su momento [CÁRDENAS,94]. Este problema puede limitar la postura competitiva de una organización. En ingeniería de procesos de negocios, por ejemplo, el software debe ser modificado rápidamente para reorganizar procesos de negocios a través de fronteras administrativas y responder a oportunidades y cambios del mercado en necesidades del cliente [CANFORA,95],[CIMITILE,92]. Evidentemente, el gran conjunto de aplicaciones sin cumplir en desarrollo y mantenimiento disminuyen la capacidad de una organización para responder a estas demandas competitivas.

Investigadores en ingeniería de software, en mejoramiento de procesos de negocios e ingeniería de información quieren drásticamente modernizar las tecnologías y los procesos el ciclo de vida de desarrollo del software para corregir esta problemática y mejorar la calidad del software [AGRESTI,86],[CANFORA,95],[OROZCO,93]. Dentro de las metas de los investigadores están el mejoramiento de los servicios al usuario a través de la conversión a nuevas plataformas y facilitar los procesos del software adoptando herramientas computacionales para su automatización. Herramientas automatizadas para el desarrollo de software, entendimiento del dominio de las aplicaciones, mantenimiento y documentación, sumados a una maduración del proceso de desarrollo de software, conducirán a una mejor calidad y confiabilidad de servicios computacionales y a una mayor satisfacción del cliente [MARTIN, 94].

Recientemente, los investigadores se han enfocado en lograr las metas anteriores a través de cambios organizacionales y metodologías para reuso de software sistemático [CARD,94],[GRISS,95]. Dichas metodologías están dirigidas a desarrollar bloques constructivos de software reusable (para lograr el reuso de software), los cuales pueden ser adaptados a otra aplicación de software y de esta manera construir las aplicaciones. El Consorcio de Productividad de Software[SPC,93] formalizó el concepto de ingeniería de dominio, un proceso para desarrollar bibliotecas de bloques constructivos de software reusable (componentes de software) para un dominio de aplicación, junto con el concepto de ingeniería de aplicación, un proceso para ensamblar automáticamente bloques constructivos de software reusable y así poder construir una aplicación en un tiempo oportuno a su solicitud [SNEED,87], y a su vez, que cumpla con los requerimientos del cliente.

Sin embargo, moverse de una práctica de software presente -dominada por grandes y complejas aplicaciones funcionando en el presente las cuales tienen un conjunto grande de solicitudes sin cumplir para desarrollo y mantenimiento- a un futuro basado en el reuso de software y a la generación automática de programas ha probado ser bastante difícil [BOYLE,84], [CIMITILE,92],[GARLAN, 94],[KRISHNAMURTHY,94].

1.2 Justificación.

El desarrollo de sistemas de información se ve sometido actualmente a grandes exigencias en cuanto a productividad y calidad, debido a la importancia creciente que adquiere la información como activo principal de las empresas y organismos, que puede suponer su principal ventaja estratégica.

La calidad en los productos de software es una necesidad creciente de todo tipo de usuarios de los mismos, y por tanto es un factor de competitividad de las empresas que los crean, ya que han de satisfacer las necesidades de sus clientes no sólo para continuar en el mercado, sino, además, para conseguir la superioridad y el liderazgo como una meta empresarial.

Sin embargo, no sólo los creadores de software han de satisfacer los factores de calidad que les demanda el mercado, también los usuarios del software, que crean, a su vez, nuevos productos y servicios, tanto para uso interno como externo a su organización, tienen la necesidad de cumplir los requisitos de calidad que les son exigidos. Hay, pues, una cadena de calidad que hay que vigilar, ya que, como toda cadena, siempre se rompe por el eslabón más débil.

Ganar la batalla de la calidad del software hoy, y en un próximo futuro, pasa por alcanzar la comprensión y el dominio de los procesos constructivos, por la implicación de los usuarios, por la gestión rigurosa de los procesos y por la tecnología de apoyo, donde se incluye genéricamente a las herramientas de ingeniería de software asistidas por computadora (conocidas como las herramientas CASE: Computer Aided/Assisted Software/System Engineering).

La complejidad de los sistemas de software es cada día mayor, y se está mostrando como un factor esencial de la calidad a las industrias y servicios que los producen y operan.

Dicho factor precisa dominarse a partir de métodos potentes de análisis y diseño, que han de verse apoyados por herramientas automáticas que faciliten su implantación. De otra manera, existen identificados un conjunto de criterios de calidad aplicables a todo tipo de software: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad que han sido aceptados por ISO(International Standard Organization).

Además, con relación a los problemas de productividad, a pesar de que la productividad del software se ha estado incrementando constantemente sobre los pasados 30 años, ésta no ha sido suficiente para cerrar la brecha entre la demanda colocada en la industria del software (por una sociedad que es cada vez más dependiente del software y cada vez menos tolerante en las fallas del software) y lo que el estado de la práctica de hacer software puede entregar [BOEHM,87],[COX,90]. Por lo tanto, se tiene actualmente la necesidad de poder producir software lo más rápido posible de tal manera que el esfuerzo para conseguirlo sea cada vez menor, y por consecuencia, poder entregar soluciones oportunas a las empresas u organizaciones.

El reuso de software proporciona algunos remedios factibles a la problemática del software [LIM, 94], pero mucho se ha cuestionado acerca de las prácticas administrativas de proyectos de software existentes, de las estructuras organizacionales y de las tecnologías para soportar dicho reuso. Hay un acuerdo general de la necesidad de un replanteamiento necesitado en la manufactura del software [BASILI,94], [COX,90], [CUSUMANO,89]. Existe también un acuerdo en que los cambios requeridos son administrativos, de cultura y de naturaleza técnica, como lo fue el caso de otras disciplinas de la ingeniería [BASILI, 91],[COX,90],[GRISS,95].

Debido a que el reuso enlaza proyectos previamente independientes, decisiones que fueron una vez de interés para un único proyecto, ahora involucra varios proyectos

[PRIETO-DIAZ,87], [SIMOS,90]. Decisiones tales como el uso de un proceso de software común, herramientas y lenguajes, así como inversiones a largo plazo en entrenamiento, construcción y mantenimiento de bibliotecas de software reusable y creación de una estructura de soporte de reuso, son elementos necesarios para lograr un proceso de adopción de reuso incremental [FRAKES,90].

Para lograr que las empresas u organizaciones consigan llevar a cabo un proceso de construcción de productos de software basado en el reuso, se llevó a cabo este trabajo el cual genera un valor agregado para los equipos de desarrollo de software, los cuales son la base del éxito de los sistemas de información computacionales.

1.3 Definición del objetivo de la tesis.

El objetivo de esta tesis es establecer las características que debe tener una organización flexible de desarrollo de software la cual pueda generar los cambios necesarios en su procesos para satisfacer los requerimientos en las aplicaciones de clientes y a la vez diseñar modelos que describan estructuras organizacionales de soporte, incorporando fundamentos de diseño organizacional y cómo pueden ser aplicados al reuso de software. Además, proponer un programa de reuso de software para conseguir la adopción del reuso de una manera incremental.

1.4 Valor agregado.

Se necesitan desarrollar y actualizar sistemas de información que cumplan con las expectativas de las organizaciones, y para ésto, los departamentos de desarrollo de software deben tener la capacidad de administrar sus procesos de construcción de software y crear productos de software con mayor calidad y a su vez tener la mayor productividad posible.

Una alternativa es por medio de implementar un proceso de construcción de software basado en el reuso, de tal manera que se institucionalice esta nueva forma de trabajar. Así, comenzar una cultura enfocada en la reusabilidad de componentes de software (bloques constructivos de software reusable), crear bibliotecas de software reusable, construir aplicaciones a partir de dichas bibliotecas, poder estructurar equipos de proyectos de software donde se definen funciones de reuso y establecer la comunicación entre estos equipos llevará a poder implementar un reuso sistemático.

1.5 Metodología.

El enfoque del presente estudio fue de carácter exploratorio y de revisión de casos, en el que nos enfocamos a clarificar y a describir cuáles son los factores organizacionales que impactan en la implantación de un programa de reuso de software, y a determinar las acciones que conducen a una adopción del paradigma de reuso de software.

Dentro del estudio se llevaron a cabo análisis de documentos donde se presentaran casos de utilización del paradigma de reuso, se utilizaron entrevistas y la observación-acción de grupos de desarrollo de software.

1.6 Organización del contenido.

El capítulo dos se concreta a dar el soporte teórico de todo lo que sustenta el trabajo de tesis, poniendo el marco de referencia de la ingeniería de software y definiendo los conceptos necesarios sobre el reuso de software. Dentro de este capítulo se dedica espacio a los conceptos del ciclo de vida para el desarrollo de software el cual es de gran importancia para poder definir los procesos de construcción de software basados en el reuso. Además, no menos importante es estudiar lo que se ha dicho sobre calidad y productividad en el software, ya que son dos elementos que intentan alcanzarse con la introducción del paradigma del reuso en los procesos de construcción de productos de software.

En el capítulo tres comienza la adaptación del trabajo de tesis, ya que presenta la necesidad de superar ciertos problemas de coordinación entre las funciones de los diferentes departamentos. Estos problemas están inmersos en la organización y deben ser resueltos con un efectivo diseño de la misma. Como las organizaciones de interés en esta tesis son aquellas que tengan que desarrollar software, se presentan las características que debe poseer una organización flexible de desarrollo de software para poder superar los problemas de coordinación de funciones y la importancia del diseño de la organización en el reuso de software.

En el cuarto capítulo se propone un modelo general que describe los elementos más importantes a considerar para instalar una estructura organizacional de soporte para el reuso de software, y que además ayudan a contener las características de una organización flexible de desarrollo de software. A lo largo del capítulo se identifican algunas variantes que pueden considerarse en el modelo, derivándose otras posibles estructuras organizacionales.

En el capítulo quinto se presenta un estudio realizado por el autor de la tesis con relación a un programa de reuso, poniendo en claro las consideraciones previas a un programa y la planeación y puesta en marcha del mismo. Se hace énfasis en identificar objetivos organizacionales y oportunidades de reuso, los cuales deberán ser considerados para un buen programa.

El capítulo seis presenta el mapeo de los modelos propuestos en casos reales de reuso de software. En este capítulo se describen dos empresas dedicadas al desarrollo de software las cuales están aplicando el reuso de software en sus procesos de definición y construcción de productos. El estudio de estos casos fue relevante porque ayudaron a la investigación a identificar las características de una organización flexible de desarrollo de software y sus elementos primordiales.

En el capítulo siete se hacen conclusiones y recomendaciones al lector, así como de una propuesta para trabajos futuros tomando como base el contenido de esta tesis.

CAPÍTULO 2. INGENIERÍA DE SOFTWARE Y REUSO DE SOFTWARE.

2.1 ¿Qué es Ingeniería de Software?

El campo de la ingeniería de software abarca todas las actividades involucradas en la solución de problemas a través del desarrollo de sistemas de computadora. Todas estas actividades implican el establecimiento y uso de principios sólidos de ingeniería, los cuales están orientados a obtener económicamente software que sea fiable y funcione eficientemente sobre máquinas reales [PRESSMAN,89]. Todo este conjunto de actividades son llevadas a cabo dentro de un proceso de desarrollo de software. Además, la ingeniería de software es la administración de expectativas, tecnologías de cómputo, personas y sus habilidades, tiempo y costos para crear un producto de software que cumpla las expectativas del cliente -quien utilizará el software- con un proceso que cumpla las expectativas del productor del software [MARTIN,94].

De acuerdo con Roger S. Pressman, la ingeniería de software abarca un conjunto de tres elementos claves -métodos, herramientas y procedimientos- que facilitan al constructor de sistemas de computadora controlar el proceso de desarrollo de software y suministrar a los que practiquen dicha ingeniería las bases para construir software de alta calidad de una manera productiva.

Métodos. Suministran el *cómo* construir técnicamente el software. Abarcan un amplio espectro de tareas que incluyen: planificación y estimación de proyectos; análisis de los requerimientos del sistema y del software; diseño de estructuras de datos, arquitectura de programas y procedimientos algorítmicos; codificación; prueba y mantenimiento. Estos métodos introducen un conjunto de criterios para la calidad del software.

■ *Herramientas de Ingeniería de software.* Suministran un soporte automático o semiautomático para los métodos. Hoy, existen herramientas para soportar cada uno de los métodos mencionados anteriormente. Cuando se integran las herramientas de forma que la información creada por una herramienta pueda ser usada por otra, se establece un sistema para el soporte del desarrollo del software, llamado ingeniería de software/sistemas asistido por computadora (CASE: acrónimo en inglés de Computer Aided Software/System Engineering). CASE combina el software, hardware y bases de datos de la ingeniería del software para crear un entorno de ingeniería del software, el cual ayuda al desarrollador en las diferentes etapas del proceso de desarrollo.

Procedimientos. Son el pegamento que mantiene a los métodos y a las herramientas unidas y facilitan un desarrollo racional y oportuno del software de computadora. Los procedimientos definen la secuencia en la que se aplican los métodos, las entregas (documentos, informes, formas, etc.) que se requieren, los controles que ayudan a

asegurar la calidad y coordinar los cambios, y las guías que facilitan a los gestores del software establecer su desarrollo.

La Ingeniería de software está compuesta de pasos que abarcan los métodos, herramientas y procedimientos tratados anteriormente. Estos pasos se denominan frecuentemente paradigmas de la ingeniería de software. Un paradigma para la ingeniería del software se elige basándose en la naturaleza del proyecto y de la aplicación, los métodos y herramientas a usar y los controles y entregas requeridos.

En lo siguiente, se describe brevemente el paradigma del ciclo de vida clásico para el desarrollo del software según la visión de Roger S. Pressman.

2.1.1 El ciclo de vida del desarrollo de software en modelo de cascada (ciclo de vida clásico).

Este modelo exige un enfoque sistemático, secuencial, del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento. Este paradigma abarca las siguientes actividades:

Ingeniería y análisis del sistema. Debido a que el software siempre es parte de un sistema mayor, el trabajo comienza estableciendo los requerimientos de todos los elementos del sistema y luego asignado algún subconjunto de estos requerimientos al software. Esta visión del sistema es esencial cuando el software debe interrelacionarse con otros elementos tales como hardware, personas y bases de datos. Aquí se abarcan los requerimientos globales a nivel sistema con una pequeña cantidad de análisis y diseño a nivel superior.

Análisis de los requerimientos del software. El proceso de recolección de los requerimientos se centra e intensifica especialmente en el software. Para comprender la naturaleza de los programas que hay que construir, el analista debe comprender el dominio de la información del software, así como la función, rendimiento e interfaces requeridas. Los requerimientos tanto del sistema como del software se documentan y revisan con el cliente.

Diseño. Es un proceso multipasos que se enfoca sobre cuatro atributos distintos del programa: estructura de datos, arquitectura del software, detalle de los procedimientos y caracterización de las interfaces. El proceso de diseño traduce los requerimientos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación. Como los requerimientos, el diseño se documenta y forma parte de la configuración del software.

Codificación. El diseño se traduce a una forma legible por la máquina, por ejemplo, haciendo uso de lenguajes de programación como de tercera o cuarta generación.

Prueba. Una vez que se ha generado el código, comienza la prueba del programa. La prueba se enfoca sobre la lógica interna del software, asegurando que todas las

sentencias se han probado, y sobre las funciones externas, esto es, realizando pruebas para asegurar que la entrada definida producirá los resultados que realmente se requieren.

Mantenimiento. El software necesitará cambios después de que se entregue al cliente. Los cambios ocurrirán debido a que se han encontrado errores, debido a que el software debe adaptarse por cambios en el entorno externo (por ejemplo, un cambio solicitado debido a que se tiene un nuevo sistema operativo o dispositivo periférico), o debido a que el cliente requiere aumentos funcionales o del rendimiento del sistema. El mantenimiento se puede aplicar a cada una de las etapas anteriores del ciclo de vida del desarrollo.

Este ciclo de vida de desarrollo, llamado el paradigma clásico del ciclo de vida, tiene un lugar definido e importante en el trabajo sobre ingeniería de software. Suministra una base en la que pueden colocarse los métodos para el análisis, diseño, codificación, prueba y mantenimiento.

2.1.2 Visión genérica de la ingeniería del software de acuerdo con Roger S. Pressman.

Independientemente del paradigma de ingeniería elegido, el proceso de desarrollo del software contiene tres fases genéricas. Las tres fases, definición, desarrollo y mantenimiento, se encuentran en todos los desarrollos de software, independientemente del área de aplicación, tamaño del proyecto o complejidad.

La *fase de definición* se enfoca en el *qué*. Esto es, durante la definición, el que desarrolla el software intenta identificar qué información ha de ser procesada, qué funciones y rendimientos se desea, qué interfaces han de establecerse, qué ligaduras de diseño existen y qué criterios de validación se necesitan para definir un sistema correcto. Por tanto, han de identificarse los requerimientos claves del sistema y del software. En esta fase se especifican tres pasos:

Análisis del sistema. Aunque descrito ya en el paradigma anterior, el análisis del sistema define el papel de cada elemento de un sistema informático, asignando finalmente el papel que jugará el software.

Planificación del proyecto de software. Una vez que el alcance del software es establecido, los riesgos son analizados, se asignan los recursos, se estimen los costos y se definen las tareas y la planificación del trabajo.

Análisis de requerimientos. El alcance definido para el software proporciona la dirección, pero antes de comenzar a trabajar, es necesario de disponer de una información más detallada del dominio de la información y de la función del software.

La *fase de desarrollo* se enfoca en el *cómo*. Esto es, durante el desarrollo, el que desarrolla el software intenta descubrir cómo han de diseñarse las estructuras de

datos y la arquitectura del software, cómo han de implementarse los detalles procedimentales, cómo ha de trasladarse el diseño a un lenguaje de programación (o lenguaje no procedimental) y cómo ha de realizarse la prueba. Tres pasos siempre deben ocurrir en esta fase:

Diseño del software. Se trasladan los requerimientos del software a un conjunto de representaciones que describen la estructura de datos, arquitectura y procedimiento algorítmico.

Codificación. Para lograr la codificación, las representaciones del diseño deben trasladarse a un lenguaje artificial (lenguaje de programación) que da como resultado unas instrucciones ejecutables por la computadora.

Prueba del software. El código ejecutable por la máquina debe ser probado para descubrir los defectos que puedan existir en la función, la lógica y la implementación.

La *fase de mantenimiento* se enfoca sobre el cambio que va asociado con una corrección de errores, adaptaciones requeridas por la evolución del entorno del software y modificaciones debidas a los cambios a los requerimientos del cliente para reforzar o aumentar el sistema. La fase de mantenimiento reaplica los pasos de las fases de definición y desarrollo, pero en el contexto del software existente. Durante esta fase se encuentran tres tipos de cambios:

Corrección. Corregir defectos encontrados.

Adaptación. Con el paso del tiempo es posible que cambie el entorno original (por ejemplo, CPU, el sistema operativo, periféricos) para el cual se desarrolló el software. El mantenimiento adaptativo se traduce en modificación del software para acomodarlo a los cambios de su entorno externo.

Aumento. Conforme se utiliza el software, el cliente/usuario conocerá funciones adicionales que podría ser beneficioso añadirlas. El mantenimiento perfectivo aumenta el software más allá de sus requerimientos funcionales originales.

Las fases y pasos relacionados descritos en la visión genérica de la ingeniería del software se complementan con varias actividades protectoras. Las revisiones se realizan durante cada paso para asegurar que mantienen la calidad. La documentación se desarrolla y controla para asegurar que toda la información del sistema y del software estará disponible para un uso posterior. El control de los cambios se instituye de forma que los cambios puedan ser mejorados y registrados.

2.2 ¿Qué es un proyecto de desarrollo de software exitoso?

Para esta sección se tomaron en cuenta opiniones de James Martin y Roger Pressman quienes han analizado la importancia de un proyecto de desarrollo exitoso.

La meta de un proyecto exitoso es producir un producto aceptable. Para entender mejor esto, definiremos algunos conceptos.

Un producto de software es un sistema de programas de computadora y todo aquello necesitado para correrlos en una máquina. El producto incluye a los programas mismos y generalmente la documentación necesitada para usarlos y cambiarlos. Podría también incluir entrenamiento y conversión de datos para ayudar al cliente a hacer los cambios de el sistema viejo a el nuevo, y servicios de mantenimiento y consulta para ayudarlo a continuar ejecutando al sistema exitosamente.

Un proyecto de desarrollo de software es un esfuerzo que resulta de la promesa del productor para crear un producto de software, que en algunos casos se obtiene el pago económico por parte del cliente/usuario. El cliente y el productor acuerdan expectativas de lo que deberá hacer el producto de software, qué tan bien lo hará, lo que el cliente necesitará para usarlo, el costo económico del producto y cuando estará disponible para ser utilizado (cuando será liberado, y por lo general está asociado a un número de versión).

El productor empieza revisando su conocimiento de sistemas similares que ya haya hecho antes, las capacidades de su personal y la confianza que tiene en ellos, una imagen de lo que va a ser producido y un plan mostrando cómo hacerlo.

Un proyecto es normalmente llevado a cabo por un equipo de personas que se reúne para producir dicho producto en particular. Este equipo de proyecto usualmente forma parte de una organización más grande, tal como una compañía de desarrollo de productos de software o de un corporativo que genera sus propias aplicaciones. Dentro de estas organizaciones más grandes, conforme algunos proyectos se concluyen y los equipos se disuelven, otros proyectos se están formando. La gente se mueve de un proyecto a otro.

Un proyecto de desarrollo de software exitoso, por lo tanto, entrega un producto de software dentro de los límites de tiempo y costo estimado cumpliendo con las expectativas que se fijaron entre cliente y productor.

2.3 ¿Qué hace un ingeniero de software?

También llamado el analista de sistemas, es quien integra las varias tecnologías de las ciencias computacionales y las habilidades de las personas para construir un sistema de cómputo y así satisfacer las expectativas definidas, como qué tan bien el producto cumple las necesidades del cliente, cuánto cuesta y cuando podrá ser entregado para su uso [McCLURE, 92]. Esto implica construir un producto de alta calidad usando un

proceso de alta productividad que controle la calidad en cada paso del proceso de desarrollo.

2.4 ¿Cuáles son las características de un proyecto de Ingeniería de Software?

Estas características son un compendio de textos publicados por los autores James Martin, Carma McClure y Ivar Jacobson.

- El proyecto está concernido con hacer un cambio a un sistema existente o construir un nuevo sistema. En ambos casos se dirá del resultado como el producto.
- El cambio no es fácil de deshacer o modificar una vez que éste es hecho. El costo de corregir errores es alto. Por lo tanto uno debe visualizar el cambio cuidadosamente antes de hacerlo.
- El proyecto no puede ser hecho por una sola persona y usualmente debe ser hecho con un equipo de gentes con diferentes habilidades. Así, se debe estar interesado en la administración de la gente y de la comunicación entre ellos.
- El producto debe trabajar con otros sistemas o el trabajo que produce el producto debe ser coordinado con el trabajo de otras personas. Así, el cómo y cuando el proyecto es hecho debe ser planeado para ajustar restricciones.
- El proyecto involucra dos culturas: el cliente quien desea el cambio y la gente quien posee la tecnología para hacer el cambio. Ellos pueden tener diferentes conocimientos previos y pueden platicar diferentes lenguajes técnicos. Por lo tanto, la comunicación entre estos dos conjuntos de personas puede ser difícil, pero a pesar de eso es muy importante.
- Los recursos necesarios deben estar comprometidos con el proyecto y los proveedores o clientes pueden expresar inquietud para saber si el valor del producto vale la pena el costo de comprometer estos recursos.
- El producto es nuevo. Nunca ha sido hecho antes. Así, no es hasta el final del proyecto que uno sabe exactamente lo que el producto es, cómo se comporta, lo que vale y lo que cuesta hacerlo. Ingeniería del software y sus clientes desearían tener toda esta información antes de que ellos decidan el inicio del proyecto. Pero ellos nunca lo hacen, así que deben ofrecer estimaciones y usarlas para tomar decisiones antes de que esta información llegue a estar disponible.
- Los clientes se comprometen a comprar un sistema que nunca han visto. Por lo tanto ellos solicitan una buena visualización o descripción de lo que ellos obtendrán antes de que estén comprometidos.
- Empezar un proyecto es una empresa arriesgada. Puede fallar.

2.5 Interés por la Calidad.

Un proyecto falla cuando no se entrega un producto de calidad. No importa qué tan duro cada quien trabajó, si el software no realiza lo que se acordó, el proyecto no

puede ser considerado como un éxito. Cuanto más conozcamos de lo que se constituye la calidad y cómo cada miembro de equipo contribuye a dicha calidad, cuanto más capaces seremos para crear el mejor producto posible [PIATTINI, 95].

Para poder especificar la calidad en los productos de software debemos basarnos en ciertos factores que nos ayuden a evaluarla, y así saber si ésta está presente en los productos que se liberan (listos para ser usados por el cliente) [MARTIN, 91]. Estos factores deberían ser especificados cuando el cliente y el desarrollador contraen hacer el proyecto. Los factores deben ser considerados durante todos los pasos del proyecto y ser evaluados cuando el producto está liberado. Los factores de calidad de acuerdo con los estudios de Roger Pressman, Rubén Prieto-Díaz y James Martin son:

- *Compleitud.* ¿El producto hace todo lo que las especificaciones dicen que debería hacer ?
- *Corrección.* ¿El producto hace su trabajo correctamente como está especificado ?
- *Confiabilidad.* ¿Hasta cuándo el producto continuará haciendo lo que las especificaciones dicen que debería hacer, especialmente cuando sea expuesto a nuevas entradas y nuevos entornos que las especificaciones dicen que el sistema debe ser capaz de manejar ?
- *Usabilidad.* ¿Qué tan fácil es para el usuario/cliente operar el sistema sin cometer errores, o alternatively, qué tan buena es la confiabilidad del sistema cuando el usuario comete errores?
- *Entendimiento/Simplicidad.* ¿Qué tan fácil puede alguien entender lo que el producto hace, cómo trabaja, y cómo controlarlo y usarlo ?
- *Robustez.* ¿Cómo el sistema sobrevive a malas entradas, identificándolas y ayudando con su corrección?
- *Eficiencia.* ¿Qué tan eficazmente el sistema usa recursos para manejar problemas como está especificado?
- *Capacidad de Supervivencia.* ¿Qué tan bien puede el producto ejecutar sus funciones cuando su entorno (por ejemplo, el hardware) se ha deteriorado?
- *Integridad.* ¿Qué tan bien el sistema se protege contra actos hostiles o acceso a personas no autorizadas?
- *Verificabilidad.* ¿Qué tan fácil es verificar que el sistema está haciendo correctamente lo que se supone debe hacer?
- *Capacidad de Mantenimiento.* ¿Qué tan barato es arreglar o modificar el producto?
- *Flexibilidad.* ¿Qué tan barato es hacer modificaciones al sistema de tal forma que haga algo no originalmente anticipado en los requerimientos?
- *Expandibilidad.* ¿Qué tan fácil es expandir las capacidades del sistema?
- *Portabilidad.* ¿Qué tan fácil es poner al sistema a operar en otro entorno (por ejemplo, sobre diferente hardware o diferente sistema operativo)?
- *Interoperabilidad.* ¿Qué tan fácil es conectar al producto con otros sistemas?
- *Reusabilidad.* ¿Qué tan fácil sería usar el producto o parte de él en otros sistemas?

Estos factores no son independientes. Por ejemplo, si un sistema es mantenible, es probable que sea flexible y extendible. Si necesito mayor eficiencia puedo sacrificar la capacidad del buen mantenimiento.

2.6 Productividad.

El concepto de productividad es muy importante en el desarrollo de proyectos de software, ya que implica conocer la proporción que existe entre los recursos que se invierten en un proyecto y el grado de avance que se obtiene en los mismos.

El desarrollo de software es una tarea compleja que incorpora varias fases y es un proceso que consume tiempo [RAO, 93]. La capacidad para poder estimar el tiempo y los costos de un proyecto de software es una necesidad que buscan todos los productores de software, además de considerar la calidad en los productos [PIATTINI, 95].

El concepto de productividad que se tiene dentro de la ingeniería de software es el mismo que maneja cualquier productor de bienes y servicios. La productividad es la cantidad producida teniendo en cuenta el trabajo efectuado y el capital invertido, además de que se busque la calidad en dicha cantidad producida (Diccionario Larousse edición 1980). Con lo anterior, podemos decir que un producto de software debe ser producido dentro un proceso de desarrollo que considere tiempos, costos y calidad en cada fase del proceso. Así, la productividad pretende lograr la liberación de productos dentro de tiempos y costos acordados con la calidad esperada por el cliente/usuario [MARTIN, 91].

También, productividad es crear y modificar aplicaciones en forma mucho más rápido, es decir, ahorrar tiempo en los procesos de construcción del software y poder actuar oportunamente a las necesidades del mercado sin implicar costos altos, los cuales podrían ser el aumentar el número de personas asignadas a un proyecto, pasarse de las fechas programadas del proyecto y mayor tiempo en el uso del equipo de desarrollo para dicho proyecto [MARTIN, 91].

2.7 El Reuso de Software.

2.7.1 Definición de reuso de software.

El término de reuso de software es susceptible a varias definiciones. La definición más apropiada para este trabajo es el uso de texto fuente escrito en algún lenguaje de programación o código compilado en un contexto diferente de aquel para el cual fue originalmente escrito [TRACZ, 94]. Si nos restringimos a el reuso de código fuente y de código máquina, sin embargo, ciertamente no cosecharíamos todos los beneficios potenciales del reuso; especificaciones de requerimientos, diseños, planes de prueba y muchos otros productos generados durante el proceso de desarrollo son capaces de ser reusados [LANERGAN, 84],[LUBARS, 91].

Existen dudas si el término de reuso de software debiera ser interpretado como incluir reuso con modificación o si debiera ser restringido a cubrir únicamente reuso sin modificación. Si se permite la modificación, entonces las ocasiones en las cuales se pueda practicar el reuso serán mayores porque hace posible la adaptación del software [TRACZ, 94]. Si no se permite la modificación se gana la ventaja de la confiabilidad por hacer uso de código ya probado [COX, 90]. En este trabajo, el término de reuso de software será interpretado como incluir el reuso con modificación, ya que es la forma de poder adaptar software al nuevo sistema.

Dentro del campo del reuso de software existen conceptos importantes por definir [TRACZ, 94]:

Reusabilidad : es una propiedad o atributo que posee el software construido para soportar el reuso. Si consideramos módulos de software con la característica de reusabilidad es porque están bien diseñados y han sido probados, en varios lugares, en diferentes aplicaciones, para minimizar el desarrollo del nuevo código.

Software reusable (componente reusable): es una representación de algún aspecto (diseños, módulos de código fuente, módulos de código objeto, aplicaciones completas) de un sistema que puede ser usado (con modificación si es necesario) en diferentes aplicaciones.

Reusabilidad del software : es el grado a el cual el software puede ser reusado por diferentes aplicaciones.

Reuso de software : es la actividad de incorporar software reusable (que quizá fue diseñado para ser reusado) dentro del proceso de construcción de software.

2.7.2 Componentes de software.

Para definir los conceptos de toda la sección 2.7.2 se utilizó la visión de Roger S. Pressman [PRESSMAN, 89] y de Ivar Jacobson [JACOBSON, 92].

Un componente de software es un elemento que debe ser incluido dentro de un todo (aplicación, módulo del sistema, etc.) para desempeñar una función. Dichos elementos tienen su origen de dos formas diferentes: cuando son diseñados con anticipación con la finalidad de ser reusados en distintas aplicaciones, y cuando surgen del desarrollo de otros proyectos, donde algún programador los extrae (usando algún método de ingeniería de software) y los aísla para que desempeñen únicamente su función como componentes. Lo que es común entre los dos es su capacidad de reusabilidad, donde a cada componente se le acompaña con la documentación indispensable para su futura utilización (como por ejemplo, entender qué hace, cómo lo hace, su ambiente de ejecución).

El software de computadora es información que existe en dos formas básicas: componentes no ejecutables por la máquina y componentes ejecutables por la

máquina. Esta información forma parte de la base de los activos de software reusables de la organización de desarrollo, los cuales se deben ir incrementando para ir adoptando una cultura de reuso.

2.7.2.1 Componentes ejecutables por la máquina.

Componentes de software ejecutables por la máquina son creados a través de una serie de traducciones que mapean requerimientos de cliente/usuario a código ejecutado por la computadora. Un modelo de requerimientos (o prototipo) es traducido a un diseño. El diseño del software es traducido a un lenguaje de programación que especifica los estructuras de datos del software, los atributos procedimentales y los requerimientos relacionados. Un traductor procesa este lenguaje convirtiéndolo el código anterior en instrucciones ejecutables por la máquina.

Con los nuevos entornos de desarrollo de software, en especial el concepto CASE (Computer Aided Software/System Engineering) son una herramienta que hacen posible la reutilización de requerimientos, diseños y de código fuente de otros proyectos, los cuales con ciertas modificaciones llevan a una generación automática de código máquina para la construcción de las nuevas aplicaciones.

Dentro de esta gama de software reutilizable que puede ser interpretado por la máquina y generar código máquina está la construcción de componentes de código fuente reusables de software, los cuales están escritos en algún lenguaje de programación. Cada componente es una unidad constructiva estándar que es usada para desarrollar aplicaciones. La idea es semejante a tener bibliotecas de funciones y procedimientos ya probados por el proveedor de algún lenguaje de programación, donde es posible utilizar el material de dichas bibliotecas para la producción de software más complejo. En los años 60's, se construyeron bibliotecas de subrutinas científicas que fueron reusables en un arreglo amplio de aplicaciones de ingeniería y en lo científico. Estas bibliotecas de subrutinas reusaron algoritmos bien definidos de una manera efectiva, pero tenían un dominio limitado de aplicación.

Los componentes son segmentos de código fuente que encapsulan algoritmos y estructuras de datos, y forman un concepto más abstracto que puede ser incorporado en otra aplicación. El desarrollador tiene la posibilidad de adaptar dicho componente para lograr que se comporte como se necesita. Por eso, es necesario considerar el reuso de software con modificación. Un componente reusable ejecutable por una máquina de los 90's encapsula tanto datos como procesamiento dentro de un único paquete (a menudo llamado clase u objeto), habilitando al ingeniero de software a crear nuevas aplicaciones de las partes reusables.

Para hacer un componente reusable en varias aplicaciones es necesario que éste sea independiente de la aplicación para la cual fue diseñado, pero no siempre es necesario que se cumpla esto; algunas veces habrá componentes que sean dependientes de la aplicación pero con un interés a ser reusados. Por lo anterior, podemos clasificar los componentes dentro de una escala que fluctue desde completamente independientes de la aplicación hasta completamente dependientes.

Cuanto más sea dependiente el componente de una cierta aplicación, cuanto más seguido se debe adaptar para usarlo.

Los requerimientos impuestos en un componente son mucho mayores que sobre cualquier software ordinario (requerimientos, diseños). Usar un componente significa que podemos ahorrar tiempo puesto que no se necesita saber cómo trabaja por dentro (en algunos casos). Únicamente se necesita saber usar. Un componente complejo que es fácil de usar eleva el nivel de abstracción para el desarrollador. Éste entonces forma una simplificación conceptual de lo que está implementado.

Como el componente debe ser usado en varios contextos, se necesita que éste sea una abstracción general para que pueda ser usado ampliamente. Esto implica que el componente esté diseñado para ser reusado. Pero un componente que nace de una aplicación en particular se puede mejorar o reprogramar para llegar a tener alto grado de reusabilidad, consiguiéndose la mencionada abstracción general.

Como un ejemplo se tienen las interfases interactivas de hoy son a menudo construidas usando componentes reusables que facilitan la creación de ventanas gráficas, menus pull-down, y una gran variedad de mecanismos de interacción. Las estructuras de datos y el detalle de procesamiento requerido para construir la interfase están contenidos dentro de una biblioteca de componentes reusables para la construcción de la interfase.

2.7.2.2 Componentes no ejecutables por la máquina.

Componentes de software no ejecutables por la máquina son toda aquella documentación que se va generando en el proceso de desarrollo: métodos de desarrollo, requerimientos del software/sistema, especificaciones, análisis de sistemas y diseños (de programas, de flujo de datos, de secuencia de tareas, de bases de datos, etc.). Toda esta información puede ser reutilizada para idear y construir una nueva aplicación.

2.7.3 Componentes ejecutables como un mecanismo de refuerzo según Pressman.

Cuando trabajamos con componentes de software, debemos verlos como parte del lenguaje de programación. Del mismo modo como tenemos funciones, procedimientos, variables, etc. en las bibliotecas del lenguaje de programación (las primitivas del lenguaje), se tienen primitivas en un nivel más alto de abstracción (los componentes reusables en las bibliotecas de reuso). De esta manera, los componentes constituyen un mecanismo de refuerzo para el lenguaje de implementación; en vez de trabajar con estructuras primitivas, trabajamos con estructuras de abstracciones de mayor nivel, como ventanas gráficas y menus pull-down. De esta manera podemos tomar ventaja de empezar a desarrollar a partir de estructuras más complejas. Esto implica que estos componentes están escondidos en el diseño, esto es, no los incluimos en los diagramas de diseño, diagramas de interacción, etc. El uso de ellos es encapsulado y se les puede llamar objetos.

Típicamente varias aplicaciones comparten un conjunto común de dichos objetos de aplicación reusables. Construir aplicaciones usando dichos objetos de aplicación es un factor clave para mejorar la productividad.

Con relación al tema, Ivar Jacobson afirma que por medio de componentes se espera dominar el diseño de un sistema de cualquier tamaño. Hay dos importantes razones para considerar a los componentes como una gran ayuda: reducir el tiempo de desarrollo (lográndose el ahorro de costos) y elevando la calidad (por utilizar software ya probado). Se reduce el tiempo de desarrollo a medida que tenemos componentes más poderosos como una base de desarrollo. Esto significa que reducimos complejidad y por lo tanto tenemos que escribir menos código. En la medida en que se usan componentes completos significa que ellos están bien probados, tanto por el tiempo que se invirtió en su creación como porque ellos han sido usados en otras aplicaciones.

2.7.4 Ventajas de reusar código fuente considerando la visión de Bindu R. Rao.

La mejor manera de desarrollar software es a base de reutilizar software ya probado. Se debería reusar software antes que desarrollar nuevo código. La programación es un proceso cargado de posibles errores. Aun los mejores programas escritos por un programador experto pueden tener unos cuantos errores. Esto es el por qué el desarrollo de software nunca está terminado sin llevar a cabo fases de pruebas extensas. Por lo tanto, cuanto más nuevo código se desarrolle, cuanto mayor llega a ser la necesidad para aplicar procedimientos apropiados de pruebas. Como un corolario de lo anterior, cuanto más código (código probado) se reuse, cuanto menor serán los errores introducidos y menores serán los costos totales del desarrollo de software.

2.7.5 Bibliotecas de software reusable.

De acuerdo con la visión de Ivar Jacobson y la ayuda de otros autores se va a describir el tema de las bibliotecas de software reusable..

Reusabilidad de código es posible si únicamente hay una forma eficiente para almacenar código probado en alguna forma de biblioteca de componentes reusables, y si existe alguna forma de acceder dicha biblioteca fácilmente y eficientemente [RAO, 93],[MARTIN, 94].

La información que puede almacenar debe cumplir con estándares mínimos autorizados para su posible utilización [PRIETO-DIAZ, 87]. Esta información incluye: componentes de código fuente, componentes de código ejecutable por la máquina (código máquina), módulos completos de software, productos completos de software, documentación de diseños, y en general, toda información producida durante cada etapa del proceso de desarrollo de software. La clasificación que se le dé a esta información es muy importante para su búsqueda eficiente [ROBSON, 92] y evitar un largo consumo de tiempo, además de que la documentación ayude al desarrollador a comprender rápidamente lo que necesita.

La construcción y evolución de una biblioteca de software reusable es una herramienta necesaria para llevar a cabo los servicios de reuso:

- Creación de nuevos componentes.
- Seguimiento de versiones de componentes.
- La búsqueda de componentes según necesidades.
- La prueba de componentes.
- Modificación de nuevos componentes.
- La clasificación y adición de nuevos componentes.

Alguien debería ser responsable por asegurarse de que la información de la biblioteca esté disponible y se extienda por toda la organización de desarrollo, y que todo el software que contiene esté accesible.

Los componentes de la biblioteca deberán preferentemente estar compartidos entre varios diferentes productos. Esto significa que la biblioteca deberá servir a varios proyectos. Las propuestas para nuevos componentes normalmente vienen de proyectos que la biblioteca soporta. De esta manera se pueden construir componentes más complejos a partir de los ya construidos.

Los proyectos deben poner a consideración componentes que podrían tener un potencial de reuso, así como de aquellos componentes generados por el desarrollo de algún producto. En ambos casos se requiere de documentación para su posterior revisión [ROBSON, 92]. Dichos componentes deben ser revisados por un grupo de diseñadores experimentados que formen algún consejo de revisión. Este consejo deberá juzgar los componentes propuestos a ser programados y los componentes ya construidos con potencial de reuso. Para lograr esto, dicho consejo formula los estándares mínimos permitidos para ser admitir componentes en la biblioteca. El consejo de revisión debe autorizar al encargado de las bibliotecas de reuso para que clasifique y agregue los componentes aprobados.

Se necesita hacer crecer a las bibliotecas para animar más al reuso de los componentes, ya que la variabilidad anima a los desarrolladores a reusar.

2.7.6 Consideraciones sobre el Reuso.

Tener la habilidad de reusar código por mucho tiempo ha sido una noción en el mundo de la ingeniería de software. Esto ha sido uno de los mayores temas de estudio que ayudan en aumentar la productividad de los desarrollos de software [WOODFIELD,87],[MARTIN,94].

Se ha considerado que el reuso de software ofrece grandes potenciales para mejorar el proceso de producción del software [MARTIN, 91]. No únicamente se espera que conduzca a incrementos en la productividad sino que también conduzca a sistemas de software más robustos y más confiables [MARTIN, 94]. Por lo anterior, esfuerzos

continuos en investigación están ocurriendo con la finalidad de mejorar los procesos de desarrollo de software basado en el reuso y en la construcción de herramientas computacionales que faciliten el reuso [McCLURE,92],[SPC, 93],[RICHTER,87].

2.8 Nuevas Estructuras Organizacionales.

Para llevar a cabo el reuso de software es necesario planear una estructura de soporte de personal que incorpore actividades de reuso en su proceso de desarrollo y que a la vez se vayan generando las bibliotecas de reuso [JOOS,94].

Es necesario que se diseñen los procesos de construcción de software basados en el reuso, así como de los procesos de creación de componentes de reuso. Estos procesos deben estar soportados por una estructura organizacional que apoye las actividades de reuso [FAFCHAMPS,94].

Se han propuesto diferentes divisiones de personal para equipos de proyectos y equipos de construcción de software de reuso [RICHTER,87]. Un modelo utilizado para crear estructuras de soporte es el siguiente:

- 1) Una relación productor-consumidor entre equipos creadores de bibliotecas de software para reuso y los equipos de proyectos, donde el equipo creador de bibliotecas es puramente responsable para producir componentes reusables.
- 2) Un arreglo compartido donde los equipos de proyectos contribuyen y consumen lo que está en las bibliotecas de reuso.

2.9 Nuevos Modelos de Procesos.

Un ciclo de vida de software es un modelo para organizar, planear y controlar las actividades asociadas con el desarrollo y mantenimiento de software. Un ciclo de vida identifica tareas de desarrollo para cada fase que se deba considerar en los procesos de producción. El ciclo de vida involucra las fases de análisis, diseño, codificación y prueba [PRESSMAN, 89]. Cuando se habla del reuso, hay dos ciclos de vida a considerar:

- 1) el ciclo de vida para desarrollar componentes reusables.
- 2) el ciclo de vida para desarrollar con componentes reusables.

Aspectos a considerar incluyen si los dos ciclos de vida son diferentes o si la disponibilidad de una base de activos reusables modifica el ciclo de vida fundamental (bibliotecas de reuso). Este depende del enfoque de reuso utilizado como de la metodología usada.

Con el enfoque de componentes ejecutables, tanto los activos reusables (componentes de las bibliotecas de reuso) como los productos desarrollados con ellos son componentes de software (con un nivel de abstracción mayor). Los componentes reusables pueden ser desarrollados paralelamente al desarrollo de productos específicos o separadamente, es decir, crear componentes que tengan un potencial de reuso o extraer y adecuar componentes de proyectos pasados.

Cuando los componentes son desarrollados paralelamente al producto de software, se sigue el ciclo de vida como si no se generaran componentes, excepto que se debe poner mayor cuidado en su construcción. Cuando una actividad separada es requerida para desarrollarlos, se necesita llevar a cabo un análisis de dominio e ingeniería de dominio.

Una de las entradas principales del análisis de dominio es un conjunto de sistemas ya desarrollados dentro de un dominio, cuyas características comunes son identificadas, comprendidas, implementadas y entonces empaquetadas [LUBARS, 91]. La identificación y comprensión de características comunes pueden tomar lugar en las primeras etapas del desarrollo, para las cuales existe una documentación adecuada. Por ejemplo, si documentación de análisis de buena calidad para los sistemas existentes está disponible, las características comunes pueden ser reconocidas en el nivel de análisis. Si no, se necesita ver en los diseños existentes o aun en fragmentos de código implementado, llevar a cabo algún proceso de ingeniería de reversa para recuperar requerimientos de los componentes/sistemas individuales, identificando características comunes, comprendiéndolas, y entonces poder diseñar e implementar los componentes [CANFORA, 95]. Con respecto a la ingeniería de dominio, es el proceso de crear una arquitectura de software específica de un dominio, consistiendo en construir grupos (bibliotecas) de clases, empezando desde el nivel más bajo, donde dichos grupos tienen alto potencial de reuso en su dominio.

Construir nuevas aplicaciones con componentes reusables no requiere seguir un ciclo radicalmente diferente de aquel que se utiliza para construir nuevas aplicaciones sin componentes reusables. [DECKRO,90]. Una de las críticas dirigidas al ciclo de vida en cascada es que cada etapa del ciclo de vida está influenciada por la etapa anterior (Top-Down), mientras que la existencia de componentes reusables requiere de algún tipo de procedimiento con visión hacia adelante para identificar oportunidades de reuso y tomar ventaja de ésto [EDWARDS, 90],[SIMOS, 90]. Se considera que ésto es principalmente un problema de documentación: Reuso tradicionalmente ha significado el reusar fragmentos de código pequeños los cuales tienen poco o ninguna documentación del ciclo de vida; si la información de análisis fuera almacenada en bibliotecas de componentes, por ejemplo, el analista podría identificar oportunidades de reuso en el nivel de análisis sin ver al código real de los componentes reusables.

CAPITULO 3.

Características de una Organización Flexible de Desarrollo de Software.

3.1 El diseño de la organización y sus problemas.

La organización lleva a cabo sus actividades por medio de un conjunto de procesos principales, los cuales están interrelacionados. Para ejecutar estos procesos se divide a la organización en entidades funcionales o departamentos, donde cada entidad ejecuta un conjunto de tareas, de tal manera de poder administrar cada tarea dentro de estos procesos. Estas entidades funcionales tienen relaciones y establecen jerarquías para la asignación de mandos (direcciones, gerencias, jefaturas). Estas jerarquías forman la estructura organizacional y el administrador debe diseñar los perfiles de puestos correctos, las tareas correctas a estos perfiles y la cadena de mandos correcta para la asignación de autoridades, donde la autoridad mayor en la organización suele llamarse la alta administración. Con la experiencia que se ha tenido en estos tipos de organizaciones, se ha visto que la ventaja mayor es la excelencia funcional. Sin embargo, su defecto central son las tareas con cruces de coordinación y cruce de coordinación de funciones. Por caso, una tarea de análisis de una aplicación que incluya actividades de reuso de software implica la introducción de dos líderes de funciones, el líder del proyecto y el líder de las actividades de reuso. Esto se manifiesta como una falta de alineación cuando un proceso atraviesa los límites de la organización (incluir actividades de reuso en el análisis de productos de software), comunicación ineficiente a través de entidades funcionales (comunicación entre el bibliotecario de componentes y los desarrolladores de aplicaciones), y falta de un entendimiento comprensivo de las necesidades del cliente (entre los productores de componentes y los consumidores).

Una organización de desarrollo de software basada en el reuso debe alinear a las entidades funcionales para poder administrar tareas con cruces de coordinación y de departamentos con cruces de coordinación, tal como llevar a cabo una correcta coordinación de tareas dentro de productores y consumidores.

Considerando el enfoque de una organización funcional, una organización de reuso de software tiene cuatro funciones que son el productor, el consumidor, el administrador de bibliotecas de reuso y el director/administrador de proyectos de software. Un efectivo diseño de una organización de reuso necesita superar los problemas de la falta de alineación entre funciones organizacionales tales como las del productor y el consumidor, comunicación ineficiente a través de límites funcionales como la retroalimentación de los consumidores a los productores, y un entendimiento compartido de necesidades de cliente lo cual es decisivo para entregar productos exitosos al cliente. Para superar estos problemas inherentes al diseño de una organización, se necesita crear una organización flexible de desarrollo de software

(OFDS) que planea los cruces de coordinación de tareas que se deben permitir, coordina dichos cruces durante los procesos de desarrollo de software y aprende de ésto y los mejora.

3.2 Características de una organización flexible de desarrollo de software.

Todo organización de desarrollo de software debe estar preparada a los cambios que se producen en su medio ambiente de negocios, tales como sistemas de información innovadores, nuevos servicios al cliente, nuevas tecnologías de hardware y software. Este tipo de organización entonces deberá ser flexible para modificar lo que sea necesario para adaptarse, y de ahí el ser una Organización Flexible de Desarrollo de Software (OFDS). Para lograr ser más competitiva, el reuso de software ofrece los mecanismos necesarios para lograrlo, pero el reuso es un aspecto de negocios que implica transición tecnológica y cambio organizacional [GRISS,95]. Para que una organización pueda soportar la transición tecnológica y el cambio organizacional se deben poseer ciertas características organizacionales que apoyen estos cambios. Estas características se estudian en las siguientes secciones.

3.2.1 La organización debe estar diseñada para estar basada en los procesos principales.

Determinar los procesos principales de la OFDS y a partir de éstos llevar a cabo su diseño organizacional. Estos procesos principales contribuyen directamente a la creación o distribución de un producto o servicio, cada etapa en el proceso debe agregar valor a la etapa siguiente. Estos procesos principales involucran reuniones de trabajo, tomas de decisiones, flujos de información, metodologías de desarrollo de software, integración de documentación de cada fase del proceso de desarrollo (requerimientos, análisis, diseños, códigos fuentes, entre otros) e integración de tecnologías (herramientas C.A.S.E., lenguajes de programación). Todos estos procesos necesitan estar reunidos en un sistema integrado para que se encuentren articulados y así poder diseñar a la organización y a sus tecnologías para soportar dichos procesos principales.

3.2.2 Mostrar comportamiento adaptativo.

Debido a que el reuso de software es un aspecto de negocios que involucra transición tecnológica y cambio organizacional, una OFDS debe proveer los mecanismos para la flexibilidad de negocios. Para que se dé la flexibilidad se necesita incorporar la retroalimentación entre entidades funcionales -como la que se debe dar entre productores de componentes reusables y los consumidores de éstos-, de tener una retroalimentación del cliente que usa las aplicaciones, del conocimiento de las nuevas tecnologías y del medio ambiente de negocios. Luego, poder generar los cambios

necesarios (cambios tecnológicos, en metodologías, en autoridades jerárquicas, en estructura organizacional, en asignación de tareas, en productos/servicios) en la organización en respuesta a dicha retroalimentación.

Reacomodar efectivamente actividades de trabajo requiere de un entendimiento operacional -tanto por la administración como por la gente que hace el trabajo- de los elementos que guían el comportamiento en una OFDS, incluyendo: estrategias de la organización, procesos de trabajo, procesos de toma de decisiones, sistemas de personal como premios y cultura, y objetivos/metast individuales. Estos elementos necesitan estar diseñados para reforzar los objetivos de una OFDS. Las personas dentro de una OFDS necesitan información acerca de estos elementos, conocimiento acerca de cómo cambiarlos, recursos para hacer cambios, y autoridad para implementar los cambios para poder adaptar a la OFDS a nuevas condiciones del entorno.

3.2.3 Soportar alineación a través de sus entidades funcionales.

El llevar a cabo actividades en paralelo provoca que la OFDS pierda alineación en sus procesos principales, por ejemplo, el poder estar diseñando sistemas y a la vez empezar a codificarlos. Para dirigir este problema, el trabajo de lo específico a lo general de un productor necesita incluir intereses de lo general a lo específico de un consumidor. Por lo tanto, es necesario incluir aspectos de lo general a lo específico durante el proceso de desarrollo de software basado en el reuso, y así los productores de componentes tienen una visión general del funcionamiento de las aplicaciones como un todo. Mecanismos que puedan soportar esta alineación serían: incluir funciones para la determinación de necesidades de cliente por medio de análisis de dominio, proporcionando reuniones de debate para que los productores y consumidores discutan planes futuros y problemas actuales, es decir, considerar requerimientos de proyectos futuros o concurrentes a medida que se desarrolla software para satisfacer los requerimientos de proyectos actuales.

3.2.4 Un enfoque en el cliente final.

El éxito de una OFDS está basado en cumplir las necesidades del cliente, es decir, todo software producido debe cumplir los requerimientos estipulados por el cliente. Así, los desarrolladores involucrados en algún producto de software, deben enfocar sus esfuerzos en la satisfacción de un cliente final de dicho producto. Por ejemplo, las actividades de los productores y consumidores de la organización de desarrollo se mantienen alineadas si se lleva el enfoque de cumplir los requerimientos y expectativas del cliente. Mecanismos que conducen un enfoque en el cliente final, y dicho enfoque considera el cruce de coordinación de funciones, soportará alineación de la OFDS. Estos mecanismos incluyen, por ejemplo, un proceso de definición de producto, donde los requerimientos del proyecto puedan ser alineados trabajando dentro de un único dominio (aplicando técnicas de análisis de dominio). Proyectos

dentro del mismo dominio probablemente llegan a tener requerimientos similares, los cuales pueden ser llenados por software similar o idéntico. A medida que números grandes de requerimientos sean similares, grandes cantidades de software pueden ser reusados.

3.2.5 Aprendizaje organizacional.

Dentro de las organizaciones de desarrollo de software el aprendizaje organizacional es una característica básica que le da ventaja competitiva a la organización. Dicha organización debe prepararse para poder adquirir conocimiento de sus procesos principales de tal manera que entienda la razón de ser de cada uno, pueda detectar situaciones problemáticas y tome las acciones apropiadas para mejorar. Cada proceso debe agregar valor al desarrollo de productos de software, y de no ser así, analizar y rediseñar el contenido y flujo de estos procesos e incorporarlos en las actividades de construcción de software. Además, debe detectar y comprender cualquier inhibidor del reuso de software que cause un bloqueo en las actividades de reuso.

3.2.6 Reusar experiencias.

Una OFDS registra el medio ambiente de negocios, las necesidades del cliente, el contenido y ejecución de los procesos principales, documenta completamente sus metodologías de desarrollo, fomenta el compartir conocimientos entre sus miembros (pues al ser compartido es un aprendizaje organizacional y su valor aumenta) y va generando una biblioteca de componentes reusables. Todos estos registros forman parte de las experiencias que la organización ha pasado y que serán la base para una utilización posterior o para su mejora continua. El reuso de formas de hacer las cosas y de las experiencias (utilización de ciertas tecnologías, metodologías, estructuras organizacionales de soporte, estrategias de desarrollo) ayudan a desarrollar una base de capacidades en la OFDS que le servirán para poder enfrentar los objetivos organizacionales.

CAPITULO 4.

Factores Organizacionales y el Reuso de Software.

4.1 Introducción.

A lo largo del tiempo, el reuso de software ha ocurrido dentro de los equipos de programación individuales y de proyectos de desarrollo de productos de software [WOODFIELD,87]. El reuso que se ha estado presentando tradicionalmente ha sido iniciado por los programadores. Ellos reusan trabajos que han hecho antes, comunican a otros programadores acerca del software que ellos conocen que existe y que puede ayudarles, y en algunos casos organizan pequeñas bibliotecas de rutinas comunes. Lo anterior se puede considerar como el inicio a un programa de reuso de software, pero el objetivo es establecer programas de reuso bien estructurados que lleven a un reuso cada vez mayor.

Los programas de reuso han asumido que las soluciones técnicas superarían barreras para un reuso efectivo. Sin embargo, en el repaso de los acontecimientos pasados de programas de reuso muestran que los factores organizacionales pueden afectar en la implementación de dichos programas de reuso [JOOS,94]. Un factor organizacional fundamental es la relación entre productores y consumidores de componentes y servicios de reuso [FAFCHAMPS,94], la cual implica una coordinación ordenada de actividades para modificar componentes reusables, crear nuevos y hacer uso correcto de dichos componentes en la construcción de aplicaciones. Toda esta serie de actividades llevan a las organizaciones a ir ganando experiencias e incrementar los beneficios que el reuso les puede dar.

Para cosechar los beneficios potenciales de un reuso cada vez mayor, donde se considera el mejoramiento de la calidad y la productividad, es necesario llevar el reuso fuera de los individuos o equipos y ubicarlo en un centro de cómputo o aún más allá a nivel empresarial, para que el reuso esté disponible en todo momento . Para lograr ésto, se necesita crear una estructura organizacional de soporte para que ayude a integrar el desarrollo de sistemas a través del reuso y a la generación de componentes de reuso.

La estructura organizacional ayuda a poder implementar un programa de reuso, donde el enfoque de esta estructura es eliminar los inhibidores no técnicos (sociológicos, psicológicos o administrativos) típicamente asociados con el reuso de software. En este trabajo, para poder organizar responsabilidades, se considera dos formas de dividir a los equipos de trabajo: 1) la relación productor-consumidor, donde los productores diseñan y desarrollan componentes reusables, y consumidores quienes diseñan y desarrollan productos de software con componentes reusables; 2) un arreglo compartido de equipos de proyectos donde en cada equipo existen productores y consumidores los cuales contribuyen y consumen lo que está en las bibliotecas de reuso.

Se han discutido y defendido las funciones de un productor y de un consumidor, así como de las interacciones entre ellos [BARNES,87], [DAVIS,93], [FAFCHAMPS,94]. Esta distinción de papeles no tiene precedentes en una estructura de ingeniería de software tradicional, y por lo tanto se tiene mucho que aprender acerca de cómo hacer para que estas nuevas relaciones trabajen efectivamente. Además, se debe escoger una estructura organizacional que apoye la manera de trabajar de la empresa y a la vez elimine los conflictos entre los equipos de trabajo [FAFCHAMPS,94].

Por lo anterior, se considera que además de la estructura de un equipo de proyectos que se dedica a la construcción de productos de software, un equipo responsable para brindar servicios de reuso -como construir y mantener una biblioteca de componentes para ser reutilizados- pudiera ser necesitado. Al final, cuál división de trabajo se debe adoptar para echar a andar un programa de reuso dependerá del alcance del programa de reuso y de las características de la organización en cuestión, incluyendo número de proyectos en paralelo, amplitud del dominio de las aplicaciones, del talento del personal de sistemas y de la estructura organizacional.

En lo que sigue, se presentan un modelo general que ayuda a entender las relaciones entre productores y consumidores, y de las características que presenta un departamento de desarrollo de software basado en el reuso. Este modelo es una propuesta de la tesis para poder entender la importancia de una estructura organizacional de soporte para implementar un programa de reuso. El modelo general describe los elementos mínimos que constituyen una estructura de soporte, y con algunas variaciones se derivan otras estructuras que pueden ayudar a organizar algún departamento de desarrollo de sistemas computacionales.

Las consideraciones tecnológicas requeridas para el trabajo de todas estas personas es tener una red de computadoras, donde cada equipo tiene acceso a un computador central que mantiene todas las bibliotecas de componentes y sistemas de reuso.

4.2 El modelo general.

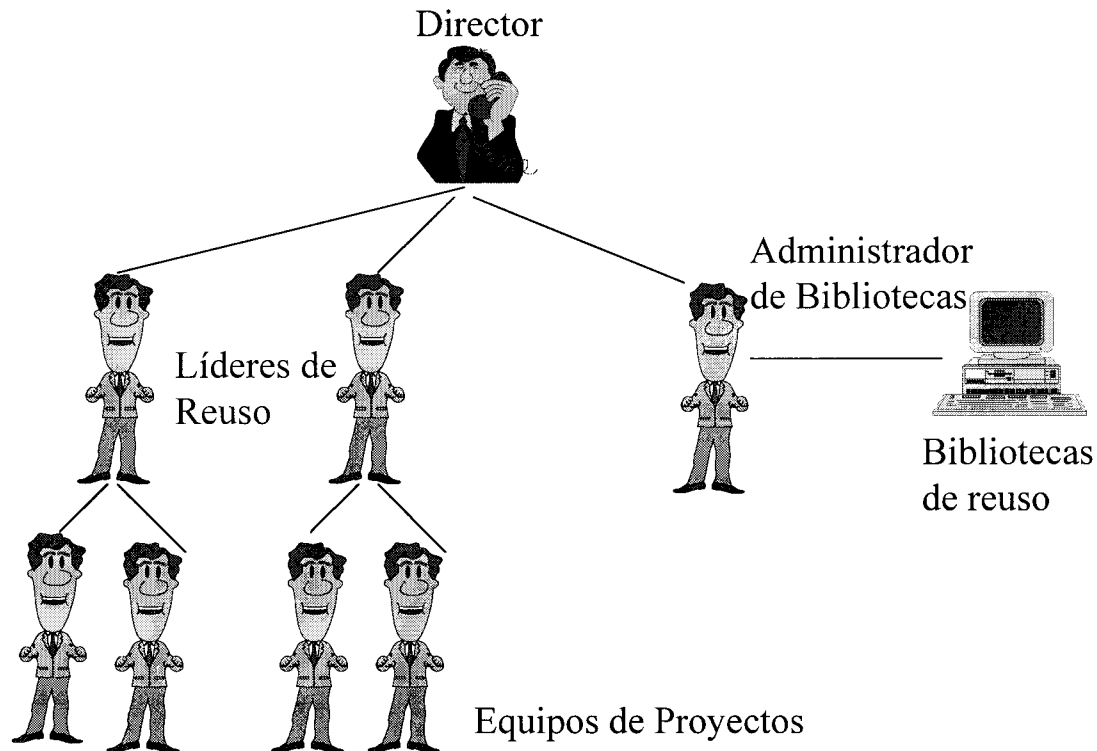
4.2.1 Introducción.

Para poder hacer el análisis de este modelo, se considera que el objetivo del departamento de desarrollo de software (departamento de desarrollo de sistemas) es liberar productos de software solicitados por algún cliente externo al departamento. Lo anterior implica que este departamento tenga una estructura divisional, en donde los diseñadores/programadores están agrupados en equipos dedicados a la liberación de un producto de software.

La estructura organizacional de soporte de este modelo general está compuesta por un director del departamento de desarrollo de software, equipos de desarrollo para cada proyecto, líderes de proyectos, un administrador de bibliotecas de reuso, un comité de evaluación de componentes y bibliotecas de componentes reusables

(bibliotecas de reuso). La descripción del modelo se desarrolla en las siguientes secciones.

ESTRUCTURA ORGANIZACIONAL DEL MODELO GENERAL



4.2.2 El director del departamento de sistemas.

Es una persona con mucha experiencia en la administración de proyectos de software, donde implica madurez en la coordinación de equipos de trabajo. Debe tener buenas bases de ingeniería de software para poder analizar muy bien los procesos de construcción del software que se llevan a cabo en su departamento. Debe conocer el paradigma de reuso de software y obtener algún método que se pueda adaptar al proceso de construcción de software basado en el reuso, ganar experiencias de reuso estudiando casos reales, aprender más acerca de sus procesos de desarrollo para que logre hacerlos cada vez más sistemáticos y pueda introducir mejoras en ellos sacando provecho del software reusable.

Por ser el encargado de la creación de todas las aplicaciones, el director atiende juntas con el equipo de planeación de aplicaciones requeridas y se genera un calendario de todas las aplicaciones a llevar a cabo y de las prioridades correspondientes. Esta planeación será la base de la carga de trabajo para el departamento de sistemas, por lo que debe administrar muy bien las prioridades.

Como encargado principal de las actividades de reuso (las cuales se presentaron en 2.7.5), el director debe cuidar la creación y el mantenimiento de los estándares mínimos de entrada a la biblioteca central de reuso (especificadas en 4.2.3), o en su defecto a la biblioteca secundaria, que deben cumplir los componentes de software reusable. Además, debe establecer con el coordinador de bibliotecas de reuso un mecanismo seguro para la administración de bibliotecas y conocer tecnologías computacionales que apoyen todo su proceso de construcción de software.

El director controla todas las actividades del proceso de desarrollo de software basado en el reuso y para lograr esto, planea, introduce, coordina y controla un programa de reuso de software. En apoyo a lo anterior, el director debe conseguir el apoyo de la alta dirección y el compromiso de todo su personal de desarrollo, ya que el apoyo en recursos y el cambio organizacional son decisivos en el éxito de un programa de reuso.

Personalmente, da pláticas e indicaciones sobre los estándares de programación y presenta nuevas técnicas y tecnologías para el desarrollo de aplicaciones basadas en el reuso. Fomenta entre los equipos el diseñar software reusable para incrementar la cantidad y variedad de componentes de reuso, haciendo crecer las bibliotecas, de tal manera que se vayan haciendo cada vez más aplicaciones con este material y aumente la productividad y la calidad en los productos.

Para la actualización de las bibliotecas central y secundaria, el director forma parte de un comité de evaluación de componentes para llevar a cabo la evaluación de componentes reusables que cumplen con los estándares mínimos establecidos.

El director informa a todo el departamento sobre la estrategia de reuso que se pretende llevar a cabo. Dicha estrategia puede estar basada en la determinación de requerimientos de proyectos individuales dentro de un único dominio o basada en la determinación de requerimientos de proyectos futuros o en paralelo. Considerando la estrategia seleccionada y para medir mejoras en los procesos de desarrollo, el director marca metas de reuso para cada proyecto, acuerda fechas de revisión de avances y determina tiempos de desarrollo.

El director hace juntas de trabajo con cada equipo de proyecto con la finalidad de comparar tiempos programados contra etapas del proyecto terminadas. Ahí mismo, determina la productividad que se está teniendo hasta ese momento y discute con el equipo la calidad del software que se está construyendo para conocer algunos beneficios temporales del programa de reuso. Además, informa de las tendencias que se están presentando con relación a los productos de software que el equipo desarrolla. Es por medio de estas juntas como el director puede apoyar la adopción incremental del reuso, ganándose experiencias y detectando errores en el proceso de desarrollo.

4.2.3 La Biblioteca de reuso central.

Una biblioteca de software reusable que contiene tanto los componentes específicos de dominio -desarrollados por un proyecto en particular- como los componentes independientes de dominio, y ambos garantizan un alto nivel de calidad y de reusabilidad.

Esta biblioteca es controlada y administrada centralmente por un administrador de bibliotecas de reuso. Los desarrolladores de cada equipo de desarrollo someten sus componentes candidatos a evaluación al comité de evaluación de componentes, para luego ser introducidos a esta biblioteca. Los componentes de esta biblioteca deben cumplir los siguientes criterios de entrada:

1. Incluir diseño del componente.
2. Código del componente.
3. Incluir casos de prueba y resultados.
4. Documentación que explique y aclare al componente.
5. Información que indique su uso, restricciones, criterios de desempeño y restricciones del medio ambiente de ejecución.
6. Información de clasificación (la información de clasificación es de utilidad para ser usada por alguna herramienta de biblioteca para una búsqueda futura).

Estos criterios garantizan componentes de la más alta calidad.

Todos los componentes en esta biblioteca deben pasar el ciclo de desarrollo de productos estándar, incluyendo las fases de inspección cuidadosa y pruebas de independencia.

Si algún defecto fuera encontrado en un componente (quizá durante su ejecución dentro de alguna aplicación) de la biblioteca, todos los usuarios de ese componente son informados de los defectos y la solución. Cada proyecto individual pueden determinar si necesitan o no integrar la solución en sus productos. En cualquier caso, todas las versiones del componente son mantenidas en la biblioteca. Si una mejora es solicitada, ésta es revisada por el comité de evaluación de componentes para asegurar que no impacta en la reusabilidad del componente. El autor original (puede ser autor o autores) incorpora las mejoras en el componente y proporciona información adicional de diseño, documentación y casos para su prueba. Si la mejora hace incompatible al componente con las versiones anteriores, éste es renombrado y considerado como un nuevo componente.

4.2.4 Biblioteca secundaria de reuso.

Esta biblioteca se requiere para almacenar aquellos componentes que tienen un potencial de reuso pero no cumplen los estándares mínimos de entrada (ver 4.2.3) de la biblioteca central de reuso. Esta biblioteca se necesita para ir dando seguimiento a estos componentes que por falta de tiempo, o por no ser necesitados en otros proyectos (no fueron diseñados como reusables) no se han podido adaptar a los estándares, pero hay interés de incluirlos en la biblioteca central si éstos son

mejorados y estandarizados. Cualquier equipo de desarrollo puede hacer uso de los componentes de esta biblioteca, pero tiene la responsabilidad de incorporar los estándares mínimos especificados por el comité al componente elegido. Al hacer ésto, el equipo puede proponerlo después como un componente que deba estar en la biblioteca central. Esto le dará puntos por reusar un componente y por generar componentes de alta calidad.

4.2.5 Biblioteca de productos terminados.

Esta biblioteca contiene todos los productos terminados, los cuales ya fueron liberados pero deben ser analizados para poder extraer posibles componentes de reuso, que pudieran ser muy específicos del área de dominio de la aplicación o componentes independientes de dominio.

4.2.6 Administrador de Bibliotecas de software.

El administrador de bibliotecas reporta directamente al director del departamento de desarrollo. Debido a que el departamento de desarrollo de sistemas se dedica a la creación de productos de software -los cuales son la razón de cada proyecto-, es claro que cada proyecto necesite de componentes de las bibliotecas de reuso. Dichos componentes deben estar almacenados de tal manera que su beneficio se vea reflejado en la utilización de ellos en varios proyectos, de ahí la utilidad de cada componente y, por consecuencia, de una biblioteca de componentes. Así entonces, la administración de componentes está basada en múltiples proyectos y para ayudar en esta correcta administración de componentes almacenados en las bibliotecas de reuso, el uso de herramientas computacionales que asistan en la administración de estas bibliotecas es conveniente, ya que proporcionan esquemas de búsqueda fácil de usar para los programadores, así como facilitar al administrador de bibliotecas el poder clasificar y agregar componentes (proporcionar oportunos servicios de reuso) [BRUCE,87],[MACCHINI,92].

Las actividades que se deben desempeñar para llevar a cabo la administración de bibliotecas son:

1. Clasificación de componentes.
2. Dar de alta componentes según su clasificación.
3. Modificación de componentes. Dar seguimiento de las versiones de cada componente, ya que durante las juntas del comité puede surgir la necesidad de renombrar un componente al presentar una versión muy diferente a éste, lo que podría impactar en las aplicaciones utilizándolo. Por lo anterior, tener cuidado en las modificaciones en el código del componente, las cuales surgen de la necesidad de ajustar componentes para su reuso.

4. Borrado de componentes. Si un componente va a ser borrado de la biblioteca central, tener cuidado si algún proyecto o producto de software lo está utilizando, ya que se perdería su información de documentación.
5. Obtener componentes a partir de otros mercados comerciales de componentes de software.
6. Evaluar los mecanismos de búsqueda, selección y prueba de componentes para determinar su facilidad de uso y poderlos mejorar.
7. Investigar nuevas tecnologías para la construcción de bibliotecas de software (por ejemplo, el uso de hipertextos para la búsqueda de componentes).
8. Difunde la información de las bibliotecas de reuso entre el personal de todo el departamento de desarrollo de un manera electrónica.

4.2.7 Los equipos de desarrollo de software.

Estos equipos están constituidos por un líder de reuso y un conjunto de desarrolladores de software. El líder es el encargado de apoyar en todas las actividades de reuso y de guiar la estrategia de reuso (ver 4.3.1.3).

Cada equipo lleva a cabo la construcción de sus aplicaciones basándose principalmente en el software reusable. Para lograr esto, el líder de reuso proporciona el apoyo para la búsqueda, selección y prueba del software reusable de las bibliotecas. Luego, el mismo equipo toma la decisión de usar o de adecuar aquel software que lo requiera.

El equipo de proyecto presenta el análisis de requerimientos al director y al administrador de bibliotecas para estimar el alcance de reuso que pueden conseguir, considerándose un porcentaje mínimo de reuso. Posteriormente, cada equipo de proyecto presenta sus diseños iniciales de su producto de software al director y al administrador de bibliotecas, de tal manera que se haga la planeación del tiempo estimado de desarrollo y poner las metas que el proyecto puede alcanzar con relación al reuso de componentes. Cualquier componente que deba ser creado es de la responsabilidad de cada proyecto el desarrollarlo desde su diseño.

El líder de reuso debe tener acceso a la documentación completa y actualizada de la funcionalidad de los componentes y de la forma de utilizarlos. El equipo se apoya en su líder para coordinarse con el administrador de bibliotecas para proporcionarles esta documentación, la cual podría estar presente en línea y de manera electrónica en sus computadoras de trabajo.

Debido a que en esta estructura son los equipos los productores de componentes y a la vez los consumidores de ellos, tienen que llevar a cabo juntas periódicas de todo el equipo para plantear actividades de reuso. En estas juntas se analizan los posibles componentes que se podrían reutilizar para la construcción de su producto

(considerando las bibliotecas de reuso), se exponen candidatos de reuso y se lleva a cabo la identificación de candidatos de reuso que se produjeron durante el desarrollo. En estas juntas el líder de reuso es el coordinador.

Con relación a la creación de componentes, los autores de cada componente desarrollado presentan su componente candidato. También, si el proyecto va en etapa de análisis o diseño, los autores presentan posibles componentes que han detectado y que consideran podrían tener un potencial de reuso. En ambos casos se debe presentar la documentación marcada por el comité de evaluación (ver 4.2.3). Luego el líder pasará esta información (quizá electrónicamente) al administrador de bibliotecas para ser considerada en la junta del comité de evaluación.

Cada equipo de proyecto debe recibir la preparación educacional especial que su proyecto le demande, por ejemplo, análisis de dominio, educación general de reuso, adecuación de las técnicas estructuradas para el reuso, ingeniería de aplicaciones, diseño basado en objetos, técnicas de programación orientadas a objetos y capacitación en herramientas de desarrollo.

4.2.8 Líderes de proyecto.

Un líder de proyecto tiene a su cargo un equipo de desarrollo. Debe ser un programador/diseñador experimentado y respetado por el departamento. Debe tener experiencia en las herramientas de desarrollo para poder dar soporte a su equipo. La formación principal de este líder está en la ingeniería de software. Dentro de sus habilidades están el tener amplia experiencia en desarrollo de software, estar bien entrenado en técnicas de ingeniería de software, coordinación simultánea de equipos de desarrollo y alta capacidad para comunicarse. Dentro de las técnicas de ingeniería de software debe conocer el paradigma de reuso y los beneficios que ésta puede generar.

Tiene a su cargo todas las actividades del equipo, donde implica el dar seguimiento a todas las etapas del proceso de desarrollo de software, considerando en cada etapa la aplicación de actividades de reuso (ver 2.7.5). Además, tiene acceso a todos los diseños y códigos fuente generados por los desarrolladores del equipo con la finalidad de supervisar estándares de programación. Está al pendiente de que el equipo haga uso de las bibliotecas de reuso y de conseguir todo el apoyo necesario para las actividades de reuso.

El líder de cada proyecto identifica cualquier necesidad educacional especial que el proyecto puede requerir, por ejemplo, educación general de reuso, diseño basado en objetos, técnicas de programación orientadas a objetos, la adecuación de técnicas estructuradas para el reuso o en la capacitación en nuevas herramientas de desarrollo. Él hace llegar una solicitud de capacitación al director, justificando en qué se debe capacitar y a quienes, aclarando el tiempo esperado de entrenamiento y la puesta en producción de los miembros capacitados del equipo.

Las funciones generales de este líder son:

1. Estar en las juntas de especificación de requerimientos en cada proyecto para luego determinar con su equipo las especificaciones de requerimientos de los componentes a requerir.
2. Administrar todas las actividades de reuso que necesite su equipo.
3. Animar y obligar al uso de componentes.
4. Asiste a seminarios, congresos o eventos en general sobre reuso de software.
5. Pertenecer al comité de evaluación de componentes.
6. Monitorear los avances en el desarrollo de software.
7. Define estrategias de reuso de un proyecto o en la creación de componentes junto con su gente técnica.
8. Establece metas de reuso junto con el director en base a un porcentaje mínimo de reuso. Dicho porcentaje es fijado por el director y el líder cuando el departamento de desarrollo ya alcanzó un equilibrio de reuso (cuando el departamento ya está reusando componentes de sus bibliotecas de reuso), entonces se puede ir ajustando periódicamente de una manera incremental.
9. Establecer tipos de reuso que su equipo pudieran incorporar como: reuso de análisis, de diseños, de rutinas compradas a un proveedor externo, reusar componentes producidos por el proyecto mismo o hacer uso de las bibliotecas central o secundaria.

4.2.9 El Comité de Evaluación de Componentes.

Este comité es responsable de llevar a cabo la evaluación de todos los componentes candidatos de reuso presentados por los desarrolladores de los mismos (autores), con la finalidad de agregar componentes a la biblioteca central (componentes de más alta calidad) o a la biblioteca secundaria. Lo anterior tiene el objetivo de ir haciendo crecer las bibliotecas de reuso. El comité considera los estándares mínimos de aceptación (ver 4.2.3) para llevar a cabo la evaluación.

Todos los miembros de este comité deben ser diseñadores experimentados, tener buenas bases de ingeniería de software y conocer el paradigma de reuso.

Los componentes candidatos y su material de documentación de soporte son sometidos al coordinador de bibliotecas para su próxima evaluación por el comité de evaluación. Dependiendo del avance en el desarrollo del componente candidato (es una propuesta de algún desarrollador o grupo de ellos donde se tiene quizá su diseño inicial y una justificación de ser reusable, o es un componente que surgió de un proyecto) el material presentado al comité varía. El autor de un componente completamente nuevo somete los objetivos y el diseño de alto nivel. Para componentes ya programados, el autor proporciona toda la documentación requerida (la estipulada en 4.2.3).

Cuando el coordinador tiene los candidatos suficientes para justificar una junta, éste convoca al comité de evaluación de componentes. Para lograr que los miembros del comité tengan un conocimiento previo de los componentes a evaluar el día de la

reunión, el material conveniente de cada componente debe ser distribuido por lo menos dos semanas antes de la junta. Durante la reunión, cada componente candidato es evaluado y uno de tres resultados es emitido: 1) el componente candidato es aceptado en la biblioteca central, 2) el componente candidato es aceptado en la biblioteca secundaria, 3) se determina que se necesita información adicional y su revisión es re-calendarizada.

Cualquier componente que no cumpla con el estándar mínimo propuesto, debe ser clasificado y guardado en la biblioteca secundaria de reuso. Si el componente es visto con alto potencial de reuso, el comité debe turnarlo a su autor para su adaptación y próxima evaluación, con la finalidad de incluirlo en la biblioteca central y así garantizar su calidad y reusabilidad.

Cualquier componente incluido en alguna de las bibliotecas deberá ponerse a disposición de cualquier equipo de desarrollo como material de reuso.

Este comité tiene además la autoridad de generar un programa de incentivos (se propone en la sección 4.4) para fomentar el reuso de software en el departamento. Su función en este programa será el establecer una tabla de asignación de puntos por contribuir a las bibliotecas de reuso y por reusar componentes/partes reusables en los proyectos. Dicha tabla debe especificar los límites de asignación de puntos que podrá ganar un individuo. Además, establecer las reglas del juego con relación a los criterios para considerar a alguien como autor de dichos componentes/partes reusables y toda la mecánica de asignación de premios (tanto de placas como de premios económicos).

4.3 Variaciones en el modelo general.

4.3.1 Introducción.

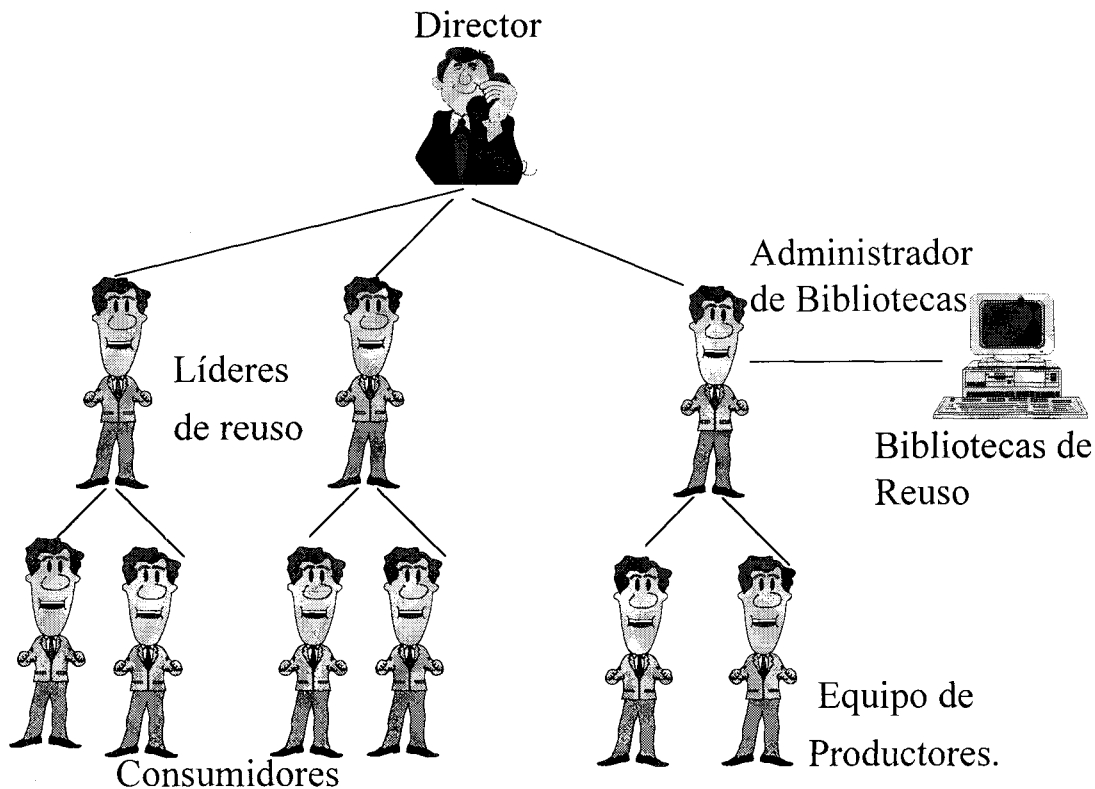
Haciendo algunas variantes al modelo podemos derivar otras estructuras organizacionales que pueden soportar un programa de reuso. Estas variantes se deben analizar ya que los departamentos de desarrollo de software tienen capacidades muy variadas dependiendo de la cantidad de proyectos que la empresa necesite manejar en paralelo, pues una empresa con muchos proyectos y de dominios variados deberá tener mayores capacidades que una empresa con pocos proyectos de software y con dominios no tan variados. Por ejemplo, cuando consideramos empresas grandes, el control en la creación y en el mantenimiento de componentes reusables (por ejemplo, control de versiones para cada componente) y a la construcción de aplicaciones a partir de éstos se vuelve complejo en la medida que crecen los proyectos en paralelo y a la diversidad del dominio de las aplicaciones.

Para estudiar estas variantes vamos a considerar los elementos presentados en el modelo general y con algunas modificaciones se definirán nuevas estructuras de soporte.

4.3.2 Director de Productores y Consumidores.

4.3.2.1 Introducción.

ESTRUCTURA DE PRODUCTORES Y CONSUMIDORES



En esta estructura se van a dividir las funciones del equipo de desarrollo de proyectos de tal manera que las relaciones de productores y consumidores estén bien separadas. La estructura tiene la intención de estudiar estas relaciones y determinar su funcionalidad y flexibilidad, ya que éste es un factor importante para lograr incluir el reuso en el desarrollo de software.

Para la estructura se considera un director del departamento de desarrollo de sistemas, un equipo de consumidores de componentes reusables (equipos de proyectos) que producen los productos finales de software, un equipo de productores de componentes reusables, un líder del equipo de productores (administrador de bibliotecas), un líder por cada equipo de consumidores, un comité de evaluación de componentes y las bibliotecas de reuso.

Para cada equipo de consumidores y productores se asigna un líder que se encargue de dirigir las actividades de reuso. Dentro del equipo de productores está su líder de productores, quien es el encargado de las bibliotecas de reuso y del servicio que los

productores brindan a cada proyecto consumidor. En cada proyecto consumidor se nombra un líder de proyecto, el cual será el encargado de introducir el reuso en el proceso de desarrollo de productos.

4.3.2.2 El director del departamento de sistemas.

Tiene a su cargo un equipo de productores para que administren las bibliotecas y los sistemas reusables. De esta manera el director ordena todas las actividades de los servicios de reuso. Dentro de este equipo, el director asigna un líder del equipo de productores para que administre las bibliotecas de reuso y el desarrollo de componentes para los proyectos que las solicitan.

El director tiene a su cargo a los equipos de proyectos (equipos consumidores de componentes), los cuales llevan a cabo la construcción de los productos de software que serán entregados al cliente final. Para ésto, asigna un líder de proyecto en cada equipo consumidor, de tal manera que haya alguien quien garantice el reuso de componentes. Para apoyar lo anterior, facilita el entrenamiento de consumidores sobre el uso de los diferentes componentes y de lo que está disponible de otros proyectos, de tal manera que se gane tiempo y calidad en la construcción de aplicaciones a partir de la adecuación e inclusión de componentes. El director y el líder de productores asisten a cada equipo consumidor para facilitar el reuso, ayudando en el diseño inicial de la aplicación y al uso de las bibliotecas de reuso, de tal manera que se haga la planeación de aquellos componentes que pueden ser reusados y aquellos que deben ser programados por el equipo de productores.

Dentro de las funciones del director está el establecer perfectamente las funciones de productores y consumidores, de tal manera que cada equipo consumidor se dedique a la construcción de productos de software a la medida de los requerimientos determinados, y a la vez cuenten con el apoyo del equipo productor para la generación de cualquier componente. Debe cuidar las cargas de trabajo entre los equipos y administrar muy bien las prioridades, lo anterior con la finalidad de que los productores puedan dar un servicio oportuno y nunca se incrementen los tiempos de entrega de aplicaciones.

Para poder generar la biblioteca central de reuso, el director y el equipo de productores acuerdan los estándares de programación y de documentación, además de que ambos forman un comité de evaluación de componentes, donde el director es coordinador oficial. Así, periódicamente este comité hace juntas para evaluar cada componente que pudiera formar parte de la biblioteca central, por lo que coordina la junta en la que se presentan los componentes, se revisan los estándares acordados y se toma la decisión de introducirlos o de mejorarlos para una posterior evaluación. Para llevar a cabo este proceso, el director debe considerar los siguientes productos de software:

1. Productos de software generados por los proyectos de los equipos consumidores.

2. Componentes específicos de una aplicación generados por productores en demanda de un equipo consumidor.
3. Componentes reusables generados por productores los cuales fueron diseñados para ser reusados.

Los productos de software están almacenados en la biblioteca de productos terminados, los cuales pudieran adecuarse para llegar a ser componentes. Los componentes específicos de una aplicación son aquellos que se programaron para satisfacer los requerimientos de un producto específico pero no se diseñaron para ser reusados, por lo que son muy específicos para funcionar dentro de sus aplicaciones. Los componentes reusables son el producto de un esfuerzo mayor de desarrollo en donde se consideran requerimientos de proyectos futuros, y estos componentes pueden ser incorporados en una área más amplia de aplicaciones o formar parte de un único dominio, pero sí están diseñados para ser reusados.

Estos tres productos de software deben ser considerados y es mediante las juntas de trabajo como se podrían identificar sus potenciales de reuso. Por lo anterior, el director puede solicitar la presencia de consumidores y productores autores de los productos siendo evaluados para ayudar en el entendimiento de su programación.

4.3.2.3 El equipo de productores.

En la estructura organizacional tienen al director como su jefe inmediato. Se recomienda que sean programadores respetados y experimentados, que tengan alta capacidad de comunicación y de relaciones interpersonales, que estén bien entrenados en técnicas de software y que estén comprometidos con los requerimientos de un producto final de software (por ejemplo, un producto comercial de interés para la empresa). Tienen la función de prestar todos los servicios de reuso dentro del departamento de desarrollo de sistemas, por lo que implica que sean cuidadosos en mantener enlaces de comunicación activos con los consumidores. Están muy de cerca de los consumidores para asistir en la búsqueda, funcionamiento, utilización, adecuación o creación de los componentes reusables.

Este equipo de productores deben recibir cualquier preparación educacional especial que sus funciones demande, por ejemplo, educación general de reuso, análisis de dominio, adecuación de técnicas estructuradas para el reuso, ingeniería de dominio, diseño basado en objetos, técnicas de programación orientada a objetos y capacitación en herramientas computacionales.

Todos forman parte del comité de evaluación de componentes para evaluar componentes candidatos a ser incluidos en la biblioteca central o secundaria. Las formas de hacer sus actividades se pueden dividir en dos:

1. Actividades sincrónicas (que suceden al mismo tiempo), las cuales son actividades iniciadas por solicitudes de los equipos consumidores, canalizadas a través del director, y éstas pueden variar desde una simple búsqueda de un componente

requerido hasta la construcción del componente desde su diseño. Esta actividad involucra el apoyar al consumidor en entregarle un componente que llene sus requerimientos.

2. Actividades asincrónicas, consistiendo de crear componentes que serán solicitados en el futuro probablemente (anticipándose a demandas futuras), o adecuar componentes generados por las actividades sincrónicas para mejorar su reusabilidad y llegar a formar parte de la biblioteca central.

Como creadores de software reusable, los productores pueden adoptar las siguientes estrategias:

1. Determinar los requerimientos de un proyecto de software pero dentro de un único dominio. Los proyectos dentro de un mismo dominio van a tener probablemente requerimientos similares que pueden ser llenados por un software similar o idéntico. Mientras mayor sea el número de requerimientos similares, mayor será la cantidad de software que puede ser reusado.
2. Estableciendo un mecanismo de visión futura, donde se consideran requerimientos de proyectos futuros o concurrentes (lo que se haga será reutilizado en otros proyectos a la vez) explícitamente mientras se desarrolla software para satisfacer requerimientos de proyectos actuales. Esto implica el anticiparse a las necesidades de los proyectos futuros mientras se consumen recursos (tiempos invertidos por los equipos en análisis, diseño y programación) en los proyectos actuales.

Este equipo incluye en cada componente la información de documentación ya establecida por el comité de evaluación para uso de consumidores o de otros productores. Además, los productores deben animar a los consumidores a proporcionar retroalimentación sobre la calidad, claridad y cobertura de la documentación.

4.3.2.4 El líder del Equipo de Productores.

Dentro de este equipo de productores hay un líder del equipo que es el jefe de todos. Él tiene a su cargo todas las actividades del equipo. Además, tiene a su cargo el computador que almacena todas las bibliotecas de reuso: biblioteca central, biblioteca secundaria y la biblioteca de los productos de software de proyectos anteriores.

Las funciones generales de este líder son:

1. Administración de las bibliotecas de reuso (ver 4.2.6).
2. Determinar con su equipo de productores las especificaciones de requerimientos que los componentes deben de cumplir de acuerdo con lo acordado con los equipos consumidores.
3. Administrar el desarrollo de componentes.
4. Publica en el departamento el contenido de las bibliotecas de reuso.
5. Asiste a seminarios, congresos o eventos en general sobre reuso de software.

6. Proponer nuevos lenguajes de desarrollo que animen el uso de componentes (por ejemplo, los lenguajes de programación orientados a objetos).
7. Monitorear avances en el desarrollo de nuevos componentes, especialmente en aquellos que son solicitados por un equipo consumidor.

4.3.2.5 El equipo de consumidores (equipos de proyectos).

Estos equipos de programadores tienen al director como su jefe directo y están agrupados en equipos de proyectos para la producción de un producto de software, donde además existe un líder de proyecto quien coordina toda la administración del desarrollo del producto.

Es en estos equipos consumidores donde se deben construir aplicaciones basadas en el software reusable. Para esto, deben estar bien enterados de toda la información que existe en las bibliotecas de reuso. Tienen la obligación de hacer uso de los componentes reusables de las bibliotecas central y secundaria. Para lograr poner metas de reuso, cada equipo consumidor debe darse tiempo para obtener los requerimientos del producto final, los cuales deben ser revisados posteriormente por una junta de trabajo con el director y el líder de productores. Esta junta es muy importante porque se deben poner en claro los requerimientos de las aplicaciones de los consumidores, y en base a esto, los productores conocer cuáles son los requerimientos a considerar para la búsqueda o creación de componentes. Esta coincidencia de requerimientos puede verse afectada por los siguientes factores:

1. La calidad y la reusabilidad del software de los productores.
2. Las habilidades y el conocimiento de los consumidores acerca del reuso y del software reusable.
3. El grado de congruencia entre los requerimientos del productos y del consumidor.

De cualquier manera, los consumidores solicitan a los productores la creación de componentes y la asistencia en la utilización de componentes.

Los consumidores deben hacer el diseño preliminar de las aplicaciones junto con el líder de productores, de tal manera que se puedan acordar cuáles partes del software serán reusadas y cuáles deberán ser pasadas al equipo de productores para su programación. Esta actividad es relevante para cumplir con las metas de reuso marcadas para el proyecto, ya que un correcto diseño de las aplicaciones ayuda a detectar las características de los componentes que se necesitan, determinándose la selección de componentes reusables (se gana tiempo de desarrollo y mejora la calidad), y de lo contrario, la construcción de los mismos.

Los consumidores deben tener acceso a una documentación completa y actualizada de la funcionalidad de los componentes y de la manera de usarlo. Ellos son más receptivos a soluciones reusables cuando se les suministra una solución de muestra, la cual sea relevante a su dominio (la solución que buscan). Un ejemplo útil cubre un conjunto de posibilidades que podrían ocurrir en la vida real. Así, los consumidores no arrancan desde cero (analizar, diseñar y programar el componente), sino únicamente

tienen que adaptar el componente a su problema. Ellos empiezan con un sistema corriendo y lo van ajustando a sus necesidades.

Los consumidores deben recibir cualquier preparación educacional especial que un proyecto demande, por ejemplo, educación general de reuso, adecuación de técnicas estructuradas para el reuso, ingeniería de aplicaciones, diseño basado en objetos, técnicas de programación orientada a objetos y capacitación en herramientas computacionales.

4.3.2.6 El Líder de un equipo de consumidores.

Para esta estructura organizacional se considera todo el perfil del líder de proyectos del modelo general. Es un líder de proyecto quien coordina toda la administración del desarrollo del producto pero incluyendo el reuso de software. Tienen la obligación de incluir el uso de los componentes reusables en la construcción del software final.

4.3.2.7 Comité de evaluación de componentes.

Para la organización productores-consumidores el comité está integrado por el director y todo el equipo de productores. Desempeña todas las funciones del modelo general.

4.3.2.8 Otros puntos importantes a considerar.

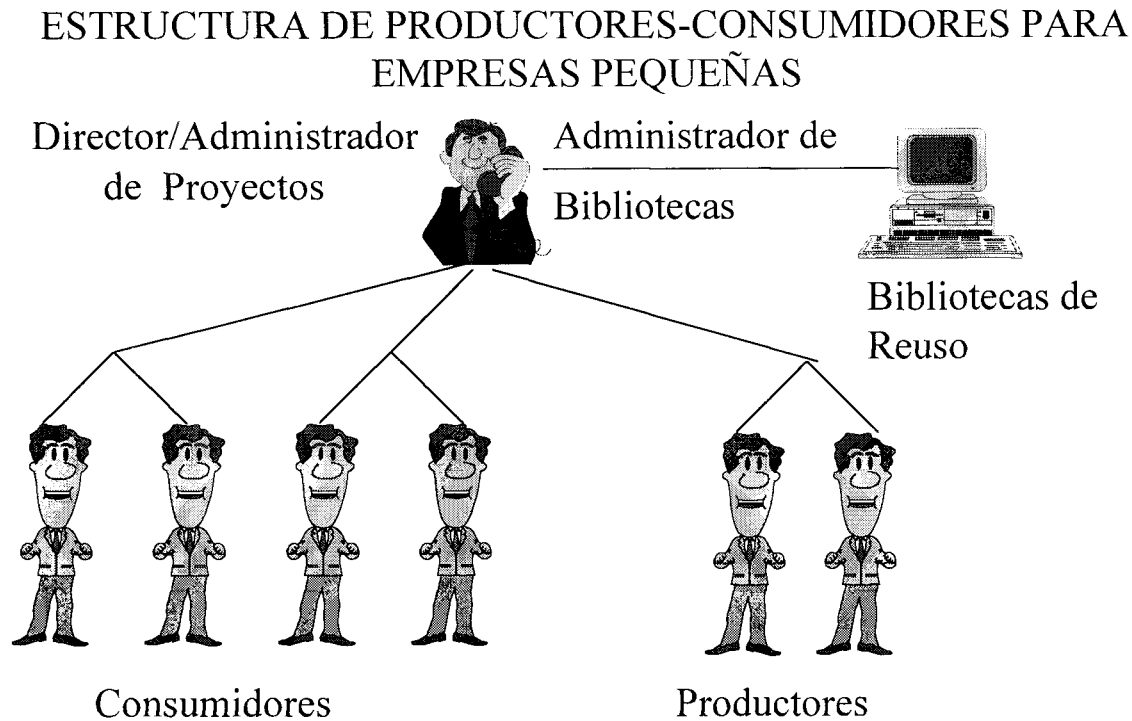
La inhabilidad para efectuar definición de producto es una mayor fuente de frustración para los productores, ya que les es muy satisfactorio contribuir con el diseño de productos finales y por lo tanto les interesa conocer la aplicación real de sus componentes. Esta redefinición de funciones de trabajo atribuye responsabilidades para la definición de productos a ambos productores y consumidores.

Practicantes e investigadores de reuso de software enfatizan la sabia separación de las funciones de trabajo de productores y consumidores como un factor clave en el éxito de un programa de reuso [FAFCHAMPS,94]. Sin una buena clasificación entre las actividades de productores y consumidores y de un entendimiento del modelo anterior, es difícil comparar programas de reuso que son utilizados bajo diferentes estructuras organizacionales considerando la relación productor-consumidor. Además, es importante poder comprender el por qué la resistencia parece más difícil en algunas estructuras organizacionales que en otras.

4.3.3 Director de productores y consumidores para empresas pequeñas.

Descripción.

La estructura propuesta es la siguiente:



Esta estructura está formada de un director principal del departamento, un equipo de productores de componentes reusables, equipos de consumidores, un comité de evaluación de componentes y las bibliotecas de reuso.

En esta estructura sucede que los proyectos son pocos y un director puede tener la capacidad de administrar directamente a los equipos de consumidores, a la vez, administrar al equipo de productores.

Con relación al equipo de productores, el director participa en la construcción de componentes y es el administrador de las bibliotecas de reuso. Además, en el desarrollo de un producto se necesita la colaboración paralela de un equipo consumidor, del equipo de productores y del director.

Con relación a los equipos consumidores, participa activamente dentro del desarrollo de los proyectos, administrando los tiempos de cada fase del proyecto y coordinando las actividades de reuso, haciendo énfasis en la utilización de los componentes reusables de las bibliotecas.

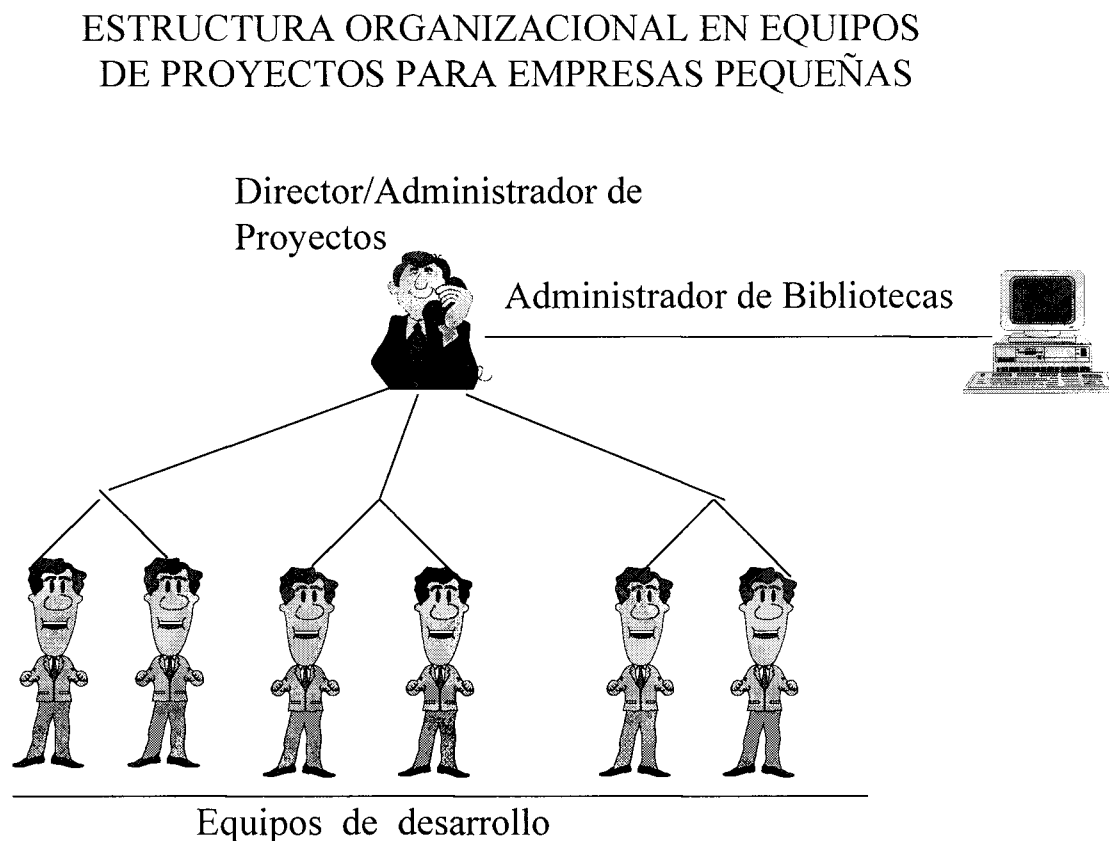
El comité de evaluación está formado por el director y todo el equipo de productores.

Por ser una empresa pequeña, las relaciones interdepartamentales no tienen problema. Se fomenta mucho la comunicación informal. Tienen a un único jefe a quien reportan tanto equipos de proyectos como equipo de productores. Si un proyecto demanda mucha atención de un productor, será el director quien resuelva este conflicto y asigne prioridades.

4.3.4 Director de equipos de desarrollo de software para empresas pequeñas.

Descripción.

La estructura propuesta es la siguiente:



Las variantes importantes en esta estructura están en considerar todo el modelo general pero asignando directamente la responsabilidad de los equipos y de las bibliotecas de reuso al director, ya que por haber pocos proyectos es posible que tenga capacidad de hacerlo. El director es quien podrá dirigir de cerca todas las actividades del desarrollo de los productos, y a la vez, apoyar en las actividades de reuso.

El director del departamento de desarrollo es el responsable de distribuir las actividades de reuso en cada equipo de desarrollo, estableciendo la división de las tareas, tanto para aquellos que desarrollarán componentes para el proyecto como aquellos que se dedicarán a la construcción del producto final. El director es entonces quien actúa como un líder de reuso de proyectos y por lo tanto tiene la total responsabilidad de incorporar actividades de reuso en su proyecto.

El director asume la responsabilidad de ser el coordinador de bibliotecas de reuso para encargarse de toda la administración de las mismas y tiene la responsabilidad de fomentar su utilización.

Cada equipo de proyecto tiene la responsabilidad de generar sus componentes y a la vez debe proponerlos al director para su posterior evaluación por el comité de evaluación.

El comité de evaluación deberá estar integrado por el director y aquellos desarrolladores más experimentados que considere el director.

4.4 Incentivos para promover reuso de software.

4.4.1 Introducción.

Una manera de incrementar la cantidad de reuso es incrementar la cantidad y variedad de software reusable disponible para los proyectos en las bibliotecas. Incrementando la cantidad de software que un programador puede reusar provoca que dicho programador más probablemente vea las bibliotecas primero.

Para llevar a cabo esto, un programa de incentivos debe ser establecido por el comité de evaluación de componentes para ayudar a poblar la biblioteca central de reuso, así como animar mayor reuso en los programadores individuales y en los proyectos. Este programa de incentivos está dividido en dos planes separados, uno para el reusador y uno para el contribuidor, haciendo énfasis en el último.

4.4.2 Incentivos para el reusador: incentivos para el reusador toma la forma de reconocimiento individual y por equipo. En una base regular (típicamente trimestral), el equipo de programación que ha reusado la mayor cantidad de software (haciendo uso de un mayor número de componentes, haber construido una aplicación con la integración de un determinado número de líneas de código reusado en proporción a las líneas de código de dicha aplicación) de una biblioteca de reuso es reconocido por el Consejo de Revisión de Reuso con un premio por equipo. Más de un equipo puede ser reconocido en el mismo período. Además del premio en equipo, cada reusador individual es animado a ganar un reconocimiento.

Se asignan puntos por cada mil líneas de código ahorradas por el reuso de componentes o partes de ellos. Los ahorros son calculados usando los beneficios estimados de reusar la parte/componente con relación al producto final. Cada ocurrencia de la parte/componente reusado constituye un reuso.

4.4.3 Incentivos para el contribuidor: El énfasis del programa de incentivos está puesto en el contribuidor, puesto que es el contribuidor individual quien debe escribir el software que cumpla el criterio de entrada a la biblioteca central de reuso. El reconocimiento del contribuidor está basado en un sistema de suma de puntos y obtención de placas. La primera vez que un individuo contribuye a la biblioteca central se le asignan puntos y un reconocimiento. Puntos adicionales son asignados cuando el software de dicho individuo es usado por otros individuos, donde el número de puntos aumenta a medida que dicho software es reusado una y otra vez. Así, la mecánica es ganar puntos por contribuir a las bibliotecas de reuso y también ganar puntos cada vez que sean reusadas las contribuciones. Al final, después de sumar una cierta cantidad de puntos, un incentivo económico debe ser asignado.

Entonces, cada vez que el mismo individuo contribuye con otro componente de software, puntos son concedidos al individuo por la nueva contribución y puntos adicionales empezarán a sumarse cada vez que esta aportación sea reusada por otros. Todos los puntos deberán estar registrados por individuo y a la vez con sus correspondientes componentes, de tal manera que se lleve un seguimiento de puntos por componente de un cierto individuo.

Puntos son concedidos cuando una parte que constituye un componente es reusada fuera del equipo del individuo. Ésta es una manera de premiar partes populares de componentes.

Cuando un equipo ahorra líneas de código por reusar un componente/parte en su proyecto, el autor individual gana puntos, es decir, tantas veces como se haya reusado la parte ganará puntos dicho autor. Ahora, si más de un autor cooperó en un componente/parte, los puntos se asignan de acuerdo con la relación al porcentaje de propiedad del componente/parte.

En una base regular (pudiera ser trimestral), se hacen referencias cruzadas en la biblioteca de reuso central para sumarizar puntos totales para cada autor por cada componente. Cuando un autor alcance una placa, será públicamente reconocido y recibirá además un incentivo económico. Para regular lo anterior, debe limitarse la cantidad de puntos que un autor puede recibir por un componente.

CAPITULO 5.

Instalando un programa de reuso.

5.1 Consideraciones previas al programa.

La alta administración de la organización debe saber que el reuso de software es un aspecto de negocios que involucra transición tecnológica y cambio organizacional. Se requiere del apoyo de la alta administración para poder soportar cada una de las fases que deberán ser llevadas a cabo en un programa de reuso. En particular, los tres elementos más críticos que conducen a un programa de reuso exitoso son el soporte y el liderazgo de la administración, el cambio organizacional y la creación de una mentalidad de reuso.

Como se ve, primero debemos asegurar el compromiso de la administración para después soportar todo el cambio de procesos que se debe de dar con la introducción de las actividades de reuso. Dado que el reuso enlaza proyectos anteriormente independientes (ya que podemos hacer uso de software generado en proyectos anteriores), decisiones que fueron una vez de interés a un único proyecto ahora involucran varios proyectos. Decisiones tales como el uso de procesos comunes de software, herramientas y lenguajes, así como de inversión a largo plazo en entrenamiento, construcción y mantenimiento de software reusable, y crear una organización de soporte pueden raramente ser llevados a cabo sin el involucramiento y soporte de la alta administración.

El conocimiento por parte de la alta administración de los beneficios que implica la introducción del reuso en los procesos de desarrollo de software es importante. Ellos deben entender los cambios organizacionales que se deben generar para implementar un programa de reuso. Dentro de estos cambios está el fomentar una cultura de reuso y desarrollar una organización orientada al reuso, donde implica educación y entrenamiento para los equipos de desarrollo, y estructuras organizacionales apropiadas.

Para llevar a cabo esta actividad hay un buen número de opciones organizacionales (por aplicación, por función-proyecto, en una estructura híbrida, separación o unión de productores y consumidores) y tecnológicas (ambientes de desarrollo, herramientas CASE) entrelazadas que deben ser consideradas, con diferentes características costo/beneficio,. La alta administración deben apoyarse en personal experto para hacer una evaluación y comparación de las posibles opciones que mejor se apliquen a su organización.

Dentro de un programa de reuso se deben marcar las actividades que se deben llevar a cabo para conseguir el proceso de adopción de reuso. Dicha adopción se va logrando de manera incremental a medida que se ganan experiencias de reuso. Estas

experiencias pueden ir dictando cambios en el mismo programa a medida que pasa el tiempo.

Así entonces, antes de empezar un programa de reuso es obligatorio seguir los siguientes puntos:

1. Incluir al reuso de software dentro de las metas de la organización.
2. Instituir un grupo de personas que investiguen la situación actual de la empresa para conocer su estado actual en el desarrollo de software:
 - Procesos de desarrollo y mantenimiento de aplicaciones.
 - Dónde se encuentra con relación a su portafolio de aplicaciones pendientes y de actualización de aplicaciones.
 - Cuáles son las habilidades de los desarrolladores y administradores de proyectos.
 - Cómo ayuda su estructura organizacional.
 - Cómo apoya su infraestructura tecnológica.
 - Qué paradigmas de desarrollo conocen y utilizan (orientado a objetos, estructurado, paradigma de reuso, lenguajes de cuarta generación).
 - Evaluar tiempos de liberación de software y estimar costos.
 - Detectar posibles factores que podrían bloquear la implementación del programa: económicos, culturales, de actitud de los diseñadores y programadores de aplicaciones, por falta de tiempo, por alta carga de trabajo, por el tiempo que duraría el programa para después empezar a ver los esperados beneficios y por implicar formas muy novedosas de hacer las cosas.
3. Luego de tener un diagnóstico de lo anterior, hacer recomendaciones acerca del reuso en la empresa con relación a lo siguiente:
 - *Recomendaciones educacionales.* Fuertes bases de ingeniería de software a los diseñadores y programadores de aplicaciones, tanto en conceptos y métodos como de aplicación de tecnológica. Cursos de entrenamiento en reuso de software tanto para diseñadores y programadores como para administradores de proyectos, de tal forma que se desarrollen habilidades y se enseñen las técnicas requeridas en reuso.
 - *Recomendaciones motivacionales.* Hacer énfasis en incentivos, métricas para reuso y trayectorias de carrera en la organización.
 - *Recomendaciones organizacionales.* El administrador general del desarrollo de software debe generar su propio programa de reuso y presentar los recursos necesarios a la alta administración.. Tener un comité que vigile el buen funcionamiento del programa. Habilitar los cambios necesarios en las estructuras organizacionales.
 - *Recomendaciones tecnológicas.* Determinar un proceso de desarrollo de software específico basado en el reuso, incluyendo desarrollo de estándares de programación y de requerimientos en herramientas computacionales.

Las recomendaciones que se hagan sirven de base para la generación del programa. En dicho programa se deben contemplar las necesidades de educación para todo el departamento de desarrollo de software y de los administradores que estén cerca del programa; los mecanismos de motivación para comenzar a crear bibliotecas de reuso y diseñar aplicaciones basadas en estas bibliotecas; las opciones de tecnología, considerando cuál es la mejor para facilitar la creación de componentes de reuso, el desarrollo de aplicaciones y la utilización de componentes en otros proyectos; y los factores organizacionales que ayuden a la correcta administración de los equipos de proyectos.

5.2 El programa de reuso.

Este paso incluye identificar objetivos organizacionales (productividad y objetivos de calidad) y oportunidades de reuso (por ejemplo, trabajar dentro de un único dominio para tener similares requerimientos y así aumentar la cantidad de software a ser reusado). Una organización puede activarse en diferentes dominios de aplicación, y el potencial de reuso en cada uno de estos dominios tiene que ser estimado. Es bueno recordar que las aplicaciones del sistema de información para la administración son bastantes estables y promedios altos de reuso son posibles [LANERGAN,84]. Sin embargo, no hay una manera fácil de descubrir qué tanto reuso es posible dentro de un dominio de aplicación sin realmente hacerlo sobre un período de años o llevar a cabo un estudio que identifique similitudes y diferencias entre sistemas relacionados dentro de un dominio. Un programa de reuso incluye los siguientes pasos:

1. Definir el alcance del programa de reuso.
2. Establecer objetivos de reuso razonables.
3. Identificar estrategias alternativas para la adopción de reuso.
4. Implementar la estrategia.

El alcance del programa de reuso consiste de escoger un dominio de aplicación que ofrezca el mayor potencial de reuso, los riesgos más bajos y los retornos sobre la inversión más rápidos. Una vez que el alcance está identificado, la organización debe establecer objetivos de reuso que el equipo pueda lograr con un esfuerzo razonable, dependiendo de una autoevaluación de sus procesos administrativos y técnicos.

La identificación de los objetivos de reuso sugiere un número de estrategias candidatas, cuyos costos y beneficios son analizados en este paso. Una estrategia de adopción es vista como una combinación de un enfoque técnico y de una estrategia que muestra los pasos a seguir para conseguir los objetivos de reuso planeados. Se recomienda empezar con una estrategia con adopción de reuso incremental, ya que es a base de las experiencias que se vayan viviendo como la organización va teniendo mayores beneficios del reuso.

Basado en los análisis comparativos de las varias estrategias candidatas de adopción, seleccionar una o hacer la combinación de estrategias. En esta etapa, un plan que

muestre los pasos a seguir bien detallado es producido. Las decisiones tales como cuánto del dominio reusable se cubrirá el primer año, el paso de adquirir los componentes reusables y algunos análisis de costos se deben hacer en esta etapa.

Para implementar el programa de reuso se debe contar ya con una estructura organizacional basadas en alguno de los modelos propuestos en este trabajo.

5.3 El programa propuesto.

1. Asignar a la autoridad más alta en el departamento de desarrollo de sistemas como el director del programa encargado de todas las actividades de reuso (director de sistemas, gerente de sistemas, jefe de sistemas). Darle completo entrenamiento en ingeniería de software y enseñarle la estrategia de desarrollo de aplicaciones basada en el reuso de software. Proporcionarle un curso de reuso para administradores de ingeniería de software. Durante el entrenamiento, es necesario que se le muestren casos de la aplicación del reuso y de los beneficios que se pueden lograr.
2. El director de sistemas escogerá una estructura organizacional basada en alguno de los modelos propuestos en este trabajo, considerando el número de proyectos a realizar y a la diversidad de dominio de las aplicaciones.
3. Proporcionar a todo el personal de desarrollo con entrenamiento en reuso, considerando el análisis y diseño orientado a objetos, reuso de software y análisis de dominio.
4. Analizar el dominio objetivo inicial junto con los equipos involucrados- el dominio de las aplicaciones que debemos abordar como resultado de las oportunidades del mercado o de las necesidades de la empresa. Se recomienda empezar con una estrategia de un único dominio para poder aprovechar más el potencial de reuso, y así ganar rápidamente una experiencia positiva en el reuso.
5. El director del programa de reuso establece los estándares con relación a la documentación y programación de cualquier desarrollo -considerando una metodología adoptada para dicha documentación, abarcando las fases de planeación, análisis, diseño e implementación/codificación-, de los componentes de reuso y de su clasificación correcta. Más adelante, al establecer un comité de evaluación de componentes reusables, se podrán revisar los estándares de documentación y generar los cambios en base a las experiencias obtenidas y a las propuestas del mismo comité.
6. Seleccionar y obtener herramientas computacionales para el desarrollo de sistemas computacionales.
7. Apoyar a los equipos involucrados en el desarrollo de las aplicaciones a dominar las herramientas computacionales.
8. El director asignará los proyectos iniciales a los equipos escogidos y se trabajará sobre los diseños preliminares de las aplicaciones.
9. Establecer un comité de evaluación de componentes reusables.
10. Al término de los diseños de las aplicaciones marcar los objetivos de reuso, considerando a la efectividad de reuso como su principal objetivo, la cual se ve como la razón de los beneficios de reuso a los costos de reuso.

11. Pasar a diseñar e implementar los componentes reusables más genéricos, cumpliendo con los estándares de programación y documentación.
12. Diseñar y poblar una biblioteca central de componentes reusables.
13. Empezar el diseño detallado y la programación de todo el software con componentes reusables.
14. Definir un conjunto de métricas para estimar beneficios actuales y beneficios de tener una biblioteca de componentes reusables y un conjunto de aplicaciones con un potencial a ser reusadas.
15. Idear e implementar un programa de incentivos para motivar el reuso de software.

5.4 Consideraciones para el seguimiento de un programa de reuso.

- a) El líder de un proyecto debe revisar los diseños detallados y la programación de aplicaciones, buscando siempre la utilización máxima de componentes reusables.
- b) Observar la actitud de los desarrolladores ante el proceso de desarrollo de software basado en el reuso.
- c) Cuestionar a los desarrolladores sobre la facilidad de utilizar componentes.
- d) Vigilar el apego a los estándares de programación.
- e) Determinar cómo va el proceso de adaptación al paradigma del reuso.
- f) Observar cómo se da la comunicación entre personal de desarrollo.
- g) Determinar si es necesario cambiar algo.
- h) Monitorear el cumplimiento de fechas de liberación de proyectos.
- i) Evaluar los beneficios esperados al término de cada proyecto.
- j) Presentar y discutir los beneficios obtenidos al término de un proyecto, determinando lo que pudo hacerse mejor.
- k) Extraer las experiencias obtenidas por los diferentes equipos y discutir las en reuniones generales del departamento de desarrollo.
- l) Si se necesita capacitación en algún proyecto, hacer lo posible por efectuarla.

CAPITULO 6.

Mapeo de los modelos en casos reales de reuso de software.

6.1 Introducción.

En este capítulo se presentan dos empresas que ayudaron en la definición de las características propuestas en el capítulo tres y en la identificación de los elementos del modelo general. Las empresas sirvieron como un medio de observación para determinar las características de una organización flexible de desarrollo de software y de los elementos que debe contener una estructura organizacional de desarrollo de software para introducir el resuso. .

6.2 INCAFI.

El Instituto Nacional de Capacitación Fiscal es una empresa de desarrollo de software requiere del trabajo de profesionistas de distintas especialidades para idear, planear y construir software para la educación y sistemas de información. Esta empresa trata de cumplir con sus clientes lo más rápido que sea posible, proporcionando productos de software de alta calidad.

Gracias a la amplia experiencia que INCAFI ha venido obteniendo de sus proyectos pasados, sus procesos principales de negocios se han venido perfeccionando para contribuir totalmente en el logro de las metas de la empresa.

Objetivo de la Empresa: Desarrollar productos de software para la educación y sistemas de información.

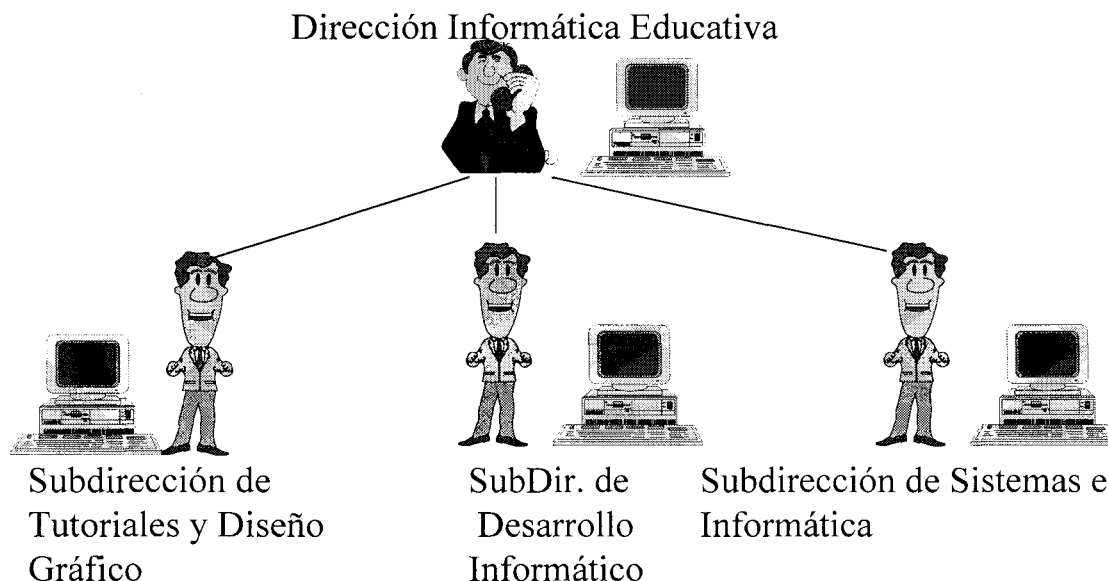
Productos: Tutoriales Convencionales y Tutoriales Multimedios, Sistemas de Información de Comercio Exterior y Autonorma (bibliotecas de leyes y reglamentos).
En los últimos dos años : se ha desarrollado 72 tutoriales y 2 sistemas de información.

Clientes: Subsecretaria de ingresos, agencias aduanales, gobiernos de los estados mexicanos (finanzas y recaudación fiscal) y sistemas de soporte a INCAFI.

La estructura organizacional de la empresa se encuentra en la página siguiente.

INCAFI

ESTRUCTURA ORGANIZACIONAL



1. Subdirección de Tutoriales y Diseño Gráfico:
Definen todo el diseño y el contenido de cada tutorial.
 - Didácticos.- Obtienen y capturan la información de tutoriales.
 - Diseñadores Gráficos.- Cualquier gráfico que deba contener un tutorial (diseños gráficos por computadora, fotografías, etc.).
2. Subdirección de Desarrollo Informático:
Arma y empaqueta tutoriales con herramientas de animación y lenguajes de programación (incluye textos, imágenes, videos).
 - Informáticos.- Crean módulos reusables y ensamblan tutoriales.
3. Subdirección de Sistemas de Información.
Se encarga del desarrollo y mantenimiento a sistemas de soporte INCAFI, sistemas de información a clientes y de un proyecto para bibliotecas electrónicas.
 - Informáticos.- Desarrollan y mantienen versiones de los sistemas; utilizan reuso.

Análisis de la Empresa con relación a las características propuestas.

1. La distribución de tareas está bien dividida: un equipo determina objetivos, funcionalidad y contenidos de tutoriales y otro se dedica a la programación y prueba del tutorial. Para ambas etapas, INCAFI investiga y adquiere tecnologías computacionales que le apoyen en sus procesos de construcción de software para tutoriales. Por ejemplo, con lenguaje Visual Basic se construyó un componente -

Editor de Tutoriales- que asiste al experto en el tema para capturar toda la información que deberá estar en el tutorial (tópicos, preguntas y respuestas, evaluaciones, casos dirigidos); para la generación de animaciones también se tienen aplicaciones necesarias. Toda la información de tutoriales y de sus componentes es clasificada y almacenada para su posible reuso futuro. Esto da un diseño basado en sus procesos principales.

2. El área de definición de tutoriales y de diseño gráfico establecen con el cliente las características y funciones del tutorial que se necesita. Aquí se está muy pendiente de lo que el cliente necesita para luego elaborar un pequeño prototipo que será mostrado al cliente para que verifique sus requerimientos. Dado que INCAFI tiene ya tiempo trabajando con sus diversos clientes, ha creado vínculos muy fuertes con ellos y el entendimiento de lo que necesita el cliente se ejecuta sin problemas. Si el cliente necesita una modificación al revisar un prototipo -ya sea que decidió incluir algunos temas adicionales-, INCAFI tiene la flexibilidad de ordenar que se regrese a una etapa de definición y luego regresar a la etapa de diseño y construcción. Para esto se trabaja muy armónicamente con las relaciones entre las subdirección de tutoriales/diseño gráfico y la subdirección de desarrollo informático. Esto nos da la característica de enfoque en el cliente y de soportar alineación a través de funciones.
3. Entre las subdirecciones se manifiesta una fluída comunicación informal e informal, lográndose un continua retroalimentación. Esta retroalimentación ayuda a mejorar la coordinación de las actividades de los equipos que ponen resultados de ciertas tareas a disposición de otros (por ejemplo, desarrollo informático solicita a los diseñadores del tutorial información sobre orden de reactivos y evaluaciones), reacomodando tareas en los procesos cuando sea necesario. Como INCAFI busca el desarrollo de tutoriales lo más paralelamente posible, hace análisis de tutoriales concurrentes y de aquellos que se deberán elaborar en el futuro para determinar el alcance del dominio y sacar ventaja de ésto. Las personas de cualquier departamento involucradas en algún tutorial hacen juntas de trabajo para acordar los requerimientos de tutoriales (considerando la estrategia de un único dominio para aprovechar mayores beneficios) y así poder considerarlos en la construcción de cada elemento del tutorial (ir de lo más general a lo específico). Esta coordinación de funciones marca la alineación a través de funciones y de que muestra un comportamiento adaptativo.
4. La rapidez en la elaboración de cualquier software o de tutoriales es una necesidad que esta empresa tiene. Para poder mejorar dicha rapidez, la empresa dedica tiempo para medir tiempos de liberación de productos y los compara contra proyectos anteriores, examinando métodos de trabajo e identificando las ventajas obtenidas al reusar proyectos anteriores. Están convencidos que el reuso de material de proyectos anteriores acelera el proceso de diseño y construcción de otros. Lo anterior indica el aprendizaje que vive la empresa, el cual está basado en sus experiencias de proyectos anteriores.

La estructura organizacional.

La división de tareas está bien marcada con relación a la conceptualización y de diseño de tutoriales y a su programación. Podemos decir que están organizados por funciones, donde cada subdirector de departamento puede estar en contacto con la evolución de un producto y a la vez estar coordinando a su gente para lograr el mayor paralelismo posible en las tareas. Así, en el departamento de desarrollo informático tenemos gente programando y ensamblando tutoriales y gente creando componentes (por ejemplo, editor de tutoriales, generador de crucigramas, captura de evaluaciones, salvador de pantallas, graficadores de resultados) que serán reutilizados por varios proyectos paralelos o futuros.

Para cada tutorial, el subdirector de diseño gráfico y de desarrollo informático hacen una junta para acordar los requerimientos y así plantear lo que podrá ser reusado y aquello que tiene que ser hecho desde su diseño. Ellos están convencidos que esta parte del análisis de tutoriales es importante para poder, por un lado, acordar lo necesario para el tutorial, y por otro, el ganar una visión de cómo hacerlo más rápido aprovechando lo que ya se tiene (reusar la estructura general de un tutorial anterior).

Cada equipo de desarrollo en la subdirección de desarrollo informático tiene almacenado el proyecto en sus computadores individuales y tienen acceso al servidor de componentes y proyectos anteriores. Ellos solicitan la asistencia a su jefe inmediato (el subdirector) para buscar información de reuso. Ahí permanece el proyecto hasta que es terminado y liberado. Antes de que un proyecto liberado sea pasado a la biblioteca de proyectos liberados y para reuso, el equipo informático lo documenta y pone a disposición del subdirector para su inclusión en dicha biblioteca (ubicada en el servidor). Si algún equipo desarrollo algún componente nuevo, éste es almacenado en la biblioteca de componentes ubicada en el servidor de la subdirección.

Para poder clasificar y almacenar ordenadamente sus componentes y proyectos, el subdirector de desarrollo informático hace funciones de administrador de bibliotecas de software reusable. Se hace una diferencia entre componentes reusables y proyectos completos, ya que los componentes que se han programado son incluidos en los nuevos proyectos en alguna barra de menus y los proyectos completos se reusan pero demandan un esfuerzo de adecuación mayor para el desarrollador.

Después de que cada producto es liberado, el subdirector de desarrollo informático recibe todo su material, verifica la documentación y se almacena en una biblioteca de proyectos -en el servidor a cargo de este subdirector- para que esté disponible para su futura consulta pero sin modificación, ya que sólo este subdirector podrá autorizar algún cambio (actualización para mejora o corrección). Si el cambio ya es muy trascendente, se genera una nueva versión del tutorial o se crea otro con diferente nombre.

Por la cantidad de tutoriales que hay que estar produciendo y modificando, el director de informática educativa tiene la responsabilidad de cuidar la calidad de tutoriales y mejorar los tiempos de liberación, es decir, mayor rapidez en la producción. Debido a

esto, él impulsa el reuso de proyectos anteriores y la planeación estratégica de dominios (comparar tutoriales y sacar provecho de similitudes).

6.3 SINERGY.

Sinergy Software Development S.C. es una empresa formada por profesionales del área informática y administrativa cuyo objetivo es conjuntar su experiencia y conocimientos para proporcionar un servicio de consultoría de alta calidad en informática y las áreas administrativas involucradas.

Tienen un compromiso con las empresas al ofrecerles una metodología integral en todas las etapas de implantación de sistemas, utilizando los recursos tecnológicos más avanzados en hardware y software, y la experiencia y soporte de un equipo de profesionales enfocados a analizar e implantar las soluciones más adecuadas para cada empresa.

Objetivo: Desarrollar, adecuar e implantar software administrativo que cubra las necesidades de las empresas mexicanas empleando las herramientas más avanzadas del mercado.

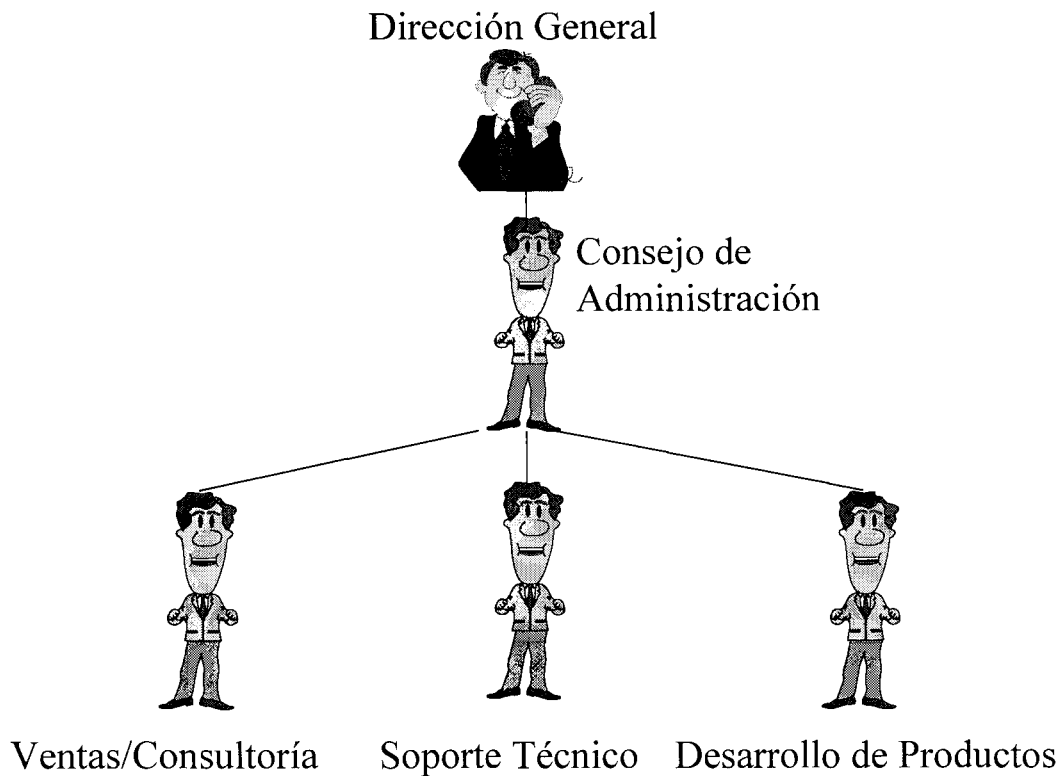
Productos: SCENARIO es sistema integral que contiene un conjunto de módulos en las áreas Comercial/Administrativa, Contabilidad y Finanzas, Recursos Humanos, Información Gerencial, Administración del sistema y Manufactura.

Clientes: Cualquier empresa pública y privada, tanto de manufactura como de servicios (Camino y Puentes Federales de Ingresos y Servicios Conexos, PEMEX Exploración y Producción, BMZ Agente de seguros, entre otros).

La Estructura Organizacional general de SINERGY es la siguiente:

1. Director General.
 - 1.2 Consejo de Administración.
 - 1.2.1 Ventas/Consultoría.
 - 1.2.1.1 Acceso a clientes.
 - 1.2.1.2 Análisis de factibilidad.
 - 1.2.1.3 Análisis de procedimientos.
 - 1.2.2 Soporte Técnico.
 - 1.2.2.1 Los propios productos de Sinergy (sistemas integrales).
 - 1.2.2.2 Herramienta de otras casas de software (Novell, LogicWoks, Centura)
 - 1.2.3 Desarrollo de productos de software.
 - 1.2.3.1 Equipos de desarrollo de sistemas.

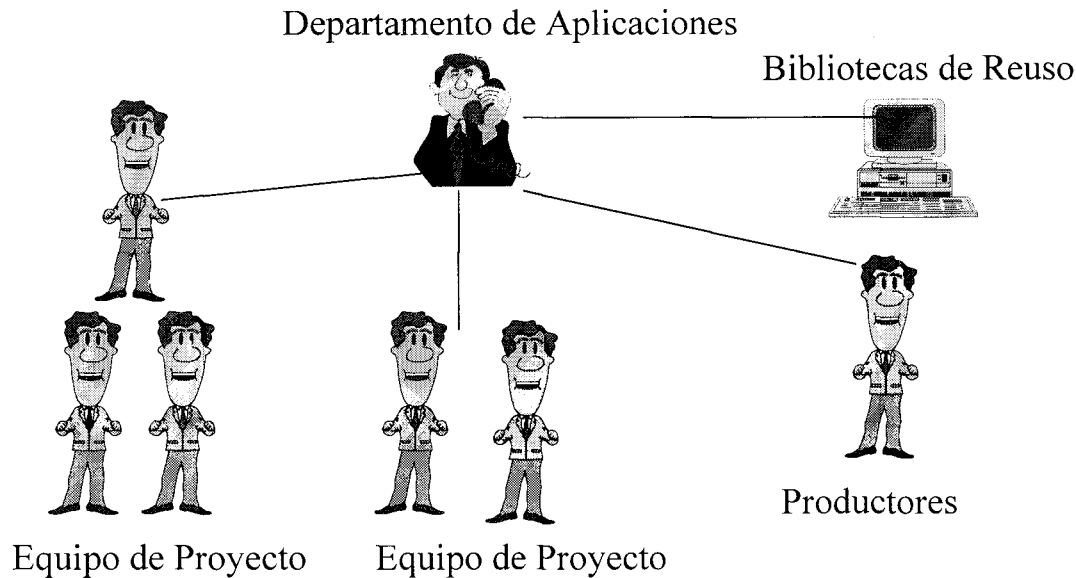
La Estructura Organizacional es la siguiente:



La estructura organizacional del departamento de desarrollo utiliza un enfoque de productor-consumidor. El jefe de este departamento es el director de todos los proyectos que se llevan a cabo en la empresa, y tiene juntas periódicas con el departamento de ventas/consultoría para determinar el plan de desarrollo de productos y asignación de prioridades en proyectos. La estructura organizacional de este departamento (desarrollo de aplicaciones/productos) es la que se muestra en la página siguiente.

SINERGY

ESTRUCTURA ORGANIZACIONAL



SINERGY tiene una estructura organizacional funcional, donde cada área funcional desempeña sus funciones de Ventas/Consultoría, Soporte Técnico de productos y Desarrollo/Adecuación/Implementación de los módulos de SCENARIO bajo la dirección del Consejo de Administración. La estructura funcional apoya la creación de equipos de proyectos, donde a cada proyecto se le da el soporte de cualquiera de las funciones que requiera de cada área. El jefe funcional del área de desarrollo asigna un líder de proyecto. Este jefe además cuenta con un equipo de desarrollo/actualización de componentes reusables los cuales son utilizados en todos los programas que constituyen a SCENARIO.

Con relación al desarrollo de sistemas, SINERGY dedica un equipo para hacer análisis e ingeniería de dominio -equipos productores- y así poder producir aquellos componentes que serán puestos a disposición de los sistemas de SCENARIO. Todos los componentes que se han producido están perfectamente bien documentados y se sigue un estándar de programación y de documentación. SINERGY está actualmente en una etapa donde ya tiene una biblioteca de componentes que le sirven para soportar el desarrollo de todos sus sistemas. El administrador de las bibliotecas es el jefe del departamento de desarrollo y además es el líder del equipo de productores. Juntos forman un comité de evaluación de todo el software que se genera en la empresa, con la finalidad de extraer componentes reusables o de identificar sistemas con un potencial de reuso. SCENARIO está diseñado a base de subsistemas que a su vez son arquitecturas de software que pueden ser utilizadas para otros sistemas, además de contener componentes genéricos de alta calidad.

Con respecto a los equipos de proyectos, el líder de cada proyecto tiene la responsabilidad de administrar todo el proyecto y de hacer un uso obligado y correcto de las bibliotecas de componentes. Si no hay líder, es el jefe de desarrollo quien lleva a cabo esta actividad.

Análisis de la Empresa con relación a las características propuestas.

Dado que la empresa maneja varios proyectos concurrentemente, existen acuerdos para coordinar actividades entre jefes funcionales y líderes de proyectos. Un líder de proyecto puede solicitar directamente el apoyo de alguien de soporte técnico o la asistencia de un productor, rebasando los límites organizacionales impuestos por la estructura organizacional. El equipo de productores conocen los requerimientos de los sistemas SCENARIO, y en base a esto desarrollan software para satisfacer sistemas actuales, paralelos y futuros (los sistemas futuros son planeados por el análisis de dominio que lleva a cabo SINERGY para sus sistemas de SCENARIO). Esto muestra la característica de alineación a través de funciones.

Al jefe de desarrollo le interesa mucho el mejorar sus procesos de desarrollo y la capacidad para reaccionar ante las necesidades de los clientes, por lo que compara la rapidez en sus desarrollos, los métodos utilizados, las herramientas, la evolución de los sistemas y la calidad y apoyo de sus componentes de reuso. Esto da la característica de aprender y reusar experiencias.

Este aprendizaje y reuso de experiencias lo aplica para reacomodar actividades de trabajo, ser flexible en el cruce de coordinación entre áreas funcionales (porque un proyecto requiere el apoyo de las tres áreas) y fomentar la retroalimentación entre clientes, proveedores de software y el mercado de su sistema SCENARIO para seguir obteniendo información. Aquí, SINERGY ha modificado actividades de trabajo haciendo uso de herramientas de desarrollo de software para equipos de trabajo, compartiéndose modelos, componentes y bases de datos. Lo anterior nos muestra la ocurrencia de la característica de comportamiento adaptativo.

La implementación de los sistemas de SCENARIO están condicionados a los requerimientos precisos de cada cliente en particular. SINERGY ha estudiado las funciones de los procesos de SCENARIO y de su arquitectura de datos para poder enfocar más sus sistemas en cualquier tipo de empresa, así como de incluir mayor capacidad de configuración en SCENARIO para adaptarse al cliente. SINERGY sabe que el éxito que ha tenido SCENARIO es por enfocarse en las necesidades de cada cliente y en comprender el área del dominio de cada subsistema. Esto nos da la característica de mantenerse enfocado en el cliente.

Objetivo: Desarrollar, adecuar e implantar software administrativo que cubra las necesidades de las empresas mexicanas empleando las herramientas más avanzadas del mercado.

La estructura organizacional de SINERGY -ventas/consultoría, soporte técnico y desarrollo- está diseñada para proporcionar todo el servicio de creación e implementación de sistemas computacionales que requiere cualquier organización. Cada área funcional contribuye directamente con un valor agregado para la generación del sistema o servicio que el cliente necesita, y de esta manera SINERGY cumple con su objetivo principal. Para poder integrar más sus funciones, SINERGY utiliza herramientas de planeación de procesos de negocios (herramientas de modelación para analizar, documentar y mejorar complejos procesos de negocios) y de análisis y diseño de sistemas de información. De esta manera se articulan las funciones y se soportan los procesos principales de SINERGY.

CAPITULO 7.

Conclusiones, Recomendaciones y Trabajos futuros.

7.1 Conclusiones y recomendaciones.

1. La Organización de personas y responsabilidades son factores relevantes para implementar estrategias de reuso.
2. La coordinación de funciones puede provocar problemas de comunicación y de ejecución de trabajos si no se estudian bien los proceso de desarrollo de software.
3. El soporte de la Alta Administración y de su conocimiento de los beneficios del reuso es un punto que se debe tratar antes de formular un programa de reuso.
4. Proporcionar educación y entrenamiento en temas de reuso a los equipos de desarrollo.
5. Proporcionar incentivos para promover el reuso entre los desarrolladores.
6. Considerar al reuso de software como una meta de la organización.

Beneficio que deja la tesis a la comunidad.

1. El beneficio que arroja esta tesis es la definición de los características que debe tener una organización flexible de desarrollo de software, y la identificación de los elementos de una estructura organizacional que soporte un programa de reuso de software.
2. Las consideraciones previas que deben tomarse en cuenta antes de planear un programa de reuso y el programa de reuso propuesto.

Beneficio que deja la tesis al autor.

1. Con este trabajo de tesis tuve la oportunidad de desarrollar mi capacidad de investigar. Lo considero como un valor agregado a mi persona porque soy un profesor en la licenciatura en Informática y la actividad de investigar es una tarea que deberé desempeñar por el resto de mi vida. Así, considero que esta experiencia fue una introducción a la investigación.
2. Con relación al tema de la tesis, aprendí que la tecnología no es el mayor impedimento para adoptar el paradigma del reuso de software, sino que hace falta estudiar los sistemas de trabajo que tienen las organizaciones para poder considerar un nuevo paradigma.

7.2 Trabajos Futuros.

Un trabajo futuro es la integración de herramientas computacionales de apoyo y de la organización del trabajo. En este trabajo se planteo la necesidad de una estructura de soporte necesaria para soportar un programa de reuso, pero es importante hacer un estudio de las distintas herramientas de apoyo a la ingeniería de software que están en el mercado para apoyar todo el ciclo de vida de productos de software.

REFERENCIAS.

- [AGRESTI,86] AGRESTI, William W. "Framework for a flexible development process". New paradigms for software development, William W. Agresti, ed. IEEE, 1986.
- [BARNES,87] BARNES, B.; Durek T.; Gaffney, J.; and Pyster, A.. "A framework and economic foundation for software reuse". Proc. Workshop Software Reusability and Maintainability, 1987.
- [BASILI, 91] BASILI, V.R. and Musa, J.D. "The future engineering of software: A management perspective". IEEE Computer. Vol. 24, No. 9. September, 1991.
- [BASILI, 92] BASILI, V.R.; Caldiera G.; McGarry F.; Pajerski, R.; Page, G; and Waligora, S. "The software engineering laboratory- an operational software experience". Proc. 14th Int'l Conf. Software Engineering, Melbourne, Australia. May, 1992.
- [BASILI,94] BASILI, V.R. and Green S. "Software process". IEEE Software. July, 1994.
- [BOEHM,87] BOEHM, B. "Improving software productivity". IEEE Software. September, 1987.
- [BOYLE,84] BOYLE, J.M and Muralidharan, M.N. "Program reusability through program transformation". IEEE transactions on software engineering, 10(5):574-588. September, 1984.
- [BUDD,91] BUDD, Timothy. An Introduction to Object-Oriented Programming. Editorial Addison-Wesley Publishing Co. Primera Edición. Impreso en Massachusetts, E.U.A.1991.
- [BRUCE,87] BRUCE, Burton." The reusable software library". IEEE Software. July 1987.
- [CALDIERA,91] CALDIERA, G. and Basili, V.R. "Identifying and qualifying reusable software components". Computer. Vol. 24, No. 2. Febrero, 1991.
- [CANFORA,95] CANFORA, Gerardo; Fasolino, Anna Rita; and Torterella Maria. "Toward reengineering in reuse reengineering processes". IEEE Software. Italy. 1995.
- [CARD,94] CARD, Dave and Comer Ed. "Why do so many reuse programs fail?". IEEE Software. September, 1994.

- [CÁRDENAS,94] CÁRDENAS, Sergio. "Una Panorámica de la Ingeniería de Software". Soluciones Avanzadas. num.8. Marzo-Abril 1994. Año 2.
- [COX,90] COX, B.J. " Planning the software revolution". IEEE Software. Vol. 7, No. 6. November, 1990.
- [CIMITILE,92] CIMITILE, A. "Toward reuse re-engineering of old software". IEEE Comp. Soc. Press. Proc. of 4th International Conference on Software Engineering and Knowledge Engineering. Capri, Italy. 1992.
- [CUSUMANO,89] CUSUMANO, M.A. "The software factory: A historical interpretation". IEEE Software. March, 1989.
- [DAVIS,93] DAVIS, T. "The reuse capability model : A basis for improving an organizations reuse capability". Advances in Software Reuse, Selected Papers from the Second Workshop on Software Reusability Advances in Software Reuse. Lucca, Italy. March, 1993.
- [DECKRO,90] DECKRO, R. and Jones, R. "The social psychology of project management conflict". European Operational Research. No. 64. 1993.
- [EDWARDS, 90] EDWARDS, J.M. and Henderson-Sellers, B. "The Object Oriented systems life cycle". Comm. ACM. Vol. 33, No. 9. September, 1990.
- [FAFCHAMPS,91] FAFCHAMPS, Danielle and Navarro, J.J. "Core Work Proces and Core System Capabilities". Accepted position paper for the European CSCW Developers Workshop. Amsterdam. 1991.
- [FAFCHAMPS,94] FAFCHAMPS, Danielle. Hewlett-Packard Laboratories. "Organizational factors and reuse". IEEE Software. 1994.
- [FISCHER,87] FISCHER, G. "Cognitive view of reuse and design". IEEE Software. July, 1987.
- [FRAKES,90] FRAKES, W.B. "An emperical framework for software reuse research". In Third Annual Workshop: Methods & Tools for Reuse. CASE Center, Syracuse University. June, 1990.
- [GARLAN, 94] GARLAN, David; Allen Robert and Ockerbloom, John. "Architectural mismatch: Why reuse is so hard ?". IEEE Software. 1994.
- [GRISS,94] GRISS, M.; Favaro, J. and Walton, P. "Managerial and Organizational Issues" in starting and running a software reuse program. IEEE Software. 1994.

- [GRISS,95] GRISS, Martin L. and Wosser Marty. "Making reuse work at Hewlett-Packard". Quality Time, IEEE Software. 1995.
- [GRUMAN,88] GRUMAN, G. "Early reuse practice lives up to its promise". IEEE Software. November, 1988.
- [HOROWITZ,84] HOROWITZ, E. and Munson J.D. "An expansive view of reusable software ". IEEE Trans. Software Engineering. Vol. 10, No. 5. 1984.
- [HUBERT,90] HUBERT, Laurence and Perdreau, Gerald. "Software Factory: using process modeling for integration purposes". IEEE Software. 1990.
- [JACOBSON, 92] JACOBSON, Ivar. Object-Oriented Software Engineering : A use case driven approach. Editorial Addison-Wesley. Revised fourth printing 93. Reprinted 94. Printed in U.S.A.
- [JONES,90] JONES, G. "Methodology/Environment support for reusability". Software Reuse: Emerging Technology, Will Tracz, ed. IEEE CS Press. 1990.
- [JOOS, 94] JOOS, Rebecca. "Software Reuse at Motorola". IEEE Software. September 1994.
- [KRISHNAMURTHY,94]KRISHNAMURTHY, Balachander. "Software Architecture and Reuse - An inherent conflict". IEEE Computer Society. 1994.
- [KRUEGER,92] KRUEGER, C.W. "Software Reuse". ACM Computing Surveys. ACM Press. Vol. 24, No.2. June, 1992.
- [LANERGAN,84] LANERGAN, R.G. and C.A. Grasso. "Software Engineering with reusable designs and code". IEEE Trans. Software Engineering. Vol.10, No. 5. September, 1984.
- [LIM, 94] LIM, W. "Effects of Reuse on Quality, Productivity, and Economics". IEEE Software. Vol. 11, No.5. September 1994.
- [LUBARS, 91] LUBARS, D. Mitchell. "Reusing designs for rapid application development". IEEE Software. 1991.
- [MACCHINI,92] MACCHINI, B. "Reusing software with ESTRO (Evolving Software Repository)". Proceedings. Fourth International Conference on Software Engineering and Knowledge Engineering. IEEE Comput, Soc. Press. Capri, Italy. June, 1992.
- [MARCOS,96] MARCOS, María del Socorro J. "Organizational Learning and Information Technology Assimilation: An Exploratory Study". Disertación Doctoral. 1996.

- [MARTIN ,91] MARTIN, James. Rapid Application Development. Macmillan Publishing Company. Impreso en U.S.A. 1991
- [MARTIN,94] MARTIN, James. Odell, James J. Análisis y Diseño Orientado a Objetos. Traductor: Oscar Alfredo Palmas Velasco, Facultad de Ciencias UNAM. Editorial Prentice-Hall Hispanoamericana. Primera Edición en Español. Impreso en México.1994.
- [McCLURE,92] McCLURE, Carma. Case is Software Automation. Prentice-Hall. U.S.A. 1992.
- [NEIGHBORS,84] NEIGHBORS. "The DRACO approach to constructing software from reusable components". IEEE Transactions on Software Engineering, Vol. SE-10, No. 5. Sept. 1984.
- [OROZCO,93] OROZCO, Octavio. "Nuevas Tecnologías para el Desarrollo de Aplicaciones". Soluciones Avanzadas. num.6. Noviembre-Diciembre 1993. Año 1.
- [OSTROFF,92] OSTROFF, F and Smith, D. "Redesigning the corporation: the horizontal organization". The McKinsey Quarterly. New York, U.S.A. 1992.
- [PIATTINI,95] PIATTINI, Mario G. and Daryanani Sunil N. Elementos y Herramientas en el desarrollo de Sistemas de Información: Una visión actual de la tecnología CASE. Addison-Wesley Iberoamericana. E.U.A., 1995.
- [PRESSMAN,88] PRESSMAN, Roger S. Making Software Engineering Happen. Prentice-Hall. U.S.A., 1988.
- [PRESSMAN,89] PRESSMAN, Roger S. Ingeniería de Software: un Enfoque Práctico. Editorial McGraw-Hill. Primera Edición en Español. Impreso en Madrid, España. 1989.
- [PRIETO-DIAZ,87]PRIETO-DIAZ, Ruben. FREEMAN, Peter. "Clasifying software for reusability". IEEE Software, January 1987.
- [RAO,93] RAO, Bindu R. C++ and the OOP Paradigm. Editorial Prentice-Hall. Impreso en U.S.A.1993.
- [RICHTER,87] RICHTER, Charles and Biggerstaff,Ted. "Reusability Framework, Assessment, and Directions". IEEE Software, March 1987.
- [ROBSON, 92] ROBSON, R.N. "Using hypertext to locate reusable objects". IEEE Software.1992.

- [SIMOS,90] SIMOS, M.A. "The domain-oriented software life cycle: Toward an extended process model for reusability software reuse: emerging technology". De. Will Tracz. IEEE Cs Press, 1990.
- [SNEED,87] SNEED, H.M. and Jandrasics, G. "Software recycling". IEEE Comp. Soc. Press. Proc. of Conference on software Maintenance. Austin, Texas. 1987.
- [SPC, 93] SPC Services Corp. Reuse-Driven Software Process Guidebook. SPC-92019-CMC. Version 02.00.03, November 1993.
- [TRACZ, 94] TRACZ, Will. Confessions of a Used Program Salesman: Institutionalizing Software Reuse. Addison-Wesley. U.S.A., 1994.
- [WOODFIELD,87] WOODFIELD, S.N. and Embley D.W. "Can Programmers reuse Software". IEEE Software. July 1987.

