Universidad Autónoma de Querétaro
Facultad de Informática
Maestría en Ingeniería de Software Distribuido

"Adaptación de estándares de calidad en el desarrollo de un proyecto de software distribuido"

TESIS

Que como parte de los requisitos para obtener el GRADO de Maestría en Ingeniería de software Distribuido

Presenta:
Nidia Gabriela Vázquez Pardo

Dirigido por:
M.C. Ricardo Chaparro Sánchez

SINODALES

M.C. Ricardo Chaparro Sánchez
Presidente

Firma

M.C.C. Teresa García Ramírez
Secretario

Firma

M.C.C. Rosa María Romero González
Vocal

Firma

M.C. Adriana Rojas Molina
Suplente

Firma

Dr. Ubaldo Chávez Morales
Suplente

Firma

I.S.C. Alejandro Santoyo Rodríguez
Director de la Facultad de Informática

Dr. Luis Gerardo Hernández Sandoval
Director de Investigación y Posgrado

Centro Universitario
Santiago de Querétaro, Qro.
Mayo 2007
México

RESUMEN

En la actualidad las empresas desarrolladoras de software se han convertido en promotoras principales para la aplicación de normas y guías establecidas por instituciones para el desarrollo de software. Este trabajo propone la creación de una metodología para el desarrollo de Sistemas distribuidos bajo un estándar diseñado por uno de los institutos con mayor prestigio y validación en trabajos internacionales, el IEEE 1047 Standard for Software Life Cycle Proceses, cuyo principal objetivo es el ajuste de los procesos para el desarrollo de sistemas distribuidos. Así también, se exponen algunos trabajos los cuales se han considerado como base para la demostración de que los estándares como IEEE y otras normas son necesarios para la creación de productos de software con calidad en sus procesos y asegurar de esta forma que la robustez de los sistemas sea completa. Dentro de este trabajo también se analizan diferentes plataformas que se utilizan para el desarrollo de sistemas distribuidos como CORBA de Borland, Java RMI de SUN y DCOM de Microsoft, entre los más importantes y de mayor uso. Con lo anterior se puede asegurar que el desarrollo de sistemas distribuidos es una área amplia hoy en día y se resalta el hecho de su creación, sin embargo; no esta bien establecida en nuestra región, por lo que es necesaria una metodología para el desarrollo de aplicaciones distribuidas con calidad que proporcione herramientas a los ingenieros y a las pequeñas y medianas empresas desarrolladoras de software, que les permitan mejores formas de implementación de proyectos distribuidos en el desarrollo de software.

(Palabras clave: Calidad de Software, Ingeniería de Software, Sistemas Distribuidos)

SUMMARY

At present, companies that develop software have become the chief promoters of the application of norms and guides established by institutions for software development. This work proposes the creation of a methodology for the development of distributed systems under a standard designed by one of the most prestigious and highly validated institutes involved in international work, the IEEE 1074 Standard for Software Life Cycle Processes, whose principal objective is adjusting processes for the development of distributed systems. We also include some works which have been considered a basis form demonstrating that standards like the IEEE and other norms are necessary in the creation of software products that insure quality in their processes and guarantee that the durability of the systems is complete. This work also analyzes different platforms used in the development of distributed systems, such as CORBA from Borland, Java from SUN and DCOM from Microsoft which are among the most important and most used. The above demonstrates that the development of distributed systems is currently a broad field and that its creation is of great importance. Nevertheless, it is not well established in our region and a methodology is therefore necessary to develop distributed applications with quality that will provide tools to engineers and small and medium-sized companies that develop software and allow for better ways to implement distributed projects in software development.

(Keywords: Software quality, software engineering, distributed systems)

A mis padres.

Mamá, gracias por creer en mí y darme esperanza.

AGRADECIMIENTOS

A Dios

A mi más grande creador, al que agradezco todos los días por su misericordia y su gran amor conmigo.

A Mis Padres.

Que siempre han estado al inicio y al final de mis luchas, incondicionalmente apoyándome con manos extendidas.

A mis hermanos Sandra, Ruth, Rhoda y David

Por su gran apoyo, cariño y por estar conmigo a pesar de las circunstancias.

A mis amigos.

Gracias por apoyarme siempre y por que estuvieron alentándome a terminar este proyecto.

A los maestros:

Ricardo Chaparro Sánchez, gracias por compartir tu conocimiento, experiencia y por ayudarme en la realización de este trabajo, muchas gracias por tu gran paciencia.

Adriana Rojas Molina, gracias por esos momentos en que me impulsaste a seguir adelante y concluir este proyecto, así como alentarme a no darme por vencida.

ÍNDICE DEL CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN

1.1 Importancia

Hoy en día ha resurgido la importancia de crear de sistemas de software capaces de compartir información a través de red ya que no solamente la información forma parte de una compañía sino que también se convierte en información compartida para otros usuarios; algunos autores comentan que es la etapa, en la que quien tenga más información se tendrán mayores oportunidades de tomar decisiones que afecten positivamente a las empresas, instituciones de gobierno e instituciones privadas.

En éste capítulo se expondrá la importancia de los sistemas distribuidos, desde sus inicios y las ventajas que ofrecen al ser utilizados. Una de las principales razones del surgimiento de los sistemas distribuidos fue por la necesidad de compartir toda la información centralizada que se pudiera tener en existencia. Desde el inicio el uso de información centralizada estaba basado en el uso de grandes computadoras. Y aunque había una gran escasez de computadoras debido a que eran grandes y costosas, en los años 70, comienzan a surgir las primeras minicomputadoras, quitando del mercado a las grandes computadoras, debido al precio y los servicios que ofrecieron, de esta forma acapararon gran parte de los usuarios comerciales y no comerciales, incluso de compañías.

El estudio de los sistemas distribuidos es de mucha importancia debido a que el problema que se resuelve, es la creación de una aplicación de estos sistemas, para lo cual se requiere un conocimiento profundo de su estructura, un adecuado levantamiento de requerimientos y desarrollo del sistema, es por ello que analizar la cuestión evolutiva de los sistemas distribuidos influye en la creación de nuevas aplicaciones para estandarizar su desarrollo.

Al extenderse el uso de las computadoras personales y los beneficios que estas proporcionaban, la conexión entre máquinas resultaba una prioridad. Esto originó la posibilidad de que se construyeran sistemas que pudiesen compartir recursos, y que estuvieran interconectados en red bajo arquitecturas Cliente-Servidor. Por lo tanto los

clientes pudieron obtener recursos que los servidores proveen, estos recursos pueden ser datos, aplicaciones y componentes de software [Torres, 2003].

Después de que se suplió la necesidad de comunicación a través de computadoras y de compartir recursos, la creación de Internet se convirtió en la red más usada y de mayor tamaño, y que hasta la fecha se mantiene en crecimiento. Además es la base de nuevos proyectos de sistemas distribuidos.

1.2 Desarrollo de lo sistemas distribuidos

En la actualidad los sistemas distribuidos se han convertido en productos de software de primera necesidad para bases de datos, recursos compartidos y la conexión de sistemas heterogéneos. Los recursos pueden ser administrados por servidores y accedidos por clientes o pueden ser encapsulados como objetos y accedidos por otros objetos clientes. Estos recursos se extienden desde los componentes hardware como los discos y las impresoras, hasta las entidades de software definidas como ficheros, bases de datos y objetos de datos de todos los tipos entre ellos de archivos, estructuras, privados [Couloris, 2001].

La creación de sistemas distribuidos no ha sido tarea fácil debido que se deben considerar varias características de éstos, como son transparencia, eficiencia, flexibilidad, escalabilidad, fiabilidad, y comunicación [Couloris, 2001].

Los sistemas distribuidos permiten dentro de otras cosas la descentralización física de los datos, permitiendo la recuperación de datos analizados como información especifica sobre el área o tema de consulta [Gokhale, 1999].

1.3 Evolución actual de los sistemas distribuidos

El crecimiento de los sistemas distribuidos, va en aumento sobre todo por la gran necesidad de crear sistemas donde el procesamiento de información a distancia, fue rápido y seguro. Esto dio inicio con la creación de grupos de investigación en algunas universidades como la de Cambridge y empresas como Xerox PARC y Sun Microsystems, por mencionar algunas y que su objetivo primordial fue el desarrollo de sistemas de

archivos distribuidos en redes locales, lo que se convertía en sistemas experimentales y que originó el comienzo de un área que a finales del siglo XXI, cubre gran parte de las comunicaciones a través de la tecnología computacional; e incluyendo en estas ultimas los dispositivos controlados vía móvil.

1.4 Creación de sistemas bajo estandarización

Con la creación de sistemas distribuidos se han abierto áreas, donde son aplicados protocolos de comunicación y aplicaciones distribuidas como lo son ABS (Architecture for Information Brokerage Service) y [López de Vergara, 1999] y ABROSE (Agent Based Brokerage in Electronic Commerce), sistemas de control y simulación. Cuando se desarrolla un sistema es importante la implementación de procesos para la administración del sistema, donde la ingeniería de software forma parte esencial, permitiendo la creación de estándares de ingeniería y modelos de calidad, para el desarrollo de sistemas. La aplicación de estándares para el desarrollo de software en la actualidad se ha convertido en una necesidad para aquellas empresas que se han dado a la tarea de realizar proyectos basados en el desarrollo integral de sistemas con calidad.

En el área de software es importante considerar otras metodologías para el desarrollo de sistemas, y que se apliquen herramientas tales como CMM, (Capability Maturity Model) herramientas de ingeniería para desarrollo rápido y estándares para el desarrollo de software.

En este trabajo se considera el uso de estándares, que son instrumentos para el desarrollo de sistemas distribuidos con el objetivo de aumentar aplicaciones con mejor eficacia y asegurando robustez. Cuando se habla de creación de sistemas en paralelo, aplicaciones web y sistemas distribuidos; la expectativa que se considera es de sistemas con cierto grado de seguridad y complejidad, por tal razón son esenciales las exigencias de una buena creación de sistemas distribuidos.

Aunque los estándares generalmente no tratan de especificar el tipo de tecnología, lenguajes de programación y plataformas, existen organizaciones importantes que se consideran como las originarias de estándares internacionales entre ellas se encuentran la

Organización de Estandarización Internacional, la Comisión Electrónica Internacional y la Unión de Comunicaciones Internacional. Actualmente dos de estas organizaciones cooperan con la Junta de Comité Técnico, ISO/IEC JTC1, responsable de la tecnología de información. Este consejo es responsable de relacionar estándares para ingeniería de software, entre los más importantes se encuentra el ISO/IEC 12207, para el Proceso de Ciclo de Vida del Software. [James W. Moore, 1999].

Estas organizaciones profesionales y organismos internacionales como IEEE e ISO/IEC han publicado normas tituladas, respectivamente, "IEEE Estándar for Developing Software Life Cycle Processes" (Estándar IEEE para el Desarrollo de Procesos y Ciclo de Vida del Software) (IEEE, 1991) y "Software life-cicle process" (Proceso de Ciclo de Vida del Software) [ISO, 1994].

La norma IEEE 1074 entiende por ciclo de vida del software una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software". En los estándares se considera una actividad como un conjunto de tareas y una tarea como una acción que transforma entradas en salidas [Piattini, 2000].

Para la creación del desarrollo de software e implementación de prácticas de ingeniería, existen estándares los cuales se adaptan para la creación y desarrollo de estos, sin embargo el diseño de sistemas distribuidos se ha convertido en una gran tarea para los desarrolladores de Software, es importante que se tome en cuenta el crear nuevas técnicas para el desarrollo de sistemas distribuidos, es por eso que en el presente trabajo se realiza la adaptación de un estándar de calidad, para asegurar que el producto distribuido contará con las características necesarias solicitadas por el usuario; validando de esta forma un buen producto, evitando el constante mantenimiento y asegurando que el levantamiento de requerimientos sea lo mas cercano a lo establecido por los usuarios.

El propósito esencial de la adaptación de un estándar es validar el desarrollo de un software y que a través de la aplicación de herramientas adecuadas se puede obtener un producto de calidad. Se considera que la creación de sistemas en la actualidad, esta sufriendo cambios y pasa por una etapa de transición necesaria en la creación de productos de software que transciendan y que a su vez pueda ser creado software con las mismas metodologías que fueron aplicadas para anteriores trabajos que tuvieron éxito.

1.5 Descripción del Problema

La problemática que presenta el siguiente trabajo es la necesidad que existe para crear adecuadamente sistemas distribuidos, no importando su arquitectura funcional, software, hardware, así como también sus recursos compartidos ó plataforma, sobre la cual se encuentre desarrollado. Por lo que es necesario en adaptar una forma global y generalizada de diseñar, desarrollar e implementar un sistema distribuido que cumpla con las características necesarias de transparencia, flexibilidad, fiabilidad, escalabilidad, eficiencia y comunicación heterogénea.

Aunque a la fecha esta problemática esta resurgiendo, con mucha mayor fuerza debido a la proliferación de diversos sistemas distribuidos, tales como Internet, Intranet, la computación móvil y ubicua, debido a que son sistemas que requieren de una buena estructura en el momento de su realización y de este modo se efectúe el objetivo de crear mejores sistemas.

Es importante enfatizar que se deberá realizar una adaptación robusta de los estándares y con un aseguramiento de que tendrán estabilidad y durabilidad mientras se haga uso de los sistemas.

CAPÍTULO 2: ESTADO DEL ARTE

2.1. Sistemas Distribuidos

En el siguiente capítulo se encuentra conceptos básicos que se consideran en un sistema distribuido, características esenciales y análisis cronológico breve del crecimiento de sistemas distribuidos hasta la actualidad, como las siguientes:

- Cronología de los Sistemas distribuidos
- Futuro de los Sistemas distribuidos
- Desarrollo Actual en los Sistemas distribuidos
- Clasificación de sistemas distribuidos.

La diferencia que existe entre una red de computadoras y un sistema distribuido radica principalmente en que este último es un conjunto de computadoras independientes, apareciendo ante sus usuarios como un sistema consistente y único. Por lo general, tiene un modelo o paradigma único que se presenta a los usuarios. Con frecuencia, una capa de software que se ejecuta sobre el sistema operativo, denominada middleware, esta capa es responsable de implementar este modelo. Un ejemplo conocido de un sistema distribuido es World Wide Web, en la cual todo se ve como un documento (una página web).

En una red de computadoras no existe esta consistencia, modelo ni software. Los usuarios están expuestos a las máquinas reales, y el sistema no hace ningún intento porque las máquinas se vean y actúen de manera similar. Si las máquinas tienen hardware diferente y distintos sistemas operativos, eso es completamente transparente para los usuarios. Si un usuario desea ejecutar un programa de una máquina remota, debe registrarse en ella y ejecutarlo desde ahí.

De hecho un sistema distribuido es un sistema de software construido sobre una red. El software le da un alto grado de consistencia y transparencia. De este modo, la diferencia entre una red y un sistema distribuido esta en el software (sobre todo en el sistema operativo), más que en el hardware [Buades, 1999].

Se conoce a un sistema distribuido como aquel en el que los componentes localizados en computadoras, conectados en red, comunican y coordinan sus acciones únicamente mediante el paso de mensajes [Colurouis, 2003].

Dentro de las principales características que tienen los sistemas distribuidos se encuentran la Concurrencia de los componentes, la carencia de un reloj global y los fallos independientes de los componentes. A continuación se describen concisamente, estas características.

La concurrencia se conoce como la capacidad que tiene el sistema para manejar recursos compartidos, esto se refiere a que cuando se tiene acceso de varios usuarios a una misma base de datos al mismo tiempo, por lo que se dice que esta concurrida.

Por lo que respecta a la carencia de un reloj global, se refiere cuando los programas necesitan coordinar sus acciones mediante el intercambio de mensajes. La coordinación estrecha depende a menudo de una idea compartida del instante en el que ocurren las acciones de los programas [Cotronco, 2000].

Finalmente se encuentra los fallos independientes de los componentes, debido a que todos los sistemas informáticos pueden fallar, es preciso planificar las consecuencias de posibles fallos. Las fallas en la red se producen finalmente entre computadoras conectadas entre si.

2.1.2 Cronología de los sistemas distribuidos

El inicio de los sistemas distribuidos tiene su origen desde la creación de grandes sistemas para trabajar en serie y facilitar el procesamiento de datos. Desde el comienzo de la computación, hasta nuestros días, ha crecido la necesidad de controlar, y poseer información, hay autores que aseguraban que para fines del siglo XXI, [Gates, B. 1995] el poder de una nación, estaría basada en la cantidad de información que tuviere, esto debido a que, cuanto mas información se pueda tener acceso el conocimiento es mayor, si se hace un buen uso de esta y la toma de decisiones es segura.

Si comenzamos a plantear el origen de los sistemas distribuidos, algunos autores [Coulouris, 1994], describen el inicio a mediados de los 70’s. Una de las principales equipos de investigación que comenzaron con el diseño de un sistema de archivos

distribuidos fue Xerox, en el que se conocía como Xerox Palo Alto Reserch Center (Centro de investigación de Palo Alto Xerox), reconocido por Xerox PARC, esto se dio durante el año de 1971 al 80.

El proyecto incluía el desarrollo de las primeras estaciones de trabajo, servidores de archivos e impresión, la primera red local rápida y varios sistemas distribuidos experimentales. En 1979 la Universidad de Cambridge, realiza el Cambridge DCS (System Computing Distributed), bajo la red de Cambridge Ring, en una computadora LSI-4M68000, dentro de este rango de sistemas operativos distribuidos se encuentran también el Apollo Domain System (1980) que trabajaba bajo una red Toke ring, el Newcastle Connection que realizaba sus funciones bajo varios tipos de red y el Locus (1980) este se encontraba bajo una red Ethernet. Aunque hubo otros sistemas distribuidos de archivos como el Grapevine también de Xerox creado en 1981 y posterior a este el Cedar creado en 1982, que trabajaban bajo una red Ethernet.

Desde el comienzo de 1980 hubo una rápida expansión en investigación y desarrollo en sistemas distribuidas, para ejemplos se encuentran sistemas operativos distribuidos como Accent, creado por Carnegie Mellon University, en 1982, basado en pase de mensajes, otros similares a éste son el Mach, Amoeba, Aarhus, V-system y finalmente en 1988 Chorus creado por Chorus System, el sistema operativo con un kernel que soportaba sistemas distribuidos en tiempo real, como se muestra en la Figura No. 2.1 donde se describen los principales sistemas distribuidos, la compañía que los realizó y el año en que tuvieron auge.

Principales Sistemas Distribuidos desde 1977 a 1988

Sistema distribuido	Compañía	1977	1979	1980	1981	1982	1983	1984	1985	1986	1988
Xerox DFS	*Xerox PARC*										
Cambridge DCS	*Universidad de Cambridge*										
Locus	*UCLA*										
Apollo Domain	*Apollo Computers*										
Newcastle Connection	*Universidad de Newcastle*										
Grapevine	*Xerox PARC*										
Cedar	*Xerox PARC*										
V system	*Universidad de Stanford*										
Argus	*MIT*										
ACCENT	*Universidad Carnegie-Mellon*										
Amoeba	*Universidad de Ámsterdam*										
UNIX BSD 4.2 + Sun NFS	*Sun Microsystems*										
Mach	*CMU*										
Chorus	*Chorus Systems*										

Figura 2. 1 Cronología de los sistemas distribuidos

Ahora volviendo a lo que en la actualidad tiene aplicación, [Coulouris, 2001] considera que el Internet es apreciado como un sistema distribuido, que permite acceso a los usuarios, donde quiera que se encuentren, del servicio como el World Wide Web, el correo electrónico, y la transferencia de archivos. Así también se considera como un ejemplo de sistema distribuido la Intranet ya que es una porción de Internet administrada separadamente y que tiene un límite que se puede configurar para hacer cumplir políticas de seguridad local.

Con respecto a la computación ubicua, es un término aplicable a la interconexión que existe entre varios dispositivos a través de una Intranet casera y el host de otra Internet desde un dispositivo controlado por vía móvil, a lo que se refiere es que se pueden realizar tareas sencillas como buscar las cotizaciones mas actuales desde un servidor mientras esta viajando, mostrar fotografías u otras personas enviándolas fotografías desde una cámara digital a una impresora adecuada en una sala de reunión.

2.1.3 Futuro de los sistemas distribuidos

Es relevante la situación en la que se encuentran los sistemas distribuidos varias de las comunicaciones que existen hoy en día son desarrolladas bajo estos sistemas, sin embargo han conseguido el tener un mayor auge en concepto de la creación de nuevas aplicaciones como lo son las aplicaciones distribuidas de bases de datos, en diferentes entidades, la comunicación a través de vía satélite y en comunicación tales como los cajeros automáticos, aplicaciones web, entre otras.

2.1.4 Desarrollo actual en los sistemas distribuidos

La concurrencia de los componentes, la carencia de un reloj global y los fallos independientes de los componentes son característicos de un sistema distribuido, un ejemplo mas tangibles es la Internet, una Intranet que es una opción de Internet gestionada por una organización y computación móvil y ubicua.

A continuación se presentan dos casos de uso donde se aplica la plataforma CORBA, en la primera se muestra una comparación de diversas técnicas utilizadas en la gestión de aplicaciones distribuidas desarrolladas en los proyectos europeos ACTS ABS y ABROSE [López de Vergara, 1999].

La segunda aplicación es la creación de herramientas comerciales basadas en CORBA mediante la construcción de un prototipo que coordina las capacidades de las herramientas Matlab/Simulink, ISaGRAF y ARTISAN RtS. Para la creación de sistemas abiertos se aplican herramientas como COTS (Comercial Off The Shelf), para que colaboren en las fases del ciclo de vida de los sistemas distribuidos y su integración.

En este último caso se considera una integración para el diseño de un sistema de control de la línea de tratamiento en proceso a través de bloques funcionales, para realizar la co-simulacion entre el modelo de proceso de Simulink y el sistema de control en ISaGRAF.

Cuando se pueden resolver problemas de este tipo, como el de integrar herramientas de otros sistemas es donde se consideran las arquitecturas distribuidas debido a que pueden proporcionar ciertas ventajas, ya que permiten llevar acabo la co-simulación en distintos

equipos aumentando las posibilidades de diseño al tener en cuenta simultáneamente distintos puntos de vista [E. Estevez, 1998].

Otras de la herramientas que se han convertido de gran importancia son la aparición de servicios Web, ya que representan una evolución en la creación de plataformas distribuidas, para modelos de negocios, estos modelos de negocios se encuentran inmersos a modificaciones de las necesidades empresariales, de las restricciones del mercado, de la ampliación de la competencia, de la búsqueda del aprovechamiento de los recursos, etc., por esta razón se convierten en necesidades

2.1.5 Clasificación de sistemas distribuidos en aplicaciones

Según [Krawczyk, Wiszniewski, 1998] muestra tres puntos de vista específicos sobre las aplicaciones de software distribuido (DSA), En la Figura No.2.2 se muestran ambientes de computación que atienden los medios básicos para cómputo distribuido, las aplicaciones de usuarios de dominio están dirigidos para ambientes de computación específicos y están soportados por varias plataformas de programación. Las plataformas de programación soportan el ciclo de desarrollo de cualquier programa de aplicación distribuida.

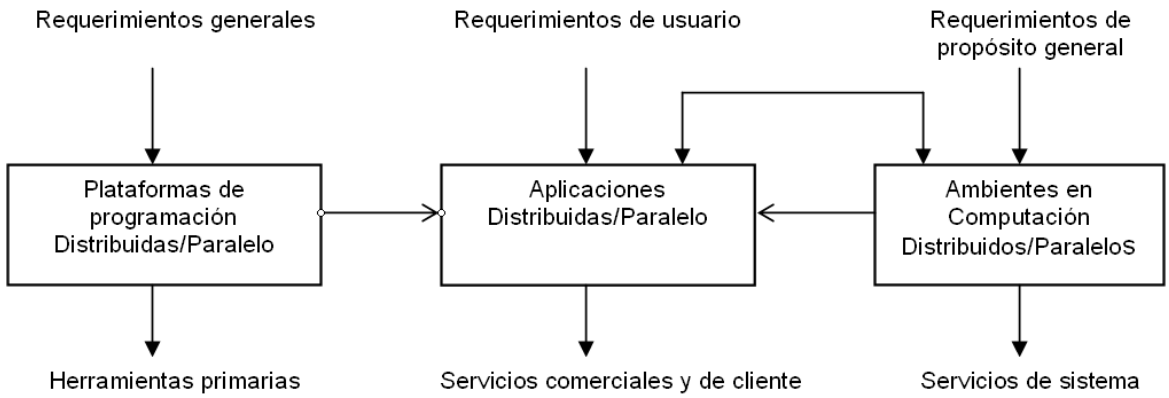

Figura 2. 2 Relación entre punto de vista sobre distribución.

Como se mencionó anteriormente las diferencias entre una red de computadoras y un sistema distribuido, es la visión en como se muestra ante los usuarios y su paradigma tiene un modelo o paradigma único que se presenta a los usuarios.

De hecho un sistema distribuido es un sistema de software construido sobre una red. El software le da un alto grado de consistencia y transparencia. De este modo, la diferencia entre una red y un sistema distribuido esta basada en el software (sobre todo en el sistema operativo), más que en el hardware [Buades, 1999].

2.2 CORBA

2.2.1 ¿Que es CORBA?

La arquitectura de Corba nace con la necesidad de comunicar sistemas con diferente interfaz de lenguajes, CORBA es propuesto por una compañía llamada OMG (Object Management Group) creada en 1989, que esta a favor de los sistemas abiertos, con interfaz estándar orientadas a objetos, construidas con hardware, redes, sistemas operativos y lenguajes de programación heterogéneos. Entre los elementos más importantes de CORBA esta IDL el lenguaje de definición de interfase, que permite la traducción entre lenguajes como Java, C++, Delphi, etc. Se habla también acerca de que CORBA es superior a Java RMI y DCOM.

La necesidad de conectar varias computadoras y solventar necesidades de aplicación y compartir información, es una las principales razones del incremento de redes y creación de sistemas distribuidos. Así también OMG, esta constituida por un conjunto de términos y definiciones para los conceptos de la tecnología orientada a objetos, el modelo de objetos que define la OMG, es un modelo matemático para las aplicaciones en términos de objetos y para el intercambio de información entre aplicaciones en basadas en mensajes, conjunto común de interfaz, protocolos y lenguajes.

2.2.2 ¿Por que utilizar CORBA?

En pocas palabras CORBA es una arquitectura de objetos distribuidos que hace posible que los objetos interactúen a través de redes de computadoras utilizando plataformas heterogéneas, sistemas de comunicación en diferentes lenguajes [Gokhale, 1999].

Los servicios que ofrece CORBA y que se encuentran disponibles en las implementaciones. Al hablar de esta arquitectura también se habla de Middleware, este es solamente el conjunto de programas distribuidos, sencillos consistentes e integrados que facilitan las tareas de diseño, programación y gestión de aplicaciones distribuidas. El

middleware una capa transparente que ofrece servicios a las aplicaciones que van ha estar en contacto directo con el usuario final.

Organiza los servicios a través de la interfaz IDL, además logra independencia de la plataforma y del lenguaje de desarrollo. Informa a los clientes y a los servidores de caídas de los canales de comunicación, integra software heredado definiendo la interfaz IDL.

2.2.3 Tipos de Middleware

El middleware son las herramientas que permiten gestionar y coordinar los mecanismos de comunicación, así también independiza el servicio y su implementación, del sistema operativo y protocolos de comunicación. Existen varios tipos de middleware, a continuación se presentan una descripción general de cada uno:

- *Base*, que se le llama al conjunto de estándares y servicios relacionados, que sirven de soporte y base para la construcción de middleware más especializado.
- *Para procesamiento de transacciones*, se encarga de la conectividad y el acceso a un gran número de usuarios, con servicios de "back-end" limitados. Se necesita de un monitor transaccional, para soporte de peticiones del cliente y servicios multihilo y administración de memoria.
- *De comunicación*, su función es proporcionar un medio de comunicaciones para que las aplicaciones puedan establecer un diálogo entre si, o lo que llamamos comunicación de programas.
- *De base de datos*, este crea una capa transparente en el acceso a base de datos, escondiendo toda la complejidad dada por la gran diversidad de servidores de bases de datos existentes.
- *De aplicación*, ofrece servicios como la ejecución que se da a través de protocolos como CGI (Common Gateway Interface), también la extensión que permite extender las funciones de un servidor de Web, e integración de otras aplicaciones.

En el caso de CORBA tiene middleware base, ORB (Object Request Broker) que establece las relaciones cliente/servidor entre objetos. Entre algunas de las arquitecturas que también manejan middleware son DCOM de Microsoft y RMI. A continuación se presentan algunos trabajos publicados recientemente, que se han realizado a través de la arquitectura CORBA y sus ventajas:

Entre las funciones de ORB una de las importantes responsabilidades es precisamente resolver peticiones de referencias a objetos, permitiendo que se establezca comunicación entre ellos, proporciona una serie de facilidades como la de localizar un objeto remoto dada una referencia a este objeto y otra es la ordenación de los parámetros y valores de retorno y de invocaciones a métodos remotos, mediante un mecanismo llamado "marshaling" para envió y "unmarshanling" para entregar los parámetros a través de la red.

CORBA ofrece interrelación con mas de 30 diferentes lenguajes de programación, debido al IDL, ya que este independiza el lenguaje de programación, no importa en que lenguaje se va implementar, el secreto esta en el compilador de CORBA-IDL, en la siguiente Tabla No. 2.1 [Peña, 2002], se muestra un comparativo entre las características de CORBA, DCOM y RMI bajo las funciones de ORB, que realiza un mapeo de la definición escrita, también CORBA permite transparencia, flexibilidad, tolerancia a fallas y escalabilidad.

Tabla 2. 1 Funciones del ORB

CARACTERÍSTICAS	CORBA	DCOM	RMI
Nivel de abstracción	4	4	4
Integración con Java	4	4	4
Soporte de SO	4	2	4
Todas las implementaciones de Java	4	1	4
Facilidad de configuración	3	3	3
Inv. Distribución de métodos	4	3	3
Inv. Guardan el estado	4	3	3
Inspección dinámica	4	3	2
Invocación dinámica	4	4	1
Eficiencia	3	3	3

Seguridad a nivel de cable	4	4	3
Transacciones a nivel de cable	4	4	0
Referencias persistentes a objetos	4	1	0
Servicio de nombre basados en URL s	4	2	2
Invocaciones multilenguaje	4	4	0
Estabilidad/Interoperabilidad	4	2	1
Estándar abierto	4	2	2

Mediante el ORB los objetos publican sus interfases, los clientes localizan a los objetos servidores y pueden invocar los servicios como si estuvieran localmente instalados.

A continuación se muestran las funciones del ORB. Un cliente necesita hacer uso del método de un objeto, y lo primero es crear una instancia del objeto, de forma transparente el ORB localiza la implementación correspondiente al objeto instanciado y devuelve una referencia al objeto remoto ubicado en algún servidor, cuando el servidor esta localizado, el ORB se asegura que el servidor está preparado para recibir la petición. Cuando el cliente invoca el método, el ORB del cliente hace uso del mecanismo del ”marshaling” envía los parámetros a través de la red, del otro lado el ORB del servidor mediante el “unmarshaling” entrega los parámetros al servidor, los parámetros de retorno se tratan del mismo modo.

2.2.4 Características de IDL

El IDL se ubica entre el cliente y la implementación de manera que este nunca llegué a interactuar directamente sobre la misma. Con ello se consigue la independencia del lenguaje de implementación, lo que permitirá al programador elegir el que más se adapte a sus conocimientos y necesidades. Fig. 2.3

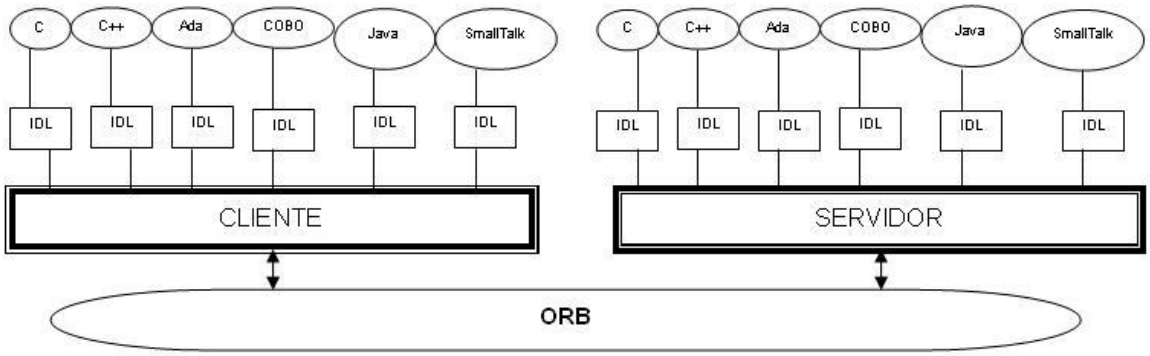

Figura 2. 3 Lenguajes de programación con entrada IDL

Las características de IDL son:

- Los identificadores que son sensibles a mayúsculas, con la restricción de que los identificadores del mismo ámbito no pueden diferenciarse únicamente en que las letras que los componen sean mayúsculas o minúsculas.
- Un objeto CORBA no soporta la noción de clase ya que implementa una interfaz remota y puede estar escrito en cualquier lenguaje.

Tabla 2. 2 Palabras reservadas en IDL

Any	Bolean	Char	In
Double	Exception	fixed	Out
Interface	Module	Octet	TRUE
Readonly	Short	Struct	Default
Unsigned	Void	Wstring	Inout
Attribute	Case	Const	Raises
Enum	FALSE	float	Typedef
Long	Object	Oneway	
Sequence	String	Switch	
Union	wchar	context	

Como CORBA actualmente ofrece soporte a más de 30 diferentes lenguajes de programación. Al definir una interfaz usando IDL, se logra independencia del lenguaje de programación, no importa en que lenguaje se va ha implementar. El elemento que se encuentra detrás de esta "mágica" forma de soportar muchos lenguajes es el compilador CORBA-IDL, que traduce y hace el mapeo de la definición de escritura usando IDL al lenguaje de programación que se usará para la implementación de la interfaz.

La transparencia de CORBA se logra en un alto grado, donde el cliente no sabe donde esta ubicado el objeto servidor, con respecto a la flexibilidad el IDL permite lograr un alto grado de transparencia en CORBA, en tolerancia a fallas CORBA no soporta directamente servicios para tolerancia de fallas, hay varios vendedores que han provisto este soporte por ejemplo, Visigenic's VisiBroker provee soporte simétrico para enlazar automáticamente a otro objeto servidor en un servidor diferente, su escalabilidad no establece un servicio especifico, por lo que utiliza Threading que permite la creación de hilos por parte de los clientes y "Tuning" que permite activar optimización para situaciones especificas.

En las siguientes tablas se hace referencia a los productos generados para conseguir el middleware de CORBA basados en el estándar de la OMG; en la Tabla 2.3, se muestran algunos productos creados en universidades, por lo tanto son de libre disposición o licencia, en la Tabla 2.4 se encuentran los proyectos gratis para su evaluación y la Tabla 2.5 los productos comerciales.

Tabla 2. 3 De libre disposición o licencia GNU

Producto	Empresa/Autor
Mico Is Corba	Univ. De Frankfurt
OmniORB2	ATT&T Cambrige
ORBit	RHAD Labs
The ACE ORB	Univ. Washington
TAO	Universidad Washington

Tabla 2. 4 Gratis para evaluación o uso no comercial.

Producto	Empresa/Autor
ORBacus	OOC
OrbixWeb	Iona
Visibroker	Inprise

Tabla 2. 5 Comerciales

Producto	Empresa/Autor
ChrusORB	Sun
Orbix	Iona
Voyager	ObjectoSpace
ORBexpress	Object Interface Systems

ORB MICO es un ORB desarrollado con fines educativos, no necesita equipos de grandes prestaciones, la Universidad de Frankfurt lo desarrollo y su rendimiento no es muy bueno. ORB BACUS es un ORB de libre distribución para fines no comerciales. Ya está actualizado a la norma CORBA 2.3. Ha superado sus pequeñas incompatibilidades en el mapeado a C++. En el caso de TAO es un ORB para tiempo real, desarrollado en la Universidad de Washington por Douglas Schmidth y su equipo. Permite la creación de servidores multihilo y tiene servicios para tiempo real.

2.2.3 Evolución de CORBA en algunas aplicaciones

CORBA es una opción para desarrollar sistemas grandes y empresariales, con una amplia variedad de lenguajes de programación para implementar el servidor y el cliente. Soporta plataformas, comunicaciones y sistemas operativos heterogéneos.

Se puede decir que los sistemas distribuidos han sido desarrollados bajo tecnologías como las de CORBA, aunque compite con otras plataformas RMI y DCOM, que han incursionado en el campo del desarrollo de nuevos sistemas, como administradores de archivos, en el área de aviación, en un sistema de Administración de configuración de aviones, la escala de aviones y la demanda de Boeing, en esta aerolínea existía la necesidad de integrar varios sistemas, para el control del mantenimiento y construcción de aviones, [Emmerich, W, 2000], para el desarrollo del sistema resolvieron la heterogeneidad de sus sistemas de mantenimiento, a través de CORBA.

2.3 Estandarización

A través de encuestas se ha encontrado referencia que existen 315 diferentes estándares, guías, y documento de mantenimiento para ingeniería de software en 46 diferentes organizaciones. Pero la disponibilidad que existe de estos estándares se convierte en algo decepcionante. Dicha encuesta también proporciona que la mayoría de estos estándares presentan diferencias detalladas, debido a que los usuarios encuentran con dificultad los estándares que satisfagan la situación en particular de proyectos a desarrollar, así también se aprecia que existen algunas

diferencias entre los estándares que los hacen difíciles de aplicarlos al mismo tiempo, por ejemplo cuando se desea conjuntar dos estándares, cada uno puede acentuar diferentes partes del desarrollo del proyecto y utilizar diversa terminología [W. Moore, 1999].

2.3.1. Evolución de los estándares en el desarrollo de software

Desde 1991 el Comité de Estándares de Ingeniería de Software ha emprendido esfuerzos para administrar la colección de estándares para promover consistencia. Esta colección ha aumentado al doble de tamaño, se ha mejorado substancialmente su integración gradual. El proceso no esta completo pero significa un progreso que ha sido, culminado con la publicación de 1999, un cuarto volumen de la publicación de Estándares de SESC.

Para algunos, el valor de usar estándares de ingeniería de software puede ser obvio, estos contribuyen a la práctica disciplinada, por lo tanto mejoran la calidad del producto. Aunque estas razones validan el uso de los estándares en el arte de la ingeniería de software, no es la única contribución que proporcionan los estándares para la ingeniería de software, también permiten la adaptación de nuevas técnicas para el desarrollo de procesos.

2.3.2 Principales estándares en el desarrollo de software.

Uno de los estándares relevantes para el proceso de ciclo de vida del Software es el ISO/IEC 12207, a continuación se presentan algunas de las aplicaciones en este estándar.

Tal es el caso del desarrollo de un estándar en la agencia espacial europea, en el cual se realiza a través de un grupo de ingeniería que se encarga de llevar a cabo una comparación entre los estándares ISO-9001/9000-3 e ISO/IEC 12207 [M. Jones, 1997].

En el siguiente trabajo se presenta una comparativa que realizó la agencia espacial Europea a mediados de 1970, comenzó con el desarrollo de software de gran tamaño, aun que no contaban con un estándar de ingeniería de software, lo que causo problemas y dificultades obteniendo una situación insatisfactoria. Para enfrentar estos problemas y

reconocer la importancia del crecimiento de software, la agencia espacial crea un consejo para establecer un estándar y validar procesos en Mayo de 1977.

El consejo se encarga de establecer y mantener una guía para los estándares de ingeniería y conseguir el desarrollo y el mantenimiento de Software. Verificar que los estándares que están establecidos sean aplicados para desarrollo de las actividades de software en la misma agencia.

Se concertó con el departamento de Asuntos legales de la agencia para salvaguardar la propiedad intelectual de los derechos relacionados al software. Actualmente el consejo tiene siete miembros, todos trabajan dentro del comité. El consejo tiene promulgado este trabajo en la forma de los estándares de PSS y en la forma de documentos de los consejos.

Los estándares de ingeniería de software fueron escritos por el consejo de la agencia espacial, donde se incluye el estándar PSS05-0, en la Fig.2.4 se muestran los documentos de PSS producidos por el consejo y varias guías de PSS-05-0, y estándares de código. Todos estos estándares están publicados como documentos PSS.

En la historia de los Estándares de la agencia especial, las primeras versiones de Estándares de ingeniería de software fue publicada en 1984, fue remplazado por PSS-050 y publicado en 1987, posteriormente el PSS-05-0, fue publicado en 1991, siguiendo una extensión del proceso de revisión involucrando a usuarios del estándar y experiencia adquirida. Desde entonces la relación de la segunda publicación, el consejo tiene escrito a un conjunto de 10 guías, cubriendo las fases de ciclo de vida y los aspectos de administración.

El estándar de PSS-05 fue publicado por la Prentice-Hall, y las guías han sido publicadas como Guías de Ingeniería de Software. Un número de organizaciones desde afuera de la agencia han implementado los estándares de ingeniería de Software así como en el ISO 9000 también aplican una comparación entre PSS-05-0 e ISO /IEC 12207 mostrando como conclusiones que el PSS-05-0 cubre los procesos de ISO/IEC 12207 en un 82% de las tareas de ISO/IEC 12207 en alcance de PSS-05-0 son cubiertas también cuenta con recomendaciones y líneas guías acerca de cómo implementar tareas de ISO/IEC 12207, con respecto a productos definidos asociados de PSS-05-0 a ISO/IEC 12207 para procesos de salida, además de que PSS-05-0 no es conflictivo con ISO/IEC 12207.

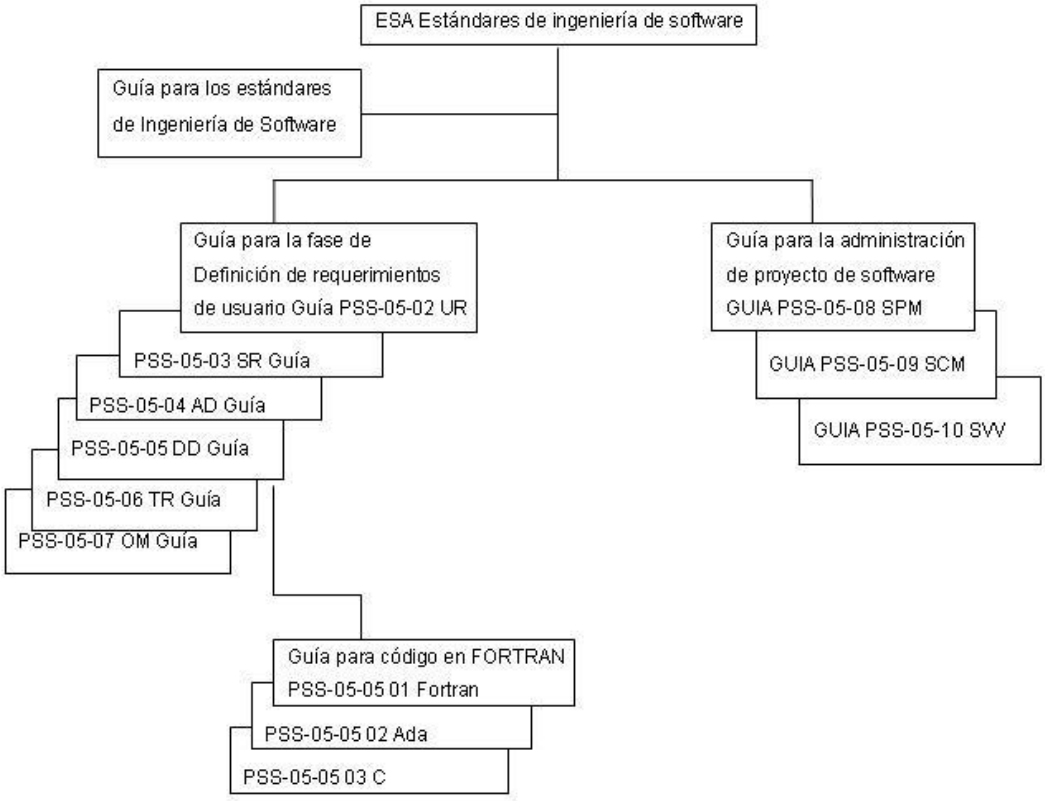

Figura 2. 4 Árbol de Documentos PSS-05-0 de ESA

El estándar PSS-05 fue escrito usando los estándares de ingeniería de software del Instituto de Ingenieros Eléctricos y Electrónicos, como recurso primario de terminología y las definiciones de productos y planes.

A diferencia de los estándares de ingeniería de software de la agencia, los estándares de ingeniería de la IEEE, son integrados en forma reducida. Los estándares describen procesos involucrados en el ciclo de vida de un proyecto de software simple desde su principio hasta las entregas de software.

Este estándar se divide en dos partes:

Los procesos de producción, basados en modelos de ciclo de vida teniendo seis fases como sigue:

- Fase de definición de Requerimientos de usuarios
- Fase de definición de requerimientos de Software

- Fase de Diseño de arquitectura.
- Fase de diseño detallado
- Fase de transferencia.
- Fase Operaciones y mantenimiento

La segunda fase es la de administración, donde se definen las siguientes actividades, cada una en término de entradas, salidas y actividades. Tienen procedimientos usados para realizar los productos, divididos en cuatro actividades:

- Administración de proyecto de software
- Administración de configuración de software
- Validación y verificación de software.
- Aseguramiento de calidad de Software.

Para el caso del PSS-05 este se convierte en un estándar pesado para proyectos pequeños, se consideran proyectos pequeños aquellos con menos de 2 hombres en un esfuerzo de desarrollo de dos años, un desarrollo simple para un equipo de cinco personas o menos ó un programa de menos de 10,000 líneas de código.

Para esto existe un estándar desarrollado por el consejo con el nombre de Estándar de ingeniería de software para proyectos pequeños. La relación que existe entre el estándar PSS-05 y para otros estándares las categorías de conformidad son: Conformidad de prácticas de estándares de ingeniera de software las cuales aseguran una conformidad para el estándar ISO. Los requerimientos del ISO el cual esta fuera del alcance de la agencia (sea de los estándares ubicados como ingeniería de software) por que después los objetivos son diferentes.

La No conformidad: Estos requerimientos del estándar ISO están en el alcance de los estándares de la agencia espacial europea aunque no son cubiertos. El ISO 9000 soporta proyectos de calidad validados, aplicados a todos los tipos de productos, hardware, software, materiales procesados y servicios. El consejo de la agencia espacial realiza un comparativo entre el ISO 9001/ISO 9000-3 y PSS-05-0, después de ello se considera a los estándares de ingeniería de software, como una excelente base para la administración de sistema de calidad de software. También considerando cubrir virtualmente todos los

requerimientos relacionados para el desarrollo de software, no tienen contradicciones con ISO 9001.

Otro de los casos de adaptación del estándar ISO/IEC 12207 fue para el desarrollo de Software Instruccional, donde el proyecto se inicia por una agencia del gobierno y los procesos son seguidos por los proveedores, durante un contrato de desarrollo de software normal. Los proceso se dirigen especialmente a pequeñas empresas desarrolladoras de software y proporcionan también una guía sobre que hacer con respecto a las expectativas del cliente. El Ministerio de Educación nacional de Turquía, presenta una problemática en la instalación y calidad de los sistemas instalados en las escuelas, es por ello que decidieron realizar un proyecto que consiste de tres principales tareas en dos fases:

Fase 1 Definir el proceso

- Definir procesos de desarrollo de software instrucción básica.
- Establecer los estándares para la entrega definida por el proceso.

Fase 2 Validación y Planeación

- Definir la evolución de criterio por pre-calificación de compañías de desarrollo de software.

El Estándar ISO/IEC 12207 se considera como uno de los estándares que se pueden utilizar para el propósito de crear software y adaptarlo a un enfoque el cual se requiere definir los procesos de desarrollo del estándar en relación con la complejidad y el tamaño del contrato.

Lo que se busca principalmente es un levantamiento de requerimientos según lo establecido en el Estándar adaptando el ISO/IEC 12207 de acuerdo a las bases guía del mismo, entre las principales tareas son: a) que sea aplicable únicamente para desarrollos de sistemas de software, b) que se proponen en este proyecto es revisar cada requerimiento por ISO/IEC 12207 y c) cada actividad del desarrollo es dejar fuera el proceso para ajustarlo al contrato de administración del proceso del Ministerio.

Considerando que una de las tareas más importantes dentro de este proyecto es establecer la especificación de Requerimientos, es decir crear un documento de requerimientos de software. Capturar los requerimientos de un sistema involucrando estrategias de especificación de aprendizaje y requerimientos técnicos.

Los requerimientos de actividades consisten de las siguientes tareas:

- Identificar y describir usuarios.
- Identificar y descripción del sistema.
- Diseño instruccional (Diseño de seudo código).
- Describir funcionalidad en detalle.
- Requerimientos de Especificación no funcional.
- Evaluar especificaciones de requerimientos.

Como se observa, los procesos del estándar que la ESA presenta fueron adaptables a sus proyectos integrando otro estándar de ISO, lograron así la creación de su estándar y entallarlo a sus necesidades.

CAPÍTULO 3: PLANTEAMIENTO DEL PROBLEMA

3.1 Descripción del problema

Algunas compañías dedicadas al desarrollo de software en México, se han dado a la tarea de incursionar en la actividad de creación de sistemas que contengan un nivel de seguridad y transparencia en el momento de su desarrollo, por lo tanto buscan robustez y calidad en los productos que se realizan, esto se ha convertido en una oportunidad de progreso y fuente de trabajo a futuro.

Aunque algunas empresas pequeñas, aún no contemplan herramientas como las tecnologías para la creación e implementación de software, debido a que apenas inician en este mercado, sin embargo existen compañías grandes y que en su mayoría son extranjeras, están implementando varios estándares y tecnologías rápidas para el desarrollo de software.

En la actualidad, la producción de software inicia y además este mercado ha sido abarcado por unas cuantas empresas. Es importante destacar que en el país se cuenta con una gran capacidad potencial para desarrollar este tipo de industria dada su cercanía geográfica y el mismo uso horario con el mercado de software más grande del mundo E.U.A [Secretaria de Economía, 2001], otra de las ventajas es que tiene conexión con tratados comerciales en varios países.

Pese a esto, la situación real con respecto al desarrollo de software es diferente, ya que nuestra nación ocupa el lugar número 50 en compra de software, así como también en la utilización de tecnologías de información y comunicación, [Secretaría de Economía, 2001].

En este trabajo se analiza la aplicación de herramientas como lo son estándares para el desarrollo de sistemas, considerando que los sistemas actualmente proporcionan la infraestructura de las aplicaciones de redes de computadoras de hoy en día. Es significativo enfatizar que la creación de sistemas requiere de cuidado, en el proceso de su elaboración y de esta forma asegurar que los proyectos cuenten con la calidad necesaria y la satisfacción de suplir la tarea que se requiera.

Tal es el caso de los sistemas distribuidos que se consideran como sistemas que requieren un carácter abierto y de heterogeneidad de sus componentes, y que no existen metodologías ad-hoc a su construcción. El propósito de este trabajo es resolver esta necesidad dentro de la región y establecer los procesos necesarios para la creación de sistemas distribuidos.

A nivel regional el desarrollo de sistemas distribuidos se ha soportado en algunas instituciones o académicamente dentro de las universidades y con ayuda de investigadores, otra de las razones, que invitan a dar importancia a estos sistemas es que hoy en día se están extendiendo en el ámbito de la computación, como lo son aplicaciones para la enseñanza a distancia, cajeros automáticos, sistemas a distancia por franquicias, entre otros.

Es necesario hacer énfasis en que el desarrollo de sistemas distribuidos con el tiempo se ha convertido en unas de las principales aplicaciones que corren bajo Internet, aunque en nuestra región la producción de software, no se considera una de las áreas que se explote en toda su totalidad, pero a pesar de esto, es un área que esta en vías de desarrollo.

En este trabajo se presentará una metodología para la creación de Procesos de desarrollo de sistemas distribuidos, a través de estándares y de esta forma se asegure la calidad de un producto de software, el propósito es elaborar una guía para la aplicación de estándares de calidad de IEEE, en el desarrollo de sistemas distribuidos, así como su validación y verificación del producto.

CAPÍTULO 4 PLANTEAMIENTO DE LA HIPÓTESIS

4.1 Hipótesis General

Mediante la aplicación de estándares de calidad en el desarrollo de software se asegura la validación y verificación de un Sistema de Software. Así mismo el uso de tecnologías modernas, como CORBA (para desarrollar software de sistemas distribuidos), obteniendo la transparencia, rapidez en la creación de sistemas, seguridad, robustez y funcionalidad.

4.2 Objetivo General

Adaptar los estándares de calidad para la aplicación de desarrollo de Software del (Instituto de Ingenieros Eléctricos y Electrónicos) IEEE en el desarrollo de un proyecto de software distribuido utilizando CORBA (Component Object Request Broker Architecture).

4.3 Objetivo Específico

- Analizar los estándares de desarrollo de Software de IEEE

4.4 Metas

- Llevar a cabo un análisis de los estándares de la IEEE comenzando por secciones.
- Realizar un estudio específico de las herramientas CORBA y RMI de JAVA.
- Revisar un proceso de ciclo de vida del software adaptable a un sistema distribuido.

CAPÍTULO 5 PROPUESTA DE SOLUCIÓN

5.1 Justificación

El crear software se convierte en una tarea difícil cuando no se cuenta con procesos de administración para controlar la documentación referente al desarrollo de software. El desarrollo de productos de software en México, esta comenzando a considerarlo fuertemente sobre todo en algunas compañías dedicadas a ello.

Las organizaciones profesionales y los organismos internacionales se han venido ocupando del ciclo de vida del software y después de varios años de trabajo, IEEE e ISO/IEC han publicado normas tituladas, respectivamente, "IEEE Standard for Developing Software Life Cycle Processes" (Estándar IEEE para el desarrollo de procesos del ciclo de vida del Software) [IEEE, 1991] y "Software life-cycle process" (Proceso del ciclo de vida Software) [ISO, 1994]. La norma IEEE 1074 entiende por ciclo de vida software "una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software". Ésta considera una actividad como un conjunto de tareas y una tarea como una acción que transforma entradas en salidas [Piattini, 1999].

El propósito general de la norma es contribuir a que las organizaciones que desarrollan o mantienen software puedan ajustarse mejor a los plazos y costos establecidos. Prevé que el usuario principal de esta especificación en un determinado proyecto de software será el 'arquitecto de procesos', el individuo cuya función es seleccionar un Modelo de Ciclo de Vida para el Software, crear el Ciclo de Vida del Software concreto del proyecto y asegurarse de que se sigue este ciclo durante el tiempo de vigencia de aquél [IEEE, 2004].

Los sistemas distribuidos se encuentran en una etapa en que su desarrollo se ha extendido, aunque a la fecha no existe alguna metodología específica para el desarrollo de procesos en la creación de estos sistemas, otra de las justificaciones que se dan en este trabajo es que los sistemas distribuidos, tienen desafíos fundamentales a los que deben enfrentarse los diseñadores, entre estos se encuentran la heterogeneidad, carácter abierto, seguridad, escalabilidad, gestión de fallos, concurrencia y la necesidad de transparencia

[Coulouris, 2001], y que a continuación se describen, debido a que deben ser tomados en cuenta para que la construcción de sistemas sea de forma completa donde los procesos del ciclo de vida de software cumplan con las características necesarias para que el desarrollo del sistema sea adecuado.

Los actuales desarrolladores se enfrentan a problemas en el momento de comenzar con el desarrollo de software dentro de estas características están, [Dollimore, Jean, 2001], la heterogeneidad, ya que los sistemas distribuidos se aplican bajo conjuntos de redes heterogéneas, así como hardware, sistemas operativos, lenguajes de programación e implementaciones de diferentes desarrolladores, otro de estos problemas es la extensibilidad, esta trata del grado en el que el sistema distribuido puede añadir nuevos servicios para compartir recursos y ponerlos a disposición. También dentro de estas se encuentra la seguridad ya que la mayoría de la información que se comparte se ofrece en los sistemas distribuidos, tiene un valor interior para los usuarios, por esta razón es importante considerar que la confidencialidad, integridad y disponibilidad sean considerados.

Así también una mas de las características es la escalabilidad, que radica en que conserve su efectividad cuando ocurre un incremento significativo en el número de recursos y el número de usuarios, pero como en todo sistema también es considerado el tratamiento a fallos, para la detección de estos existen algunas técnicas, donde el reto es detectar las anomalías que no pueden ubicarse pero que si pueden esperarse y como existe la posibilidad de que varios clientes necesiten acceder a un recurso compartido al mismo tiempo, a lo que se llama concurrencia, en este aspecto los objetos de sistemas distribuidos deben de estar sincronizados para que los datos permanezcan consistentes, tal como se muestra en la siguiente figura, Figura 2.5.

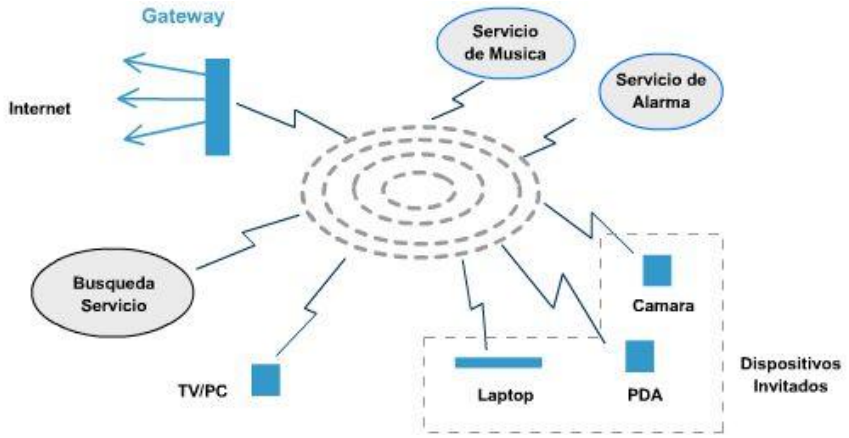

Figura 2. 5 Distribución de dispositivos en una red.

Uno de las últimos problemas a los que se enfrentan los desarrolladores es la transparencia ya que se convierte en un reto desde el punto en que se necesita ocultar al usuario y al programador aplicaciones de la separación de los componentes en un sistema distribuido, de forma que se perciba el sistema como un todo y no como una colección de componentes independientes, esto tiene mucho peso durante el diseño del software del sistema.

El desafío de implementar todas estas características en un sistema distribuido, es lo que se lleva a crear metodologías para su construcción, y que estas técnicas sean adaptables y verificables al ciclo de vida del software.

Debido a todo esto, se tiene una visión hacia el futuro del desarrollo de software en México, donde existen esfuerzos y acciones por parte de centros de investigación que propician el aprovechamiento de vínculos académicos y de negocios de grupos de ingenieros mexicanos que trabajan en empresas líderes y universidades [Secretaría de Economía, 2001], y que abren puertas para la creación de nuevas tecnologías especificas para la creación de sistemas distribuidos y asegurar su calidad desde el momento de comenzar su desarrollo.

La necesidad que existe de contar con el apoyo de conocimiento, herramientas avanzadas, plataformas y nuevas tecnologías relacionados con la producción de software, originará una rápida y eficiente inserción en los mercados de trabajo, tanto de empresas que

destinen sus servicios y productos al mercado nacional, como las dedicadas a la exportación. [Secretaria de Economía, 2001].

5.2. ¿Por que utilizar CORBA?

Debido a que en el desarrollo de sistemas distribuidos se aplican herramientas de lenguaje aptas para su creación, a continuación se describe la razón por la que se considera CORBA como arquitectura apta para la creación de sistemas distribuidos.

El papel que tienen las arquitecturas distribuidas en el desarrollo de software tales como CORBA, RMI de Java y DCOM de Microsoft son algunas que en la actualidad han proliferado, [Peña, 2002], es preciso resaltar que estas permiten la replicación de servidores y que cuenta con un servicio de tolerancia a fallos por lo que respecta a el modelo cliente /servidor. Es sin duda, que herramientas como estas son necesarias para resolver varias de las necesidades que se están generando en el ámbito de las comunicaciones de redes, aplicaciones web y distribución de datos.

La combinación de los entornos distribuidos, con las técnicas de orientación a objetos, está dando lugar a un nuevo concepto de sistemas que, requieren una infraestructura de comunicación que soporte la ligas entre objetos situados en diferentes nodos de las redes, y probablemente sobre diferentes máquinas y entornos [TCPSI, 2004].

CORBA constituye hasta hoy el único estándar independiente encaminado a solventar este problema, y dispone ya de implementaciones comerciales que cubren prácticamente todos los entornos informáticos y de comunicaciones imaginables. Junto a esta infraestructura, CORBA está definiendo servicios adicionales, tales como seguridad, transacciones distribuidas, y otros, así como su integración con Internet, lo que permite considerarlo como una de las plataformas básicas de desarrollo de sistemas profesionales.

Esta es una de las razones por las que se considera que CORBA, es una arquitectura distribuida, para el desarrollo del sistema en investigación y de esta forma asegurar la creación de sistemas distribuidos.

CAPÍTULO 6 DESARROLLO

6.1 Introducción

Como se ha mencionado anteriormente la importancia del uso de estándares y modelos de calidad que han sido creados específicamente para el desarrollo de sistemas, se describen a continuación. Estas herramientas se están utilizando en distintas compañías que desarrollan software, entre los mas importantes se encuentran las estándares de IEEE: 610.12 Standard Glosary of Software Enrgineering Trminology (Estándar para el Glosario de Terminología de Ingeniería de Software), IEEE 1058 Standard for Sopftware Proyect Management Plans, (Estándar para el Plan de administración del proyecto del Software), IEEE/IEA 12207 Software Life Cycle Processes (Procesos de Ciclo de vida de software), IEEE 830 Recommended Practice for Software Requeriments Specifications (Prácticas recomendadas para las Especificaciones de Requerimientos de Software), IEEE 1016 Recommended Practive for Software Design Descriptions (Práctica Recomendada para las descripciones de Diseño de Software), IEEE 829 Standard for Software Test Documention (Estándar para la prueba de Documentación de Software).

Dentro de este mismo propósito se encentra ISO con sus estándares ISO/IEC 12207 Software Life cycle processes (Proceso de ciclo de vida del software), creado en 1995 y el ISO/IEC TR 15846 Software Engineering Software Life Cycle Process-Configuration Management for Software (Ingeniería de software-Proceso de ciclo de vida para la administración y configuración del software).

Así también se encuentran los modelos como CMM-SEI creado por Watts S. Humphrey, en 1986, este modelo evalúa la madurez de los procesos de administración para el desarrollo de sistemas, así también se encuentran los modelos como TSP (Team Software Process), PSP (Personal Software Process) estos últimos modelos fueron desarrollados después que CMMI. En caso de MoProSof fue desarrollado en México.

Actualmente se han desarrollado a nivel internacional una gran cantidad de modelos para crear y desarrollar software con calidad. En esta sección se analizaran en que consisten

los más importantes, así como los detalles de los puntos específicos que interesan para el desarrollo de este proyecto de investigación.

El modelo de CMMI-SEI Capability Maturity Model Integration (Integración del Modelo de Capacidad de Madurez) el propósito de este modelo es la evaluación de los procesos, esto a través de niveles de madurez, dicho modelo proporciona cinco niveles de madurez, Nivel 1 Inicial, conocido Nivel 2 Repetible, Nivel 3 Definido, Nivel 4 Gestionado, Nivel 5 Optimizado, donde cada nivel se sujeta a establecer tareas y agruparlas en áreas llamadas KPA's, áreas claves de proceso, en la cual cada área de proceso se define un conjunto de buenas prácticas que habrán de ser definidas, provistas, ejecutadas, medidas y verificadas. [Humphrey, 1986].

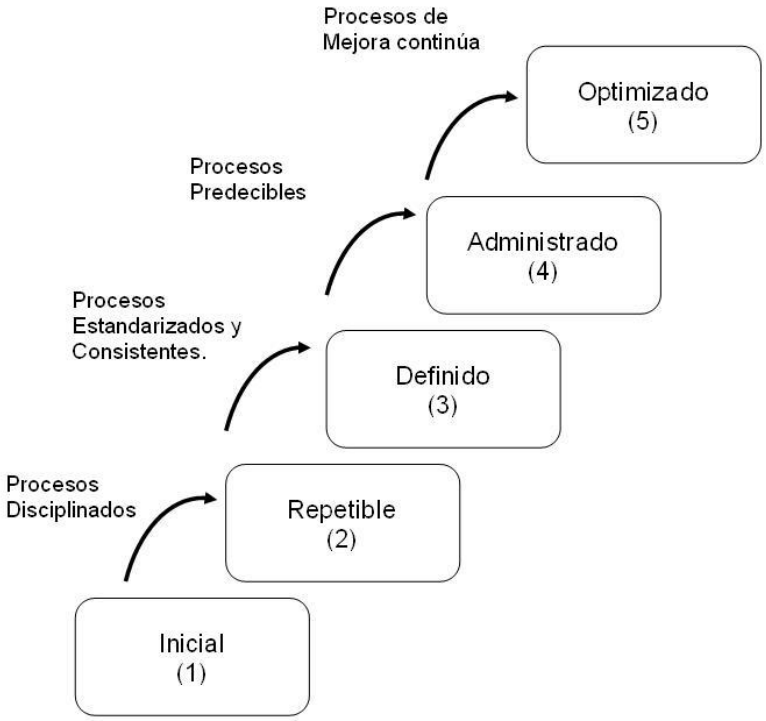

Figura 6. 1 Los cinco niveles de Madurez de los Procesos de Software

Otros de los modelos para el desarrollo de software es SPICE (Software Process Improvement Capability and dEtermination), conocido como ISO/IEC 15504. Este estándar proporciona un aseguramiento a los procesos de software, permite a organizaciones involucradas planear, administrar, monitorear controlar y mejorar la adquisición, suministro, desarrollo, operación, evolución y soporte de software. [SPICE, 1998].

A nivel nacional también se han hecho esfuerzos por parte del gobierno para desarrollar un modelo para el desarrollo de software. Tal es el caso de MoProSof (Modelo de procesos para la industria del software) este surge como base a la Norma Mexicana para la Industria de Desarrollo y Mantenimiento de Software, donde el propósito es presentar un modelo de procesos para la industria de Software en México que fomente la estandarización de superación a través de la incorporación de mejores practicas en gestión de ingeniería de software, que nace en Mayo del 2003, en un convenio entre la Universidad Nacional Autónoma de México y las Secretaría de Economía [Secretaría de Economía, 2003].

Sin embargo técnicamente se requiere de un esfuerzo mayor y más enfocado a los aspectos técnicos del desarrollo de software. En este ámbito una de las organizaciones internacionales de estandarización provee algunos documentos (estándares) para el desarrollo de software que a continuación se describen.

El ISO/IEC 12207 es un estándar que promociona un proceso para el ciclo de vida de Software, donde se muestra desde la adquisición y suministro de productos y servicios de software hasta su validación y verificación del mismo, el marco de trabajo proporciona control y mejora de los procesos. Lo que se considera como ventaja para este estándar es que puede ser adaptado para un proyecto o producto de software por lo que se considera como unos de los más completos debido, a su alcance para implementar cada proceso en el desarrollo de software.

El Instituto de Ingenieros en Eléctrica y Electrónica, IEEE por sus siglas en inglés provee un conjunto de herramientas técnicas igualmente factibles para el desarrollo de software, IEEE es un organismo que en la actualidad se dedica a la creación de estatutos para establecer buenas doctrinas que sujeten a una mejor organización los proyectos de electrónicos, mecánicos y electricistas.

Este organismo se dedica a promover la teoría y la participación en la creación de investigaciones en electro tecnología, y forman mejores productos para asegurar una mejor calidad de vida.

Dentro de los estándares de IEEE para el desarrollo de software se encuentran varios dentro de los principales, esta el IEEE/IEC 12207.0 [IEEE, 1996], que se encarga de

los procesos de desarrollo de software, el IEEE 830 [IEEE, 1998], que es necesario para el levantamiento de requerimientos, el IEEE 1074 [IEEE, 1997] que su función principal es administrar los procesos de software del proyecto.

Uno de los que más resalta es el estándar (IEEE Standard for Developing Software Life Cycle Processes) conocido también por su acrónimo IEEE-1074, este estándar es para proceso de ciclo de vida del software, es útil para cualquier organización que es responsable de administrar y desempeñar proyectos de software y se puede usar donde el software es el sistema total o donde el software es parte de un gran sistema.

En la siguiente sección se realiza un breve análisis de todo el estándar seleccionado, IEEE 1074 (Standard for Developing Software Life Cycle Processes). Este estándar permitirá, en las fases posteriores de este trabajo la aplicación del estándar en el desarrollo de proyectos de software distribuido. A partir de este análisis es como se ha tomado la decisión de elegir el estándar como base para desarrollar este trabajo.

La decisión de proponer un estándar como IEEE 1074 es que éste se ajustará para un modelo de ciclo de vida de software, de acuerdo a las cláusulas de Administración de un proyecto desde su inicio, así también por que es independientemente de las herramientas de lenguaje de programación, sin restringir la plataforma que se utilice ya sea CORBA, RMI y DCOM abren la posibilidad de que el desarrollo del producto de software sea abierto para los sistemas distribuidos.

6.2 Análisis del Estándar 1074 de IEEE

El estándar proporciona una estrategia para crear un proceso Ciclo de Vida de Software (SLCP). Esta dirigido principalmente a la arquitectura de procesos para un proyecto de software. Define la forma en que es creado un proceso. Es útil para toda organización que es responsable en la administración y realización de proyectos de software. Puede utilizarse donde el software es el sistema total o donde el software es parte de un gran sistema. [IEEE 1074,1997].

Esta metodología comienza con la selección de un modelo de ciclo de vida del software para uso específico de un proyecto. La organización de este estándar se resume en cinco cláusulas principales ver Tabla No.6.1

Tabla 6. 1 Organización del estándar

Elemento	Titulo
Cláusula 1	Vista general
Cláusula 2	Referencias
Cláusula 3	Definición de acrónimos
Cláusula 4	Conceptos claves
Cláusula 5	Implementación del Estándar
Anexo A (Normatividad)	Actividades
Anexo B(informativa)	Ejemplo de planeación
Anexo C(informativa)	Información de planear una Plantilla
Anexo D (Informativa)	Bibliografía

La cláusula cuatro que se refiere a los conceptos claves, describe una explicación previa a los conceptos claves que serán utilizados en el estándar, finalmente la cláusula cinco que es la Implementación del Estándar son las que detallan mayor información acerca de los procesos que se tienen que implementar.

6.2.1 Conceptos Claves del Estándar

La cláusula cuatro presenta cuatro secciones que se dedican a:

1) Actividades.- En esta parte se consideran las entradas y salidas de la información, que tendrá cada actividad, será completa una Actividad cuando se han procesado todas las entradas de información y se han generado todas las salidas de información.

a) Formato. En esta parte se determinan las partes en que consiste cada actividad. Comenzado por la información de entrada, descripción y salida de información.

b) Criterios de entrada y salida.- Para introducir uno de los principales criterios para comenzar una actividad es que se deberá presentar por lo menos un elemento específico de entrada. Para que una salida de información sea completada, deberá ser procesada toda la información de entrada, así también se espera que cada proyecto determine el flujo de requerimientos durante la planeación de las actividades del SLCM.

c) "Si aplica" la Actividad.- Se marcan como aplicables aquellas actividades que se consideran obligatorias, cada Actividad se considera aplicable cuando la información de salida viene a ser disponible para ser usada por otras actividades.

d) Estructura Organizacional.- Menciona que a cada persona se le deberá de asignar su responsabilidad para el desempeño de las actividades y para la calidad de las entradas y salidas del grupo de información.

2) Elementos del proceso del ciclo de vida: En la Figura 6.2 se presenta la descripción de los conceptos claves en el desarrollo de un SLCP.

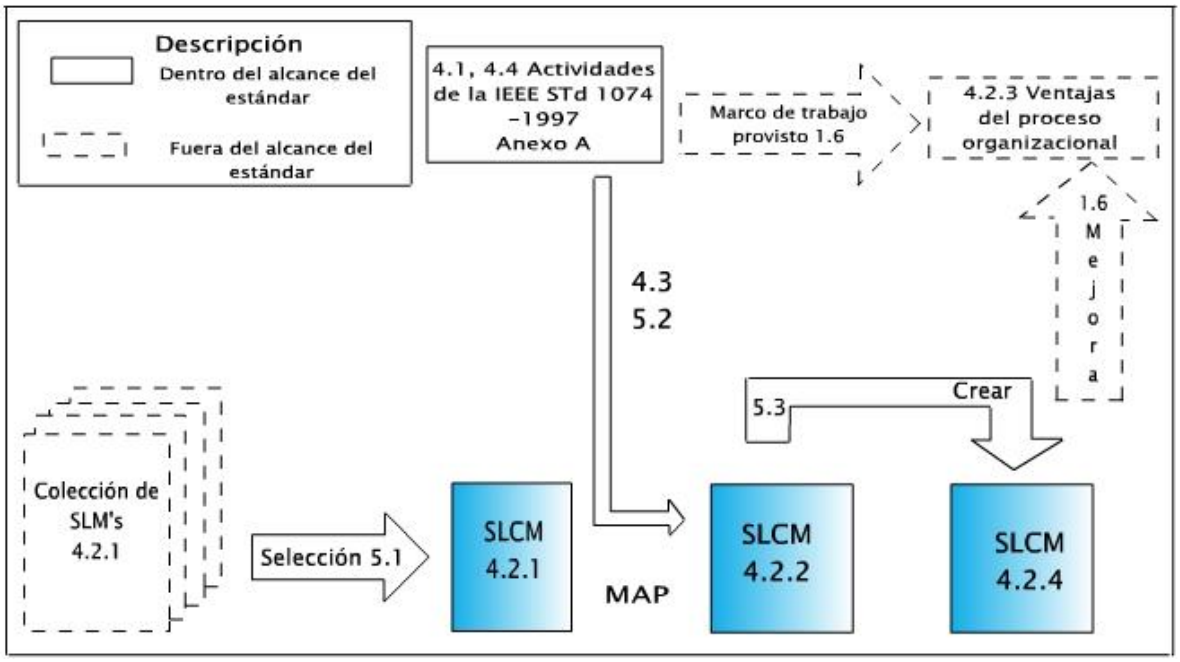

Figura 6. 2 Desarrollo de SLCP

a) SLCM.- El modelo de ciclo de vida del software se deberá seleccionar para un proyecto. Esta sección esta basada sobre los atributos del proyecto y capacidad organizacional.

b) SLC.- El SLC es una secuencia ejecutable de actividades que son realizadas durante el proyecto, es creado para planear las actividades provistas en el Anexo A del estándar.

c) OPA's.- Se consideran como los artefactos, que definen el ambiente de una organización para los proyectos de software. Estos artefactos se seleccionan y se adaptan para un proyecto en particular.

d) SLCP.- Este es creado para aumentar el SLC con los OPA's que son seleccionados para el proyecto. Provee de un enfoque específico para ser usado en el proyecto.

3) Planeación.-Al planear la secuencia ejecutable de las actividades en este estándar desde un SLCM seleccionado. Las actividades se pueden planear en tres formas: Instancia, Iteración e Invocación.

a) Instancia.- Una actividad se planea como una instancia, si sus entradas específicas, son procesadas y sus salidas especificas producidas.

b) Iteración.- Una actividad se planea como una iteración si al menos una de las entradas de información es procesada y alguna de su información de salida es creada.

c) Invocación.- Esta se refiere al llamado que se realiza a las actividades, desde otros grupos de actividades que son paralelos.

4) Entrada de información, salida de información: Las tablas de entradas y salidas muestran el flujo de información entre las actividades en el anexo A. La información se muestra a través de flujos de entrada y salida.

Convenciones.- la entrada de información y de salida, para cada Actividad está lista en un formato de tres columnas. La fuente o destino de la información del grupo de actividades se muestra en dos columnas.

Información Externa.- La fuente de información externa y destino están fuera del alcance de este estándar. Este tipo de información esta dividido en dos partes en Información Externa de salida y Externa de entrada, donde esta ultima no es requerido para plantear una actividad y cuando no existe este tipo de información no debe ser usada.

Información Genérica.- En varios casos, la información de entrada y salida de las columnas en las tablas designa la información especifica que entra o sale de la actividad. Sin embargo, algunas actividades tienen información de salida la cual es destinada a resguardarse según la actividad del grupo que le corresponda.

Información vs Documentos.- Este estándar no requiere de la creación de documentos específicos. La información que resulta de la ejecución de las actividades se espera para estar coleccionada en cualquier forma es consistente con el SLCM seleccionado y los OPA’s.

6.2.2 Implementación del Estándar

La cláusula cinco establece tres secciones, esta es llamada de Implementación del Estándar:

1) Selección de los modelos del ciclo de vida del software (SLCM)

Inicialmente el proceso de arquitectura debe identificar el SLCM para el cual se planean las actividades. Este paso abarca localizar, evaluar, seleccionar y adquirir un SLCM. Esto es posible para una organización que tiene múltiples SLCM, sin embargo, solamente se debe seleccionar un modelo para un proyecto.

Los procesos de arquitectura deben seguir los siguientes cinco pasos para evaluar y seleccionar un SLCM.

- Identificar todo los SLCM que están disponibles para el desarrollo del proyecto.
- Identificar los atributos que aplicaran para el efecto deseado del sistema y el ambiente desarrollado.
- Identificar cualquier obligación que se deba imponer sobre la selección.
- Evaluar varios SLCM basados sobre las experiencias pasadas y capacidades organizacionales.
- Seleccionar el SLCM que satisfaga mejor los atributos y obligaciones del proyecto.

2) Selección del ciclo de vida del software (SLC).

Se identifican las actividades del Anexo A este debe ser planeado sobre SLCM.

a) Ubicar las actividades en secuencia ejecutable.

El orden en el cual las actividades serán realizadas se determina por tres principales factores:

i) El SLCM seleccionado será dictado una orden inicial de actividades.

ii) Las obligaciones pueden requerir la coincidencia de actividades en el SLCM. En este caso las actividades pueden estar en ejecución.

iii) El orden de las actividades puede ser impactado por la entrada y salida de criterio de las actividades asociadas. La disponibilidad de la información de salida es una actividad podría afectar el comienzo de otra actividad. La segunda actividad puede requerir, tantas entradas como sean requeridas, para la obtención de salidas en cada actividad.

b) Desarrollo y justificación a una lista de Actividades no usada.

Todas las actividades, que no son aplicables para este proyecto deben ser identificadas y explicadas en la lista de Actividades no usadas.

c) Verificar el mapa.

Los procesos de arquitectura deben asegurar que las actividades están completamente planeadas sobre el SLCM. Los resultados contenidos en SLC de las actividades para completar exitosamente un proyecto de software.

3) Establecer uno Proceso de Ciclo de Vida del Software (SLCP).

Los pasos anteriores deberán asegurar para desarrollo del SLC. Como el siguiente paso, se debe aplicar la disponibilidad de OPA's, para las actividades SLC, y se debe reconciliar la obligación. Las salidas de información generadas por cada actividad deben asignarse para los documentos apropiados. El resultado es lo establecido por el SLCP.

Finalmente el estándar anexa cuatro elementos o secciones que proporcionan ayuda para el desarrollo de sistemas.

Los Anexos B, C y D son informativos e incluyen información útil, pero no requerimientos, mientras que el Anexo A es normativo.

Los componentes del Proceso de Ciclo de Vida consisten de 65 actividades. Estas actividades están incluidas en el Anexo A y están organizadas en 17 Grupos de Actividades, los grupos de Actividad son además agrupados dentro de cinco secciones como muestra en la Tabla No. 7.1.

Tabla 6. 2 Grupos de Actividades

Sección del Titulo	Cláusula	Grupos de actividad
Administración de Proyecto	A.1	Iniciación de Proyecto Planeación de Proyecto Control y monitoreo de proyecto.
Pre-Desarrollo	A.2	Exploración de Concepto Asignación de Sistema Importación de Software
Desarrollo	A.3	Requerimientos Diseño Implementación
Post-Desarrollo	A.4	Instalación Operación y soporte Mantenimiento Cambios o Retiro
Integración	A.5	Evaluación Administración de configuración de software Desarrollo de configuración Entrenamiento

Las actividades que se realizan adecuadamente están planeadas dentro de un SLCM y así mismo las actividades que están invocadas por otras actividades.

6.3 Propuesta de solución al proyecto

En esta sección se considera la propuesta del proyecto, para el desarrollo de este trabajo se propone utilizar la cláusula cinco del estándar donde se implementa esta cláusula, en conjunto con el anexo A, y que en este último se establecen las actividades que son adaptables para el proceso de Administración de Proyecto. El anexo A (normativo) del estándar, contiene las cláusulas donde se encuentran los grupos de actividades, estas actividades serán expuestas para ser planeadas sobre el SLCM seleccionado.

Se propone el uso del anexo A para la creación de actividades en el desarrollo del sistema, cada sección de Administración de Proyecto consta de grupos de actividad, así como cada grupo de actividad contará con sus tareas a realizar. Este tipo de tareas ayudaran a la mejor administración de desarrollo y diseño, asegurando de esta forma adaptabilidad, desempeño, utilidad, seguridad y confiabilidad.

La finalidad es ajustar dichos procesos, donde a través de ellos se implementen la Administración de proyecto, pre-desarrollo, desarrollo, post-desarrollo e integración del sistema.

Estos procesos se ajustan adecuadamente al desarrollo de un sistema distribuido debido a las líneas guías que proporciona un estándar como el IEEE 1074, ya que completa todas fases del ciclo de vida del software y ayuda a una mejor comprensión de la administración del proyecto. Así también se considera este estándar debido a que las actividades están directamente orientadas al refuerzo del desarrollo.

En el caso de construir software distribuido busca, se satisfagan las necesidades primordiales, además de que la estructura del sistema, soporte las necesidades que requiera la aplicación, es por ello que se necesita que se construyan adecuadamente procesos, bien establecidos y completos para seguir una construcción de software integra.

En los estándares de desarrollo de software su función principal es crear un modelo de ciclo de vida para el Software, por otro lado este no contempla los procesos para la contratación, adquisición o desarrollo de hardware, solo se basa en crear el desarrollo de software, no dirige un método de un modelo de ciclo de vida de software específico.

6.4 Planteamiento del proyecto de conformidad con el estándar

En esta sección se presenta la propuesta del Modelo de Desarrollo de Sistemas Distribuidos, donde se describirá de forma gráfica el seguimiento que tendrá el desarrollo de las actividades para el modelo propuesto, esto con el fin de presentar de manera clara la implementación de un proyecto de software y los procesos que se llevaran acabo a través de su creación.

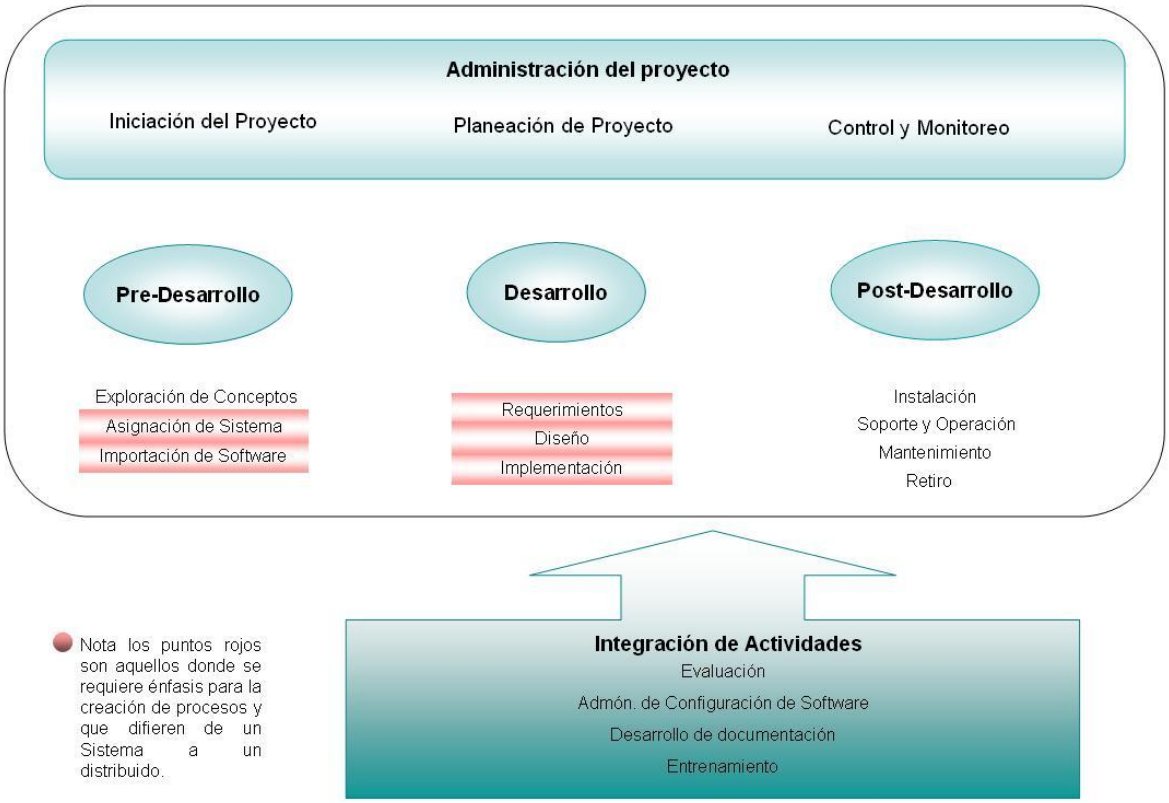

Figura 6. 3 Modelo propuesto para desarrollo de Sistemas distribuidos.

El modelo que se presenta en la parte superior tiene cinco divisiones Administración del proyecto, Pre-Desarrollo, Desarrollo, Post-Desarrollo e Integración de Actividades, donde la Administración del proyecto regirá a las tres áreas principales de creación del producto que son Pre-Desarrollo, Desarrollo y Post Desarrollo, la ultima de las áreas Integración de Actividades estará dedicada a la evaluación de procesos y verificaciones que se apliquen durante el proceso de ciclo de vida del Software.

Comenzaremos por describir el área de Administración del proyecto esta parte se divide en:

Iniciación del Proyecto.- Esta encargada de la iniciación de los procesos de ciclo de vida de Software.

Planeación del Proyecto.- Se encarga del plan de evaluaciones y de la administración del plan de configuración, esta encargado de la instalación del sistema, documentación, entrenamiento e integración del sistema.

Control y Monitoreo.- La siguiente estructura del modelo, involucra todos aquellos procedimientos que llevan a cabo la parte administrativa y control de métricas del desarrollo del sistema, como es el caso de evaluaciones dentro de la planeación, definición de requerimientos del desarrollo del sistema, esto se refiere al seguimiento de vigilancia y verificación de los requerimientos solicitados.

La siguiente sección es Pre-Desarrollo en esta parte se secciona en Exploración de Conceptos, Asignación de Sistema, Importación de Software. Las partes de Localización de sistema e Importación de Software difieren de la administración de un sistema a un distribuido, es importante hacer hincapié que será necesario ubicar las necesidades principales de requerimientos del sistema distribuido ya que se convierte en un puente entre el la fase de exploración y la definición de Requerimientos.

Posterior a este se encuentra la fase de Desarrollo, la cual consta de Requerimientos, Diseño e Implementación. Esta fase de requerimientos es de importancia para la creación de sistemas distribuidos, ya que se dará principal interés al área de estructura del proyecto distribuido, debido a que la localización de clientes y de servidores requieren de saber donde estarán ubicados sus componentes.

Otra de las secciones posteriores es la parte del Post-Desarrollo, donde se divide en Instalación, Operación y Soporte, Mantenimiento y Retiro.

La última de las Fases es la de integración de Actividades, se divide en:

- Evaluación
- Administración y configuración de Software
- Desarrollo de documentación
- Entrenamiento

En esta sección se realizaran las actividades que durante el proceso de ciclo de vida del Software son diseñadas para cubrir los defectos en el producto y los procesos que se usan para el desarrollo del producto.

CAPÍTULO 7 APLICACIÓN DEL ESTÁNDAR

7.1 Aplicación del Estándar de la IEEE 1074

En este capítulo se realizará la aplicación del modelo propuesto, por lo que a continuación se presenta un análisis detallado de éste y de cada una de las actividades que serán consideradas para sistemas distribuidos (SD), como se explicará más adelante se realiza un estudio detallado de las actividades que son consideradas para la creación de actividades en proyectos distribuidos, ya que los requerimientos para el desarrollo de SD abarcan otras características, como la planeación del sistema del cliente y del servidor, desarrollo de arquitectura e implementación de estos.

El modelo del ciclo de vida para el desarrollo de sistemas distribuidos se dividirá en tres partes:

Actividades del grupo de Administración del proyecto donde se gestionan todos los documentos que controlaran cada parte del Pre-Desarrollo, Desarrollo y Post-Desarrollo, es necesario ubicar los documentos para cada uno de los procesos en la iniciación, control, monitoreo y planeación de proyecto para la creación de un sistema distribuido.

Actividades del grupo de pre-desarrollo, desarrollo y post-desarrollo, se consideran dentro del mismo nivel, bebido a que las actividades que se realizan están unidas para la creación del software y que se encuentran dentro de las actividades para el modelo del ciclo de vida del software SLCM. En las actividades de desarrollo se toman en cuenta las actividades para el cliente y el servidor del sistema distribuido, ya que para la creación del sistema distribuido el desarrollo del servidor en el sistema tiene otras características para la arquitectura y diseño del sistema, al igual que el cliente o clientes.

Las Actividades del grupo de integración, consta de evaluar, controlar y definir cuales serán los objetivos de mejora, estas actividades se aplican también para el cliente y servidor en el desarrollo de SD, el grupo de actividades de evaluación tiene como objetivo la evaluación de pruebas al sistema distribuido, en cada una de las entradas de información que son requeridas para asegurar la completa tarea de cada actividad y de esta forma no

carezca de buen funcionamiento del sistema, así como también se asegura la calidad de las funciones del proyecto.

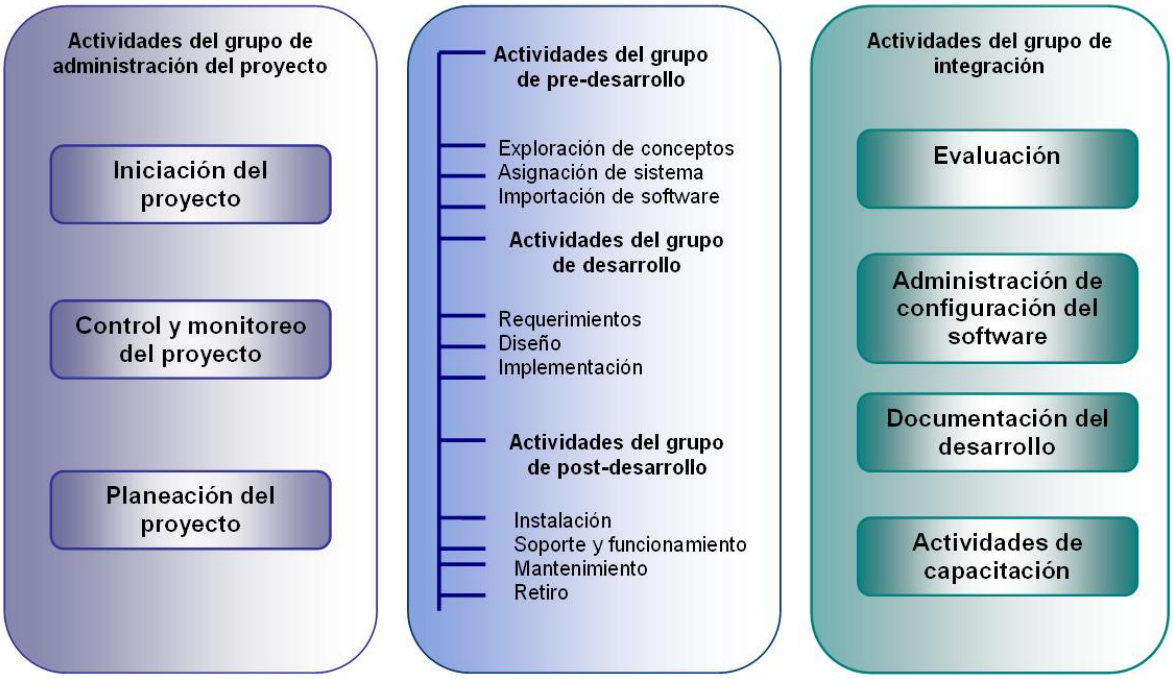

Figura 7. 1 Modelo propuesto para las Actividades de grupo

7.2 Actividades del grupo de Administración del proyecto

Como se muestra en la siguiente Figura No. 7.2 se describe cada una de las fases de las Actividades del grupo de Administración de proyecto, para la creación de SD, es importante resaltar que en esta actividad la planeación del proyecto es esencial en el proceso de desarrollo del proyecto.

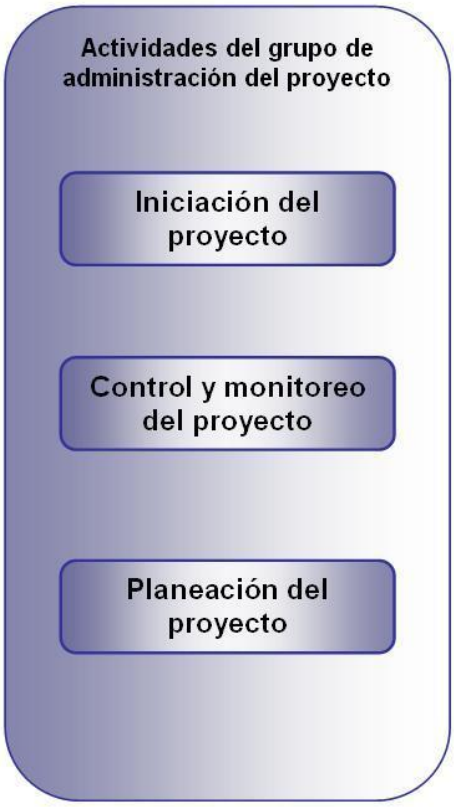

Figura 7. 2 Actividades del grupo de administración del proyecto

Las actividades de la administración de Proyecto llevan consigo la parte fundamental de cómo se completan cada uno de los procesos y actividades. Este se describe en el transcurso de la siguiente sección:

7.2.1 Iniciación del Proyecto.

En esta iniciación se administran tanto las actividades para la creación e infraestructura de un desarrollo de sistema distribuido y mantenimiento del mismo. Es la base para el SLCP (Proceso de ciclo de vida del Software) completo. En las actividades siguientes se aplican al igual que a un diseño de sistema no distribuido, sin embargo en las siguientes líneas se describen las actividades que son realizadas conforme a los anexos de la actividad 1.1. La fase consta de cuatro grupos de actividades preponderantes Creación del proceso de ciclo de vida de software, Estimación del desempeño, Asignación de recursos al proyecto, Definición de métricas, estas actividades son utilizadas para el comienzo de

actividades en el proyecto, como la asignación del modelo que se utilizará para el inicio de la administración de los procesos, se consideran recursos, datos de estimación en los tiempos de desarrollo, además de que estas actividades son descritas en los anexos y que se escriben en la siguiente Tabla 7.1.

Tabla 7. 1 Actividades de iniciación estándar IEEE

Actividades de Iniciación del Proyecto	Para aplicación al proyecto distribuido
Creación del proceso de ciclo de vida de software	Anexo A A.1.1.1
Estimación del desempeño	Anexo A A.1.1.2
Asignación de recursos al proyecto	Anexo A A.1.1.3
Definición de métricas	Anexo A A.1.1.4

Se toman los anexos que se muestran en el estándar IEEE Std 1074-1997, especificando que las actividades importantes para la iniciación del sistema distribuido, así como las entradas de información que se reciben para la realización de cada tarea.

En el caso de la Creación del proceso de ciclo de vida de software, se considero el modelo de ciclo de vida, en cascada donde se comienza desde la administración, levantamiento de requerimientos, diseño e implementación, el proceso para un sistema distribuido como para un sistema que centralizado (no distribuido) se considera igual en esta fase del estándar, ya que se mantiene la misma línea de aplicación para ambos tipos de sistemas.

La aplicación de las actividades que se describen a continuación son únicamente para la creación de SD, esto haciendo hincapié en la forma en como se construye el software cliente y el servidor para dicho sistema distribuido. Las actividades del grupo de planeación del proyecto y control de monitoreo, requieren de mayor cuidado en el momento de establecer los requerimientos para el sistema distribuido, así también los grupos de Pre-Desarrollo, Desarrollo y Post-Desarrollo, del mismo estándar, es por ello que más adelante se describe de forma detallada como se aplicaron para el desarrollo del sistema distribuido

7.2.2 Planeación del proyecto

Esta dirige la planeación para toda la administración del proyecto, incluyendo contingencias. Estas actividades se pueden hacer como están descritas en el mismo estándar

IEEE 1074, sin embargo para su elaboración en la creación de sistemas distribuidos se detalla cuales son las actividades que son modificadas debido al tipo de información requerida, las actividades que serán documentadas en este anexo son el de Evaluaciones, Administración de configuración, sistema de transición, instalación, documentación, capacitación, administración de proyecto e integración, en ellas se describen la planeación de configuración de cómo se administrara cada documento de instalación y transición para cada actividad en los procesos del proyecto, se presentan las actividades y los anexos en la Tabla 7.2.

Tabla 7. 2 Actividades de Planeación del proyecto

Actividades de Planeación del proyecto	Para aplicación al proyecto distribuido
Plan de Evaluaciones	Anexo B 1.2.1
Plan de administración de configuración	Anexo B 1.2.2
Plan de sistema de transición	Anexo B 1.2.3
Plan de Instalación	Anexo B 1.2.4
Plan de documentación	Anexo B 1.2.5
Plan de capacitación	Anexo B 1.2.6
Plan de la administración de proyecto	Anexo B 1.2.7
Plan de integración	Anexo B 1.2.8

Es importante resaltar que las actividades descritas en las tablas, muestran aquellos grupos de actividades que fueron adaptadas para el desarrollo de sistemas distribuidos y que son descritas en los anexos de la misma forma definidos en la columna de aplicación al proyecto distribuidos.

7.2.3 Control y monitoreo de proyecto

La administración de los procesos de esta actividad, deben controlar e identificar la evaluación de tareas que son necesarias para asegurar que el producto de software y los esfuerzos de desarrollo conozcan sus metas, en la Tabla No. 7.3 se describen las actividades correspondientes Manejo de riesgos, Control de proyecto, Identificación de necesidades de mejora, Coleccionar y analizar datos métricos, en estas actividades se muestran los controles y manejo de riesgos de las diferentes actividades a realizar, por ejemplo la identificación de necesidades en el sistema distribuido comenzaría por conocer el alcance

con el que contara el sistema y cuantos equipos controlara el SD, estas actividades se encuentran descritas en el anexo B.

Tabla 7. 3 Actividades de control y monitoreo de proyecto

Actividades de Control y monitoreo de proyecto	Para aplicación al proyecto distribuido
Manejo de Riesgos	Anexo B 1.3.1
Control del proyecto	Anexo B 1.3.2
Identificación de necesidades de mejora en el proceso de ciclo de vida del software	Anexo B 1.3.3
Coleccionar y analizar datos métricos	Anexo B 1.3.5

Estas actividades son usadas para trazar y administrar el proyecto, abarcan la colección y análisis de las métricas de software del proyecto, el resguardo de registros y la identificación de oportunidades de mejora en los procesos del ciclo de vida del software. Como se describieron anteriormente las actividades de la planeación de proyecto, así se presentan las actividades de control y monitoreo con sus respectivos diagramas de entradas y salidas de información.

7.3 Actividades del Grupo de Pre-desarrollo

Dentro de las Actividades de Pre-desarrollo, estas actividades corresponden al anexo B de actividades 2.1 y son las siguientes, Exploración de conceptos, Asignación de sistema e Importación de Software, como se describen a continuación:

7.3.1 Exploración de conceptos

Las cuatro actividades de exploración de conceptos Tabla 7.4, Identificar ideas o necesidades, Formulación de aproximaciones potenciales, Estudio de la conducta de viabilidad, Refinar y fijar las ideas ó necesidades; son actividades esenciales en el desarrollo del proyecto distribuido, estas actividades son para establecer cuales serán los medios mas viables que deberán utilizarse para el desarrollo del SD, esto se refiere a que el conocer las características del sistema, como ubicación, plataforma, distribución del servidor y de los clientes, será administrado por esta parte de las actividades y que se muestran en el anexo B.

Tabla 7. 4 Actividades de exploración de conceptos

Actividades de exploración de conceptos	Para aplicación al sistema distribuido
Identificar ideas o necesidades	Anexo B 2.1.1
Formulación de aproximaciones potenciales	Anexo B 2.1.2
Estudio de la conducta de viabilidad	Anexo B 2.1.3
Refinar y fijar las ideas o necesidades	Anexo B 2.1.4

7.3.2 Asignación del sistema

En el área de Actividades de Pre-desarrollo, se encuentran las actividades de Asignación de sistema, con las actividades de: Funciones para analizar, Desarrollo de la arquitectura de sistema y Descomponer requerimientos de sistema, y que corresponden al anexo B de actividades 2.2, las actividades son mostradas en la Tabla 7.5. Para el caso de sistemas distribuidos se consideran cada una de estas partes necesarias para el levantamiento y requerimientos de todos aquellos elementos como hardware, distribución de equipo, arquitectura con la que se cuenta, comunicación en la distribución de información. Por lo tanto, es necesario que sea definida la parte de Software, como protocolos de comunicación, sistemas operativos que interactúan, entre otras aplicaciones, la asignación de sistema, consta de la descripción funcional del sistema, la parte esencial es que la arquitectura del sistema viene a hacer la base para el grupo de actividades de diseño.

Tabla 7. 5 Actividades de Asignación del sistema

Actividades de asignación de sistema	Para la aplicación de sistemas distribuidos
Funciones para analizar	Anexo A A.2.2.1
Desarrollo de la arquitectura de sistema	Anexo A A.2.2.2
Descomponer requerimientos de sistema	Anexo A A.2.2.3

7.3.3 Importación de Software

Las actividades 2.3 de Importación de software corresponden al anexo B y se muestran en la Tabla 7.6. Estas actividades consisten en describir las librerías de código, controladores de dispositivos, utilerías y sistemas que son completamente funcionales que están para ser integradas dentro del actual proyecto. Dentro de este se encuentra la evaluación de Importación del software, esta actividad se considera para determinar los requerimientos de migración de software y que sean satisfechos para usarse a partir de otro

proyecto con la organización, incluyendo artículos de librerías anteriores que se requieren para el desarrollo de las nuevas herramientas, así también fuentes de modelos de software fuera de la organización que desarrolla el proyecto.

Tabla 7. 6 Actividades de importación del software

Actividades de importación de software	Para la aplicación de sistemas distribuidos
Identificar los requerimientos de software a importar	Anexo B 2.3.1
Evaluar las fuentes de importación de software	Anexo B 2.3.2
Definir métodos de importación de software	Anexo B 2.3.3
Importación de software	Anexo B 2.3.4

7.4 Actividades de Grupo de Desarrollo

Se establecen tres actividades, estas se describen como se muestra en la siguiente Tabla 7.7, Definición y desarrollo de requerimientos de software, definir requerimientos de interfase y priorizar e integrar requerimientos de software, estas actividades se definen en el anexo B adaptado para SD y son aplicables para cada uno de los requerimientos de la interfaz y la integración de componentes en el proyecto, el levantamiento de requerimientos que se realice en estos procesos deberán incluir detalladamente cual será el funcionamiento del cliente y del servidor, pero descritos en los documentos para su levantamiento, tales como formatos, guías de procedimientos y diseños previos de interfaz.

7.4.1 Requerimientos

Esta actividad comprende las actividades 3.1 del anexo B, siendo las siguientes: Definición y desarrollo de requerimientos de software, Definir requerimientos de interfaz y Priorizar e integrar requerimientos de software. En este se deben de tomar en cuenta requerimientos de hardware, humanos y componentes de software, se consideran parte de los requerimientos funcionales del sistema, tales como la interfaz, así también es necesario definir cuales son los requerimientos prioritarios para la integración del SD, el proceso que se realiza para este tipo de proyectos se muestra en el anexo B, donde se observa que es necesario terminar la información de salida de las actividades de la Asignación de sistema y administración del proyecto, para utilizarla como información de entrada en esta actividad.

Tabla 7. 7 Actividades de Grupo de Desarrollo

Actividades de Requerimientos	Aplicación para sistemas distribuidos
Definición y desarrollo de requerimientos de software	Anexo B 3.1.1
Definir requerimientos de interfase	Anexo B 3.1.2
Priorizar e integrar requerimientos de software	Anexo B 3.1.3

7.4.2 Diseño

Las actividades de Diseño corresponden al punto 3.2 del anexo B, así como una breve descripción de como trabajan estas actividades y lo que realizan en cada parte del proyecto. Como se muestra en la siguiente Tabla 7.8, donde las actividades son las siguientes: Realizar diseño arquitectónico, Diseño de la base de datos, Diseño de interfaz y Realizar diseño detallado, estas actividades son ajustadas y consideradas para el desarrollo del SD, donde se debe dar mayor importancia en la estructura básica del sistema ya que depende de la creación de cómo se encuentre diseñado, desde las bases de datos, interfaz, establecimiento del lugar de los clientes y los servidores, las actividades de diseño utilizan como información de entrada parte de las actividades de la Planeación del proyecto, del desarrollo de la arquitectura del sistema, Requerimientos del software a importar y por ultimo de Priorizar e integrar los requerimientos de software.

Tabla 7. 8 Actividades del Grupo de Diseño

Actividades de Diseño	Aplicación para sistemas distribuidos
Realizar diseño arquitectónico	Anexo B 3.2.1
Diseño de la base de datos	Anexo B 3.2.2
Diseño de interfaz	Anexo B 3.2.3
Realizar diseño detallado	Anexo B 3.2.4

7.4.3 Implementación

Las actividades de implementación corresponden al anexo B 3.3, resultan de la transformación del diseño detallado del producto de software en el lenguaje de realización. La siguiente Tabla 7.9 muestra las siguientes actividades Crear código ejecutable, Crear documentación de funcionamiento y Realizar integración, donde se deberá especificar la creación del código ejecutable y son generados usando el SLCP, están involucradas para el

control y revisión de integración, por la parte de las actividades de Administración de control.

Tabla 7. 9 Actividades del Grupo de Implementación

Actividades de Implementación	Aplicación para sistemas distribuidos
Crear código ejecutable	Anexo B 3.3.1
Crear documentación de funcionamientos	Anexo B 3.3.2
Realizar integración	Anexo B 3.3.3

7.5 Actividades del grupo de Post-desarrollo

Esta área de actividades esta encargada de desempeñar la instalación, operación, soporte, mantenimiento y retiro del producto de software. Donde se proponen las siguientes actividades para su implementación, Instalación, Soporte y funcionamiento, Mantenimiento y Retiro del sistema, estas actividades se pueden establecer directamente como se encuentra el estándar utilizado (IEEE 1074), mientras que las actividades antes mencionadas en las tablas son actividades adaptadas para sistemas distribuidos.

7.5.1 Instalación

Se proponen las actividades de distribuir, instalar y aceptar el software en el ambiente operacional. Estas se describen a continuación para las actividades recomendadas a desarrollar.

En estas actividades se encuentran la planificación del proyecto, administración del plan del proyecto, documentación de operación y la evaluación de prueba en la instalación del software.

7.5.2 Soporte y funcionamiento

Proporcionar asistencia en la operación del producto incluye consulta al usuario y soporte de requisitos para el mantenimiento, monitoreo y control.

El aceptar el software que se arroja como producto, debe de consistir de un análisis de la información de reporte de evaluación, de acuerdo al usuario de aceptación planeada para asegurar que el software instalado. Cuando el análisis de resultados satisface los

requerimientos del usuario, entonces el sistema es aceptado por el usuario. En el caso del sistema distribuido se requiere aplicar esta actividad para ambas aplicaciones del cliente y el servidor.

7.5.3 Mantenimiento

Este grupo de actividades se encarga de la identificación de los errores de software, fallas y faltas. Los requerimientos para el mantenimiento de software inician con los cambios de SLCP, las actividades del mantenimiento constan de identificar las necesidades para las mejoras de software y salidas de las recomendaciones de mejora de software de acuerdo al SPMI.

7.5.4 Retiro

Estas actividades involucran remover un sistema existente desde su activación de soporte o uso por su operación o soporte, o por remplazar con un nuevo sistema o una versión del sistema existente. La información planeada del retiro con lleva las actividades de generar una migración de un sistema anterior.

7.6 Anexos guía para la adaptación del estándar.

En los documentos que se presentan en la parte posterior de este trabajo, son tres de los cuales el primer *Anexo A*, corresponde al estándar IEEE 1074 del proceso de Ciclo de vida de software, como se muestra es una copia fiel del anexo del estándar y que proporciona una vista de lo establecido de sus actividades y tareas, los otros dos anexos se realizaron como parte del desarrollo de esta tesis y se muestran al final de este documento.

El siguiente documento *Anexo B*, este muestra el modelo propuesto con las actividades de cambio para el diseño de sistemas distribuidos, la forma en que presentan dichas actividades es a través de diagramas, donde las actividades de entradas están definidas por rectángulos sombreados del lado izquierdo mientras que las actividades; de salida son las del lado derecho, igualmente en rectángulos sombreados. Cada actividad esta

descrita en el anexo, y se muestran cuales son las actividades que se recomiendan para el desarrollo de sistemas distribuidos.

Por ultimo se encuentra el *Anexo C*, el cual contiene aquellas actividades que son aplicables para el proyecto, por cada actividad se presenta una vista de lo que es aplicable para el desarrollo y se describen únicamente aquellas actividades presentadas en el modelo del estándar adaptado.

CAPÍTULO 8 RESULTADOS

Dentro de los resultados preliminares, para la implementación del sistema se encuentra que a partir del levantamiento de requerimientos es necesario que se considere la división del sistema en Cliente y Servidor, para la elaboración de los sistemas distribuidos, cabe mencionar que el modelo propuesto en este trabajo, esta constituido por una división de tres grupos para la administración del proyecto, los cuales son tomados del estándar logrando adaptarse al medio para dar importancia a aquellas actividades que involucran a los sistemas distribuidos.

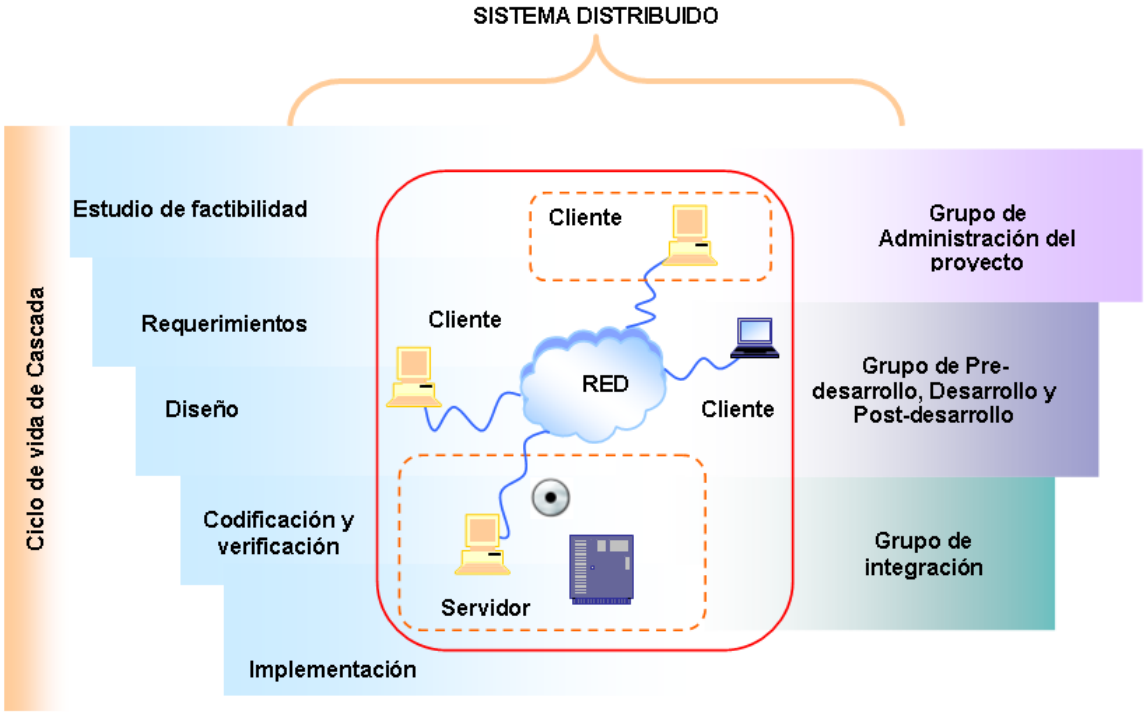

Figura 8. 1 Modelo de adaptación para sistemas distribuidos.

El origen de construir un sistema distribuido de forma independiente, se puede ver en la figura No. 8.1 donde se representa el modelo de adaptación para sistemas distribuidos, se conoce que la construcción de estos puede generarse de dos formas la primera es a través del desarrollo independiente, con esto se refiere a elaboración de documentos y actividades de desarrollo de forma separada, tanto para el sistema del cliente como para el sistema del servidor, el área marcada con líneas punteadas, muestra la separación para ambos sistemas,

siempre y cuando las actividades sean realizadas dentro de los grupos de administración del proyecto, pre-desarrollo, desarrollo y post-desarrollo, esto lleva a realizar de forma rápida otras actividades como el levantamiento de requerimientos y de planeación.

La segunda parte corresponde a realizar el desarrollo del sistema distribuido por integración, en la figura anterior se aprecia una línea continua roja la cual muestra la integración de las actividades del modelo de adaptación y que se involucran en el ciclo de vida del software para dichos SD.

La integración de las actividades del modelo de adaptación del estándar 1074 de IEEE, se lleva acabo a través de grupo de actividades, estas son separadas en la siguiente forma por tareas que son integradas, en otras actividades, estas a su vez completan otras actividades de los mismos grupos, esta división ayuda a completar de forma de integración aquellas actividades al grupo de administración que se encuentren en las actividades de grupo de pre-desarrollo, mientras que otras forman parte de las actividades del grupo de desarrollo.

Las actividades de cada grupo, se integran durante la elaboración de cada tarea en el proceso del desarrollo de software podemos apreciar en los diagramas de actividades para cada grupo, un ejemplo de integración se muestra en la siguiente Figura No. 8.2, ejemplo: en las actividades de planeación del proyecto en la actividad 1.2.8 están las actividades de varios grupos que deben ser integrados en actividades de planeación de administración del proyecto y del grupo de pre-desarrollo.

A continuación, en esta figura se aprecia la integración de las actividades en el grupo del Plan de integración, tal es el caso del grupo de actividades de entradas que se presentan las integraciones como una ventaja para el avance de las actividades de trabajo, vemos que en el plan de integración (1.2.8), las actividades del plan de evaluaciones (1.2.1) y manejo de riesgos (1.3.1), control del proyecto (1.3.2) pertenecen al grupo de Administración del proyecto, donde se muestra que la integración de actividades y documentos para el análisis de requerimientos para ambos sistemas servidor y cliente, en esta parte se complementan la administración para controlar el levantamiento de requerimientos, así mismo las actividades de priorizar e integrar requerimientos de software (3.1.3), realizar diseño detallado (3.2.4) y realizar integración (3.3.3), son utilizados para

concluir el diseño de ambos sistemas del servidor y del cliente, y que forman parte del grupo de desarrollo.

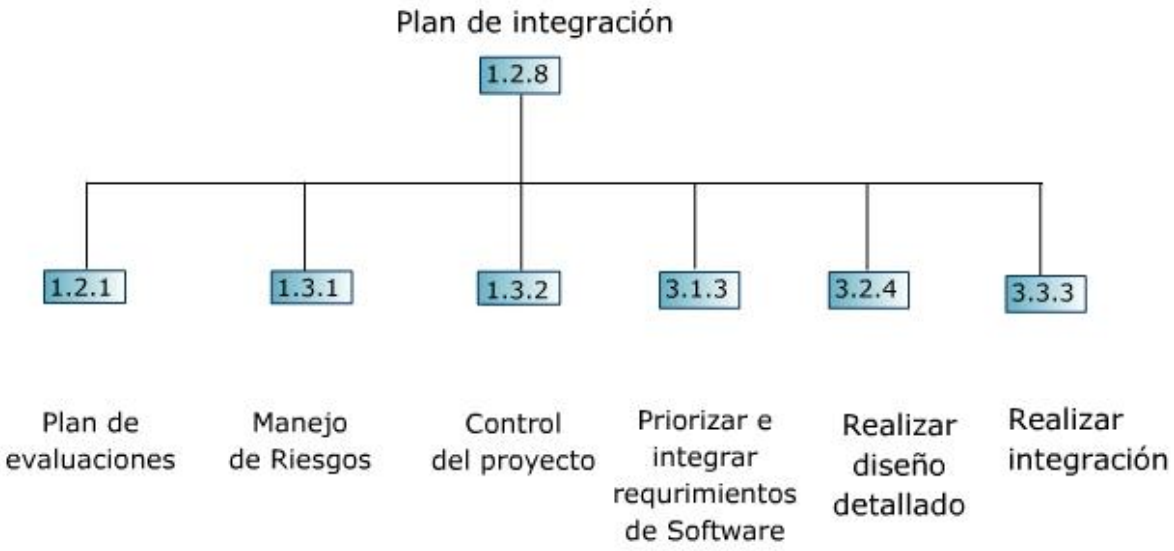

Figura 8. 2 Integración de actividades para el plan de integración.

La separación de los grupos del modelo propuesto en el estándar para desarrollo de SD, que se presentaron anteriormente: actividades de administración del proyecto, actividades de pre-desarrollo, desarrollo, post-desarrollo y finalmente de actividades de grupo integración, muestra que puede administrarse un proyecto distribuido de forma guiada por un ciclo de vida de software, donde se aprecia un conjunto de actividades tales como plan de evaluaciones, manejo de riesgos, control del proyecto, integración de requerimientos de software, realizar diseño detallado y realizar integración y que se administran de forma separada sin dejar de integrar las actividades de otros grupos de administración.

Aquellas actividades que se logran separar facilitando el determinar que documentos deberán ser integrados en cada grupo de actividades siempre y cuando correspondan al SD, es importante mencionar que las actividades que se aplican en el modelo solo corresponden al grupo de actividades de administración del proyecto y de pre-desarrollo, desarrollo y post-desarrollo, dando importancia a las partes que serán administradas desde la planeación del sistema hasta la integración del mismo SD, como se muestra en la figura No. 8.2

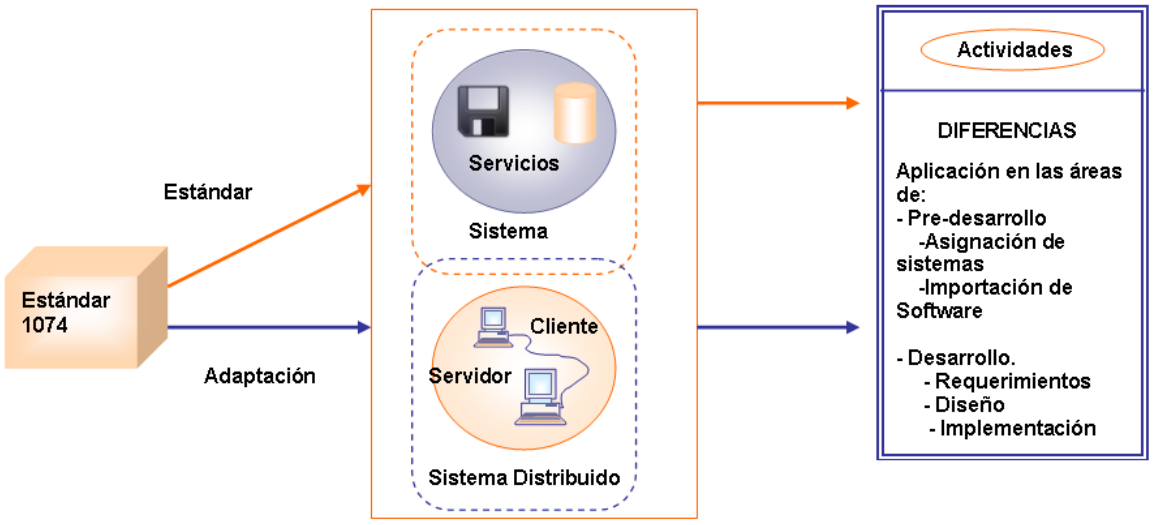

Figura 8. 3 Diferencias de adaptación a los sistemas distribuidos

La Figura 8.3 presenta las diferencias que existen entre el aplicar el estándar a un desarrollo de sistema convencional ó cliente-servidor y en establecerlo para sistemas distribuidos, entre las cuales se encuentran diferencias en el Predesarrollo, asignación de sistemas e importación de software, así como en el desarrollo de requerimientos, diseño e implementación.

Es claro que el desarrollo a un SD a un sistema integral, consta de necesidades de implementación, que se destacan de aquellas que son para un sistema de software tradicional, así también, en la presente guía se muestra cuales son aquellas actividades que son adaptables para la creación del proyecto presentado.

Lo que se puede apreciar en la guía es que todo aquello que corresponde a la implementación del servidor y requerimientos, como son planeación del proyecto, administración del proyecto y levantamiento de requerimientos será necesario integrar cada actividad que se encuentre relacionada a la creación de diseño y estructura del SD, esto proporciona una percepción mas adecuada para dar importancia a aquellas actividades que requieran cuidado para la implementación del servidor y el cliente del proyecto.

Por ultimo es notable ver que estas actividades que se presentan para cada grupo de desarrollo y post-desarrollo del modelo, tendrán una administración a través del control de

riesgos y el manejo de riesgos, dichas actividades contienen un apartado para la administración de cada actividad lo que se conoce como plan de administración de configuración, este a su vez se lleva acabo en conjunto con el modelo de ciclo de vida.

CAPÍTULO 9 CONCLUSIONES

En este trabajo se obtienen las ventajas del desarrollo de los sistemas distribuidos a través del modelo de desarrollo de sistemas con el estándar presentado; por esta razón el proporcionar esta guía, ayuda a realizar de forma óptima la administración de proyectos distribuidos. La guía propuesta puede utilizarse de forma integrada para el desarrollo del servidor y del cliente como se mostró en este trabajo, además proporciona una visión de las ventajas en el desarrollo de forma integral de aquellas actividades como los requerimientos del diseño, la administración de configuración, desarrollo de la arquitectura del sistema así como realizar diseño arquitectónico y diseño de interfases. La integración de las actividades antes mencionadas, son involucradas en cada uno de los grupos, esto ayuda a que el desarrollo tenga una línea con el diseño de la estructura del SD.

Se considera una ventaja el poder utilizar un modelo a partir de un estándar evaluado, debido a que estos estándares son altamente calificados como herramientas necesarias para el diseño estructurado de proyectos de software, sin embargo a la fecha se requiere de modelos dedicados a la creación de proyectos distribuidos y no existen modelos que administren y sean dedicados al desarrollo. La ventaja que presenta el estándar 1074, de forma integrada para cada actividad permite hacer una adaptación a la creación de sistemas SD y tomar en cuenta que actividades de los requerimientos serán utilizados para integrar adecuadamente cada actividad que se va realizando.

Los beneficios al utilizar las actividades de cada grupo, proporcionan el control de forma gradual de los procesos en cada tarea que se realiza conforme van avanzando cada una de las actividades, así mismo proporcionan que otros nuevos desarrolladores puedan continuar con la administración del proyecto en caso de que el proyecto no este concluido. Los SD comienzan a tomar parte del mercado lo que beneficia a las micro y pequeñas empresas dedicadas al desarrollo de software.

Sin embargo es importante resaltar que la mayoría de estas empresas carecen de guías establecidas para el desarrollo, es por ello que la propuesta presentada en este trabajo puede servir de guía para aquellos desarrolladores que se encuentren construyendo proyectos bajo plataformas distribuidas, sin dejar de aplicar el ciclo de vida del software otra de las ventajas que presenta, es el modelo, el cual puede ser modificable e incluso dar

uso hacia el desarrollo de proyectos educativos en las instituciones, como aportación a enseñanza de apoyo a nuevas tecnologías de ingeniería y de comunicación, es de relevancia el uso de tecnologías como estándares de calidad, permite utilizar un basto contenido de herramientas administrativas para el control de proyectos.

La diferencia de utilizar una guía establecida para el desarrollo de software, significa que los desarrolladores podrán establecer sus propias líneas de trabajo en la creación de documentos para la administración de sistemas distribuidos. El trabajo a futuro para este modelo considera que la implementación del modelo de adaptación para un SD, proporcione ayuda en la construcción de proyectos que requieran de una administración y control de elaboración de documentos de forma segura.

BIBLIOGRAFÍA

Baudes Gabriel, 1999, Sistemas Distribuidos, Ingeniería de Software III URL (http:dmi.uib.es~bbuades/sistdistr/34)

Couloris, George, Dollimore, Jean, Dollimore, Kindberg;, 1999, Distributed Systems, Concepts and Design, Addison –Wesley.

Contronco Domenico, Russo Stefano, Savy Carlo, 2000, An Integrated Approach to Design Complex CORBA Systems, Dipartimento di Informatica e Sistemistica, Universita di Napoli Federico II, Napoli, Italy.

Onur Demirôrs, Elif Demirôrs, Ayca Tarhan, 1999, Tailoring ISO/IEC 12207 for Instructional Software Development, Associate Professor, Middle East Technical Unv. Informatics Institute, Ankara-Turkey. Pagina 302.

E. Estévez, U. Gangoiti, S. Calvo, M. Marcos, D. Orive, J Portillo, N. Iriondo, I. Cabanes, I. Sarachaga, F. Artaza, I. Calvo, F. López, 1998, Entorno de Co-simulación basado en CORBA para sistemas de control industrial: Aplicación a una línea de tratamiento en caliente, URL.(http://www.shef.ac.uk/acse/utc/flexicon/publications/papers/cosimulation_paper.pdf)

Eduardo G. Jara, 2000, Definición de un sistema de Aseguramiento de Calidad para actividad de Titulación en un Currículum de Ingeniería de Software. Universidad del Bío-Bío, Departamento de Sistemas de Información, Concepción, Chile. URL (http://www.willydev.net/descargas/prev/sistemaAesgura.pdf)

Gokhale, Aniruddha, Schmidt Douglas C. “Techniques for Optiminizing; CORBA Middleware for Distributed Embedded Systems”, Department of Computer Science, Washington University, 1999.

IEEE 1074 Standard for Developing Software Life Cycle Processes.

ISO/IEC 12207, Software life cycle processes, First edition 1995-08-01.

Krawczyk,Wiszniewski, 1998, Análisis and Testing of Distributed Software Applications, Henryk Krawczyk, Bogdan Wiszniewski, Research Studies Press LTD.

Jorge E. López de Vergara, Victor A. Villagrá, Juan I. Asensio, José I Moreno, Julio J. Berrocal., 1999, Experiencias en la gestión de aplicaciones distribuidas, URL (http://jungla.dit.upm.es/~jlopez/publicaciones/telecomid99.pdf)

M. Jones , U.K. Mortense, J.Fairelough, 1997, The ESA Software Engineering Standard: Past, Present and Future, URL(http://www.ieee.org)

Peña Cabañas, Luis Miguel; 2002, Técnica de Desarrollo de aplicaciones distribuidas tolerantes a fallos sobre Arquitecturas CORBA y Java RMI, Universidad Complutense de Madrid, Mayo, URL (http://grasia.fdi.ucm.es/sensei/docs/sensei.pdf)

Piattini, Mario, Ruiz Francisco, Calero Coral, Polo Macario, “MANTEMA:a Software Mainatenance Methodology Base don the ISO/IEC 12207 Standard”, Grupos ALARCOS, Escuela superior de Informática, Universidad de Castilla-La Mancha, 1999.

Secretaría de Economía, 2001, Programa para el Desarrollo de la industria del Software http://www.economia-sniim.gob.mx/ofvir/pdis.pdf “Prosoft”, URL (http://www.economia.gob.mx/?P=1128)

James W. Moore, Noviembre/Diciembre 1999 An integrad Collection of Software Engineering Standares, The Mitre Corporation, Pag. 52, Focus, IEEE Software.

ACRÓNIMOS

CGI	Common Gateway Interface
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integrated
COM	Common Object Model
CORBA	Common Object Request Broker Architecture
DCOM	Distribuited Component Object Model
EUA	Estados Unidos de America
IDL	Interface Definition Language
IEC	International Electro technical Comisión
IEEE	Intitute of Electrical and Electronic Engineering
ISO	International Organization for Standarization
MOPROSOFT	Modelo de Procesos para la industria del Software
OMA	Object Manajement Architecture
OMG	Object Manajement Group
OPA	Organizational Process Assets
ORB	Object Request Broker
RMI	Remote Method Invocation
SD	Sistema distribuido
SEI	Software Engineering Institute
SLC	Software Life Cycle
SLCM	Software Life Cycle Model
SLCP	Software Life Cycle Process