INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

"MODELO DE DESARROLLO DE SOFTWARE BASADO EN INGENIERÍA DE DOMINIO"

TESIS

QUE PARA OBTENER EL GRADO DE

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

M. EN C. RAMÓN MARÍN SOLÍS

DIRECTORES:
Dr. AGUSTÍN F. GUTIÉRREZ TORNÉS.
Dr. SERGIO SUÁREZ GUERRA.

México, D.F. **Enero del 2009.**

INSTITUTO POLITÉCNICO NACIONAL

SIP-14

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D. F. siendo las 18:00 horas del día 5 del mes de Diciembre de 2008 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

"MODELO DE DESARROLLO DE SOFTWARE BASADO EN INGENIERÍA DE DOMINIO"

Presentada por el alumno:

MARÍN	**SOLÍS**	**RAMÓN**
Apellido paterno	materno	nombre(s)

Con registro:

B	0	4	1	2	5	7

aspirante al grado de: ***DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN***

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente

Dr. Alexandre Felixovich Guelboukh Kahn

Primer vocal
(Director de Tesis)

Secretario

Dr. Grigori Sidorov

Segundo vocal
(Director de Tesis)

Dr. Sergio Suárez Guerra
Tercer vocal

Dr. Agustín Francisco Gutiérrez Tornés
Suplente

Dr. Napoleón Conde Gaxiola

Dr. Cornelio Yáñez Márquez

EL PRESIDENTE DEL COLEGIO

Dr. Jaime Álvarez Gallegos

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

INSTITUTO POLITECNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESION DE DERECHOS

En la Ciudad de **México D.F.** el día **10** del mes **diciembre** del año **2008**, el que suscribe **M. en C. Ramón Marín Solís**, alumno del Programa de **Doctorado en Ciencias de la Computación** con número de registro **B041257**, adscrito a **Centro de Investigación en Computación**, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del **Dr. Agustín F. Gutiérrez Tornés y** del **Dr. Sergio Suárez Guerra** y cede los derechos del trabajo intitulado **"MODELO DE DESARROLLO DE SOFTWARE BASADO EN INGENIERÍA DE DOMINIO"**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección **rmarin@computer.org** .Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Ramón Marín Solís

RESUMEN

"MODELO DE DESARROLLO DE SOFTWARE BASADO EN INGENIERÍA DE DOMINIO"

La problemática a la que se enfrentan las personas involucradas en el desarrollo de sistemas de información y software ha rebasado los enfoques tradicionales. Los modelos diseñados para proporcionar productos y servicios de calidad a las organizaciones son inviables la mayoría de las veces. La Ingeniería de Software se ha constituido como una disciplina fundamental en este ámbito, pero el paradigma que ha desarrollado se encuentra rebasado por la complejidad, el caos y la crisis. Han surgido una gran variedad de métodos y metodologías, las cuales prometen manejar la complejidad en la que las organizaciones e individuos están inmersos, pero esta cantidad de enfoques disímbolos han incrementado la confusión entre clientes y desarrolladores. El presente trabajo muestra los elementos clave para el rediseño del paradigma actual de la Ingeniería de Software. Se propone un paradigma sistémico, en el que concurren la Ingeniería de Sistemas, de Software y la Hermenéutica Analógica.

Se instrumenta un modelo heurístico, interpretativo y contextual aplicado al proceso de desarrollo de software. Se inicia formulando la problemática a la que se enfrenta el área de desarrollo de software, se revisa el paradigma sistemático, el cual se emplea en el área y se propone un paradigma sistémico basado en la Ingeniería de Dominio.

El modelo propuesto es holístico (contextual), heurístico (innovador), crítico (relevante) y ético. Coadyuva en la solución de la problemática a la que se enfrentan los ingenieros de software, arquitectos, analistas y diseñadores de sistemas. Este modelo toma en cuenta a todos los interesados del proceso de desarrollo de software, respetando su tradición, su cultura y sus valores. Aplica la hermenéutica como recurso de indagación y comunicación entre los distintos entes involucrados en el proceso de desarrollo de software y con esto, se proporcionan herramientas que facilitan la comprensión de los problemas a resolver; se contextualizan en un escenario relevante y se plasman en un sistema de información. Se brindan además, herramientas de comunicación entre todos los involucrados.

ABSTRACT

"DOMAIN ENGINEERING-BASED SOFTWARE DEVELOPMENT MODEL"

The problematic faced by people involved in information systems and software development has overcame the traditional approaches. The designed models are, most of the time, unviable. Software Engineering has become a fundamental discipline, but the developed paradigm has been exceeded by complexity, chaos and crisis. Numerous methods and methodologies have emerged, promising to manage the complexity of organizations and individuals. But this wide amount of different approaches has increased the confusion among clients and developers. The present Thesis shows the key elements in order to redesign the current paradigm of software engineering. We propose a systemic paradigm, composed by systems and software engineering and analogical hermeneutics.

We instrument a heuristic, interpretative and contextual model, applied in software development. We started diagnosing the problematic of the software development area and reviewing the systematic paradigm applied on software development. We propose a domain engineering-based paradigm.

The proposed model is holistic (contextual), heuristic (innovative), critic (relevant) and ethical. It helps to solve the problematic faced by software engineers, architects, analysts and systems designers. This model involves all stakeholders of a system, respecting their tradition, culture and values. It applies hermeneutics as an inquiry and communication tool among all stakeholders involved in the software development process. In addition, we provide tools that help us to understand the problems to solve and contextualize them in a relevant scenario in order to put it in an information system. Besides, we provide tools for communication among them.

"… Las intelecciones más valiosas son las que más tarde se encuentran; pero las intelecciones más valiosas son los métodos…"

Friederich Nietzsche

AGRADECIMIENTOS

Muchas son las personas que han contribuido al presente trabajo, directa o indirectamente.

Agradezco en primera instancia al **Dr. Agustín Francisco Gutiérrez Tornés**, mi Director de Tesis. Por compartir sus conocimientos e inquietudes; por su gentileza, su apoyo y su paciencia. Gracias por dirigir este trabajo y orientarme siempre.

Al **Dr. Sergio Suárez Guerra**, mi Director de Tesis en el CIC. Por todo su apoyo incondicional, sus observaciones siempre certeras y por su trato fino y atento.

Al **Dr. Miguel Ángel Mora Espinosa †**

A los miembros de mi Comité Doctoral al **Dr. Grigori Sidorov, Dr. Napoleón Conde Gaxiola, Dr. Cornelio Yáñez Márquez y Dr. Alexandre Felixovich Guelboukh Kahn**. Por sus revisiones rigurosas al trabajo y las observaciones que permitieron cohesionar esta Tesis.

Al **Dr. Mauricio Beuchot Puente** del Instituto de Investigaciones Filológicas de la UNAM. Por todo su apoyo en estos años y por recibirme en sus seminarios de Filosofía, Ética y Hermenéutica

A mi amigo el **M. en C. David Flores Vasconcelos.**

A mi estimada amiga la **Mtra. Martha Diana Bosco Hernández** de la Facultad de Filosofía y Letras de la UNAM.

A la **Dra. Frida Díaz Barriga Arceo** del Posgrado Facultad de Psicología, UNAM.

A mi **esposa Tere**, quien sufrió junto conmigo todas las desveladas y presiones, gracias por tu amor y comprensión. A su familia, que ha pasado a ser la mía también: a su a su Mamá **Teresa**, a su Papá **Jesús**, y a sus hermanos: **Jesús**, **Alejandro** y **Memo**. Les agradezco el cariño que me han brindado.

A mi **Mamá Micaela Solís Saldaña**, gran parte de lo que soy es gracias a ella. A mis hermanos **Israel**, **Noé**, **Gloria** y **Gerardo**. Gracias por todo su apoyo y cariño.

A mis amigos de toda la vida, la **Familia Gómez Martínez**, así como a la nueva familia de **Eva** y **Alejandro**. Gracias por su amistad sincera a lo largo de los años.

A la **Familia Zetina Hernández**. Gracias **Paz**, por tus consejos y tu presencia siempre amable. **Román,** te agradezco enormemente toda tu amistad, apoyo y solidaridad.

Al **Ing. José Antonio Benítez y a Paty.**

Al **personal de la UTE**, por apoyarme en todos los trámites y gestiones.

Al **CONACyT** por la beca para estudios doctorales No. 192497

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

GLOSARIO DE TÉRMINOS Y ABREVIATURAS

ACM - Association for Computing Machinery (Asociación para maquinaria de cómputo). Fundada en 1947, la *ACM* es la mayor fuerza dedicada al avance en las habilidades de los profesionales y estudiantes del campo de las Tecnologías de la Información a nivel mundial. Actualmente cuenta con 78,000 miembros, y es reconocida por su portal, líder en literatura de computación, publicaciones arbitradas y en conferencias innovadoras, lo cual le dan un liderazgo en este siglo 21. (ACM 2004)

Cliente/Servidor, Cómputo Cliente/Servidor (Client/Server computing).
(1).Paradigma o modelo de la interacción entre procesos ejecutándose concurrentemente.
(2). Una forma de Procesamiento Distribuido, en la cual un Cliente solicita servicios y un Servidor los proporciona. El Cliente y Servidor son transparentes entre ellos sin importar la localización o plataforma.

Economía Digital. En la economía Norteamericana se habla de *Economía Digital*. En el caso de la *Sociedad de la Información* (concepto adoptado por la Unión Europea), la clave reside en el papel que las tecnologías convergentes han de jugar para conformar una sociedad europea más cohesionada, mientras que el punto de vista norteamericano enfatiza su aporte a la productividad y los cambios que introducen en el funcionamiento y eficiencia de los mercados.

IEEE. (Institute of Electrical and Electronics Engineers)
*Instituto de Ingenieros Electricistas y Electrónicos***.** Es una sociedad técnico-profesional internacional que sirve al interés público y está formada por miembros de las ramas eléctricas, electrónica, computación, informática y otras tecnologías. El IEEE tiene Comités técnicos (TCOS) que formulan estándares de cómputo y comunicaciones, y es el responsable de muchos de los principales estándares de cómputo y comunicaciones en la industria (IEEE 2005b). Cuenta con más de 365,000 miembros en cerca de 150 países, alrededor del 40% fuera de EUA, y con cerca de 68,000 miembros estudiantiles (IEEE 2005a)

IEEE-CS. Sociedad de Computación de la IEEE es la organización de profesionales de la computación más grande del mundo, cuenta con alrededor de 100,000 miembros. Fue fundada en 1946, y es la sociedad más grande dentro de la IEEE, de las 37 con las que cuenta el Instituto de Ingenieros Electricistas y Electrónicos (IEEE). (IEEE-CS 2004)

INCOSE -International Council on Systems Engineering. (Consejo internacional en Ingeniería de Sistemas). Es una organización con más de 4,800 miembros de ingenieros en sistemas y otros interesados en la Ingeniería de Sistemas. Su propósito es fomentar la definición, comprensión, y práctica de una Ingeniería de Sistemas de clase mundial en la industria, el gobierno y la academia. INCOSE está compuesto de capítulos localizados en varias ciudades alrededor del mundo. (INCOSE 2004)

ISO, the International Organization for Standardization. Organización internacional para la estandarización. Es la más grande desarrolladora y editora de estándares internacionales.

ISO/IEC the International Organization for Standardization / IEC, the International Electrotechnical Commission. Organización internacional para la estandarización / Comisión Electrotécnica Internacional.

OCDE. *Organización para la Cooperación y el Desarrollo Económicos*. Organismo internacional que promueve políticas diseñadas para: 1) Lograr el más alto crecimiento económico sustentable, empleo y aumentar la calidad de vida en los países Miembros, manteniendo estabilidad financiera, y de esa manera, contribuir al desarrollo de la economía mundial. 2) Contribuir a una expansión económica firme en los países Miembros y no miembros en el proceso de desarrollo económico; y 3) Contribuir a la expansión del comercio mundial de forma multilateral y no discriminatoria de acuerdo a las regulaciones internacionales. México forma parte desde el 18 de mayo de 1994.

OECD. Organization for Economic Co-operation and Development. Ver *OCDE*.

Proceso de Negocios (Business Process). El manejo de una actividad como una entidad que tiene una duración limitada, con un alcance, regida por metas y políticas de negocios.

Protocolo. Un conjunto formal de convenciones que rigen el formato y control de datos. Un conjunto de procedimientos o reglas para establecer y controlar transmisiones desde un proceso o dispositivo fuente a un proceso o dispositivo destino.

Requerimientos. En el desarrollo de sistemas, los objetivos principales que el sistema o conjunto de sistemas deben cumplir.

Sociedad de la Información. Es el estado en que se encuentran las sociedades en que se implanta y generaliza el uso de las Tecnologías de la Información y las Comunicaciones en los distintos ámbitos de la vida de los ciudadanos, de las empresas y las instituciones. A todos ellos se les permite acceder a la información y productos que se encuentran en formato electrónico sin limitaciones de tiempo y espacio.

Stakeholder. Involucrado. Parte implicada, parte afectada, parte interesada. En un sistema, proyecto, empresa, o proceso.

TI o TIC. *Tecnologías de la Información y Comunicaciones*. Se le llama así al sector de la economía, que proporcionan manufactura de equipos o servicios, para las funciones de procesamiento de información y comunicaciones que incluyen la transmisión y la visualización por medios electrónicos. Este término es usado comúnmente por la *OCDE* y se basa en la División: 72 – "Computer and related activities" de la ISIC - International Standard Industrial Classification, la que incluye: Consultoría de Hardware; Consultoría y venta de Software; Procesamiento de datos; Actividades de bases de datos; Mantenimiento y reparación de equipos de oficina, contables y de cómputo; Otras actividades relacionadas con cómputo.

Capítulo 1. Introducción

1.1 Problemática

El área de desarrollo de software se encuentra en un escenario de crisis. Al menos un 65% de los proyectos que se implementan, fracasan o no cubren con las expectativas de los usuarios (Ambler 2004). Existe una búsqueda continua de opciones novedosas para atacar la problemática del desarrollo de software, la cual no ha brindado todos los frutos esperados.

La Ingeniería de Software es una disciplina emergente del campo de las Ciencias de la Computación y ha pretendido dar solución a este escenario de crisis. Sus orígenes se remontan a la primera conferencia de Ingeniería de Software de la OTAN en 1968 -*NATO* Software Engineering Conference *1968* (Naur and Randell 1969).

Una de las causas de esta problemática es la falla en la comunicación cliente-vendedor la cual provoca la mayoría de los problemas en el proceso de desarrollo de software. Las empresas por su parte tratan de justificar sus fallas culpando a los cambios tecnológicos o a una pobre especificación de requerimientos. Los usuarios demandan cada día más una atención justa por parte de los desarrolladores de software. Quieren que sus proyectos se entreguen a tiempo, con la funcionalidad requerida y en el precio acordado. De igual forma las empresas de Tecnologías de la Información (TI) quieren que los proyectos que desarrollan cubran las funciones que el cliente les pidió, quieren entregarlos a tiempo o antes de tiempo y que les generen una utilidad razonable (Donaldson y Siegel 2001).

Las empresas del ramo de TI desarrollan sus proyectos de forma improvisada, tomando decisiones en el corto plazo. Esto funciona si el proyecto es pequeño, pero conforme el proyecto es más grande la dificultad y los errores crecen, provocando que estos proyectos se desarrollen de forma caótica (Fowler 2003).

México no es la excepción, las empresas del ramo no cuentan con procedimientos ni procesos para la administración (análisis, implementación y control) de proyectos (Prosoft 2004).

En el desarrollo de software son muchos los que piensan que el software es un fin por sí mismo, y no comprenden que sólo es un medio para atender problemáticas. Esto produce que el enfoque del área sea completamente mercantilista. El desestimar los elementos culturales, políticos y sociales, inherentes al trato humano, ha propiciado que gran parte de los proyectos de desarrollo de software no cubran con las expectativas que generan, ni con el presupuesto y el tiempo asignado.

Vivimos en una época de crisis, y ésta se enmarca en distintos ámbitos: el económico, el político y el social. La dependencia que tenemos, tanto de capital (económica), como intelectual (ciencia y tecnología) provoca un atraso significativo en nuestro desarrollo. Para superar esta crisis hay que transformar nuestras organizaciones. El presente trabajo propone una forma que coadyuva a lograrlo.

1.2 Planteamiento del Problema

Si la Ingeniería de Software comprende "todos los aspectos de la producción de software desde las primeras etapas de la especificación del sistema, hasta el mantenimiento de éste, después de que se utiliza" (Sommerville 2005). Y Boehm (2006) por su parte señala que, al ser la Ingeniería de Software la aplicación de la ciencia y las matemáticas para construir software útil para las personas. La frase útil para las personas, implica que las ciencias relevantes incluyen a las ciencias del comportamiento, las ciencias de la administración, la economía, además de las Ciencias de la Computación.

Y si diversos estudios (Ambler 2004; Charette 2005; El Emam y Koru 2008) señalan que la Ingeniería de Software aún no logra cumplir la promesa de producir software de alta calidad. Se supone que el enfoque rígido y organizado que adoptaron para lograrlo ha resultado ser inefectivo. Ya que el desarrollo de software todavía es una actividad caótica, lo cual es una clara evidencia de que este paradigma ha quedado rebasado para solucionar problemas complejos en un entorno cambiante y en escenarios de crisis.

Nuestro problema queda definido de la siguiente forma:

Las teorías de la Ciencias de la Computación y las metodologías tradicionales de desarrollo de software no siempre pueden aplicarse a problemas prácticos, reales y complejos. Éstas no abarcan todos los aspectos de la producción de software incluyendo la especificación del sistema. Aquí

subyace el factor fundamental: los usuarios no pueden definir con claridad lo que realmente quieren, y aquellos que pueden definir lo que quieren, resulta que esto no es lo que necesitan.

La relación entre usuarios y desarrolladores se puede tipificar de la siguiente forma:

a) **Problemas de Capacidad.**
   i. Los usuarios no pueden definir con claridad lo que realmente quieren,
   ii. Aquellos que pueden, resulta ser que no es lo que necesitan.

b) **Problemas ético-legales.**
   i. Los usuarios no quieren definir lo que creen que necesitan.
   ii. Los usuarios ocultan lo que creen que necesitan

Y el problema se da también en sentido inverso, con desarrolladores que pueden desconocer cuál es la solución viable y por cuestiones de ética, al conocer las implicaciones de la solución, no querer brindársela al cliente.

1.3 Justificación

La Ingeniería de Software en conjunción con la Ingeniería de Sistemas constituirá una de las disciplinas principales del Siglo XXI brindándoles a las organizaciones el soporte para proveer productos y servicios (Boehm 2006). El trabajo conjunto de la Ingeniería de Software y de Sistemas ya es un hecho que se ve reflejado en los estándares publicados por la ISO/IEC: Systems and software engineering, Software life cycle processes (12207:2008) y System life cycle processes (15288:2008). Resultado del trabajo del IEEE (Institute of Electrical and Electronics Engineers- Instituto de Ingenieros Electricistas y Electrónicos) con otros organismos como el INCOSE (International Council on Systems Engineering – Consejo Internacional de Ingeniería de Sistemas).

Existen varias iniciativas y grupos de trabajo en el campo de las Tecnologías de la Información (TI's) que comparten nuestra preocupación por una visión más integral de lo que son las Ciencias de la Computación y la Ingeniería de Software para el desarrollo de sistemas y software útil, por ejemplo:

1. El Comité de Estándares de Ingeniería de Software y Sistemas (S2ESC- IEEE Systems and Software Engineering Standards Committee) incluye la Ingeniería de Sistemas cuando se usa software intensivamente (IEEE 2004). Forman parte de este Comité además, el INCOSE y el DHS (Departament of US Homeland Security- Departamento de seguridad interna de los

EU) y el SEI (Software Engineering Institute- Instituto de Ingeniería de Software) de la Universidad Carnegy Melon (CMU) (IEEE S2ESC 2008)

2. El grupo de trabajo JTC 1/SC7 de estandarización de Ingeniería de Software y de sistemas de la ISO/IEC. Cuyo mandato es: la estandarización de procesos, herramientas de soporte y tecnologías para la ingeniería de productos de software y sistemas (ISO/IEC JTC1/SC7 2008)
3. El SEI (Software Engineering Institute- Instituto de Ingeniería de Software) de la Universidad Carnegy Melon (CMU- Carnegy Melon University) es un Centro de investigación y desarrollo financiado por el gobierno federal de los Estados Unidos de América. El SEI desarrolla trabajos en las áreas de: Ingeniería de Software, seguridad en cómputo y mejora de procesos (SEI 2008).
4. La iniciativa de la Agile Aliance (2008). Publicaron el manifiesto ágil para desarrollo de software y practican los siguientes valores: individuos e interacciones sobre procesos y herramientas; software útil en lugar de documentación detallada; colaboración con el cliente en lugar de negociación de contrato y responder al cambio en lugar de seguir un plan (Manifesto for Agile Software Development 2001).
5. La iniciativa "Services Science, Management, and Engineering" –SSME (ciencia, administración e ingeniería de servicios) de IBM. La cual se enfoca en el apoyo a los servicios. Ya que estos representan en los países industrializados casi el 80% de sus economías (Chesbrough y Spohrer 2006). Por tal motivo, IBM ha lanzado un proyecto de investigación tendiente a establecer la disciplina SSME y señalan que el futuro de los servicios depende del soporte de las Ciencias de la Computación, y del área de Tecnologías de la Información en general, las cuales les proporcionarán el procesamiento, almacenaje y transferencia de información, así como de herramientas que les permitirán comunicarse y construir relaciones con sus clientes (Roland y Miu 2006). Indican además, que si los científicos de la computación trabajan con modelos formales de algoritmos y cómputo, algún día, los científicos de servicios, podrán trabajar con modelos formales de sistemas de servicios (Maglio, Srinivasan, Kreulen, y Spohrer 2006)
6. El proyecto Competisoft en América Latina. Que proporciona a la industria de software de Latinoamérica un marco de referencia para la mejora y la certificación de sus procesos de software(Oktaba et al. 2007)

7. El Programa para el Desarrollo de la Industria del Software -Prosoft (2004) en México. Que tiene como meta el desarrollar una industria del software competitiva internacionalmente y asegurar su crecimiento en el largo plazo. Y esperan situar a México como líder de esta industria en Latinoamérica para el 2013 y convertirlo en líder desarrollador de soluciones de TI de alta calidad y uso de software en Latinoamérica.

Las crisis contables, administrativas y de credibilidad en varias empresas de los Estados Unidos de América, han hecho patente la necesidad de prestar atención a los aspectos legales y éticos en el campo de las Tecnologías de la Información y Comunicaciones (Chesbrough y Spohrer 2006). Por tal motivo, se rescatan y aplican en esta propuesta, los aspectos éticos en la formación y en la práctica de los profesionales del área. Para ello se usan los códigos de ética y práctica profesional de las dos principales organizaciones del campo de la computación, la ACM y la Sociedad de Computación de la IEEE (IEEE-CS y ACM 1999).

Nuestro campo de aplicación es amplio, ya que actualmente las empresas de TI se enfocan exclusivamente a modelar la estructura de las organizaciones, olvidándose del componente cultural inherente al trato humano, lo cual ha conducido a que muchos de los proyectos fracasen. Y no contemplan las expectativas y necesidades de todos los involucrados, concentrándose más bien, en el cumplimiento conforme a tiempo y presupuesto asignados.

1.4 Marco Conceptual de la Tesis

En el presente trabajo se hace una puntualización acerca de la problemática del proceso de desarrollo de software desde dos dimensiones, la racional y la ética. La primera urdida en torno al concepto de método y calidad, o mejor, métodos para la calidad, y la segunda en términos de valores, creencias, hábitos y operaciones. Este enfoque innovador conduce a una situación compleja, que requerirá de una investigación, con cuatro vertientes principales:

1. Las Ciencias de la Computación y la Ingeniería de Software, enfocadas a desarrollar productos de calidad;
2. La Ingeniería de Sistemas, enfocada en el diseño y su implementación;
3. La Heurística;
4. La Ética y la Hermenéutica.

Al caracterizar la problemática de proceso de desarrollo de software como un sistema, se puede establecer que concurren en él disciplinas directas (Ciencias de la Computación y la Ingeniería de Software), afines (la Ingeniería de Sistemas y la Heurística) y las marginales (la Hermenéutica, la Ética), las cuales brindan posibilidades de resolver tal problemática (ver Fig. 1.1).

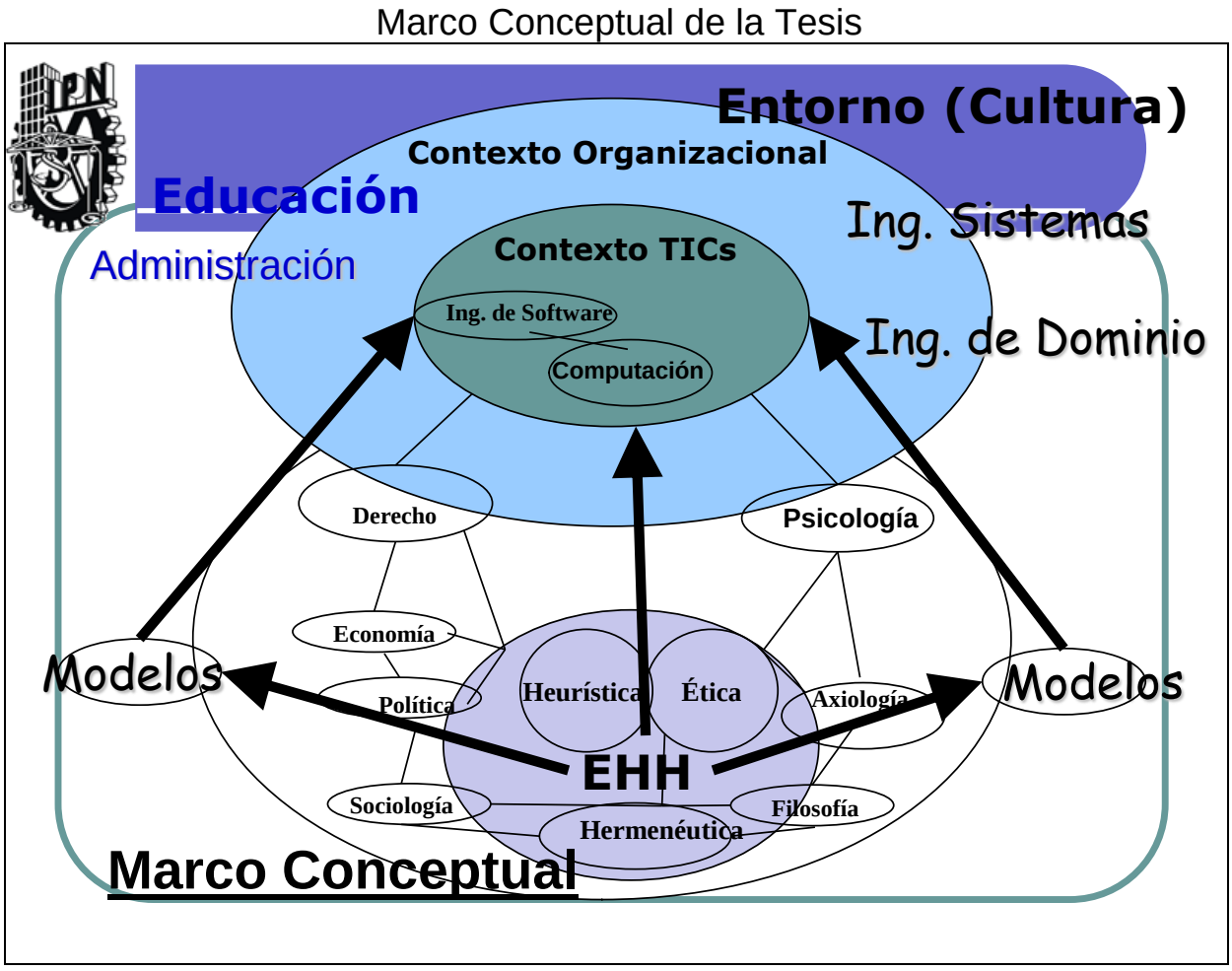

Figura 1.1 Marco Conceptual de la Tesis

Fuente: Elaboración Propia

1.5 Marco Metodológico

Este trabajo es transdisciplinar *(*Ackoff 1999a) y heurístico. Entendiendo con esto que se sabe lo que se quiere obtener, se reconoce que no se cuenta con un método completo y explícito para lograrlo. Por lo que se aplican disciplinas cuyo objeto de estudio, estado del arte y método formal de investigación son ajenas a nuestro campo, lo cual permitirá, presentar una nueva visión, una definición y una concepción novedosas de lo que son las Ciencias de la Computación y la Ingeniería de Software, mediante su articulación con la Ética, la Hermenéutica Analógica y la Heurística.

El trabajo de investigación surge como una investigación exploratoria de las disciplinas directas, afines y marginales. Y a partir de éstas se construye un modelo.

1.6 Preguntas de Investigación

Estas fueron las preguntas que originaron la presente investigación:

¿El paradigma sistemático aplicado al desarrollo de software es el más adecuado para un entorno complejo y en constante cambio, con participantes con intereses diferentes?

¿Las fallas en la comunicación de los distintos participantes involucrados en un sistema, inciden en el proceso de desarrollo de software?

¿Los aspectos ético-culturales inciden en el proceso de desarrollo de software?

¿Es posible modelarlos e incorporarlos en el proceso de desarrollo de software?

¿Se puede implementar una herramienta que modele los aspectos ético-culturales y que mejore la comunicación de los participantes del sistema?

1.7 Hipótesis

La hipótesis de investigación para el proyecto es:

Al usar un modelo heurístico, interpretativo y contextual se mejora el proceso de desarrollo de software y los productos resultantes inciden en la sustentabilidad de las organizaciones.

Se definirán y evaluarán los aspectos heurísticos, interpretativos (hermenéuticos) y contextuales (holísticos) y su relación en el proceso de desarrollo de software. Nuestra hipótesis de trabajo es que estos aspectos, generen un proceso de desarrollo adaptativo, y que los productos de este proceso (el software) incida en la sustentabilidad de las organizaciones.

1.8 Objetivos

Los objetivos de la investigación son:

- **General:** Diseñar e instrumentar un modelo heurístico, interpretativo y contextual aplicado al proceso de desarrollo de software.
- **Específicos**:
    - Formular la problemática a la que se enfrenta el área de desarrollo de software.
    - Proponer un modelo que coadyuve a solucionar tal problemática, tomando en cuenta a todos los interesados, respetando su tradición, su cultura y sus valores.
    - Aplicar la Hermenéutica como recurso de indagación y comunicación entre los distintos entes involucrados en el proceso de desarrollo de software.

1.9 Aportaciones

El presente trabajo propone un modelo sistémico, para lo cual tendrá que cubrir tres imperativos: i) totalidad, ii) innovación y iii) mérito, que planteados en forma conceptual, corresponden a las nociones de holística (totalidad), heurística (innovación) y crítica (mérito); con esto se busca propiciar la transformación de las organizaciones, mostrándoles una ruta para que aprovechen todos los recursos con que cuentan o los que necesitan, poniéndose con ello en camino de operar y desarrollarse de forma sustentable.

El diseño del modelo parte de las arquitecturas de cómputo empresarial y se transforma en una Arquitectura Sistémica, y posteriormente integra a la Ingeniería de Dominio.

Nuestras aportaciones son:

1. Un modelo heurístico, interpretativo y contextual aplicado en el proceso de desarrollo de software.

2. Una herramienta que facilita la comunicación entre los distintos involucrados en el proceso de desarrollo de software.
3. Una aplicación del modelo interpretativo (hermenéutico) en el campo de las Ciencias de la Computación.
4. Un mecanismo de evaluación de los aspectos heurísticos, interpretativos, contextuales y éticos propuestos.

La investigación para construir el modelo de desarrollo de software ha sido progresiva, y los avances de la misma se han presentado en varios congresos nacionales e internacionales.

1.10 Esquema de la Tesis

La tesis se encuentra estructurada de la siguiente forma:

En el presente Capítulo se hace el planteamiento de la investigación.

En el Capítulo 2, Estado del Arte, se hace una revisión de la problemática del desarrollo de software, así como de sus metodologías y modelos. Señala que para afrontar esta problemática se están conjuntando las Ingenierías de Software y de Sistemas. Se muestran algunos acercamientos de ambas disciplinas así como la aplicación del enfoque hermenéutico en los Sistemas de Información. Se hace una revisión de la ética y los aspectos legales en la Ingeniería de Software, así como de las distintas nociones de Arquitectura. Se hace una revisión de las Metáforas de la Organización, y por último se presentan la Ingeniería de Dominio y la Ingeniería de Software Basada en Evidencias. Además de ser una revisión del estado del arte, el Capítulo 2 sirve como Marco Teórico, pues aquí se presentan las teorías, definiciones y enfoques que serán empleados a lo largo del trabajo.

En el Capítulo 3, Modelo Propuesto, se desarrolla el modelo, el cual es heurístico, interpretativo y contextual. Y se aplica en el proceso de desarrollo de software basado en Ingeniería de Dominio.

En el Capítulo 4, Resultados, se muestran los resultados de la evaluación del modelo. Se propone una rúbrica como método de evaluación de modelos de Ingeniería de Software. Se aplica al modelo propuesto y a otro modelo de desarrollo de software y se comparan sus resultados.

En el Capítulo 5, Conclusiones y Trabajos Futuros, se revisan los resultados de este trabajo de investigación.

Capítulo 2. Estado del Arte

Autor: Dirk Hagner
IEEE-CS Software Vol 19 No.5

El Capítulo se inicia con una revisión de la problemática del desarrollo de software, así como de sus metodologías y modelos. Señala que para afrontar esta problemática se están conjuntando las Ingenierías de Software y de Sistemas. Se muestran algunos acercamientos de ambas disciplinas así como la aplicación del enfoque hermenéutico en los Sistemas de Información. Se hace una revisión de la Ética y los aspectos legales en la Ingeniería de Software, así como de las distintas nociones de Arquitectura. Se hace una revisión de las Metáforas de la Organización, y por último se presentan la Ingeniería de Dominio y la Ingeniería de Software Basada en Evidencias.

Además de ser una revisión del estado del arte, el Capítulo 2 sirve como Marco Teórico, pues aquí se presentan las teorías, definiciones y enfoques que serán empleados a lo largo del trabajo.

CAPÍTULO 2. ESTADO DEL ARTE

Nada cierto podemos afirmar del mundo objetivo y del sujeto que lo mira,
salvo que uno y otro son haces de percepciones instantáneas e inconexas
ligadas por la memoria [recuerdos] y la imaginación [anhelos y sueños]
El mundo es imaginario, aunque no lo sean las percepciones en que,
alternativamente [momentos] se manifiesta y se disipa

La realidad no son sino <<descripciones del mundo>>
Octavio Paz

2.1 Problemática y Propuesta de Rediseño al Paradigma de la Ingeniería de Software

La problemática a la que se enfrentan las personas involucradas en el desarrollo de sistemas de información y software ha rebasado los enfoques que se siguen para proporcionar productos y servicios de calidad a las organizaciones. La Ingeniería de Software se ha constituido como una disciplina fundamental en este ámbito, pero el paradigma que ha desarrollado se encuentra rebasado por la complejidad, el caos y la crisis. Han surgido una gran variedad de métodos y metodologías, las cuales prometen manejar la complejidad en la que las organizaciones e individuos están inmersos. Pero esta cantidad de enfoques disímbolos han incrementado la confusión entre clientes y desarrolladores. El presente trabajo muestra los elementos clave pare el rediseño del paradigma actual de la Ingeniería de Software, en el que se propone un paradigma sistémico, en el que concurren la Ingeniería de Sistemas, de Software y la Hermenéutica Analógica.

2.1.1 Convergencia entre la Ingeniería de Software y la Ingeniería de Sistemas

La convergencia entre la Ingeniería de Software y la Ingeniería de Sistemas más evidente cada día. Por ejemplo, ha llevado al Comité de estándares de Ingeniería de Software de la IEEE (SESC- Software Engineering Standards Committee) a ampliar sus objetivos y cambiar su nombre al "Comité de estándares de Ingeniería de Software y Sistemas" (S2ESC- IEEE Systems and Software Engineering Standards Committee) para incluir la Ingeniería de Sistemas cuando se usa software intensivamente (IEEE 2004). Forman parte de este Comité además, el INCOSE, el DHS (Departament of US Homeland Security- Departamento de seguridad interna de los EU) y el SEI

(Software Engineering Institute- Instituto de Ingeniería de Software) de la Universidad Carnegy Melon (CMU- Carnegy Melon University) (IEEE S2ESC 2008).

La misión del S2ESC (IEEE 2004b) es:

-Desarrollar y mantener una familia de estándares de Ingeniería de Software y sistemas, que sea relevante, coherente, comprensible y efectiva en su uso. Esos estándares serán para el uso de profesionales, organizaciones y educadores y tienen como finalidad: mejorar la efectividad y la eficiencia de sus procesos de Ingeniería de Software, mejorar las comunicaciones entre los clientes y proveedores, y mejorar la calidad del software y los sistemas que contienen software.

-Desarrollar conocimiento que ayude a los profesionales, las organizaciones y los educadores en la comprensión y aplicación de estos estándares.

-Soportar y promover: un cuerpo de conocimiento (Body of Knowledge) de Ingeniería de Software (IEEE-CS 2004), así como, mecanismos de certificación para los profesionales de la Ingeniería de Software.

Destaca dentro de la misión, el integrar la Ingeniería de Software y de sistemas, así como, mejorar la comunicación entre los clientes y proveedores. El presente trabajo propone la integración de ambas disciplinas bajo un enfoque sistémico y propone además, el uso de la Hermenéutica Analógica para establecer un modelo de comunicación entre los distintos participantes en el proceso de desarrollo de software.

2.1.2 El Paradigma de la Ingeniería de Software

La Ingeniería de Software es:"La aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software, esto es, la aplicación de la ingeniería al software; así como el estudio de esos enfoques" (IEEE 1990)

Según Sommerville (2005) es: "una disciplina que comprende todos los aspectos de la producción de software desde las etapas iníciales de la especificación del sistema, hasta el mantenimiento de éste, después de que se utiliza".

Cuando propone la forma de trabajo refiere que: "en general los ingenieros de software adoptan un enfoque sistemático y organizado en su trabajo, ya que es la forma más efectiva de producir software de alta calidad. Sin embargo, aunque la ingeniería consiste en seleccionar el método más apropiado para un conjunto de circunstancias, un enfoque informal y creativo de desarrollo podría ser [más] efectivo en algunas circunstancias. Y muestra un ejemplo de un caso: "El desarrollo informal es apropiado para el desarrollo de sistemas de comercio electrónico basados en Web que requieren una mezcla de habilidades de software y de diseño gráfico" (Sommerville 2005).

Al tratar de diferenciarla de otras disciplinas, señala que, la diferencia entre la Ingeniería de Software y Ciencias de la Computación es que esta última "se refiere a las teorías y métodos subyacentes a las computadoras y los sistemas de software, mientras que la Ingeniería de Software se refiere a los problemas prácticos de producir software [de alta calidad]… los ingenieros de software a menudo utilizan enfoques ad hoc para desarrollar el software. Las elegantes teorías de la Ciencia de la Computación no siempre pueden aplicarse a problemas reales y complejos que requieren [de] una solución de software". (Sommerville 2005)

Cuando intenta establecer la diferencia entre la Ingeniería de Software y la Ingeniería de Sistemas, confunde a esta última con una de sus áreas de estudio –los Sistemas de Información- dentro de la cual se incrusta la Ingeniería de Sistemas Basados en Computadoras, que "se refiere a todos los aspectos del desarrollo y de la evolución de sistemas complejos donde el software juega un papel principal. Por lo tanto la Ingeniería de Sistemas [Basados en Computadoras] comprende el desarrollo de hardware, políticas y procesos de diseño y distribución de sistemas, así como la Ingeniería de Software. Los ingenieros de sistemas están involucrados en la especificación del sistema, en la definición de su arquitectura y en la integración de las diferentes partes para crear el sistema final. Están menos relacionados con la ingeniería de los componentes del sistema (hardware, software, etc.). La Ingeniería de Sistemas es más antigua que la de software, por más de 100 años, las personas han especificado y construido sistemas industriales complejos, como trenes y plantas químicas. Sin embargo, puesto que se ha incrementado el porcentaje de software en los sistemas, las técnicas de la Ingeniería de Software se utilizan en el proceso de Ingeniería de Sistemas [Basados en Computadoras]" (Sommerville 2005).

2.1.3 La Problemática en la Ingeniería de Software

Como demuestran los hechos, el objetivo de producir software de alta calidad está lejos de cumplirse. Tepper (2002) cita a un reportero de tecnología que en 1997 escribió: 'el software se hace muy mal, a menudo está terriblemente mal hecho. Es desarrollado de una forma despreocupada e irresponsable, que sería inmoral si se aplicara a la construcción de puentes, automóviles y quizás hasta en trabajos de plomería'. Y señala que cinco años más tarde [en el 2002] la situación en lugar de mejorar ha empeorado. Dentro de las características que hacen que el desarrollo de software sea deficiente destaca: la creciente complejidad del mismo y que los desarrolladores de software no consideraren las necesidades del usuario promedio ya que para el usuario final el software es sólo un medio para desarrollar una tarea y no un fin en sí. El desdén por los usuarios llega hasta el punto de considerarlos estúpidos, perjudicándolos ya que obtienen software repleto de características que no usan y no tienen las que necesitan.

Por su parte Ambler (2004) es más tajante, y dice que "dependiendo de la fuente, la industria de Tecnologías de la Información tiene un rango de fracaso en proyectos de misión crítica del 65% al 85%. Y aún el panorama más alentador, del 65%, que publica el Standish Group en su reporte del *CHAOS*, es aterrador". A pesar de las críticas por la falta de transparencia en los reportes del Standish Group, de acuerdo a una encuesta realizada en el 2005 y el 2007, el porcentaje de fracasos aún ronda entre el 26% y el 34%. Y se señalan como causas los cambios en los requerimientos y el alcance del proyecto; así como sobrepasar el presupuesto, la falta de compromiso de la alta dirección y falta de habilidades administrativas (El Emam y Koru 2008).

En cuanto a la forma en la que se desarrolla el software, Fowler (2003), remarca que la mayoría del desarrollo de software es una actividad caótica, caracterizada a menudo por la frase 'codifica y arregla'. El software se escribe sin basarse en un plan y el diseño del sistema es improvisado. Se toman decisiones para el corto plazo, lo cual funciona si el sistema es pequeño, pero conforme el sistema crece, aumenta la dificultad para agregar nuevas características al mismo. Además los errores se incrementan y son más difíciles de arreglar.

Donaldson y Siegel (2001) apuntan, que la '"forma' en la que la industria de software desarrolla sus productos debe cambiarse, esta afirmación no es sólo una opinión, ya que es un punto de vista que comparten muchos en la industria. Las siguientes solicitudes y reclamos, son ejemplos que hacen eco de la necesidad de cambio:

- Entrega el software a tiempo y dentro del presupuesto
- Deja de hacer solicitudes de último minuto para nuevas características [del software]
- Haz que el software haga lo que pedí
- Ayúdame –define lo que realmente quieres
- Deja de darme soluciones a corto plazo para problemas de largo plazo
- Deja de aplazar la fecha de entrega
- Proporciónanos los recursos para una prueba adecuada
- Establece una arquitectura en la cual podamos trabajar
- Reduce la dependencia de los individuos en nuestra organización
- Desarrolla nuevos sistemas con costos más bajos
- Proporciona los recursos para implementar una nueva forma de hacer negocios

Los ejemplos anteriores suenan como partes de diálogos cliente-desarrollador o de diálogos dentro de una organización que se dedica al negocio del desarrollo de software.

Los clientes quieren que el software satisfaga ciertos criterios, quieren sistemas que:
1) hagan lo que se supone que tienen que hacer;
2) que se entreguen a tiempo y;
3) que se entreguen en el precio acordado.

De la misma forma, los vendedores (por ejemplo las compañías que desarrollan software) quieren que los sistemas que desarrollan:
1) hagan lo que los clientes quieren;
2) se entreguen antes de tiempo o a tiempo y;
3) generen una utilidad razonable.

El software 'bueno' (de calidad) es aquel que satisface el criterio del cliente y del vendedor, y que ambos quieren que sus criterios se satisfagan de forma consistente [confiable-constante]". El desarrollo exitoso del software significa la capacidad de producir software de calidad de forma confiable. E indican "que la industria de software ha demostrado claramente que la falla en la

comunicación cliente-vendedor provoca la mayoría de los problemas en el desarrollo de software" (Donaldson y Siegel 2001)

Los empresarios de la industria de desarrollo de software requieren de personal competente en el área, y dicen que "En estos días, no puedes pagar a empleados que no entiendan el negocio como un todo. En la compañía, estamos tratando de contratar desarrolladores que comprendan, que están tratando de solucionar problemas de negocios" (McLaughlin 2003)

Con base en lo anterior, se hace patente que el paradigma actual de la Ingeniería de Software, aplicado a problemas complejos, en un entorno de constante cambio, y con participantes en el proceso de desarrollo con intereses diversos, y más aún, de distintas culturas, ha quedado rebasado. Para rediseñarlo se propone, hacerlo bajo un enfoque sistémico.

2.1.4 Disciplinas que Concurren en el Desarrollo de Software

"Un paradigma surge en el marco de una comunidad, de una cultura" (Beuchot 2005a) por lo que para realizar el rediseño al paradigma de la Ingeniería de Software se requiere un acercamiento conceptual y del lenguaje entre las disciplinas que convergen en el desarrollo de software y de sistemas de información.

2.1.4.1 Ingeniería de Sistemas

Concierne a la Ingeniería de Sistemas el ofrecer diseños de sistemas aptos para resolver las tres faenas esenciales del quehacer gerencial: formular políticas, tomar decisiones y controlar operaciones (Beer 1972*)*. A su vez, el quehacer gerencial aludido tiene lugar en sistemas de una especie singular a los que denominaremos organizaciones. Éstas se conforman de dos componentes: su estructura y su cultura. Se considera que el diseño es una actividad humana primitiva y espontánea, que se desglosa en tres disciplinas: aprendizaje, liderazgo, previsión. Y están en concordancia con tres dimensiones o categorías preexistentes, esto es, anteriores e independientes del hombre: la complejidad, el caos, la crisis. (Mora 2004)

La Ingeniería de Sistemas en su carácter de disciplina científica presenta una cuádruple característica, consistente en ser heurística, hermenéutica, ética y teleológica, puesto que usa modelos en la forma de sistemas, incluye modos de interpretación, busca producir beneficios y se orienta al futuro. (Mora 2004)

La Ingeniería de Sistemas ha pasado por distintos enfoques, de acuerdo al contexto (complejidad) y a la naturaleza (unitaria, pluralista, coercitiva) de los participantes del sistema. (Jackson 2003)

2.1.4.2 El Enfoque Analítico: La Era de la Máquina

"El universo era una máquina que fue creada por Dios para realizar Su obra. Se esperaba que el hombre, como parte de esa máquina, cumpliera con los designios de Dios, que hiciera Su voluntad. De aquí se infirió que el hombre tenía que crear máquinas para que hicieran su trabajo. La revolución industrial fue un producto de esa inferencia. Los hombres en esa época, confrontaron la naturaleza con un temor reverente, y con la admiración y la curiosidad de un niño, e intentaron descifrar sus misterios analíticamente." (Ackoff 1999)

2.1.4.3 La Era de los Sistemas

"La Segunda Guerra Mundial, sacó a la ciencia y a los científicos de sus laboratorios y los metió en el 'mundo real' en un esfuerzo por resolver importantes problemas que surgían en organizaciones grandes y complejas –militares, gubernamentales y corporativas. Los científicos descubrieron que los problemas que enfrentaban no podían descomponerse en otros que encajaran exactamente en una disciplina particular y que la interacción de las soluciones particulares de las partes separadas era de mayor importancia que las soluciones consideradas por separado. Esto llevó a su vez a la creación de trabajos interdisciplinarios. A finales de los años treinta del siglo XX, surgió de la institución militar británica la 'Investigación de Operaciones', una actividad interdisciplinaria, dirigida a resolver los problemas en la administración y el control de sus complejas operaciones.

Para los años cincuenta proliferaban las actividades científicas interdisciplinarias. Éstas incluían las Ciencias de la Administración, las Ciencias de las Decisiones, las Ciencias de la Computación, las Ciencias de la Información, la Cibernética, las Ciencias Políticas y muchas otras. Los intereses compartidos y las similitudes en sus prácticas llevaron a la búsqueda de un tema común a todas ellas, y éste fue el comportamiento de los sistemas. "(Ackoff 1999)

2.1.5 La Hermenéutica Analógica y su Contribución en el Desarrollo de Software

La hermenéutica o el arte de interpretar textos, es una "teoría de la verdad y el método que expresa la universalización del fenómeno interpretativo desde la concreta y personal historicidad"[1].

[1] Diccionario de la Real Academia de la Lengua Española.

La hermenéutica tiene como fin poner un texto en un contexto relevante. Beuchot (2005) señala que un texto no es sólo un escrito, también son textos los diálogos, y cualquier acción significativa, incluido el silencio. En nuestro caso, el texto cobra varias formas, por ejemplo:

- Los sistemas como texto, los cuales se tienen que contextualizar en un escenario relevante para la organización.
- Los requerimientos de los usuarios como texto, los cuales requieren de una interpretación válida, consensuada y relevante.
- Los modelos y diseños.
- Los programas de cómputo.

Dentro de la hermenéutica hay varias tradiciones:

1. Hay una tradición univocista [analítica-positivista],
2. otra equivocista [subjetiva-relativista] y,
3. otra analógica.

En la hermenéutica unívoca sólo hay una interpretación válida, y ésta pretende juntar al autor, al lector y al texto en un contexto único. En la hermenéutica equívoca cualquier interpretación es válida, aunque éstas sean distintas, se contrapongan o estén fuera del contexto. La Hermenéutica Analógica pone límites a la interpretación equívoca y considera como ideal la interpretación unívoca, aunque reconoce que en la mayoría de los casos, ésta no se podrá alcanzar (ver Figura 2.1).

La interpretación unívoca es inalcanzable las más de las veces; la interpretación equívoca es inútil; sólo queda la interpretación analógica, aproximativa.

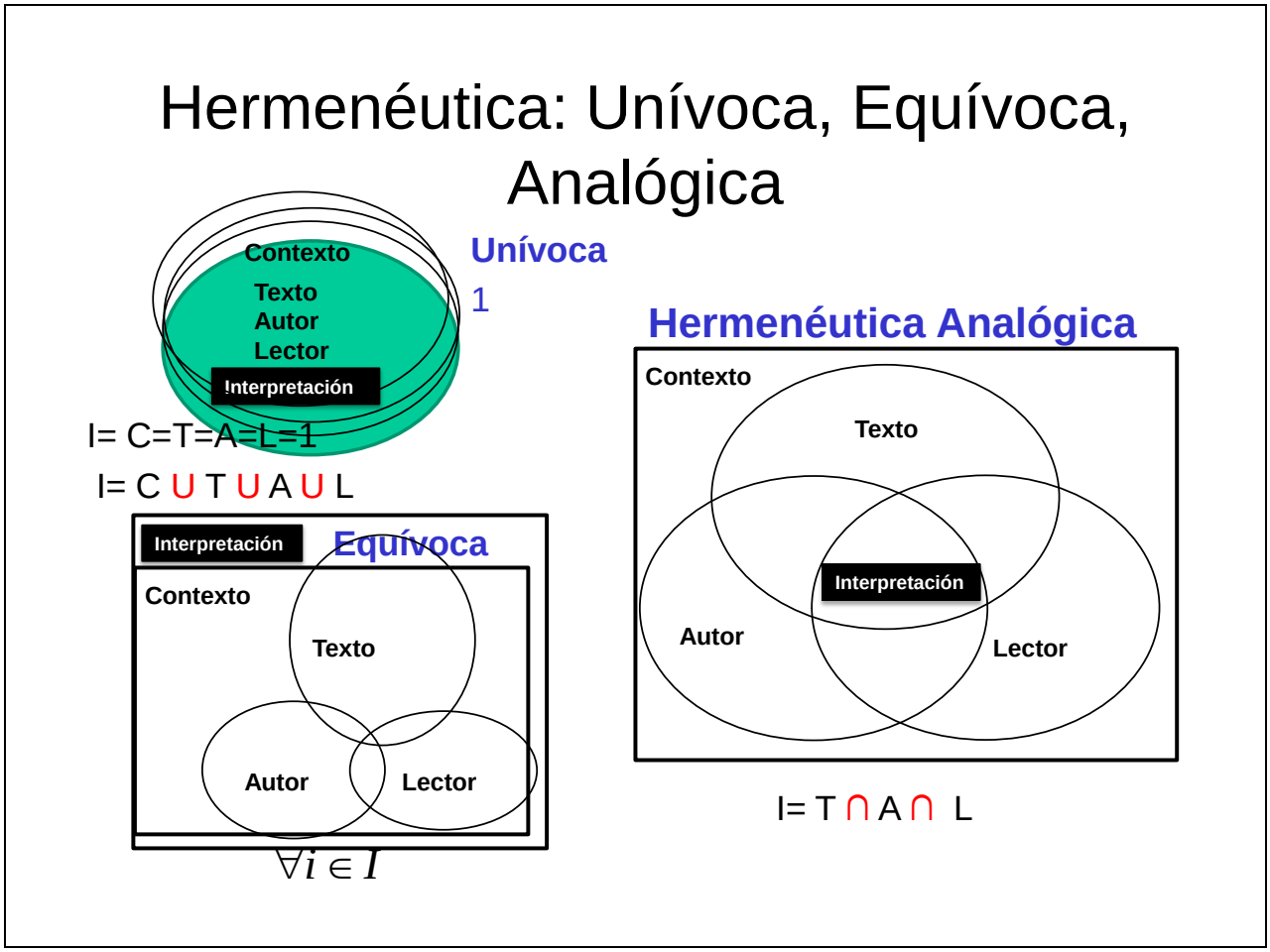

Figura 2.1 Hermenéutica: Unívoca, Equívoca y Analógica.

Fuente: Elaboración propia

El modelo propuesto se basa en la hermenéutica analógico-icónica del Dr. Mauricio Beuchot (2005; 2005a), ya que la analogía se usa para estudiar (comprender y explicar) procesos, seres vivos e intencionales. Y el ícono es el único signo que con un fragmento permite conocer el todo. Con un fragmento remite al significado. Una heurística interpretativa tendría los siguientes pasos:

1. Plantearse una **pregunta interpretativa** – es la guía. Una buena pregunta es la mitad de la solución. Cuidarse de no darle solución al problema incorrecto. Diferenciar entre eficacia y eficiencia.
2. Lanzar una o varias ***hipótesis interpretativas***.
3. Deliberar cuál es la mejor de ellas.
    a. Si es una, validarla (verificarla o falsarla), ver si funciona y por qué.
    b. Si son varias, cuál es la mejor y por qué.

4. Dictar un ***juicio interpretativo***. Este se obtiene al des-condicionar la hipótesis. La hipótesis siempre es un condicional, o está sujeta a un antecedente y un consecuente.
    a. Evitar, lo más que se pueda, que el juicio interpretativo sea unívoco. (Aunque en computación se busque siempre la exactitud).
    b. No ser rígido en las posibilidades de interpretación, este es el primer obstáculo que se enfrenta, el tratar de simplificar en extremo, juntando el horizonte, la tradición, las perspectivas, las disciplinas, la cultura del autor (el ingeniero de software) y del lector (el usuario o cliente) en el texto (el sistema).
    c. Evitar la equivocidad.
5. Conseguir una **interpretación analógica**. Un juicio interpretativo analógico, lo más permisivo, a la vez que objetivo.
    a. Primero con analogía de proporción.
        i. Las necesidades anteriores o de otros, intertextualidad, para saber si se ajusta el caso al hecho (contexto).
        ii. Por la experiencia, sintetizando casos que ya se han trabajado y estudiado
    b. En la medida de lo posible, establecer una analogía de atribución. Jerarquizar las distintas interpretaciones (o hipótesis) que haya hecho, o de alguien más (la tradición, los estándares, las prácticas aceptadas).
        i. Encontrar el analogado principal.
        ii. Ordenar por su grado de factibilidad (o verificabilidad dependiendo el proceso) el conjunto de interpretaciones (o hipótesis).
6. Seguir el esquema de la frónesis (la prudencia), ya que la interpretación es una virtud.
    a. Deliberación. Ver los Pros y contras de los juicios (o las hipótesis)
    b. Juicio prudencial o consejo (recomendación). El que es bueno para aconsejar es prudente.

La prudencia evita los extremos. Es la capacidad de distinguir, ponderar y encontrar una tercera opción dónde sólo había dos, como se muestra en la Figura 2.2

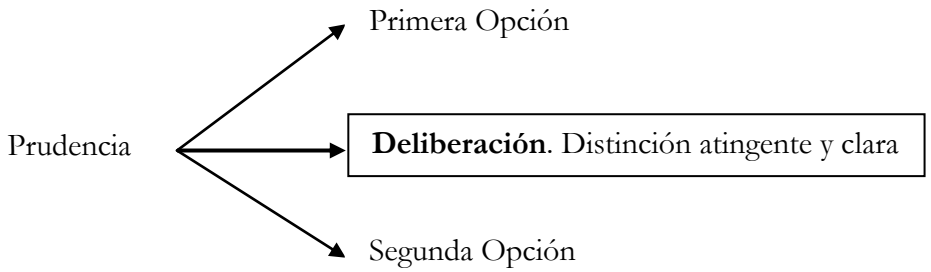

Figura 2.2 Esquema de la Prudencia.

Fuente: Elaboración propia

La hermenéutica analógico-icónica trata de juntar el explicar (lo objetivo, claro y distinto) y el comprender, tratando de que no se preste a la equivocidad: el "no te entiendo" o "tú no entiendes".

2.1.6 Rediseño al Paradigma de la Ingeniería de Software

"Los Sistemas de Información están en la edad de piedra, el paradigma que manejan está fuera de toda realidad, ya que se basan en ideas sobre la organización de los 60's. En las cuales se presenta el modelo de la administración como la toma de decisiones en la búsqueda de objetivos establecidos, un modelo del que cualquiera con experiencia real en la vida de las organizaciones sabe que es absurdo e inadecuado. Debe haber serios problemas para el campo de los Sistemas de Información en los que hay confusiones conceptuales y que está atado a un modelo caduco acerca de la naturaleza de la administración y las organizaciones… En donde se considera que la función principal de los Sistemas de Información es servir como herramientas en la toma de decisiones". (Checkland y Holwell 1998)

"El 'Desarrollo de sistemas de información' debe de empezar con una definición cuidadosa del servicio que proporcionará y partiendo de esta definición decidir qué información se necesita y cómo la tecnología nos puede ayudar a proporcionarla. (Ocurre lo contrario en las organizaciones de hoy en día –con pobres resultados que nos llevan a los espectaculares encabezados acerca de 'otro fracaso de las tecnologías de la información)" (Checkland y Holwell 1998)

El campo de los Sistemas de Información muestra dos corrientes principales de pensamiento: el 'duro' y el 'suave' (ver Tabla 2.1), los cuales producen un cuerpo de conocimiento y métodos de investigación muy distintos. El enfoque 'duro' asume que las organizaciones son entidades que buscan lograr objetivos y que el rol de la información es ayudar a la toma de decisiones; la investigación puede tomar la forma de experimentos de prueba de hipótesis, como lo hacen las ciencias naturales. El enfoque 'suave' o interpretativo, toma una visión de la organización más parcial y las ve como entidades que manejan relaciones (entre las cuales lograr objetivos es un caso especial). La información es relevante para tomar conciencia (con la toma de decisiones como un caso especial) y los métodos de investigación –aún no bien establecidos- derivarán de una ciencia social interpretativa y de la hermenéutica; estos explorarán cómo ocurre la toma de conciencia (intelección) en circunstancias particulares" (Checkland y Holwell 1998:62)

	La Tradición 'Dura'	La Tradición 'Suave'
Concepto de la organización	Entidades sociales que definen y buscan lograr objetivos, con máquinas humanas	Entidades sociales que buscan manejar relaciones
Concepto del sistema de información	Una herramienta para la toma de decisiones en búsqueda de lograr objetivos	Una parte de la interpretación del mundo. Tomar conciencia respecto a éste. Referente a la administración de las relaciones
Pensamiento de sistemas utilizado	Pensamiento de sistemas 'Duros': El mundo se asume como **sistemático**[2]	Pensamiento de sistemas 'Suaves': el proceso de búsqueda en el mundo se asume que es posible de organizarse como un sistema
Proceso de investigación y búsqueda	Basado en la prueba de hipótesis, cuantitativas si es posible	Basado en ganar comprensión y entendimiento
Teoría social	Funcionalismo[3]: Todos los aspectos de la Sociedad (instituciones, roles, normas, etc.) tienen un propósito –función- y todos ellos son indispensables para la supervivencia a largo plazo de la sociedad. *La Sociedad como un organismo*	Interpretativa
Filosofía	Positivismo	Fenomenología

Tabla 2.1. Las dos Tradiciones en las que se Basa el Trabajo de Sistemas de Información
(Checkland y Holwell 1998:48)

El desarrollo de los principios metodológicos para la elección y la explotación de la tecnología continúan hasta el día de hoy. Y hay que notar que "en un campo dominado por la tecnología dónde la tecnología cambia rápidamente, la teoría siempre estará detrás de la práctica". (Checkland y Holwell 1998)

[2] La tabla se refiere a este elemento como sistémico, pero si se contrasta con lo expuesto por el autor original cuando expone su paradigma de Investigación Activa (Checkland y Holwell 1998:24), podrá notarse claramente que el pensamiento de sistemas 'duros' asume que el mundo es sistemático.

[3] http://search.britannica.com/search?ref=A01015&query=functionalism+soc+sci&exact

2.1.7 El Cambio como Generador de Crisis

El cambio siempre ha sido acelerado. Esto no es ninguna novedad, sin embargo hay factores que hacen que los cambios que se están experimentando afecten a todos de una forma considerable:

- Los cambios habían ocurrido con la lentitud suficiente para permitir que la gente se adaptara, haciendo pequeños ajustes ocasionales o bien acumulando la necesidad de hacerlos y transfiriéndola a la siguiente generación.
- El ritmo actual es tan rápido que los retrasos para responder a él pueden ser muy costosos, incluso desastrosos. Todos los días hay compañías y gobiernos que desaparecen por que no han podido adaptarse a él o por que lo hicieron con demasiada lentitud.
- La adaptación a los acelerados cambios actuales, requiere ajustes frecuentes y extensos en lo que hacemos y cómo lo hacemos.
- Los seres humanos buscan la estabilidad y son miembros de grupos, organizaciones, instituciones y sociedades que buscan estabilidad. Puede decirse que su objetivo es la 'homeóstasis' pero el mundo en el que se persigue ese objetivo es más dinámico e inestable.
- Debido a la interconexión y la interdependencia creciente entre los individuos, grupos, organizaciones, instituciones y sociedades que se derivan de los cambios de las comunicaciones y el transporte, nuestro entorno se ha hecho más extenso, más complejo y menos predecible; en resumen se ha vuelto más turbulento. (Ackoff 1999)

Otra característica única de los cambios que experimentamos es que:

- Conforme se incrementa el ritmo del cambio, aumenta también la complejidad de los problemas que nos confrontan. Entre mayor sea la complicación de estos problemas, más tiempo tomará resolverlos
- Entre más rápido sea el ritmo del cambio, mayores serán las variaciones de los problemas que nos confrontan y más corta será la vigencia de las soluciones que encontremos para ellos.
- Por lo tanto, para cuando se tiene la cura a muchos de los problemas que nos confrontan, generalmente los más importantes, éstos ya han cambiado tanto que sus arreglos han dejado de ser relevantes o efectivos; han nacido muertos. En otras palabras, muchas de las soluciones son para dificultades que ya no existen en la forma en que se resolvieron.
- En consecuencia, nos estamos rezagando cada vez más.

El cambio más importante que está teniendo lugar, se da en la forma en la que intentamos comprender el mundo, y en nuestra concepción de su naturaleza. (Ackoff 1999)

Hay que tomar en cuenta, que en un mundo con cambios constantes y rápidos, la información sobre los sucesos no se obtiene de forma instantánea (en tiempo real). Existe un retraso, y si la información no se encuentra actualizada, las decisiones que se tomen serán erróneas.

2.1.8 Las Insuficiencias del Paradigma Sistemático

Ackoff (1999) muestra algunas ideas obvias, sobre el paradigma sistemático (ordenado, analítico y disciplinar), que son erróneas:

- Que al mejorar el rendimiento de las partes de un sistema, tomadas por separado, se mejorará necesariamente el rendimiento del todo.
- Los problemas caen naturalmente en una disciplina. La investigación efectiva no es disciplinaria, interdisciplinaria o multidisciplinaria; es transdisciplinaria.
- Lo mejor que se puede hacer con un problema es solucionarlo, falso. Lo mejor es disolverlo, rediseñar la entidad que lo tiene o su entorno, para eliminar el problema.
- La mayoría de los sistemas sociales están buscando objetivos diferentes a los que proclaman, y éstos están equivocados.

De aquí que el enfoque sistémico y transdisciplinar, aplicado en la disolución de problemas, que tome en cuenta el punto de vista de todos los participantes sea la clave para el rediseño del paradigma de la Ingeniería de Software.

La relevancia del pensamiento de sistemas para el trabajo en sistemas de Información se muestra de forma retórica en los trabajos sobre el área, pero aún no hay un esfuerzo persistente para forjar el enlace entre el pensamiento de sistemas y la provisión organizada de información en las organizaciones que es el rol de los Sistemas de Información. Checkland y Holwell (1998) destacan la relación de cuatro disciplinas relevantes para ello: Sistemas de Información, Pensamiento de Sistemas, Tecnologías de la Información y Teoría de la Organización.

La principal área del campo [Sistemas de Información] debe de ser: proveer datos e información dentro de la organización usando TI, y que esa información sea relevante para las actividades siempre cambiantes de la organización y/o de sus miembros. (Checkland y Holwell 1998)

La investigación de la realidad social debe de convertirse en un descubrimiento organizado de cómo los agentes humanos toman conciencia de sus mundos que perciben, cuáles son sus significados compartidos, y como estas percepciones cambian a través del tiempo y difieren de una persona o grupo a otro. (Checkland y Holwell 1998)

Y es en la transmisión de significados entre individuos y culturas, cuando el uso de la Hermenéutica Analógica cobra sentido ya que cuando se cambia un paradigma, tiene que hacerse conmensurable con le paradigma anterior, so pena de no ser entendido. Se ha dicho que la conmensurabilidad se da al traducir elementos de un paradigma a los de otro. Precisamente en ese trabajo de traducción es donde más se da la analogía. Hay una traducción univocista [analítica-positivista], otra equivocista [subjetiva-relativista] y otra analógica. La traducción unívoca es inalcanzable las más de las veces; la traducción equivoca es inútil; sólo queda la traducción analógica, aproximativa. (Beuchot 2005a)

El paradigma sistemático que se aplica en la Ingeniería de Software ha quedado rebasado, los cambios constantes en el entorno de las organizaciones no permiten planearlos y predecirlos. Cada vez más los usuarios demandan ser escuchados, por lo que se requiere de un lenguaje común entre todos los involucrados en el desarrollo de software. Aún con el acercamiento entre las Ingenierías de Software y de Sistemas, no se ha logrado trasladar los conceptos, modelos y metodologías claves, que lleven al desarrollo de sistemas viables, que logren adaptarse a los cambios constantes en el entorno, y que sean capaces de sobrevivir en escenarios de crisis. Y para ello se propone emplear el paradigma sistémico en el desarrollo de software, utilizando disciplinas de la Ingeniería de Sistemas y de la filosofía, como la Hermenéutica Analógica.

La Hermenéutica Analógica proporciona comprensión sobre la realidad, el poder de explicarla y transmitirla, lo cual ayuda en la traducción de las necesidades de los usuarios en diseños, productos y servicios.

Al mostrar la problemática y cómo el paradigma actual no puede resolver este conjunto de problemas, se pretende iniciar el diálogo entre la Ingeniería de Sistemas y la de software, para cumplir con la promesa de brindarles a los usuarios productos y servicios de calidad, en tiempo y a un precio accesible.

2.2 Sistemas Ético Profesionales en las Ciencias de la Computación: Hacia un Diseño Sistémico.

En el campo profesional de los sistemas, hay que tomar en cuenta varios factores para elaborar diseños viables. Se proponen tres imperativos: holismo, heurística y crítica. Los factores holísticos y heurísticos se han estudiado desde hace tiempo, los aspectos críticos son aquellos que determinan la relevancia de un sistema y los medios que se eligen para conseguir los objetivos. En ingeniería y en sistemas hay varios códigos de ética que guían nuestra práctica profesional, pero como se verá, algunos de éstos son códigos de etiqueta, no códigos de ética. Se revisa el código de práctica profesional de la ACM/IEEE-CS (Association for Computing Machinery -Asociación para Maquinaria de Cómputo / Institute of Electrical and Electronics Engineers- Computer Society – Sociedad de Computación del Instituto de Ingenieros Electricistas y Electrónicos) y el Código de Ética Profesional del Ingeniero Mexicano. Se proponen las bases para un sistema ético profesional que conduzca hacia diseños sistémicos. Para hacerlo se requiere de la incorporación de los resultados de estudios recientes en el campo de las ciencias sociales, como la Ética y la Hermenéutica Analógica.

A pesar de que los aspectos éticos y estéticos del diseño, han sido abordados por Russell L. Ackoff de manera recurrente en sus trabajos (1974, 1989, 2003), la relación y la reacción de los profesionistas del área de las Ciencias de la Computación y las ingenierías, con aquellos de áreas de la filosofía, los cuales abordan estos temas, es muy escéptica. No obstante, estos aspectos son importantes en el diseño tecnológico, ya que animan o restringen cierto tipo de comportamiento dentro y fuera de las organizaciones, por ende, afectan su desarrollo y desempeño, en otras palabras, su viabilidad.

Parte del problema, es la mala interpretación de los conceptos: "moral", "ética" o "ético"; la cual lleva a situaciones en la que los interesados tienen puntos de vista encontrados y a veces irreconciliables, siendo que lo único que hay que hacer es cambiar la clave del discurso, y desde esta nueva posición continuar con el diálogo constructivo: sistémico-transdisciplinar.

En el presente apartado se muestra cómo la ética contribuye a este diálogo transdisciplinar, conjuntando las aspiraciones individuales y colectivas, y cómo éstas conducen a un escenario distinto en donde cobran relevancia la dignidad humana, la solidaridad y la armonía.

Se inicia precisando los conceptos señalados. Posteriormente se mostrará la relación de la ética con otras disciplinas. Después se mostrará un código de práctica profesional, y se señalarán los aspectos necesarios para pasar de un código de práctica profesional (código de etiqueta) a un código de ética. Y se mostrará un código de ética profesional. Una vez establecidos los cimientos éticos se definirán los imperativos sistémicos: holísticos, heurísticos y críticos; y finalmente se aplicará todo lo anterior en la conformación de una ética del control o *Cibern-ética*.

2.2.1 La Ética

La ética forma parte del pensamiento humano y junto con otras disciplinas, da sentido y razón a la conducta humana, es decir, dirige al hombre hacia realizaciones concretas por medio de la ideología, y ésta es "una representación de la relación imaginaria de los individuos con sus condiciones reales de existencia. Esta representación está formada por un sistema de ideas que dominan el espíritu del hombre o de un grupo de hombres (sociedad). Por esa razón la ideología es la base de la ética." (Varela 1995)

El cuestionamiento señalado sobre el sentido y significado de nuestra existencia se da a través de la reflexión. Bajo esta luz tomamos decisiones y guiamos nuestra conducta (pasividad, destructividad, locura, enajenación y productividad entre otras). El hombre, al ser un ser pensante, con conciencia de si mismo ha evolucionado y creado: 1) *Cultura*. Mediante el cultivo de las facultades intelectuales y artísticas, el hombre ha creado manifestaciones de su presencia en el mundo con la música, la pintura, literatura, escultura, danza, ciencia, etc. 2) *Historia*. La capacidad de retener en la memoria los sucesos y de poder expresarlos verbalmente y por escrito, lo cual, ha hecho posible conocer el desarrollo de la humanidad, y 3) *Política*. Al establecer formas de reglamentar las relaciones en sociedad, con estructuras de gobierno, apoyados por la fuerza física, habilidades, poderío económico, la democracia o por la vía armada. (Varela 1995)

Ética (del griego *ethos* 'costumbre' y sufijo *ica* 'perteneciente a'), expresa "costumbre, hábito adquirido por la repetición de actos de la misma especie y que han adquirido fuerza de precepto". Se define como "La teoría o ciencia del comportamiento moral de los hombres en sociedad", también se define como "Una ciencia práctica y normativa que estudia racionalmente la bondad y la maldad de los actos humanos" (Varela 1995).

La ética explica la moralidad, estudiándola sistemáticamente, explicando o investigando una realidad dada. La moral es "el conjunto de normas que regulan el comportamiento individual y social" y consta de dos partes: el normativo, que se refiere al deber (tipo de comportamiento) y el fáctico, que se refiere a los hechos morales en sí. Por lo que la moral "es el conjunto de normas, reglas o deberes impuestos por una sociedad, y la moralidad es el conjunto de actos concretos efectuados por el hombre de acuerdo a la moral dominante en una sociedad determinada". La ética puede considerarse una ciencia normativa, ya que estudia la moral como un hecho real y cultural, mientras que la moral es normativa, pues prescribe normas de conducta y exige su cumplimiento (Varela 1995).

2.2.2 Relación de la Ética con otras Ciencias

La ética tiene una relación recíproca con otras ciencias, ya que por un lado, recibe aportaciones de los avances del conocimiento científico para confirmar o refutar sus postulados, y por otra parte proporciona a las ciencias una orientación en el ejercicio práctico de sus disciplinas (a través de la ética profesional). La ética, por lo tanto, se alimenta y enriquece del conocimiento científico del hombre y proporciona un camino para la praxis (Varela 1995).

2.2.3 Los Códigos de Práctica (Etiqueta) Profesional

Una profesión impone a sus practicantes ciertas obligaciones, entre las principales destacan (Varela 1995):

a) Que los *conocimientos* y las *habilidades*, relativas a la profesión, atribuidas a un practicante realmente se tengan.
b) Responsabilidad de guardar el *secreto profesional*[4] respecto de la información que se adquiere en virtud de la profesión ejercida.
c) *Solidaridad*[5] profesional, en cuanto al respeto de la clientela de los colegas.
d) *Justicia social*, cobro de honorarios justos por el servicio prestado.
e) *Humanismo*, no ver a los clientes sólo objetos en una transacción mercantil, respetar al ser humano como tal.

[4] No confundir el *secreto profesional*, con acuerdos de no divulgación (*non disclosure*), en los cuales se prohíbe, en algunos casos, hasta el uso del conocimiento adquirido por el practicante, lo cual limita la cooperación y el avance del área, de forma parecida a las patentes aplicadas a los procesos de negocios y al desarrollo de software. Y está relacionado con el siguiente apartado, la *Solidaridad*.

[5] Cabe destacar que el concepto de *Solidaridad* va más allá del respeto del mercado de los colegas.

A continuación se muestra un código de ética relacionado con las áreas de ingeniería y sistemas.

2.2.4 Código de Práctica Profesional en el Campo de la Informática-Computación

Las dos agrupaciones principales del campo de la Computación, la ACM (Association for Computing Machinery -Asociación para maquinaria de cómputo) y la IEEE-CS (Institute of Electrical and Electronics Engineers- Computer Society –Sociedad de Computación del Instituto de Ingenieros Electricistas y Electrónicos), establecieron un comité conjunto para elaborar un código de ética para el área de la Ingeniería de Software, este código fue aprobado por los miembros de ambas instituciones.

2.2.4.1 Código Práctica Profesional 5.2 de la ACM y la IEEE-CS

El código de ética y práctica profesional de la Ingeniería de Software, en su versión corta, fue elaborado por el comité conjunto de la Sociedad de Cómputo de la IEEE (IEEE-CS) y la ACM. (IEEE-CS/ACM 1999)

PREÁMBULO

La versión corta del código resume las aspiraciones [del código] en un nivel alto de abstracción. Las cláusulas que se incluyen en la versión completa[6] proporcionan ejemplos y detalles acerca de cómo estas aspiraciones modifican nuestra manera de actuar como profesionales de la Ingeniería de Software. Sin las aspiraciones los detalles pueden convertirse en tediosos y legalistas. Sin los detalles las aspiraciones pueden convertirse en frases rimbombantes pero vacías. Las aspiraciones y los detalles forman en conjunto un código consistente.

Los ingenieros de software deberán comprometerse a convertir el análisis, especificación, diseño, implementación, pruebas y mantenimiento de software en una profesión respetada y benéfica. De acuerdo a su compromiso con la salud, seguridad y bienestar social, los ingenieros de software deberán sujetarse a los ocho principios siguientes:

1. *Sociedad.* Los ingenieros de software actuarán en forma congruente con el interés social.
2. *Cliente y Patrón.* Los ingenieros de software actuarán de manera que se concilien los mejores intereses de sus clientes y el patrón, congruentemente con el interés social.

[6] La versión completa puede consultarse la misma referencia (IEEE-CS/ACM 1999)

3. *Producto*. Los ingenieros de software asegurarán que sus productos, y modificaciones correspondientes, cumplen los estándares profesionales más altos.

4. *Juicio*. Los ingenieros de software mantendrán integridad e independencia en su juicio profesional.

5. *Administración*. Los ingenieros de software, gerentes y líderes, promoverán y se suscribirán a un enfoque ético en la administración del desarrollo y mantenimiento de software.

6. *Profesión*. Los ingenieros de software incrementarán la integridad y reputación de la profesión congruentemente con el interés social.

7. *Colegas*. Los ingenieros de software apoyarán y serán justos con sus colegas.

8. *Personal*. Los ingenieros de software participarán toda su vida en el aprendizaje relacionado con la práctica de su profesión y promoverán un enfoque ético en la práctica de la profesión.

Hay que considerar que en el mundo profesional nos enfrentamos con varios problemas y a diversas situaciones las cuales rebasan el ámbito de los códigos de práctica profesional o códigos de etiqueta. Por ejemplo la falta de equidad que señala Ackoff (1989): "la preocupación principal de los que dirigen las empresas es asegurarse el ambiente de trabajo y el nivel de vida que desean [para ellos]. El comportamiento de los directivos puede entenderse mucho mejor si se acepta este supuesto en lugar de pensar que su objetivo es maximizar las ganancias o el crecimiento de la empresa […] lo único malo es su alcance [el cual] debería ampliarse hasta cubrir el ambiente de trabajo y el nivel de vida de todos los empleados". Siendo que lo más justo en este sentido sería "proporcionar a los empleados de cualquier nivel un trabajo interesante, bien remunerado y que brinde oportunidades para el desarrollo profesional, [lo cual] debería ser parte de la misión de toda empresa." (Ackoff 1989)

El anterior es un ejemplo de cómo los códigos de etiqueta quedan rebasados, ya que no brindan ningún marco de referencia para temas como la calidad de vida, dignidad humana, autorrealización, etc.

2.2.5 ¿Códigos de Ética o de Etiqueta?

Beuchot (2004) señala que varios códigos de ética son en realidad códigos de etiqueta, ya que la acción humana moral se despliega buscando, primeramente una finalidad: la felicidad, la vida feliz. Y en ese sentido, la ética "nos confronta, nos cuestiona, nos responsabiliza y a la vez nos hace crecer como personas." (Beuchot 2004). En ese sentido coincide con Adela Cortina (1996), quien señala

que la ética es "un tipo de saber que nos orienta para forjarnos un buen carácter, que nos permita enfrentar la vida con altura humana, que nos permita, en suma, ser justos y felices […] de ahí que ética y moral nos ayuden a labrarnos un buen carácter para ser humanamente íntegros."

Y es en este punto, en donde hay que señalar un cambio de clave importante, en el discurso tradicional sobre la ética y la moral, el cual es, no contrastar "moral-inmoral" sino en una contraposición menos dogmática: "moral-desmoralizado", Ya que "decir que alguien es inmoral es acusarle de no someterse a unas normas, de lo cual puede incluso sentirse orgulloso sino las respeta como suyas; pero a nadie le gusta estar desmoralizado, por que entonces la vida le parece una losa y cualquier tarea, una tortura" (Cortina 1996)

Con este cambio de clave, se pueden superar discusiones estériles y se cuestiona la validez de la ideología actual, ya que "vivimos en una época en dónde rige la ética protestante, en dónde se hace creer a los hombres que mientras más sufran en el trabajo, mejor sería para sus almas" Ackoff (1974)

2.2.6 Consideraciones para un Sistema Ético Profesional

Los aspectos (elementos) que conforman a un sistema ético profesional, según Etxeberria (2002) son los siguientes:

- Plantear cómo se realiza y cómo se controla en la actividad profesional la autonomía de las personas implicadas: la del propio profesional, la de los clientes o usuarios, la de los afectados.
- Concretar cómo se realizan las exigencias de la justicia desde y gracias a las actividades profesionales.
- Definir los principios y normas por los que debe regirse la profesión.
- Hay que precisar cómo la actividad profesional remite al bien. En primer lugar, al bien y autorrealización del propio profesional, que debe poder ver en el ejercicio de su profesión un componente decisivo de su proyecto de felicidad, de su ideal de vida, de su horizonte de plenitud. En segundo lugar, hay que remitir el conjunto de las profesiones al bien social, al bien común.
- Hay que relacionar el ejercicio profesional con la realización de las virtudes, especialmente de aquellas virtudes que más conexión tienen con cada una de las profesiones en función de sus especificidades.

- Respetar la pluralidad social, entre los clientes y entre los propios profesionales.

2.2.7 Código de Ética Profesional del Ingeniero Mexicano

El Código de Ética Profesional del Ingeniero Mexicano se publicó el 1 de julio de 1983, y firmó como testigo el C. licenciado Miguel de la Madrid Hurtado, Presidente Constitucional de los Estados Unidos Mexicanos, el cual se transcribe a continuación (UMAI 1983):

"CONSIDERANDO QUE:

1. El ingeniero mexicano sustenta su conducta en el respeto y amor a la patria.
2. El ingeniero en nuestro país ha logrado la práctica de su profesión gracias a la oportunidad que le brinda la nación mexicana.
3. Por su preparación tiene un mayor compromiso para coadyuvar a satisfacer las necesidades y elevar la calidad de vida de los mexicanos, con la convicción y responsabilidad moral de sostener un desarrollo con justicia social.
4. Es un deber propiciar el desempeño de la actividad de acuerdo con un Código de Ética que precise las obligaciones sociales, que hacen posible el respeto de cada profesional para con los demás, en busca de una justa y armoniosa convivencia humana dentro de cada nación y entre las naciones.
5. Los principios universales y nuestras mejores tradiciones consideran un alto deber la solidaridad internacional y el respeto a los valores morales de otros pueblos, en particular donde el ingeniero amplíe su preparación o eventualmente ejerza la profesión.
6. Los diversos códigos de ética profesional de colegios y asociaciones de ingenieros confluyen en una misma concepción.
7. La unión de ingenieros mexicanos se ha dado en torno a principios y normas de conducta.

Al adoptar el Código de Ética Profesional del Ingeniero Mexicano: El ingeniero reconoce que el mayor mérito es el trabajo, por lo que ejercerá su profesión comprometido con el servicio a la sociedad mexicana, atendiendo al bienestar y progreso de la mayoría.

Al transformar la naturaleza en beneficio de la humanidad, el ingeniero debe acrecentar su conciencia de que el mundo es la morada del hombre y de que su interés por el universo es una garantía de la superación de su espíritu y del conocimiento de la realidad para hacerla más justa y feliz.
El ingeniero debe rechazar los trabajos que tengan como fin atentar contra el interés general. De esta manera evitará situaciones que impliquen peligros o constituyan una amenaza contra el medio ambiente, la vida, la salud y demás derechos del ser humano.

Es un deber ineludible del ingeniero sostener el prestigio de la profesión y velar por su cabal ejercicio. Asimismo, mantener una conducta profesional cimentada en la capacidad, la honradez, la fortaleza, la templanza, la magnanimidad, la modestia, la franqueza y la justicia, con la conciencia de subordinar el bienestar individual al bien social.

El ingeniero debe procurar el perfeccionamiento constante de sus conocimientos, en particular de su profesión, divulgar su saber, compartir su experiencia, proveer oportunidades para la formación y la capacitación de los trabajadores, brindar reconocimiento, apoyo moral y material a la institución educativa en donde realizó sus estudios; de esta manera revertirá a la sociedad las oportunidades que ha recibido.

Es responsabilidad del ingeniero que su trabajo se realice con eficiencia y apoyo a las disposiciones legales. En particular, velará por el cumplimiento de las normas de protección a los trabajadores establecidas en la legislación laboral mexicana.

En el ejercicio de su profesión, el ingeniero debe cumplir con diligencia los compromisos que haya asumido y desempeñará con dedicación y lealtad los trabajos que se le asignen, evitando anteponer su interés personal en la atención de los asuntos que se le encomienden, o asociarse para ejercer competencia desleal en perjuicio de quien reciba sus servicios.

Observará una conducta decorosa, tratando con respeto, diligencia, imparcialidad y rectitud a las personas con las que tenga relación, particularmente a sus colaboradores, absteniéndose de incurrir en desviaciones y abusos de autoridad y de disponer o autorizar a un subordinado conductas ilícitas, así como de favorecer indebidamente a terceros.

Debe salvaguardar los intereses de la institución o persona para la que trabaje y hacer buen uso de los recursos que se le hayan asignado para el desempeño de sus labores.

Cumplirá con eficiencia las disposiciones que en ejercicio de sus atribuciones le dictaminen sus superiores jerárquicos, respetará y hará respetar su posición y trabajo; si discrepara de sus superiores tendrá la obligación de manifestar ante ellos las razones de su discrepancia.

El ingeniero tendrá como norma crear y promover la tecnología nacional; pondrá especial cuidado en vigilar que la transferencia tecnológica se adapte a nuestras condiciones conforme al marco legal establecido. Se obliga a guardar secreto profesional de los datos confidenciales que conozca en el ejercicio de su profesión, salvo que le sean requeridos por autoridad competente."

2.2.8 Aspectos Legales: Derechos de Autor y Propiedad Industrial

> "No hemos entendido que (hoy) la riqueza real es el conocimiento. La riqueza ya no está en la tierra ni en la fábrica: la riqueza está hoy en las patentes. Eso es lo que da a una economía su vitalidad, y ahí tenemos un déficit tecnológico-científico histórico"
>
> (Sanguinetti 2004 en Blanco 2004)

Los aspectos jurídicos son importantes en las relaciones mercantiles. Una correcta explotación de los procesos creativos nos beneficiará, y las Leyes afectan (promueven o inhiben) el desarrollo tecnológico.

2.2.8.1 Derechos de Autor (Derechos de Copia –Copyright) en la Legislación Mexicana

En México, el Instituto Nacional del Derecho de Autor (INDAUTOR), organismo desconcentrado de la Secretaría de Educación Pública, es el encargado de administrar y aplicar los derechos de autor.

La Ley Federal del Derecho de Autor- LFDA, tiene por objeto "la salvaguarda y promoción del acervo cultural de la Nación; protección de los derechos de los autores, de los artistas intérpretes o ejecutantes, así como de los editores, de los productores y de los organismos de radiodifusión, en relación con sus obras literarias o artísticas en todas sus manifestaciones, sus interpretaciones o ejecuciones, sus ediciones, sus fonogramas o videogramas, sus emisiones, así como de los otros derechos de propiedad intelectual."

En su Artículo 11, la LFDA, señala que "El derecho de autor es el reconocimiento que hace el Estado en favor de todo creador de obras literarias y artísticas previstas en el artículo 13 de esta Ley, en virtud del cual otorga su protección para que el autor goce de prerrogativas y privilegios exclusivos de carácter personal y patrimonial. Los primeros integran el llamado derecho moral y los segundos, el patrimonial." (Congreso de la Unión 2003)

Las obras a las que se refiere el Artículo 13 (Congreso de la Unión 2003) abarcan las siguientes ramas:

"I. Literaria;
II. Musical, con o sin letra;
III. Dramática;
IV. Danza;
V. Pictórica o de dibujo;
VI. Escultórica y de carácter plástico;
VII. Caricatura e historieta;
VIII. Arquitectónica;
IX. Cinematográfica y demás obras audiovisuales;
X. Programas de radio y televisión;
XI. Programas de cómputo;
XII. Fotográfica;
XIII. Obras de arte aplicado que incluyen el diseño gráfico o textil, y
XIV. De compilación, integrada por las colecciones de obras, tales como las enciclopedias, las antologías, y de obras u otros elementos como las bases de datos, siempre que dichas colecciones, por su selección o la disposición de su contenido o materias, constituyan una creación intelectual.

Las demás obras que por analogía puedan considerarse obras literarias o artísticas se incluyen en la rama que les sea más afín a su naturaleza."

Las obras relevantes para el área son los programas de cómputo. Éstos son importantes para las empresas de TI. Los demás diseños (modelos) en cuanto a creaciones, caen en la rama de creaciones literarias (en forma de un libro, manual o guía).

Los planos o esquemas de maquinaria se consideran partes de máquinas, y caen dentro de la Ley de la Propiedad Industrial, por lo que no hay una rama de ingeniería.

El Artículo 14, señala que no son objeto de la protección como derecho de autor a que se refiere la LFDA (Congreso de la Unión 2003):

"I. Las ideas en sí mismas, las fórmulas, soluciones, conceptos, métodos, sistemas, principios, descubrimientos, procesos e invenciones de cualquier tipo;
II. El aprovechamiento industrial o comercial de las ideas contenidas en las obras;
III. Los esquemas, planes o reglas para realizar actos mentales, juegos o negocios;
IV. Las letras, los dígitos o los colores aislados, a menos que su estilización sea tal que las conviertan en dibujos originales;
V. Los nombres y títulos o frases aislados;
VI. Los simples formatos o formularios en blanco para ser llenados con cualquier tipo de información, así como sus instructivos;
VII. Las reproducciones o imitaciones, sin autorización, de escudos, banderas o emblemas de cualquier país, estado, municipio o división política equivalente, ni las denominaciones, siglas, símbolos o emblemas de organizaciones internacionales gubernamentales, no gubernamentales, o de cualquier otra organización reconocida oficialmente, así como la designación verbal de los mismos;
VIII. Los textos legislativos, reglamentarios, administrativos o judiciales, así como sus traducciones oficiales. En caso de ser publicados, deberán apegarse al texto oficial y no conferirán derecho exclusivo de edición;
Sin embargo, serán objeto de protección las concordancias, interpretaciones, estudios comparativos, anotaciones, comentarios y demás trabajos similares que entrañen, por parte de su autor, la creación de una obra original;
IX. El contenido informativo de las noticias, pero sí su forma de expresión, y
X. La información de uso común tal como los refranes, dichos, leyendas, hechos, calendarios y las escalas métricas."

Aquí es importante señalar que el Artículo 14 de esta Ley, indica que no son susceptibles de protección como derecho de autor, lo cual no impide que tengan protección bajo otras Leyes, como la de propiedad industrial, la cual se revisa en el siguiente apartado.

La consideración de los derechos de autor, en el ámbito de TI, es algo que se ha descuidado. Estos son importantes para establecer un acervo de nuestros recursos, ya que las obras (diseños, programas, manuales, etc.) junto con los creadores (ingenieros, desarrolladores, empleados) conforman el capital intelectual de la empresa, el cual bien administrado, brinda ventajas a las instituciones. Esta ley también sirve como marco para establecer la propiedad de los diseños, programas desarrollados dentro de la empresa, por terceros y aquellos que desarrollamos para nuestros clientes.

Las obras están protegidas en el momento de su creación, y el trámite de registro es optativo, aunque recomendable en la mayoría de los casos.

2.2.8.2 Propiedad Industrial (Patentes, Marcas y Secretos Industriales) en la Legislación Mexicana

El Instituto Mexicano de la Propiedad Industrial, organismo descentralizado con personalidad jurídica y patrimonio, es la autoridad administrativa en materia de propiedad industrial.

Uno de los objetos de la Ley de la propiedad industrial (Congreso de la Unión 2004) es "Proteger la propiedad industrial mediante la regulación y otorgamiento de patentes de invención; registros de modelos de utilidad, diseños industriales, marcas, y avisos comerciales; publicación de nombres comerciales; declaración de protección de denominaciones de origen, y regulación de secretos industriales."

En su Artículo 9, la Ley de la propiedad industrial (Congreso de la Unión 2004), estable ce que "La persona física que realice una invención, modelo de utilidad o diseño industrial, o su causahabiente, tendrán el derecho exclusivo de su explotación en su provecho, por sí o por otros con su consentimiento, de acuerdo con las disposiciones contenidas en esta Ley y su reglamento[…]El derecho a que se refiere el artículo anterior se otorgará a través de *patente* en el caso de las invenciones y de *registros* por lo que hace a los modelos de utilidad y diseños industriales.

"Se considera invención toda creación humana que permita transformar la materia o la energía que existe en la naturaleza, para su aprovechamiento por el hombre y satisfacer sus necesidades concretas". Artículo 15. (Congreso de la Unión 2004)

No se considerarán invenciones para los efectos de la Ley (Artículo 19, (Congreso de la Unión 2004):

"I.- Los principios teóricos o científicos;
II.- Los descubrimientos que consistan en dar a conocer o revelar algo que ya existía en la naturaleza, aún cuando anteriormente fuese desconocido para el hombre;
III.- Los esquemas, planes, reglas y métodos para realizar actos mentales, juegos o negocios y los métodos matemáticos;
IV.- Los programas de computación;
V.- Las formas de presentación de información;
VI.- Las creaciones estéticas y las obras artísticas o literarias;
VII.- Los métodos de tratamiento quirúrgico, terapéutico o de diagnóstico aplicables al cuerpo humano y los relativos a animales, y
VIII.- La yuxtaposición de invenciones conocidas o mezclas de productos conocidos, su variación de uso, de forma, de dimensiones o de materiales, salvo que en realidad se trate de su combinación o fusión de tal manera que no puedan funcionar separadamente o que las cualidades o funciones características de las mismas sean modificadas para obtener un resultado industrial o un uso no obvio para un técnico en la materia."

Al igual que en la LFDA, la Ley de la propiedad industrial no considera invenciones a los elementos descritos, lo cual no impide que no puedan protegerse por otras vías. Como se observa en la legislación actual no son sujetos de patente los programas de cómputo, los métodos matemáticos (algoritmos) ni los procesos de negocio.

Otra forma de protección a tener en cuenta, son los secretos industriales, en el Artículo 82, de la misma Ley (Congreso de la Unión 2004), se considera secreto industrial a "toda información de aplicación industrial o comercial que guarde una persona física o moral con carácter confidencial, que le signifique obtener o mantener una ventaja competitiva o económica frente a terceros en la realización de actividades económicas y respecto de la cual haya adoptado los medios o sistemas suficientes para preservar su confidencialidad y el acceso restringido a la misma.

La información de un secreto industrial necesariamente deberá estar referida a la naturaleza, características o finalidades de los productos; a los métodos o procesos de producción; o a los medios o formas de distribución o comercialización de productos o prestación de servicios.

No se considerará secreto industrial aquella información que sea del dominio público, la que resulte evidente para un técnico en la materia, con base en información previamente disponible o la que deba ser divulgada por disposición legal o por orden judicial. No se considerará que entra al dominio público o que es divulgada por disposición legal aquella información que sea proporcionada a cualquier autoridad por una persona que la posea como secreto industrial, cuando la proporcione para el efecto de obtener licencias, permisos, autorizaciones, registros, o cualesquiera otros actos de autoridad"

El secreto industrial es información confidencial de acceso restringido, la cual se considera brinda una ventaja comercial o industrial.

En ambos casos, la LFDA y la Ley de Propiedad Industrial, prevén que los derechos obtenidos se puedan ceder (renunciar) o licenciar (permitir su explotación).

2.2.9 Imperativos Sistémicos

Una vez establecido lo que es un código de ética y que se han introducido las leyes mexicanas de derechos de autor y de propiedad industrial, se definen las consideraciones (imperativos) sistémicas que se deben de cumplir en las organizaciones, tanto para la elaboración (diseño), la operación, el control y la evaluación (diagnóstico) de los distintos procesos y actividades que se desarrollan dentro y fuera de la misma.

2.2.9.1 Holístico

Se define como el proceso de enfocar directamente el todo, y sus características como un todo, sin considerar sus partes (vista global). También se dice que es una posición filosófica la cual establece que: a) Los todos (Holón-wholes) no se pueden separar y b) que cualquier Holón (whole) puede entenderse únicamente en el contexto de otro Holón más grande que lo contiene, también da pie al concepto de sinergia: El todo es más que la suma de sus partes. Esta posición implica el establecimiento de jerarquías. (Principia Cibernética 2003)

2.2.9.2 Heurístico[7]

Heurística se define como el arte de inventar o el arte de descubrir soluciones. De modo breve, incluye preguntas relativas a lo que «necesito para hacer», si «tengo lo que necesito», «qué tengo actualmente», y por último «si me sirve lo que tengo».

a) ¿Qué necesito para hacer?- esta pregunta radica en la obtención de bases, antecedentes (información) y herramientas (procedimientos) que visualiza una alternativa para la solución del problema.
b) ¿Tengo lo que necesito?- es referente a la disponibilidad de las cosas y al conocimiento de las mismas para adecuar su uso a mis necesidades y no depender posteriormente de ellas.
c) ¿Qué tengo actualmente?- asociado con lo que cuento para desarrollar la solución del problema sin llegar a clasificar su importancia o jerarquía.
d) ¿Me sirve lo que tengo?- en este punto entra la aceptación o el descarte de piezas relacionadas con el problema, tanto de ideas como de objetos, se acomodan por importancia y modo de empleo.

Una vez analizadas y respondidas estas preguntas, el profesional se encuentra en el centro del problema y puede entender la vinculación de la metodología y del problema y decidir si seguir con una investigación, cuando el problema consiste en responder una pregunta, o con una acción, cuando se trata de producir un cambio.

2.2.10 El Sentido Común y el Sentido Crítico[8]

La Ley de Occam dice que "De dos soluciones igualmente válidas a un problema, la más sencilla es la que tiene precedencia".

El sentido común es aquel que produce una noción trivial derivada de una impresión inmediata. Utilizan datos de la experiencia sensible haciendo asociaciones simbólicas. Es la primera impresión que se produce en nuestro contacto con el mundo.

[7] Resumen de Mora (2001)

[8] Resumen de Mora (2001)

Resultantes de actitud derivadas del sentido común, son el respetar el entorno, la naturaleza, las cosas tal y como están cuando llegamos, así como el respeto por uno mismo, conscientes de nuestras propias limitaciones; es el comienzo para pretender una experiencia gratificante.

El buen sentido, designado en el presente trabajo como sentido crítico, produce una noción sustancial derivada de una impresión profunda. Concierne a esa versión de la conciencia humana que no puede ser resuelta con la lógica de la dominación, por el contrario, es una combinación compleja de sentido bueno y malo. Un ámbito contradictorio de ideas y conducta en la que los elementos de la acomodación y la resistencia existen en un inestable estado de tensión.

Más específicamente, la perspectiva del sentido común apunta hacia un modo de subjetividad caracterizado por formas de conciencia discursiva imbuidas con ideas dentro de la realidad social así como con las creencias distorsionadas que las mistifican y las legitiman. En adición, el sentido común se efectúa y se manifiesta a sí mismo en una conducta no discursiva marcada por la misma combinación de elementos de acomodación y resistencia. Lo que comparten estos dos momentos contradictorios, es que como condiciones para el conocimiento y para la conducta, funcionan sin el beneficio de la investigación crítica.

El sentido común representa un modo limitado de autoconciencia en contradicción con la naturaleza y mal equipado para captar la fuerza detrás de él o sus efectos en la totalidad social. Desorden más que armonía, caracteriza al sentido común. Contiene una interacción lógica entre creencias y prácticas superiores y perspicaces.

Aunque la participación no desaparece en este supuesto, carece de la dinámica de la conciencia propia necesaria para resolver sus contradicciones y tensiones o para extender sus planteamientos a una perspectiva crítica coherente a partir de la cual pueda comprometer sus propios principios.

El hombre de las masas activo, tiene una actividad práctica, pero no tiene conciencia teórica clara de su actividad práctica, la cual incluye la comprensión del mundo y la razón por la que éste la transforma.

Su conciencia teórica puede verdaderamente estar históricamente en contradicción con su actividad. Uno puede casi decir que él tiene dos conciencias teóricas (o una conciencia contradictoria); una que está implícita en su actividad y que en la realidad lo une con todos sus compañeros trabajadores en la

transformación práctica del mundo real: y otra, superficialmente explícita o verbal, que ha heredado del pasado y que ha absorbido sin criticar".

El sentido común está constituido por categorías de lo dado por hecho y por la actividad práctica divorciada de los agentes y las condiciones que las producen. En esta perspectiva el Sentido Crítico funciona para desenmascarar los mensajes revelados en el sentido común, para reubicar históricamente los intereses que los estructuraban, y para investigar sus reclamos verdaderos y sus funciones sociales. La conciencia histórica, como instancia de sentido crítico, funciona para percibir el pasado en una forma que (hace) al presente visible como momento revolucionario. En otras palabras, el sentido crítico, fundamenta la producción de conocimiento, incluyendo a la ciencia, en un marco de referencia normativo vinculado con intereses específicos. Por lo tanto, implica un proceso por el cual el significado es producido, representado y consumido. El aspecto crítico de ese proceso representa una comprensión reflexiva de los intereses incluidos en el proceso mismo y cómo estos intereses pudieran ser transformados, desafiados o mantenidos a fin de promover, más que de reprimir, las dinámicas del pensamiento y de la acción crítica. Esto significa, separar las ideas y estructurar los principios en un artefacto cultural y compararlos en un marco de referencia distinto que permita dejar a la vista los límites de ideas específicas y las propiedades formales, mientras simultáneamente permita descubrir los elementos nuevos y vitales de éstos.

El vínculo entre el sentido crítico y la noción de la verdad se localiza en la distancia entre el intérprete y lo material por un lado, y la brecha entre el presente y la posibilidad de un futuro radicalmente diferente, por el otro.

En conclusión, el Sentido Común se utiliza casi a diario. Las acciones que ocurren a nuestro alrededor las interpretamos a nuestro modo. Una ventaja es que no generaliza sobre un tema o realidad, se dice como le sucedió a uno, es el clásico dicho "Cada uno habla de cómo le fue en la feria". Esto es cierto porque cada uno dice lo que sintió de acuerdo a una acción. Pero no todo es bueno en este aspecto, el sentido común nos hace cometer errores, ya que al ver o estar en una situación la interpretamos a nuestro modo, y eso no es bueno en ocasiones, porque podemos concluir mal y ver lo que queremos ver y no lo que en realidad sucede. Este se demuestra de afuera hacia adentro. Mientras que el Sentido Crítico nos enseña lo que realmente ocurre, se deriva de una impresión profunda, es decir, se concluye después de hacer un breve o sustancioso análisis de lo que ocurre a nuestro alrededor, este actúa de adentro hacia fuera, de manera subjetiva. El inconveniente

es que está íntimamente relacionado con el Idealismo y por eso muchas veces no se toma en serio porque al idealismo se le tacha como defecto, siendo que muchas veces sirve para resolver problemas ya que es utilizado por la Metodología de Sistemas.

2.2.11 De la Conciencia Ordinaria (Utilitaria) a la Conciencia Poiética (Creadora)[9].

El hombre común y corriente se halla en una relación directa e inmediata con las cosas –relación que no puede dejar de ser consciente-, pero en ella la conciencia no destaca o separa la práctica como su objeto propio, para darse ante ella en un estado teórico, es decir, como objeto del pensamiento. La conciencia ordinaria piensa los actos prácticos, pero no hace poiésis [10] (actividad social transformadora).

Así el hombre común y corriente se ve así mismo como el ser práctico que no necesita de teorías; los problemas encuentran solución en la práctica misma, o en esa forma de revivir una práctica pasada que es la experiencia. Pensamiento y acción, teoría y práctica se separan.

El hombre común y corriente, inmerso en el mundo de intereses y necesidades de la cotidianidad, no se eleva a una verdadera conciencia[…], incapaz de rebasar el límite estrecho de su actividad práctica para ver, sobre todo ciertas formas de ella –el trabajo, la actividad política, etc.- Es decir, no acierta a ver hasta qué punto, con sus actos prácticos, está contribuyendo a escribir la historia humana, ni puede comprender hasta que grado la praxis [y con más razón la poiésis] es menesterosa de la teoría o hasta que punto su actividad práctica se inserta en una poiésis humana social con lo cual sus actos individuales implican los de los demás [como en un sistema], y, a su vez, los de estos se reflejan en su propia actividad [retroalimentación]. Ahora bien, la superación de esa concepción de la poiésis que la reduce a una actividad utilitaria, individual, y autosuficiente (con respecto a la teoría) es una empresa que rebasa las posibilidades de la conciencia ordinaria.

[9] Resumen del apartado "La Conciencia Ordinaria de la Praxis" en la Introducción de Sánchez (2003a)

[10] Sánchez (2003a) se refiere a este término como *praxis*, y es el nombre que le da titulo a su libro "Filosofía de la Praxis", aunque el término más adecuado para el proceso que el quiere designar es *poiésis*, y el mismo lo señala así en las Págs. 27 y 28. En dónde el indica que por *praxis* el se refiere a *poiésis* y, por lo tanto, su filosofía sería una "filosofía de la poiésis", a pesar de la contundencia del término Sánchez renuncia a nombrarlo así a lo largo de su obra. Nosotros rescatamos este término puesto que su uso es extendido en el ámbito de la Ingeniería de Sistemas.

2.3 Arquitecturas

En este apartado se muestra la evolución del concepto Arquitectura dentro del campo de la computación y la Ingeniería de Software. Se muestra su importancia en el desarrollo de tecnología, así como sus componentes clave.

Originalmente, Arquitectura era "La estructura organizativa de un sistema o componente" (IEEE 1990). Para construir una arquitectura, era preciso establecer un marco de referencia, o Diseño de Arquitectura, el cual es: "El proceso de definir un conjunto de componentes de hardware y software, y sus interfaces, para establecer la estructura (framework) en el desarrollo de un sistema computacional." y forma parte del diseño funcional, el cual es un proceso en el que se definen las relaciones entre estos componentes. (IEEE 1990)

Cuando se le agrega un adjetivo al término arquitectura, se presenta confusión, ya que hay arquitecturas de sistemas, de hardware, de software, de red, de cómputo, de TI y empresariales. Una arquitectura de software describe el esquema de los módulos de software y las conexiones y relaciones entre ellos. Una arquitectura de hardware puede describir como están organizados los componentes del hardware. Estas definiciones se pueden aplicar a una sola computadora, o a una familia de sistemas de información. Por lo que el término arquitectura puede tener una amplia cantidad de significados y puede abarcar distintas áreas y componentes, dependiendo de la persona que este definiendo el término. La arquitectura empresarial es una meta-arquitectura que abarca muchos sistemas de información y sus relaciones (infraestructura técnica), y además contiene otras vistas de la empresa (tareas, funciones e información), se considera que la arquitectura empresarial es el nivel más alto de arquitectura. (Armour, Kaisler y Simon 1999)

Según Sommerville (2005), existen arquitecturas generales y arquitecturas de dominio específico. Las arquitecturas generales comprenden los modelos de depósito, el modelo cliente-servidor, el modelo de máquina abstracta, los modelos de control y los modelos de descomposición modular (modelo orientado a objetos y modelo de flujo de datos). Las arquitecturas de dominio específico incluyen los modelos genéricos (abstracciones de varios sistemas reales) y los modelos de referencia que sirven como medio para comparar y comprobar si distintos sistemas en un dominio se apegan a un estándar, un ejemplo de modelo de referencia es el modelo *OSI* para la interconexión de sistemas abiertos.

En la Figura 2.3 se muestra una comparación entre el modelo *OSI* y el conjunto de protocolos *TCP/IP*. Cabe señalar que aunque el estándar de jure es el modelo *OSI*, la implementación extensiva del conjunto de protocolos *TCP/IP* (El protocolo Internet) lo han convertido en el estándar de *facto* en la industria, y es parte primordial de la arquitectura de red.

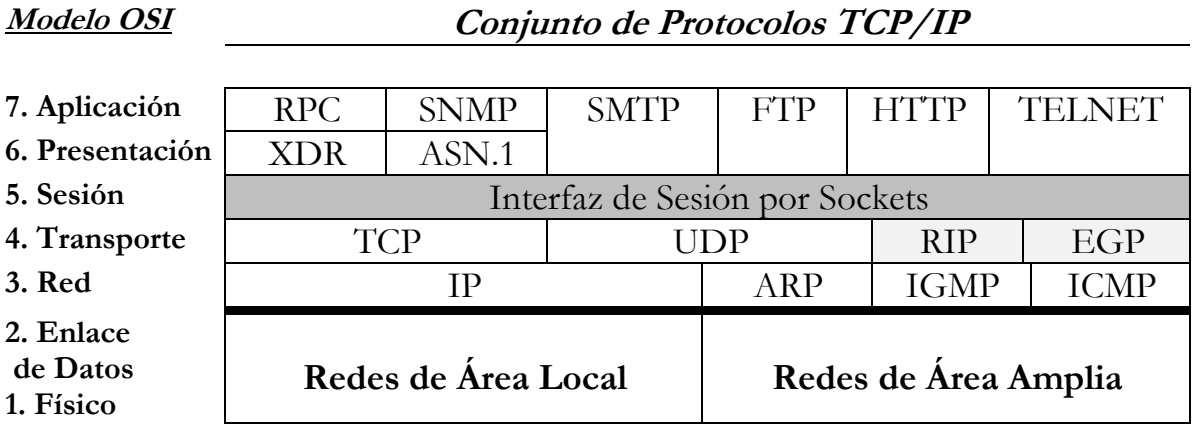

Figura 2.3 El Modelo *OSI* y el Conjunto de Protocolos *TCP/IP*.

Fuente: Adaptado de Marín (1995)

2.3.1 Evolución de las Arquitecturas en el Cómputo Empresarial

La evolución del término arquitectura, dentro del cómputo empresarial, se dio a partir del surgimiento de las redes de telecomunicaciones y del uso de los sistemas de cómputo distribuido.

El término arquitecturas de cómputo empresarial se usaba para describir un conjunto de plataformas (hardware y software) y las redes de datos que soportaban las necesidades de información de la organización. Se consideraba que el diseño de componentes intercambiables facilitaba la comparación entre distintas tecnologías. (Nezlek, Jain y Nazareth 1999). En este contexto, la arquitectura de cómputo empresarial agrupa e integra todos los recursos de cómputo y comunicaciones. Todas las computadoras, sus periféricos, y todas las redes en la organización están integradas. El objetivo primordial es hacer que todos ellos funcionen como una sola entidad (Renaud 1993).

Por analogía, se trasladó el concepto de la arquitectura de una computadora al de un grupo de computadoras integradas. En la arquitectura de una computadora, el CPU, la memoria y los controladores de entrada/salida están interconectados con el canal de comunicación (bus) del

sistema. Los controladores de E/S proveen acceso a los datos y conectividad hacia afuera de la computadora. En la Computadora Empresarial, se usa el mismo modelo en una escala más amplia (ver Figura 2.4). Los sistemas Cliente y Servidor están interconectados con la red global. Los Servidores proveen acceso a los datos y conectividad fuera de la organización, es la misma arquitectura pero con diferente enfoque. (Marín 1995)

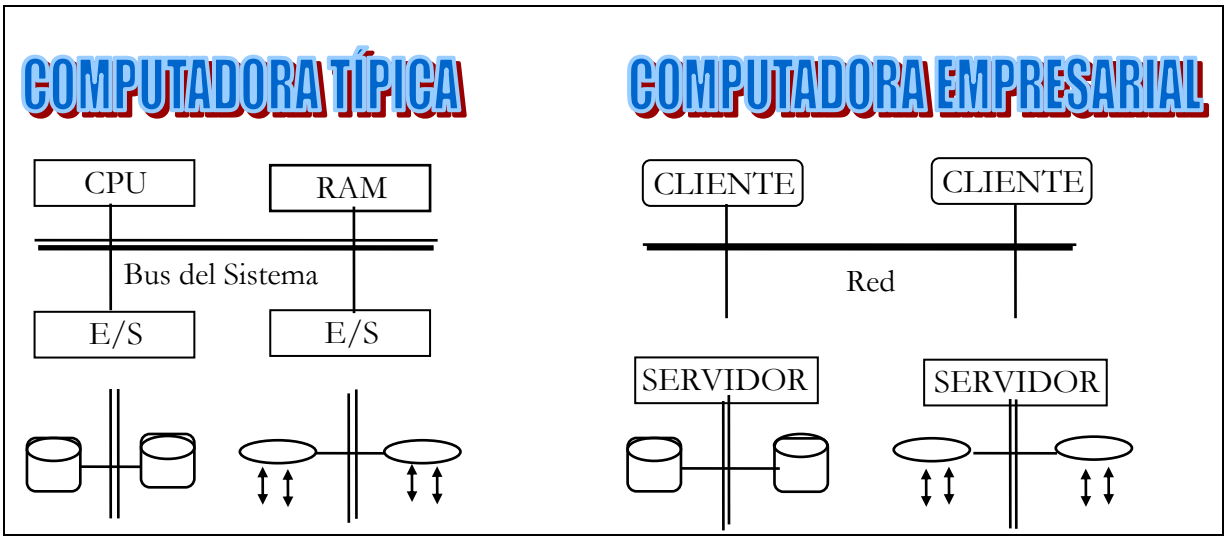

Figura 2.4 Arquitecturas de una Computadora Típica y una Empresarial

Fuente: Adaptado de Marín (1995)

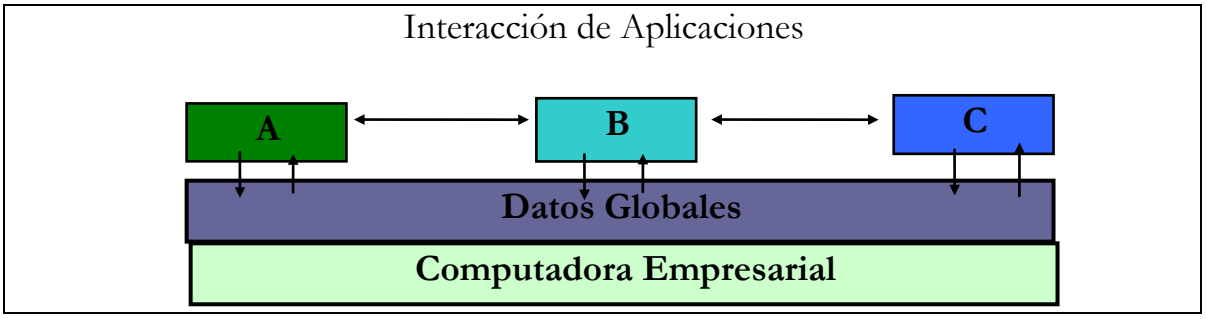

Figura 2.5 Arquitectura de la Aplicación.

Fuente: Adaptado de Marín (1995)

La arquitectura de la Computadora Empresarial abarca más componentes que los de una computadora. En lugar de un sólo CPU, hay muchos sistemas Cliente distribuidos en la organización. Cada Cliente tiene acceso potencial a todos los Servidores en la organización. Esos Servidores interactúan con sus Clientes, extendiendo la funcionalidad del Cliente con los servicios

que proveen. La interacción entre Clientes y Servidores es transparente a los usuarios, que tienen la ilusión de que todo está ocurriendo en el Cliente. Visto de esta perspectiva (Figura 2.5).

El escritorio de un usuario aparecería en la cima de la infraestructura de cómputo empresarial. Los repositorios empresariales de datos (contenidos en los servidores) son accesibles desde las aplicaciones que se están ejecutando en el escritorio. (Marín 1995)

2.3.2 Arquitecturas de Software

En programación (Clements 2003), el término arquitectura era, en 1969, una "estructura conceptual de una computadora…desde el punto de vista del programador", posteriormente se definió como "la especificación completa y detallada de la interfaz de usuario" y se hacía una clara distinción entre arquitectura (qué) e implementación (cómo). El tema de estudio se abocó a la estructura del software, cómo dividirlo y estructurarlo, en lugar de sólo programarlo para que produzca un resultado correcto. El objetivo era obtener un almacén de componentes que se pudieran usar repetidamente y que estos se pudieran compartir. Así, la idea de la arquitectura de software era codificar los componentes comunes de una familia de programas, de tal forma que las decisiones de diseño de alto nivel, inherentes a cada miembro de una familia de programas, no necesitaran ser reinventadas, revalidadas y reescritas.

La definición actual para la arquitectura de software de un programa o sistema de cómputo es "la estructura o estructuras del sistema que comprenden los elementos de software, las propiedades visibles externamente de esos elementos, y las relaciones entre ellos" Las propiedades visibles externamente son las suposiciones que otros elementos pueden hacer de un elemento como los servicios que proporciona, las características de desempeño, el manejo de fallas, el uso de almacenamiento compartido, etc.

La arquitectura es:

1) una abstracción del sistema que omite detalles de los elementos que no afectan cómo se usan, qué otros elementos utilizan, con cuales se relacionan o interactúan.
2) está compuesta por estructuras de distintos tipos, con distintas clases de elementos, relaciones y contextos.
3) todo sistema computacional, con software, tiene una arquitectura de software, aunque ésta no es necesariamente conocida por cualquiera, lo que revela la diferencia entre la arquitectura de un

sistema y su representación, lo cual muestra la importancia de la documentación de la arquitectura.

4) el comportamiento de cada elemento es parte de la arquitectura, en la medida de que ese comportamiento pueda observarse o discernirse desde el punto de vista de otro elemento, y esto es lo que permite a los elementos interactuar unos con otros. (Bass, Clements, Pasman 2003)

Las arquitecturas de software son importantes, según Bass *et al.* (2003), por tres razones:

1. **Comunicación entre los distintos participantes (*stakeholders*) del proceso de desarrollo.** Ya que representan una abstracción de un sistema que se puede usar como base para el entendimiento mutuo, negociación, consenso y comunicación.
2. **Decisiones de diseño anticipadas**. La arquitectura de software expone las primeras decisiones acerca de un sistema, y estos primeros lazos son importantes e influyen sobre el desarrollo del sistema, su implementación y mantenimiento. Es además, el primer punto en el que se pueden analizar las decisiones de diseño que regirán el sistema a construir.
3. **Abstracción transferible de un sistema**. Constituye un modelo relativamente pequeño e intelectualmente comprensible de cómo está estructurado un sistema y de cómo sus elementos trabajan juntos, y este modelo se puede transferir a lo largo de los sistemas. Se puede aplicar a otros sistemas que muestren de atributos y requerimientos funcionales similares, lo que puede promover la reutilización a gran escala.

2.3.2.1 Arquitectura de Software o Arquitectura de Sistemas

Según Bass et al. (2003), cuando se habla de arquitectura de software, se quiere resaltar la naturaleza crucial de las decisiones en cuanto a software, que hace un arquitecto, concernientes a la calidad total del producto. También indica que cuando se diseña la arquitectura de software se tienen en consideración los elementos del sistema computacional completo, principalmente el hardware, las interfaces y la red. Y señala que los ingenieros de software prefieren hablar de arquitectura de software y no de arquitectura de sistema, puesto que la libertad del arquitecto reside en la elección del software, no en la del hardware, ya que esto último no está dentro del ámbito de control del

arquitecto. Ya sea por restricciones tecnológicas o por cuestiones económicas, políticas, legales o por cumplir con los estándares; o que estos factores probablemente cambiarán a través del tiempo[11].

2.3.2.2 Estructuras Arquitectónicas y Vistas

Así como en el área médica, los distintos especialistas, el neurólogo, el cardiólogo, el ortopedista y el dermatólogo tienen una perspectiva (view) diferente de la estructura del cuerpo humano, por que se concentran en un subsistema del mismo. Hay otros, como el psiquiatra, que se encargan de la organización del sistema. Aunque estas perspectivas (vistas) son imágenes distintas y tienen propiedades distintas, todas están relacionadas: juntas describen la arquitectura del cuerpo humano. Lo mismo ocurre en el software, dado que la complejidad ha aumentado, es difícil comprender el sistema en su totalidad. Por lo que se centra la atención en un pequeño número de las estructuras de software del sistema, entonces, para comunicarnos de forma efectiva, debemos señalar claramente qué estructura se está discutiendo en un cierto momento, qué vista se está tomando de la arquitectura. (Bass et al. 2003)

Una vista es "una representación de un conjunto coherente de elementos arquitectónicos, escritos y entendidos por los interesados del sistema". Una estructura es "el conjunto de elementos por si mismo, software y hardware". (Bass et al. 2003)

Las estructuras arquitectónicas, pueden dividirse en (Bass et al. 2003):

- ✓ Estructuras de módulos. Unidades de código estructuradas en módulos.
- ✓ Estructuras de componentes y conectores. Comportamiento del conjunto de elementos (componentes) y sus interacciones (conectores)
- ✓ Estructuras de asignación. Relación con otras estructuras que no son de software como el CPU, el sistema de archivos, las redes, los equipos de desarrollo, etc.

Las vistas principales más comunes, son las presentadas por Kruchten (1995), denominadas "Cuatro más Uno" (*4+1*), las cuales fueron la base conceptual del Proceso Unificado de Rational (Rational Unified Process- RUP), las cuatro vistas son:

[11] Esta renuncia de las arquitecturas de software, y su incapacidad para mantener una arquitectura viable a través del tiempo debido a los factores que señalan, no permitirían garantizar la calidad total del producto, por lo que es ahí donde cobra relevancia nuestro modelo de arquitectura sistémica.

1) Lógica. Los elementos son abstracciones clave, las cuales se manifiestan como objetos o clases de objetos (en el mundo orientado a objetos).
2) Procesos. Esta vista trata la concurrencia y la distribución de la funcionalidad, es la vista de componentes y conectores.
3) Desarrollo. Esta vista muestra la organización de los módulos de software, bibliotecas, subsistemas y unidades de desarrollo. Es una vista de asignación, que relaciona el software con el entorno de desarrollo
4) Física. Esta vista relaciona otros elementos con el procesamiento y los nodos de comunicación, también es una vista de asignación (conocida como vista de despliegue).

En ese mismo año Soni, Nord, y Hofmeister (1995) propusieron otras vistas: la conceptual, interconexión de módulos, ejecución y codificación. Que corresponden a los modelos de módulos, componentes y conectores, y asignación.

La lista de estructuras ha continuado creciendo, sin embargo hay que considerar que uno de los deberes del arquitecto es comprender cómo éstas estructuras conducen a atributos de calidad y entonces, elegir las mejores para esos atributos.

2.3.3 Arquitecturas Empresariales

Las Arquitecturas Empresariales "definen los componentes principales de una organización, y cómo estos componentes funcionan juntos para lograr los objetivos definidos del negocio. Estos componentes incluyen personal, procesos de negocio, tecnología, información financiera y otros recursos" (McGovern, Ambler, Stevens, Lin, Sharan y Jo 2004)

Por empresa (enterprise), The Open Group's Architecture Forum (TOGAF 2003), se refiere a "un grupo de organizaciones que tienen un conjunto de objetivos o fines comunes. En este sentido una empresa puede ser una agencia gubernamental, una compañía completa, una división de una compañía, un departamento, o una cadena de organizaciones geográficamente distantes unidas por un propietario común. El término empresa, en el contexto de arquitectura empresarial, se puede usar para referirse a una empresa completa, incluyendo todos sus sistemas de información, y a un área de competencia específica (dominio) dentro de la empresa, en ambos casos, la arquitectura cruza varios sistemas y grupos funcionales."

Las arquitecturas empresariales están conformadas por dos bloques principales:

a) ***La Plataforma.*** Conjunto de Hardware y Software; y es la base de la Infraestructura Tecnológica, la cual puede representarse a través de una arquitectura de cómputo empresarial, cómo se muestra en el Capítulo 4. La plataforma incluye: equipo y controladores de dispositivos (Hardware); servicios del sistema; repositorio de datos; aplicaciones y clientes (usuarios).

b) ***El Contenido***. Formado por parte de las aplicaciones, los datos e información, que están relacionadas con los procesos de negocio, además de la experiencia y conocimiento de los clientes.

La Figura 2.6 Muestra una arquitectura empresarial típica. La cual da un orden jerárquico, y muestra los elementos principales de interés, dentro del ámbito de las Tecnologías de la Información (TIs), para las organizaciones.

Figura 2.6 Arquitectura Empresarial

Fuente: Marín (2005)

La información en particular, plasmada en esta arquitectura, nos proporciona una visión general de la empresa. Esta visión es importante en las etapas iníciales de desarrollo tecnológico, ya que nos sirve como una herramienta valiosa de comunicación y de negociación.

Actualmente las empresas en consultoría de TI, se limitan casi exclusivamente a atender los problemas de las instituciones en esos dos grandes grupos: las cuestiones tecnológicas (plataforma) y en cierto grado las cuestiones de administración (organización). Lo que da como resultado que la mayoría de los 'soluciones' que proponen, al estar limitadas en cuanto alcance, no cubran con las expectativas que generan, y si lo hacen, es por muy poco tiempo, lo cual conduce a un ciclo interminable de 'actualizaciones' y de nuevos proyectos, ninguno de los cuales puede ser capaz de proporcionar un soporte a las instituciones, ni pueden coadyuvar a la organización efectiva de las mismas, pues se desatienden cuestiones fundamentales, como los aspectos legales (derechos de autor y propiedad industrial), los aspectos político-sociales y los aspectos culturales (éticos y estéticos), además de que no abarcan la organización (el sistema) completa, ni atacan los aspectos relevantes (importantes) que afectan o afectarán a la misma.

2.3.4 Resumen de Arquitecturas

La evolución en el concepto arquitectura radica en la expansión del ámbito al que éstas se refieren. Las arquitecturas de cómputo empresarial establecen y definen la relación e interacción de componentes tecnológicos (hardware, software y telecomunicaciones). Las arquitecturas de software ponen el énfasis en cómo estos elementos se juntan en el ciclo de vida del desarrollo de software e incluyen vistas múltiples de la empresa (para hacer frente a la complejidad). Por último, las arquitecturas empresariales agregan aspectos de la organización y financieros.

Una arquitectura es una abstracción que muestra las interrelaciones entre los elementos (componentes) de un sistema. Permite reutilizar estos elementos, ocultar detalles innecesarios en las primeras fases de desarrollo y sirve como una herramienta de comunicación entre los distintos participantes involucrados (stakeholders) en un proyecto.

En las arquitecturas de software, y en las de cómputo, se renuncia a abarcar el sistema completo, por lo que entra en contradicción con el supuesto de garantizar la calidad total de sus productos. Las arquitecturas empresariales agregan más vistas de la organización. Sin embargo, ambos enfoques no muestran cómo solventar las cuestiones económicas, políticas y legales, ni cómo, mantener una

arquitectura viable a través del tiempo. Es ahí donde nuestro modelo de arquitectura sistémica cobra relevancia.

2.4 Metáforas de la Organización

Morgan (1996), en su libro "Imágenes de la Organización", explora el "impacto de las metáforas y los paradigmas cognitivos en la comprensión del mundo que nos rodea" en dónde tiene como objetivo el "demostrar cómo pueden utilizarse las ideas creativas generadas por las metáforas para crear nuevas formas de comprender la organización" (Morgan 1996)

La premisa básica de Morgan (1996) es que "el empleo de la metáfora implica un modo de pensar y un modo de ver que traspasa el cómo comprendemos nuestro mundo en general […] y que empleamos la metáfora siempre que intentamos comprender un elemento de experiencia en términos de otro". Señala que este proceso de comprensión, no es más que una interpretación de la realidad, y que en el análisis de la organización, varias de estas metáforas se dan por supuesto, ya que frecuentemente se ve a las organizaciones "como si fueran máquinas diseñadas para conseguir determinados objetivos[,] y [se espera] que [éstas,] operen fluida y eficientemente" por lo que se intentan organizar y dirigir de esa forma, cómo máquinas. (Morgan 1996).

Morgan (1996) explora ocho metáforas (imágenes de la organización):

- ✓ ***Las organizaciones como máquinas***. Organizaciones burocráticas, cada parte tiene establecida claramente su rol dentro de la operación del conjunto. Con objetivos preestablecidos por los directivos, con un alto valor de la eficiencia con la que estos objetivos se cumplen. Esta metáfora "descuida el papel fundamental de los individuos en las organizaciones y los diseños que produce son demasiado rígidos para un ambiente volátil" (Jackson 2003)
- ✓ ***Las organizaciones como organismos***. Teoría moderna de la dirección. Centra su atención en la comprensión y gestión de las 'necesidades' de la organización y las relaciones con el entorno. Las organizaciones crecen, se desarrollan, declinan, y mueren. Además se adaptan (pasivamente) a los entornos variables y cambiantes. Las organizaciones como un todo estructurado de partes interrelacionadas. Estas partes funcionan de tal forma que garantizan la supervivencia de la organización. La supervivencia remplaza al logro de objetivos como la razón de ser de la empresa, pero falla ya que olvida que los individuos o

grupos pueden no compartir los objetivos de la organización, además de que esconde los conflictos y los cambios internos (Jackson 2003)

- ✓ ***Las organizaciones como cerebros***. Se centra en la importancia del proceso de la información, al aprendizaje y a la inteligencia. Proporciona un marco de referencia para comprender y evaluar la dirección moderna. Se trata al cerebro como un tipo de computadora que procesa información y como un holograma. Esta metáfora "deriva de la cibernética, resalta el aprendizaje activo en lugar de la adaptación pasiva que caracteriza a la metáfora orgánica. Falla en que no considera a los individuos y sus motivaciones, su capacidad de poder y conflicto, y cómo se establecen los propósitos " (Jackson 2003)
- ✓ ***Las organizaciones como culturas***. La organización se ve como un lugar que conjunta las ideas, los valores, las normas, los rituales y las creencias. Forman significados compartidos que sostienen las organizaciones como realidades sociales. Se dice que esta metáfora "distrae la atención de otros aspectos importantes para el éxito de la organización, como el logro de los objetivos, el diseño de estructuras apropiadas, administración de recursos, y que puede conducir a la manipulación ideológica de los empleados" (Jackson 2003)
- ✓ ***Las organizaciones como sistemas políticos***. Enfoca los distintos intereses, conflictos y juegos de poder, que configuran las actividades de la organización, de acuerdo a reglamentos. "Las posibles relaciones políticas entre los participantes se designan como: unitarias, pluralistas o coercitivas. El abuso de esta perspectiva puede politizar la vida de la organización, no permitiendo enfocarse a otros factores cruciales para la organización" (Jackson 2003). Otra subdivisión aplicada en la relación del poder y las Tecnologías de la información es la propuesta por Jasperson et al. (2002), quienes las agrupan en: racionales, pluralistas, interpretativas y radicales.
- ✓ ***Las organizaciones como prisiones psíquicas***. Las personas están atrapadas por sus propios pensamientos, ideas y creencias, o por preocupaciones originadas en la parte inconsciente de la mente.
- ✓ ***Las organizaciones como flujo de cambio y transformación***. El cambio en la vida social. Las organizaciones como: 1) **Sistemas auto-productores** que se crean ellos mismos según su propia imagen, 2) como fruto de la **retroalimentación positiva o negativa**, y 3) producto de la **lógica dialéctica** por la que cada fenómeno tiende a generar su opuesto. "Los administradores, deben de encontrar ciclos de retroalimentación positiva y negativa, así como, entender los patrones de comportamiento de los sistemas. Los críticos de esta

metáfora dudan de que existan leyes estructurales profundas, que las organizaciones sociales deban obedecer, y se preocupan por el poder desmedido que se de a los expertos para que convenzan a otros que estas leyes existen" (Jackson 2003)

- ✓ ***Las organizaciones como instrumentos de dominación***. El foco está en los aspectos potenciales de explotación, de cómo las organizaciones utilizan a sus empleados, los bienes nacionales y mundiales, para conseguir sus propios fines, y como la esencia de la organización descansa en un proceso de dominación dónde ciertas personas imponen su voluntad sobre otras. Se critica que esta metáfora sea demasiado ideológica y politizada, y que se vea a quienes la defienden como "guardianes de lo 'políticamente correcto'" (Jackson 2003)

Alversson y Deetz (1996 en Jackson 2003) proponen una novena metáfora:

- ✓ ***Las organizaciones como carnavales***. En un carnaval el orden se suspende y se anima la creatividad, la diversidad y la ambivalencia, y es un lugar para divertirse. Se critica los aspectos 'irracionales' de este enfoque.

Las metáforas son una vista o imagen de la organización. Éstas ayudan a comprender las organizaciones, las interacciones entre sus participantes y el entorno, sin tener que entrar en detalles. Morgan (1996) señala que si verdaderamente se desea comprender una organización, hay que considerar que las organizaciones son complejas, ambiguas y paradójicas. Empleando las metáforas para entender la organización no se precisa de memorizar teorías complejas o largas listas de conceptos abstractos. Sin embargo, hay que considerar que pueden existir muchas interpretaciones, todas ellas válidas, de una organización.

2.5 Ingeniería de Dominio

La Ingeniería de Dominio surge como una alternativa para la reutilización de componentes de software y evoluciona a ingeniería de líneas de productos (Product Line Engineering) cuando se incluye además del software cualquier activo reutilizable de la organización, principalmente el capital intelectual, los conocimientos de las personas y los productos que generan como software, manuales, procedimientos, etc. La reutilización de software (software reuse) es la utilización de componentes de software o de conocimientos existentes para construir software nuevo (Frakes & Kyo, 2005).

La Ingeniería de Dominio es el proceso para crear competencias en la ingeniería de la aplicación para una familia de sistemas similares, cubre todas las actividades para construir activos de software, estas actividades son: identificar uno o más dominios, capturar las diferencias dentro de un dominio (análisis de dominio), construir un diseño adaptable (diseño del dominio), y definir los mecanismos para trasladar requerimientos en sistemas, creados con componentes reutilizables (implementación del dominio). Los productos o activos de software, de estas actividades son los modelos del dominio, modelos de diseño, lenguajes específicos del dominio (domain-specific languages DSL), generadores de código y componentes de código (SEI 2007). Ver Figura 2.7

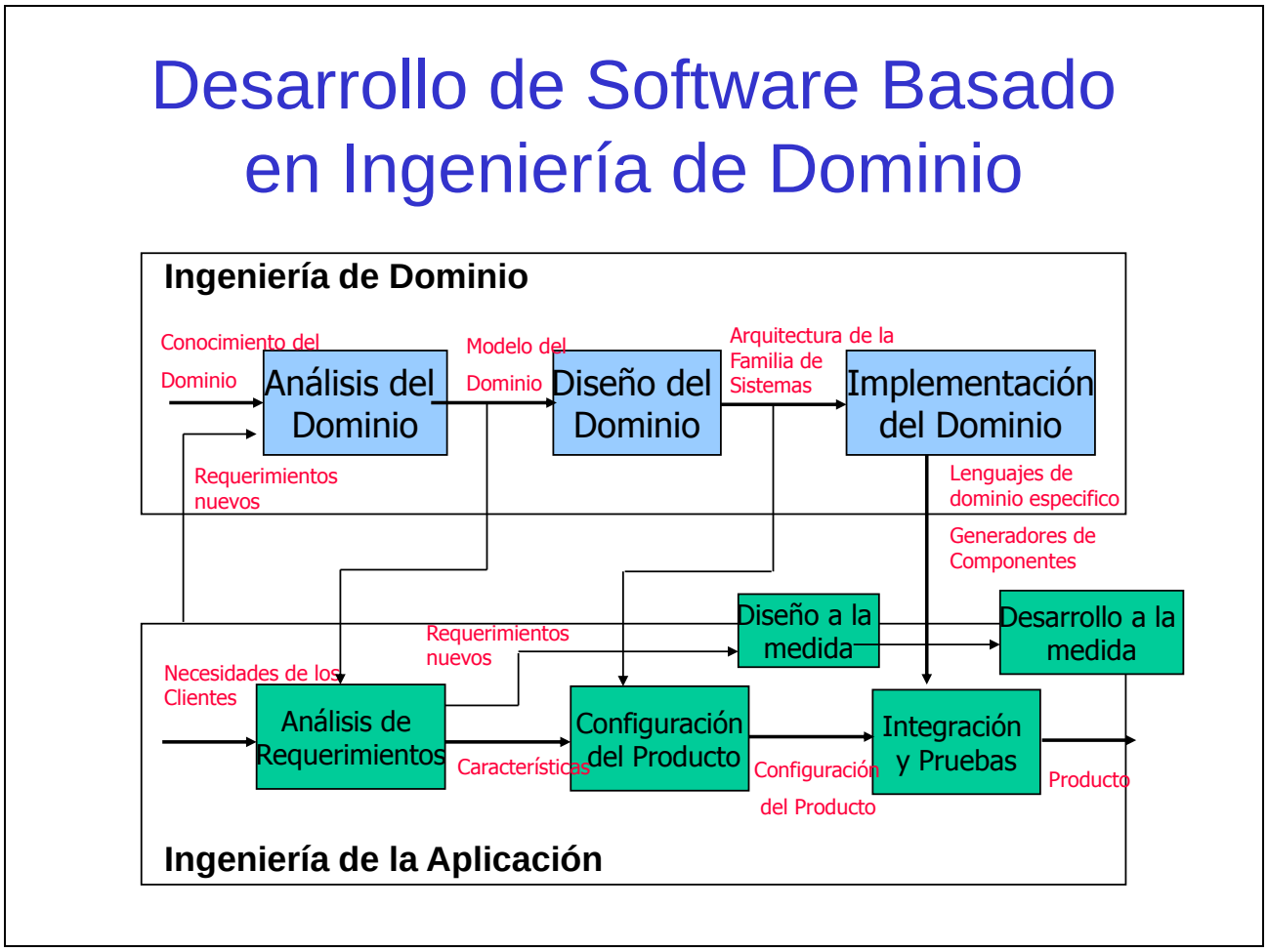

Figura 2.7 Desarrollo de Software Basado en Ingeniería de Dominio.

Fuente: Adaptado de SEI (2007)

Las tres dimensiones (García, Barras, Lagunas, Marqués 2002) en las que se pueden descomponer los conceptos involucrados en las líneas de productos son:

1. Arquitectura, componente y sistema.
2. Negocio, organización, proceso y tecnología.
3. Desarrollo, instanciación y evolución.

La arquitectura de software es el principal *activo* de la línea de productos, ya que abarca todas las características (componentes) que comparten todos los productos (sistemas) de la organización.
Los principales activos reutilizables (SEI 2008a) comprenden:

- Los requerimientos y modelos del dominio
- La arquitectura de software que los productos compartirán
- Los componentes de software
- La documentación del diseño
- Los modelos de desempeño
- Los planes y casos de prueba
- Los procesos para la utilización de los activos
- Capacitación (relativa a la línea de productos)
- Los procesos de administración técnica

Estos activos no están presentes en cada línea de productos, con la excepción de la arquitectura de software, de ahí su importancia, y es la razón por la cual forma parte importante de nuestra propuesta.

2.6 Ingeniería de Software Basada en Evidencias

Kitchenham, Dyba, y Jorgensen (2004) proponen que la Ingeniería de Software sea análoga a la práctica médica y que se use el enfoque basado en evidencias dado el éxito que ha tenido este en la práctica médica en los años 80's y 90's. A su propuesta la denominaron Ingeniería de Software Basada en Evidencias (Evidence-Based Software Engineering - EBSE)

La Ingeniería de Software Basada en evidencias consta de cinco pasos (Dyba, Kitchenham, & Jorgensen, 2005):

1. Convertir un problema relevante o necesidad de información en una pregunta que pueda ser respondida.
2. Buscar en la literatura la mejor evidencia para responder la pregunta.

3. Evaluar críticamente la evidencia por su validez, impacto y aplicación
4. Integrar la evaluación de la evidencia con la experiencia práctica y los valores del cliente y las circunstancias para tomar una decisión práctica.
5. Evaluar el desempeño y buscar formas de mejorarlo

Mucho de lo que se necesita para practicar EBSE ya existe en la mejora de procesos de software (software process improvement- SPI), por ejemplo (Dyba et al., 2005):

1. Identificar un problema.
2. Proponer una tecnología o procedimiento para solucionar el problema.
3. Evaluar la tecnología propuesta en un proyecto piloto.
4. Si la tecnología es adecuada, adoptarla e implementarla.
5. Monitorear la organización después de implementar la nueva tecnología.
6. Regresar al paso 1

Capítulo 3. Modelo Propuesto

COVER ILLUSTRATION: DIRK HAGNER
IEEE-CS Software Vol 19 No.4

En este Capítulo se desarrolla el modelo, el cual es heurístico, interpretativo y contextual. Y se aplica en el proceso de desarrollo de software basado en Ingeniería de Dominio.

CAPÍTULO 3. MODELO PROPUESTO

> "En el racionalismo contemporáneo hay una paradoja esquizofrénica en la concepción del progreso científico: por un lado, se reconoce que la creatividad, la pasión por la innovación y el descubrimiento son factores esenciales del cambio progresivo. Pero, por otro lado, se considera que estos componentes heurísticos son eminentemente irracionales, pues carecen de todo rigor metodológico."
>
> Velasco (2000)

3.1 Modelo Propuesto para la Ingeniería de Software: Heurístico, Interpretativo y Contextual

En este apartado se desarrolla el modelo, el cual es heurístico, interpretativo y contextual. Y se aplica en el proceso de desarrollo de software basado en Ingeniería de Dominio.

Parte de la propuesta de Ingeniería de Software Basada en Evidencias (Evidence-Based Software Engineering - EBSE) de Kitchenham, Dyba, y Jorgensen (2004). Se complementa con la con la heurística interpretativa definida en el Capítulo 2 la cual está basada en la Hermenéutica Analógica-icónica del Dr. Mauricio Beuchot (2005; 2005a).

Con estos elementos, se pasa de los enfoques sistemático y clínico al enfoque sistémico, y se pueden ponderar correctamente el entorno y los aspectos y procesos culturales, sociales y éticos.

El modelo propuesto para el desarrollo de software es el siguiente:

1. Usar la Metáfora Sistémica
2. Elegir la metáfora de la organización o elaborar una nueva.
3. Formular la problemática
4. Contrastar la situación actual de la organización con la situación deseada.
5. Validar que los objetivos, visión, misión y estrategias de la organización sean congruentes con los puntos 1 al 4.
6. Ingeniería de Dominio e Ingeniería de la Aplicación
    a. Diseñar e implementar la arquitectura de software
7. Establecer los aspectos éticos, legales y políticos aplicables

8. Diseñar la arquitectura sistémica (holística, heurística, crítica y ética) aplicable al dominio. Los elementos se conforman de los resultados de los puntos anteriores (1 a 8)
9. Implementar
10. Evaluar
11. Retroalimentar y reiniciar proceso

La representación gráfica del modelo se ve en la Figura 3.1.

3.2 Paso 1: Usar la Metáfora Sistémica

El modelo elaborado deberá cumplir los imperativos sistémicos mencionados previamente, deberá ser: holístico, heurístico y crítico. De tal forma que nuestra metáfora S*istémica,* deberá representar y cubrir los imperativos sistémicos (ver Tabla 3.1).

Imperativos Sistémicos	Metáfora Sistémica
Holística	Sistema Intencional, Sociocultural, con cinco elementos básicos: hombre, naturaleza, sociedad, entorno y redes. Debe satisfacer las necesidades y requerimientos (económicos, sociales, políticos, tecnológicos y culturales – éticas y estéticas) de las distintas fuerzas interesadas (usuarios y otros). Tiene un equilibrio dinámico.
Heurística	Sigue un desarrollo y evolución heurística, con ciertos patrones de evolución a través del tiempo: nacimiento, desarrollo y extinción; el hombre puede intervenir en el proceso.
Relevante / Ético	Fin último, la felicidad del hombre y mantenerlo seguro. Respeta la tradición y la cultura.

Tabla 3.1 La Metáfora Sistémica

Fuente: Elaboración propia

Con base en esta metáfora se define el proceso desarrollo de software Sistémico como: un sistema evolutivo que consta de cinco elementos básicos: hombre, naturaleza, sociedad, contexto y redes; el cual, debe de satisfacer los requerimientos económicos, sociales, políticos, tecnológicos y culturales de los distintos interesados. El sistema tiene un equilibrio dinámico y es afectado por las intervenciones que se le hagan. Su finalidad es la felicidad y seguridad de la comunidad, respetando la tradición, la cultura y a los Otros. El cual requiere además de un trabajo poiético (transformador-creador).

Modelo de Desarrollo de Software Propuesto

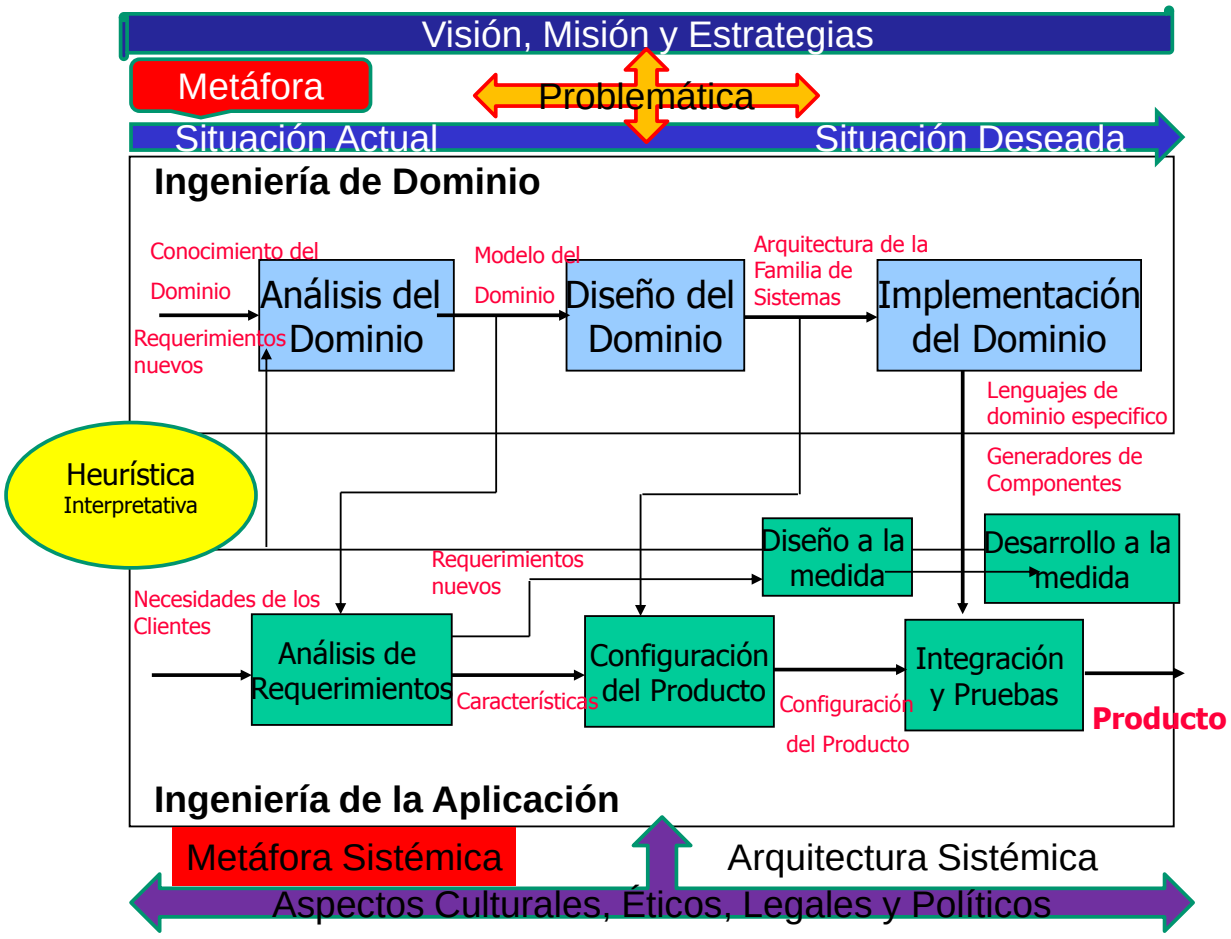

Figura 3.1 Modelo de Desarrollo de Software Propuesto.

Fuente: Elaboración propia

3.3 Paso 2. Elegir la Metáfora de la Organización o Elaborar una Nueva.

Para elegir la metáfora, se considera a la organización en la cual se desarrolla software como:

a. **Máquina**. Si la organización es rígida y burocrática, con objetivos y roles preestablecidos.
b. **Organismo**. Si funciona como un todo estructurado de partes interrelacionadas que garantizan la supervivencia de la organización.
c. **Cerebro**. Si se centra en los procesos de la información, el aprendizaje y la inteligencia.

d. **Cultura**. Si se comparten significados y dan prioridad a las ideas, los valores, las normas, los rituales y las creencias.
e. **Sistema político**. Si se enfocan en los distintos intereses, conflictos y juegos de poder, que configuran las actividades organizacionales, de acuerdo a reglamentos.
f. **Prisión psíquica**. Si consideran que las personas están atrapadas por sus propios pensamientos, ideas y creencias, o por preocupaciones originadas en la parte inconsciente de la mente.
g. **Flujo de cambio y transformación**. Si se basan en leyes estructurales y patrones de comportamiento de los sistemas.
h. **Instrumento de dominación**. Si utilizan a sus empleados, los bienes nacionales y mundiales, para conseguir sus propios fines. Y si imponen su voluntad sobre otras personas e instituciones, explotándolas.
i. **Carnaval**. Si el orden se suspende y se anima la creatividad, la diversidad y la ambivalencia, y es un lugar para divertirse.

Para construir una nueva metáfora de la organización se puede usar la guía anterior y poner un ejemplo concreto de la misma. Sino, hay que partir de los enfoques de desarrollo de software: rígidos (unívocos), arbitrarios (equívocos) y heurísticos (Figura 3.2). Éstos también son aplicables a las organizaciones. Sus características se muestran en la Tabla 3.2

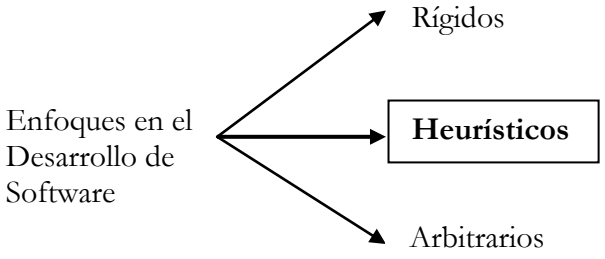

Figura 3.2 Enfoques de Desarrollo de Software

Fuente: Elaboración propia

Características de los Modelos Rígidos y de los Arbitrarios

Modelos Características	Rígidos	Arbitrarios
Organización	Estrategia, Estructura, Procesos y Prácticas bien definidas. Finanzas fuertes y personal competente	Sin dirección ni estructura
Paradigma empleado	Sistemático	Por casos (casuístico)
Enfoque	Por áreas (disciplinar) / Liderazgo en Mercados, costos y publicidad (Mercantil y utilitario) / Ventaja competitiva	Oportunista y Disciplinar /Utilitario/ No Competitivo (aunque puede serlo sólo en precio)
Productos y Servicios	De calidad (Eficientes) / Potencialmente Innovadores, ventajas de costo	Cumplen los requisitos mínimos (Deficientes en otros factores) / Tradicionales, costos altos

Tabla 3.2 Características de los Modelos Rígidos y de los Arbitrarios
Fuente: Elaboración propia

3.4 Paso 3: Formular la problemática

Ackoff (2001) define problemática como "un conjunto interactivo de amenazas y oportunidades, los cuales forman un sistema de problemas (mess)". La formulación de la problemática es un **diagnóstico** que determina cómo se destruiría la organización de continuar con su comportamiento actual, esto es, si la organización no puede adaptarse a un entorno volátil, aún en el caso que este esté predicho a la perfección.

La formulación de la problemática abarca la preparación de:

a) ***un análisis del sistema***, una descripción detallada de cómo opera el sistema actualmente (la Tabla 3.2 sirve como referencia);

b) ***un análisis de obstrucciones***, identificación de las características y propiedades de la organización que obstruyen su progreso (se puede hacer uso de la técnica FODA-Fortalezas, Oportunidades, Amenazas y Debilidades, ver Tabla 3.3); y

c) ***un escenario de referencia***, una descripción de cómo y por qué se destruiría la organización por sí sola, si los supuestos hechos fueran correctos. (Este escenario es una síntesis de lo aprendido en (a), (b) y (c))

Oportunidades y Amenazas

Factor	Oportunidades	Amenazas
Entorno	Cambio (Si logra aprovechar su ventaja)	Cambio (Si no logra aprovechar su ventaja)
Tecnología	Crecimiento de la capacidad de los distintos equipos	Incompatibilidad de aplicaciones y protocolos
Economía	Crecimiento	Crisis recurrentes y espasmódicas
Mercado TI (Productos y Servicios)	Mercado en expansión constante	Monopolios, patentes y licenciamientos

Tabla 3.3 Amenazas y Oportunidades de las Empresas de Consultoría en TI
Fuente: Elaboración propia

Escenario de Referencia del Desarrollo de Software	
Cultural	Se fomenta una sociedad neoliberal y competitiva (posmoderna)
Socio-Político	Se fomentan monopolios y dependencia tecnológica
Legal	Licencias que restringen el uso, copia, modificación y distribución de software. Patentes de software en algunos países que limitan el uso de ideas, procesos y algoritmos
Económico	Costos de desarrollo e insumos mayores
Tecnológico	Tecnología cerrada, sólo se puede "personalizar", por lo que no puede aprender

Tabla 3.4 Escenario de Referencia del Desarrollo de Software
Fuente: Elaboración propia

3.5 Paso 4. Contrastar la Situación Actual de la Organización con la Situación Deseada

Planear es importante pues permite "diseñar un futuro deseado y nos proporciona los medios efectivos para realizarlo. Es un instrumento usado por el sabio. La sabiduría es la facultad de vislumbrar con mucha anticipación las consecuencias de las acciones actuales, es estar dispuesto a sacrificar los logros a corto plazo a cambio de mayores beneficios a largo plazo, y la habilidad de controlar lo que es controlable y no desgastarse con lo que no es. Por lo tanto, la esencia de la sabiduría es la preocupación por el futuro, aunque no es el mismo tipo de interés en el futuro que tienen los adivinos, quienes sólo tratan de predecirlo, la intención del hombre sabio es controlarlo. (Ackoff 1972, 1999)

Obtener control sobre el futuro. Se basa en la creencia de que el futuro de una organización depende de cómo se construya ese futuro desde el momento presente y de lo que se haga para alcanzarlo, por lo que esta planeación consiste en el diseño de un futuro deseable y en la selección o invención de las formas para producirlo tan fielmente como sea posible (Ackoff 1999). Este futuro se crea, cerrando la brecha entre el lugar que se encuentra la organización en cualquier momento y a dónde le gustaría estar. (Ackoff 2001)

3.6 Paso 5. Validar que los Objetivos, Visión, Misión y Estrategias de la Organización Sean Congruentes con los Puntos 1 al 4

En los pasos anteriores pueden encontrarse inconsistencias entre aquello que está planeado y los procesos que se desarrollan en la organización. Además, puede darse el caso que la planeación no lleve a la organización a la situación deseada. Por lo que en este paso se revisa la problemática de la organización con los nuevos elementos que nos brindan los pasos 4 y 5, y realizan ajustes tanto a la planeación (administración) de la organización como a sus procesos.

3.7 Paso 6: Ingeniería de Dominio y de la Aplicación

Las tareas y los productos de la Ingeniería de Dominio son los siguientes:

Actividades	***Tareas***	***Productos***
Análisis de Dominio	Identificar uno o más dominios, capturar las diferencias dentro de un dominio	Modelos del dominio
Diseño del Dominio	Construir un diseño adaptable	Modelos de diseño
Implementación del Dominio	Definir los mecanismos para trasladar requerimientos en sistemas, creados con componentes reutilizables.	Lenguajes específicos del dominio (domain-specific languages DSL), generadores de código y componentes de código

Tabla 3.5. Actividades, Tareas y Productos del Desarrollo de Software
Elaboración Propia Basado en SEI (2007)

Se evalúan y definen los mecanismos para trasladar los requerimientos en sistemas, los cuales son creados con componentes reutilizables (implementación del dominio). Evalúa la viabilidad en la implementación de los lenguajes específicos del dominio (domain-specific languages DSL), de los generadores de código y de los componentes de código.

Una parte fundamental de la Ingeniería de Dominio es la Arquitectura de software. La Figura 3.3 muestra un modelo de referencia que puede servir como punto de partida para desarrollar la arquitectura.

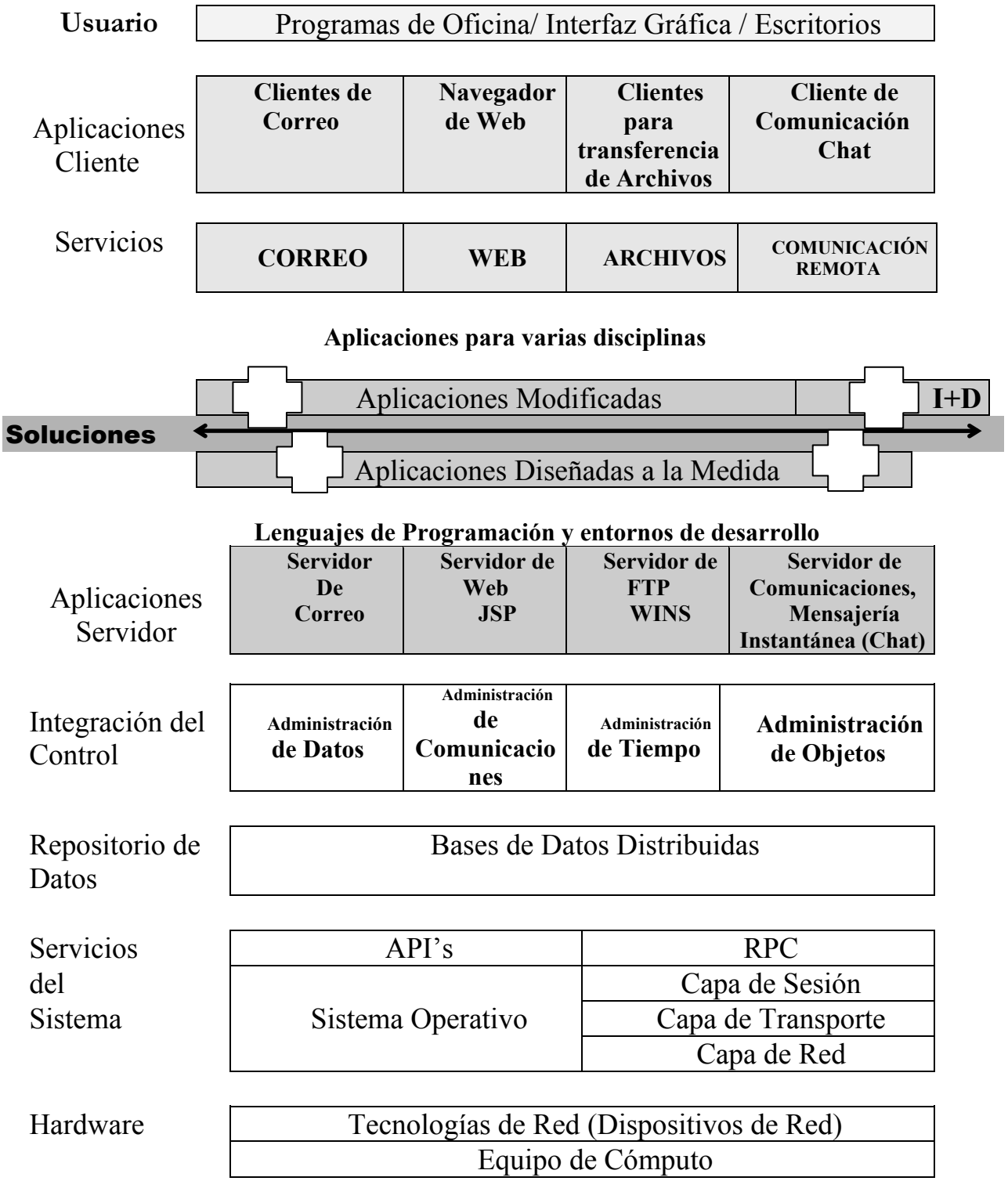

Figura 3.3 Modelo de Arquitectura de Software

Fuente: Marín (2005)

La Figura 3.4 muestra un ejemplo de una arquitectura de desarrollo que usa Software Libre.

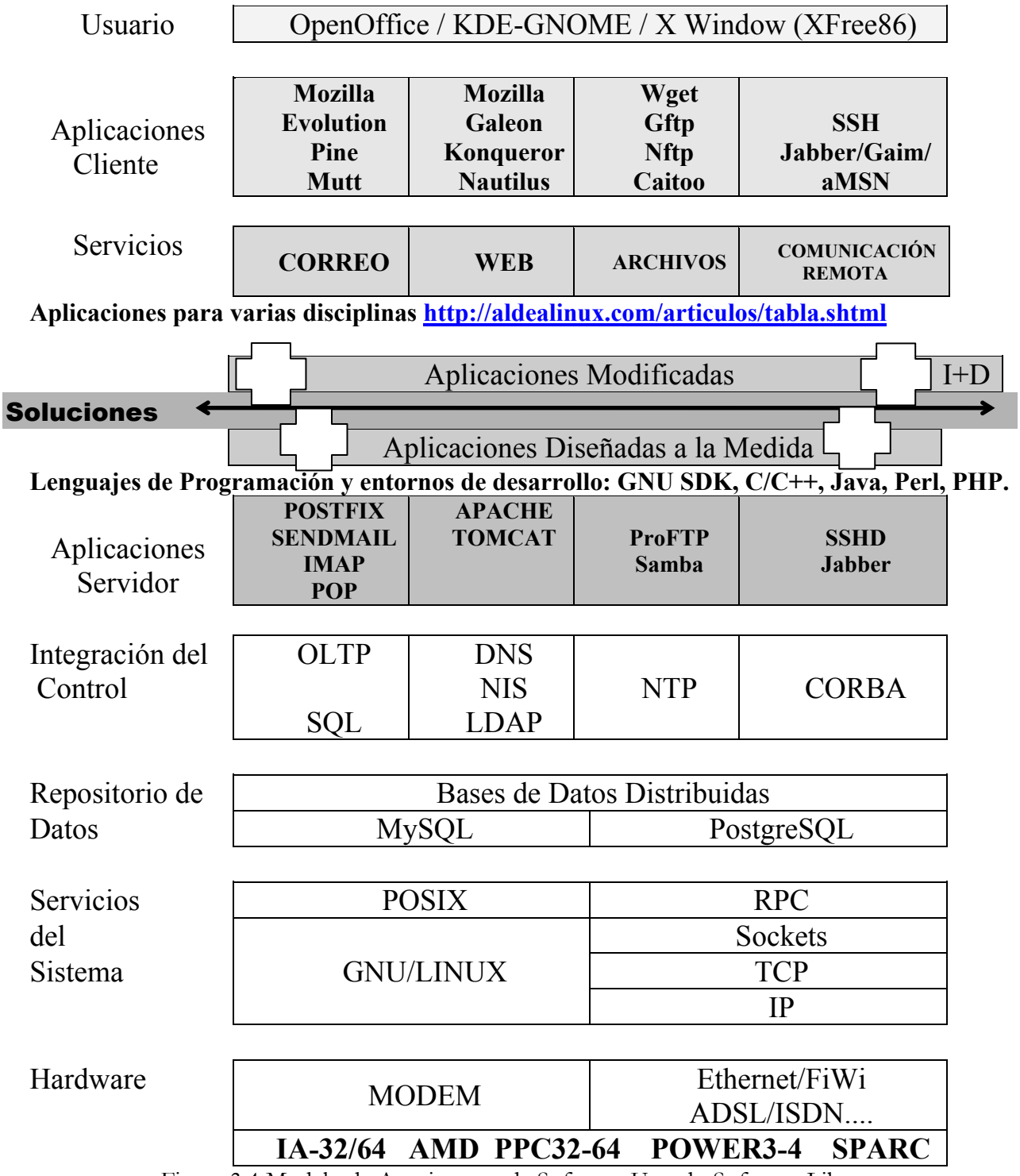

Figura 3.4 Modelo de Arquitectura de Software Usando Software Libre

Fuente: Marín (2005)

3.8 Paso 7: Establecer los Aspectos Éticos, Legales y Políticos Aplicables

Al sintetizar los elementos que se revisaron en el Capítulo 2: la Ética, los Códigos de Ética, los imperativos sistémicos y la conciencia creadora, se pone de manifiesto que los ingenieros de software deben ser competentes por sus aptitudes (formación técnica), líderes por sus actitudes (formación crítica) y con sentido humano por su discernimiento de fines y medios (formación ética). (Mora 2003)

El contar con un código de ética profesional acorde a nuestra área profesional es importante, ya que alentará o restringirá ciertos tipos de comportamiento dentro y fuera de las organizaciones, y éstos afectan su viabilidad.

En el mundo profesional nos enfrentamos con varios problemas, y diversas situaciones que rebasan el ámbito de los códigos de práctica profesional, o códigos de etiqueta, como la falta de equidad, justicia, dignidad.

Como se analizó en el Capítulo 2 varios códigos de ética profesional son en realidad códigos de etiqueta, ya que no consideran aquello que nos confronta, nos cuestiona y nos responsabiliza para forjarnos un buen carácter y crecer como personas; para lograr con ello la primer finalidad humana: la felicidad, la vida feliz. Ser justos y felices, permaneciendo humanamente íntegros. Después de indicar la diferencia entre los códigos de ética y los códigos de práctica profesional, se presentó el código profesional de la ACM/IEEE-CS, el cual pretende ser un código de ética y práctica profesional, y dado su alcance y limitaciones se concluye que sólo es un código de etiqueta (de práctica profesional). Se presentó, posteriormente el "Código de Ética Profesional del Ingeniero Mexicano", y éste sí contiene los aspectos relevantes de un código de ética.

Una vez que establecido como se conforma un código de ética, se propuso la aplicación de la Metáfora Sistémica en las organizaciones tendientes a generar una transformación social. Esta transformación es importante, ya que en el ámbito posmoderno no hay sensibilidad para el deber ni para la solidaridad, más bien se tiende al egoísmo y al hedonismo que lleva al indiferentismo. Se mostró además que la hermenéutica tiene un papel importante para la ética, ya que se requiere de una interpretación del hombre y de la sociedad, es decir, de la cultura. En la actualidad la ética vuelve a conectarse con varios aspectos de los cuales se había desconectado. La ética fue separada de la política, ya que estorbaba, pero hay voces que se alzan para llamar nuestra atención, indicándonos que una

sociedad justa funciona mejor. La ética fue separada de la economía, pero hoy se afirma que la economía sirve mejor a la sociedad si está regida por directrices morales. (Beuchot 2004)

El papel de los ingenieros de software y de sistemas dentro de la sociedad es de una gran importancia, acuden a ellos personas y organizaciones de todo tipo, con problemas diversos. Sus actos deben de guiarse bajo un comportamiento ético, buscando el bien común: el de sus clientes y el bien propio. La complejidad y los distintos intereses de los participantes en los proyectos de desarrollo de software requieren de habilidades nuevas. Por último, debemos de estar atentos ya que "la Sociedad adora a sus nuevos ídolos: la productividad, la competitividad y la competencia; está limitada por tremendas desigualdades, requiere de obediencia –lo cual coarta la libertad, la dignidad y a veces la justicia, lo que nos desalienta y nos lleva a la renuncia de rediseñar la Sociedad y darle solución a la problemática de ésta". (Sánchez 2003)

También, en el Capítulo 2 se revisa la relación que guardan los asuntos éticos y morales con los preceptos jurídicos. Las normas que de ellas emanan guían de cierta forma nuestro comportamiento, sin embargo, hay que tomar en cuenta otro tipo de regulación, la cual se da de manera silenciosa en las organizaciones y en los gobiernos, ésta es la regulación por medio de las Arquitecturas. Como se vio, las decisiones de arquitectura son la base para el desarrollo posterior de sistemas, las cuales formarán estándares de facto (y en otros casos estándares de jure) en las organizaciones. Si no se está consciente de ello se estará atado a un proveedor de servicios de cómputo, a un tipo de sistema o plataforma en particular, lo cual puede fomentar monopolios, aumentar el costo de los insumos, y generar dependencia a largo plazo.

La regulación vía arquitectura (de software, de cómputo, empresarial, etc.) es más poderosa que los otros tipos de regulación, esto es, las leyes, el mercado y las normas sociales, como lo apunta Kiyoshi Tsuru (Cámara de Senadores 2003), ya que la Arquitectura me puede permitir o negar realizar alguna acción, sin que yo como usuario tenga el poder de evitarlo, todo ello por elecciones que alguien realizó con anterioridad, la mayoría de las veces sin una discusión abierta entre todos los interesados o afectados. Y además, para restringir o constreñir una determinada arquitectura, los gobiernos y las organizaciones se sirven de las leyes o reglamentos para obligar o limitar nuestras elecciones.

Por todo ello, es necesario que en el desarrollo de políticas dentro y fuera de la organización, se tome en consideración que los elementos con los que construyan sus distintas arquitecturas, ya que sus

estructuras, limitarán o facilitarán el desarrollo de cierto tipo de sistemas, por lo que se requiere que las elecciones que se realicen en el proceso del diseño de la arquitectura se discutan y acuerden, tomando en consideración a los diferentes interesados del sistema.

El esfuerzo es doble, si se considera que en primera instancia, la teoría siempre va detrás de la práctica (Checkland y Hollwel 1998), y la Ley se queda rezagada varios pasos atrás. Por lo que se tiene que encontrar la forma para desarrollar tecnología de forma viable y que la Ley fomente este desarrollo.

3.8.1 Ejemplo de Aplicación de los Códigos de práctica Profesional en la Ingeniería de Software

Para explicar lo anterior se toma un caso de los cinco que proponen Naveda y Seidman (2006) en su guía de estudio para los profesionales del software. Para el Apartado de ética (B), del área prácticas de negocios e ingeniería económica (I).

El primer ejemplo. 1 (I.B). Ponen el siguiente caso:

Juana está muy complacida con el trabajo que un proveedor ha hecho para su empresa y lo ha recomendado a varias de sus compañías asociadas. Queriendo mostrar su gratitud, el proveedor ha ofrecido actualizarle la red de su casa con un descuento. Lo anterior no sería un problema si:

a) El descuento es parte de una promoción ofrecida para todo público.
b) El proveedor firma un contrato con Juana.
c) El Proveedor ha realizado trabajos para otros en la empresa.
d) Si al jefe de Juana le han hecho el mismo trabajo.

La respuesta que dan es **a**

Y para la explicación nos remiten al los puntos 4.04 y 4.05 del código de ética que elaboró el comité conjunto de la Sociedad de Cómputo de la IEEE (IEEE-CS) y la ACM (IEEE-CS/ACM 1999). Estos señalan que los ingenieros de software:

> 4.04. No deberán involucrarse en prácticas financieras fraudulentas tal como corrupción, facturación doble u otras prácticas financieras impropias.
>
> 4.05. Expondrán a todas las partes involucradas aquellos conflictos de interés que no puedan evitarse o evadirse razonablemente.

Los restantes cuatro casos también remiten al citado Código.

En el caso de México, algunas de estas prácticas son castigadas penal y civilmente. Y en el caso de los Funcionarios Públicos tienen prohibido aceptar regalos con valor superior a 10 días de salario mínimo vigente en el Distrito Federal. Art. 88 de LEY Federal de Responsabilidades de los Servidores Públicos (Congreso de la Unión 2003a).

Las prácticas prohibidas de los códigos de etiqueta, se pueden tipificar como delitos. No ocurre de la misma forma con los Códigos de Ética, ya que por ejemplo no se puede obligar a nadie a ser solidario, a compartir, a ser fraterno, etc.

3.8.2 Ejemplo de Aplicación de Las Leyes de Derechos de Autor y de Propiedad Industrial en la Ingeniería de Software

Del libro antes referido, se toma el caso 6 (I.C) del apartado práctica profesional:

Recientemente has creado un nuevo producto de software, el cual aplica un método único. Quieres proteger este método único.

Pregunta: De acuerdo al escenario anterior, ¿qué se debe de obtener?

a) Un Copyright (derechos de autor)
b) Una Patente
c) Registrar el producto
d) Una Marca Registrada (Trademark)

La respuesta que nos dan es **b (Una patente)**

Y nos remiten a la literatura legal estadounidense. Y efectivamente en Estados Unidos de América y en otros países sí se puede patentar software, algoritmos, procedimientos y modelos de negocios; en México no.

3.9 Paso 8: La Arquitectura Sistémica

Al integrar los aspectos culturales, éticos, legales y políticos, con la arquitectura de software, se conforma la Arquitectura Sistémica. La cual está representada de forma gráfica en la Figura 3.5:

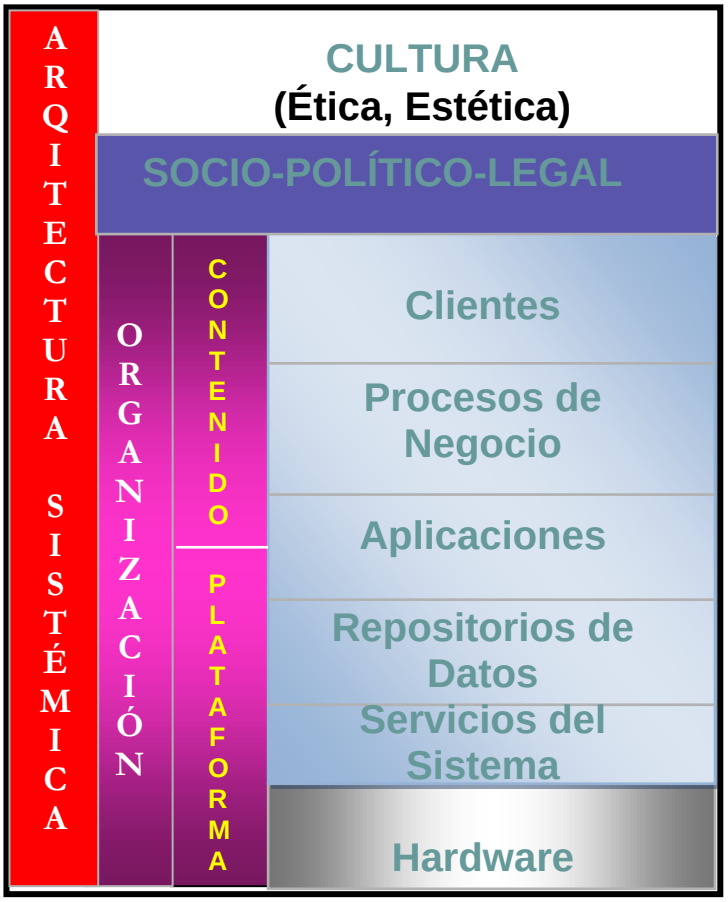

Figura 3.5 La Arquitectura Sistémica

La Arquitectura Sistémica comprende:

1) Los componentes tecnológicos de la organización (representados en la Figura 3.5 por la plataforma y el contenido), obtenidos a partir de las Arquitecturas de Cómputo Empresarial;
2) Los aspectos económicos, sociales y políticos, necesarios para el crecimiento y desarrollo de las organizaciones, en conjunción con los aspectos legales y;
3) Los aspectos culturales (éticos y estéticos), bajo un enfoque filosófico y sistémico.

Los últimos pasos del modelo son:

Paso 9: Implementar

Paso 10: Evaluar

Paso 11: Retroalimentar y reiniciar proceso

3.10 Aplicación del Modelo Propuesto con Otros Enfoques

El modelo propuesto se puede aplicar en otros enfoques de desarrollo emergentes como MDA (Model Driven Architecture- arquitectura dirigida por modelos) (OMG 2008). Figura 3.6.

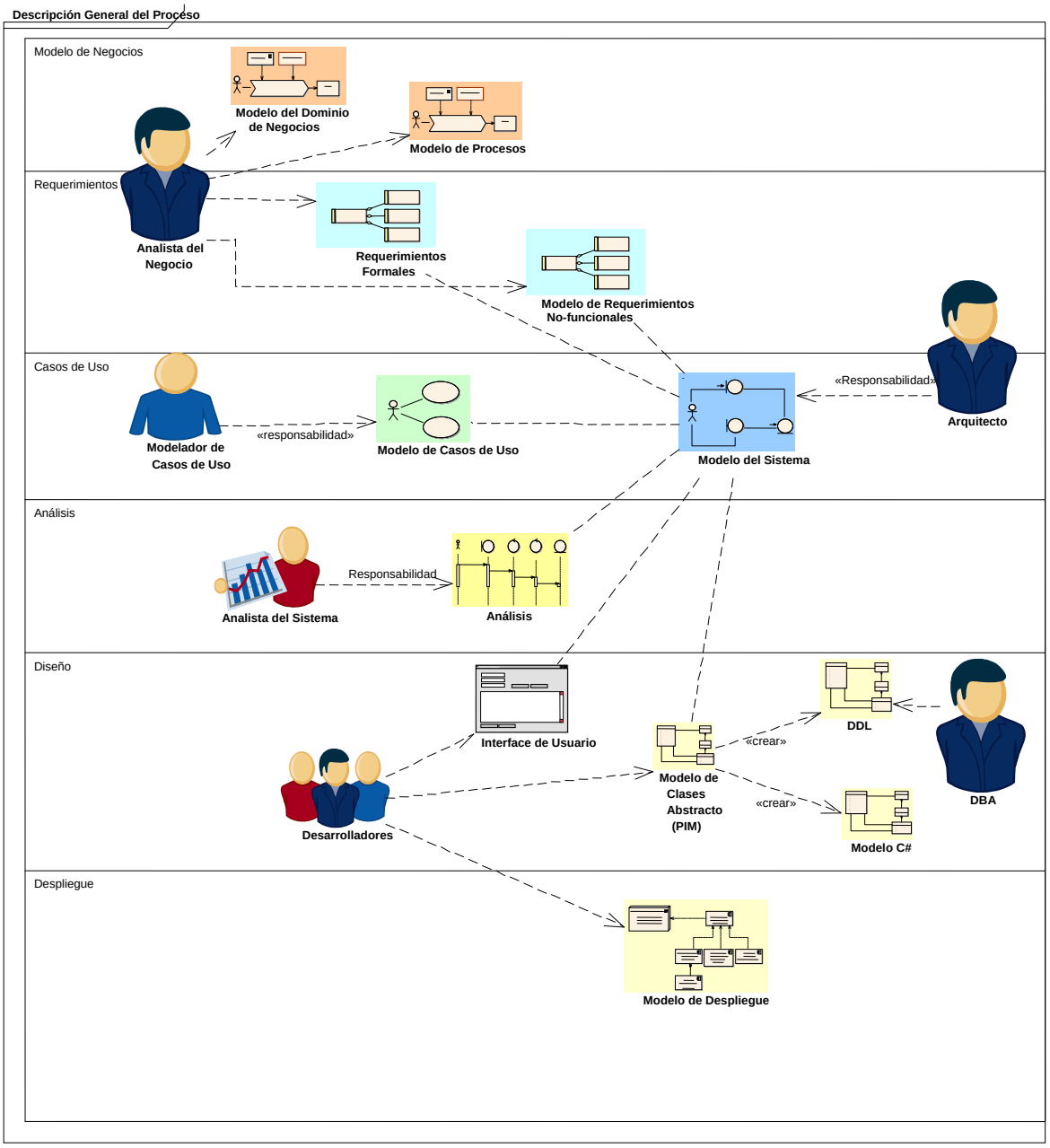

Figura 3.6 Descripción General del Proceso de Desarrollo de Software y los Modelos Resultantes

Fuente: Adaptado de Sparx (2005)

Capítulo 4. Resultados

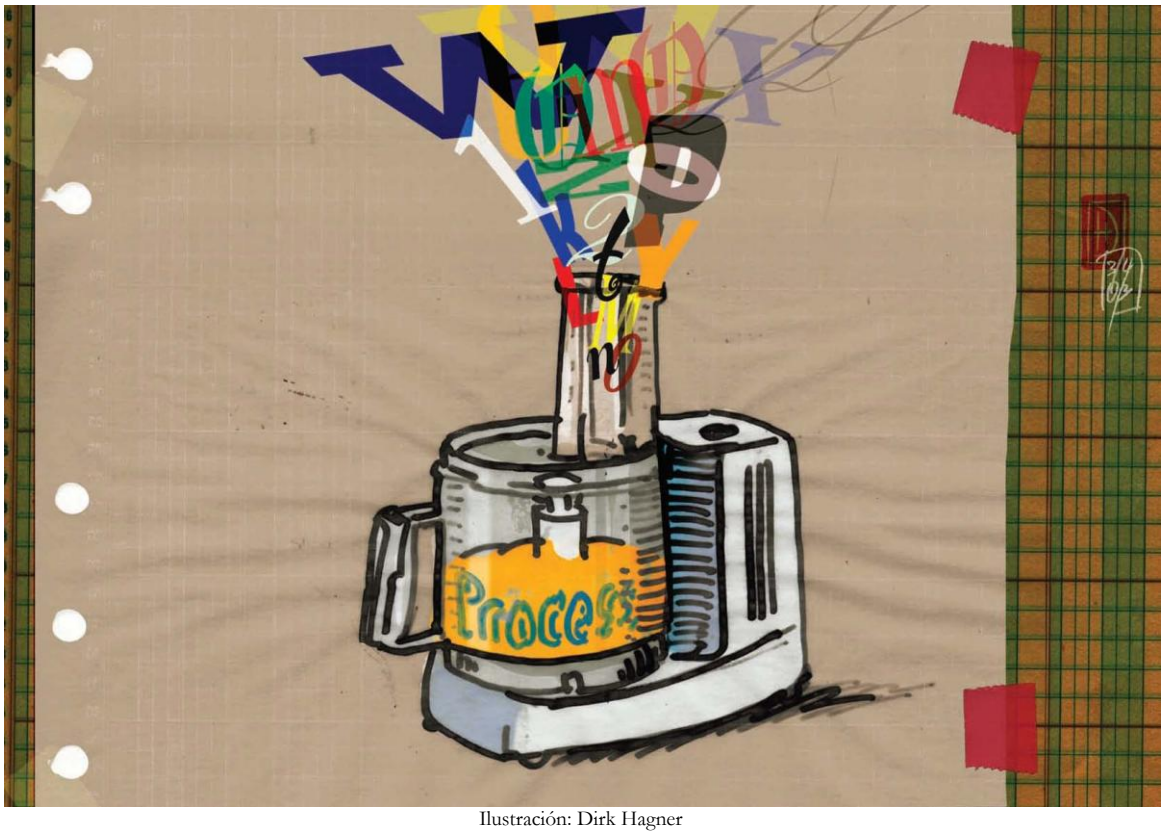

Ilustración: Dirk Hagner
IEEE-CS Software Vol20 No.2

En este Capítulo se muestran los resultados de la evaluación del modelo. Se propone una rúbrica como método de evaluación de modelos de Ingeniería de Software. Se aplica al modelo propuesto y se compara con los resultados de la evaluación, empleando la misma rúbrica propuesta, a otro modelo de desarrollo de software.

CAPÍTULO 4. RESULTADOS

4.1 Evaluación del Modelo

Kitchenham (1996) identifica nueve métodos de evaluación en la Ingeniería de Software:

- Análisis de Características, a través de
  - Experimentos formales
  - Estudios de caso
  - Encuestas
  - Revisión rápida de distintos métodos y herramientas (screening-mode)
- Evaluación cuantitativa
  - Experimentos formales
  - Estudios de caso
  - Encuestas
- Análisis de efectos cualitativos
- Benchmarking

La mayoría de las investigaciones en Ingeniería de Software no son experimentos. En un sondeo de 5,453 artículos de este campo, de las 12 conferencias y revistas (journals) más representativas, se encontró que sólo 103 pueden categorizarse como experimentos. Los casos de estudio y las encuestas son los dos tipos de estudios que más se realizan (Kitchenham, et al., 2006)

Sólo un trabajo reciente (Cole y Avison, 2007) señala el uso potencial de la hermenéutica, aplicada al desarrollo de sistemas de información (incluida la Ingeniería de Software) pero no se proponen mecanismos de evaluación.

Para evaluar el modelo propuesto, se diseñó una rúbrica de evaluación, la cual se expone a continuación.

4.2 Evaluación de Modelos de Desarrollo de Software Usando Rúbricas

Una rúbrica de evaluación es una tabla de verificación, con instrucciones de evaluación contextualizada, que define e identifica los niveles de desempeño en los aspectos, áreas o dimensiones de un proceso o producto. Ayuda a establecer los criterios de supervisión,

autoevaluación y autorregulación de una situación determinada, fija metas y ayuda a reflexionar sobre las fortalezas y deficiencias (áreas que requieren de apoyo) de los procesos y productos.

La rúbrica de evaluación se aplica en las distintas actividades de desarrollo de software basado en Ingeniería de Dominio:

- Como diagnóstico y evaluación de la actividad análisis de dominio y su producto: El modelo de diseño. Para evaluar la identificación de los dominios y la captura de las diferencias dentro de un dominio.
- Para evaluar el diseño del dominio y el modelo del dominio. Evalúa la adaptabilidad (viabilidad) del diseño.
- Evalúa y define los mecanismos para trasladar requerimientos en sistemas, creados con componentes reutilizables (implementación del dominio). Evalúa la viabilidad en la implementación de los lenguajes específicos del dominio (domain-specific languages DSL), de los generadores de código y de los componentes de código.

Esta rúbrica de evaluación, sirve por lo tanto como diagnóstico, al inicio de un proceso u actividad, ya que establece los ideales regulativos, calidad y características de los procesos y productos. Puede ser normativa, para regular las entradas, los procesos y las salidas del desarrollo de software. Evalúa además, las características deseables de las personas que intervendrán en el desarrollo de software, no sólo de los insumos (requerimientos), los procesos (actividades, tareas) y los productos (software, documentación). Y puede aplicarse en los distintos niveles jerárquicos de la organización.

A través de la rúbrica de evaluación se fomentará en los participantes involucrados en el desarrollo de software una visión sistémica teniendo como fin último la sustentabilidad. Se formará en ellos un enfoque holístico (contextual), heurístico (ponderar entre medios y fines; tareas repetitivas e innovación); crítico (establecer los factores relevantes de su práctica profesional, de acuerdo a las necesidades de su organización, comunidad o contexto) y ético.

La rúbrica de evaluación propuesta (§4.3) define cinco niveles de desempeño: nulo, incipiente, en desarrollo, maduro y ejemplar. Define los cuatro aspectos desarrollados a lo largo de la presente tesis en correspondencia con los niveles de desempeño. Y define dos meta-competencias, la prudencia y la sutileza. La rúbrica evalúa las actividades dentro del proceso de desarrollo de software y se puede

modificar de forma muy fácil para evaluar y establecer niveles de desempeño de diagnósticos (entradas), productos (resultados) y personas.

META-COMPETENCIAS.

PRUDENCIA. Sensibilidad que brinda moderación, ponderación, proporción, mesura, armonía. La prudencia es "La Virtud de las Virtudes" y por lo tanto se convierte en una meta-competencia. **SUTILEZA**: Capacidad de hacer distinciones finas, puede establecer una nueva opción donde se creía que no había o que éstas se habían agotado, capacidad para comprender situaciones complejas.

Ética

Definida como:

Ética = [(Fortaleza + Templanza) * Justicia]* Prudencia

Donde:

Fortaleza: valor y la constancia para perseverar en una **obra buena** hasta el final

Templanza: Búsqueda inteligente del ***Bien* propio**, respetando a lo que tiene derecho el ***otro***.

***Justicia*:** dar a cada uno lo que le *corresponde* o pertenece. A menudo asociada con el **bien común**.

Prudencia: Sensibilidad que brinda moderación, ponderación, proporción (*correspondencia*), mesura, armonía. La prudencia es "La Virtud de las Virtudes" y por lo tanto se convierte en una meta-competencia.

La Visión holística incluye los aspectos hermenéuticos, dado que la hermenéutica es poner un texto (sistema) en un contexto relevante.

4.3 Rúbrica de Evaluación de Modelos en Ingeniería de Software

Instrucciones: Evalúe el modelo de acuerdo al nivel de desempeño de cada uno de los aspectos: visión holística, práctica: heurística, crítica y ética. Coloque el número correspondiente (0, 1, 2, 3, ó 4) en el apartado de puntuación

Niveles de Desempeño / Aspectos	NULO 0	INCIPIENTE 1	EN DESARROLLO 2	MADURO 3	EJEMPLAR 4	Puntuación
Holístico	Desconoce el Contexto y la problemática actual del sistema, de los requerimientos y de las necesidades de los usuarios y de los distintos involucrados en el de desarrollo	Contempla la obtención de requerimientos y las necesidades de los usuarios y de los distintos involucrados en el de desarrollo	Contempla los requerimientos y las necesidades, pero no indica claramente cómo contextualizarlos en el desarrollo de software	Contempla los requerimientos y las necesidades, permite contextualizarlos y nos permite proponer una solución de acuerdo a la problemática de la organización	Permite comprender, desarrollar y explicar las actividades de desarrollo de software y contextualizarlas en un escenario relevante. Es capaz de generar en todos los involucrados en el proceso la conciencia y reflexión del equilibrio dinámico del sistema en el contexto (lo que está alrededor del sistema y su organización) de acuerdo a las intervenciones (acciones) que ellos realizan. Mostrándoles y motivando la reflexión de como éstas inciden además en otros sistemas.	
Heurístico	Actividades caóticas, sin métodos, modelos ni procedimientos	Actividades sistemáticas e indiscriminadas. Fomenta el aquí y ahora.	Actividades sistemáticas, generalmente descontextualizadas	Fomenta en las partes involucradas el desarrollo de trabajos progresivos, da retroalimentación constante de los mismos. Conforme avanza el trabajo les muestra las distintas ponderaciones para que las apliquen con posterioridad.	Heurística: Desarrollo progresivo, contextualizado y reflexivo sobre las actividades en el desarrollo de software. Fomenta en los involucrados una visión diacrónica (a través del tiempo), que cada uno determine los momentos en los cuales es posible intervenir. Los ayuda a elegir entre seguridad vs. innovación, en distintas situaciones. Los ayuda a establecer y elegir entre distintos escenarios de la evolución de una situación (a través del tiempo)	
Crítico (Relevante)	Desarrollo de actividades sin crítica alguna. No cuestiona y asume todo por cierto y verdadero	Establece los criterios mínimos de las actividades, principalmente sobre cuestiones de contenido, forma, procesos, jerarquías, y evaluación	A parte de establecer los criterios mínimos de las actividades, desarrolla algunos criterios profesionales	Explica y aplica los criterios mínimos adecuados para la actividad, de acuerdo a su contexto	Desarrolla en los involucrados en el proceso de desarrollo un espíritu crítico-reflexivo. Los guía para que sean capaces de establecer por sí mismos los criterios relevantes en sus actividades, de acuerdo a cada situación, respetando la Tradición y la Cultura, y fomentando la innovación	
Ético	Desconoce los códigos de práctica profesional (códigos de etiqueta)	Fomenta en los involucrados el conocimiento de los códigos de práctica profesional y hace una revisión textual.	Aplica los códigos de práctica profesional en el proceso de desarrollo de software	Comprende los códigos de práctica profesional y de ética; los aplica casuísticamente	Fomenta en los interesados la aplicación critico-reflexiva de los códigos de práctica profesional y de ética, ponderando: • El Contexto (general) y el caso (particular) • El beneficio propio y el ajeno	
					TOTAL	

Fuente: Elaboración propia

4.4 Comparación del Modelo de Desarrollo Propuesto y el Modelo Zachman

A continuación se evalúa el modelo de desarrollo propuesto y posteriormente se compara con el modelo de desarrollo de software de Ostadzadeh, Aliee, y Ostadzadeh, A. (2007). El cual combina la Arquitecura Zachman con la arquitectura dirigida por modelos MDA-Model Driven Architecture (OMG 2008).

Cada uno de los aspectos de la rúbrica (holístico, heurístico, crítico y ético), se evalúa de acuerdo al nivel de desempeño mostrado. Las características a cubrir para alcanzar cierto nivel de desempeño se encuentran definidas para cada uno de los aspectos.

4.4.1 Rúbrica de Evaluación del Modelo de Desarrollo Propuesto

Instrucciones: Evalúe el modelo de acuerdo al nivel de desempeño de cada uno de los aspectos: visión holística, práctica: heurística, crítica y ética. Coloque el número correspondiente (0, 1, 2, 3, ó 4) en el apartado de puntuación

Niveles de Desempeño / Aspectos	NULO 0	INCIPIENTE 1	EN DESARROLLO 2	MADURO 3	EJEMPLAR 4	Puntuación
Holístico	Desconoce el Contexto y la problemática actual del sistema, de los requerimientos y de las necesidades de los usuarios y de los distintos involucrados en el de desarrollo	Contempla la obtención de requerimientos y las necesidades de los usuarios y de los distintos involucrados en el de desarrollo	Contempla los requerimientos y las necesidades, pero no indica claramente cómo contextualizarlos en el desarrollo de software	Contempla los requerimientos y las necesidades, permite contextualizarlos y nos permite proponer una solución de acuerdo a la problemática de la organización	Permite comprender, desarrollar y explicar las actividades de desarrollo de software y contextualizarlas en un escenario relevante. Es capaz de generar en todos los involucrados en el proceso la conciencia y reflexión del equilibrio dinámico del sistema en el contexto (lo que está alrededor del sistema y su organización) de acuerdo a las intervenciones (acciones) que ellos realizan. Mostrándoles y motivando la reflexión de como éstas inciden además en otros sistemas.	3
Heurístico	Actividades caóticas, sin métodos, modelos ni procedimientos	Actividades sistemáticas e indiscriminadas. Fomenta el aquí y ahora.	Actividades sistemáticas, generalmente descontextualizadas	Fomenta en las partes involucradas el desarrollo de trabajos progresivos, da retroalimentación constante de los mismos. Conforme avanza el trabajo les muestra las distintas ponderaciones para que las apliquen con posterioridad.	Heurística: Desarrollo progresivo, contextualizado y reflexivo sobre las actividades en el desarrollo de software. Fomenta en los involucrados una visión diacrónica (a través del tiempo), que cada uno determine los momentos en los cuales es posible intervenir. Los ayuda a elegir entre seguridad vs. innovación, en distintas situaciones. Los ayuda a establecer y elegir entre distintos escenarios de la evolución de una situación (a través del tiempo)	3
Crítico (Relevante)	Desarrollo de actividades sin crítica alguna. No cuestiona y asume todo por cierto y verdadero	Establece los criterios mínimos de las actividades, principalmente sobre cuestiones de contenido, forma, procesos, jerarquías, y evaluación	A parte de establecer los criterios mínimos de las actividades, desarrolla algunos criterios profesionales	Explica y aplica los criterios mínimos adecuados para la actividad, de acuerdo a su contexto	Desarrolla en los involucrados en el proceso de desarrollo un espíritu crítico-reflexivo. Los guía para que sean capaces de establecer por sí mismos los criterios relevantes en sus actividades, de acuerdo a cada situación, respetando la Tradición y la Cultura, y fomentando la innovación	4
Ético	Desconoce los códigos de práctica profesional (códigos de etiqueta)	Fomenta en los involucrados el conocimiento de los códigos de práctica profesional y hace una revisión textual.	Aplica los códigos de práctica profesional en el proceso de desarrollo de software	Comprende los códigos de práctica profesional y de ética; los aplica casuísticamente	Fomenta en los interesados la aplicación critico-reflexiva de los códigos de práctica profesional y de ética, ponderando: • El Contexto (general) y el caso (particular) • El beneficio propio y el ajeno	3
					TOTAL	13

Fuente: Elaboración propia

4.4.2 Rúbrica de Evaluación del Modelo Zachman

Instrucciones: Evalúe el modelo de acuerdo al nivel de desempeño de cada uno de los aspectos: visión holística, práctica: heurística, crítica y ética. Coloque el número correspondiente (0, 1, 2, 3, ó 4) en el apartado de puntuación

Niveles de Desempeño / Aspectos	NULO 0	INCIPIENTE 1	EN DESARROLLO 2	MADURO 3	EJEMPLAR 4	Puntuación
Holístico	Desconoce el Contexto y la problemática actual del sistema, de los requerimientos y de las necesidades de los usuarios y de los distintos involucrados en el de desarrollo	Contempla la obtención de requerimientos y las necesidades de los usuarios y de los distintos involucrados en el de desarrollo	Contempla los requerimientos y las necesidades, pero no indica claramente cómo contextualizarlos en el desarrollo de software	Contempla los requerimientos y las necesidades, permite contextualizarlos y nos permite proponer una solución de acuerdo a la problemática de la organización	Permite comprender, desarrollar y explicar las actividades de desarrollo de software y contextualizarlas en un escenario relevante. Es capaz de generar en todos los involucrados en el proceso la conciencia y reflexión del equilibrio dinámico del sistema en el contexto (lo que está alrededor del sistema y su organización) de acuerdo a las intervenciones (acciones) que ellos realizan. Mostrándoles y motivando la reflexión de como éstas inciden además en otros sistemas.	3
Heurístico	Actividades caóticas, sin métodos, modelos ni procedimientos	Actividades sistemáticas e indiscriminadas. Fomenta el aquí y ahora.	Actividades sistemáticas, generalmente descontextualizadas	Fomenta en las partes involucradas el desarrollo de trabajos progresivos, da retroalimentación constante de los mismos. Conforme avanza el trabajo les muestra las distintas ponderaciones para que las apliquen con posterioridad.	Heurística: Desarrollo progresivo, contextualizado y reflexivo sobre las actividades en el desarrollo de software. Fomenta en los involucrados una visión diacrónica (a través del tiempo), que cada uno determine los momentos en los cuales es posible intervenir. Los ayuda a elegir entre seguridad vs. innovación, en distintas situaciones. Los ayuda a establecer y elegir entre distintos escenarios de la evolución de una situación (a través del tiempo)	2
Crítico (Relevante)	Desarrollo de actividades sin crítica alguna. No cuestiona y asume todo por cierto y verdadero	Establece los criterios mínimos de las actividades, principalmente sobre cuestiones de contenido, forma, procesos, jerarquías, y evaluación	A parte de establecer los criterios mínimos de las actividades, desarrolla algunos criterios profesionales	Explica y aplica los criterios mínimos adecuados para la actividad, de acuerdo a su contexto	Desarrolla en los involucrados en el proceso de desarrollo un espíritu crítico-reflexivo. Los guía para que sean capaces de establecer por sí mismos los criterios relevantes en sus actividades, de acuerdo a cada situación, respetando la Tradición y la Cultura, y fomentando la innovación	2
Ético	Desconoce los códigos de práctica profesional (códigos de etiqueta)	Fomenta en los involucrados el conocimiento de los códigos de práctica profesional y hace una revisión textual.	Aplica los códigos de práctica profesional en el proceso de desarrollo de software	Comprende los códigos de práctica profesional y de ética; los aplica casuísticamente	Fomenta en los interesados la aplicación critico-reflexiva de los códigos de práctica profesional y de ética, ponderando: • El Contexto (general) y el caso (particular) • El beneficio propio y el ajeno	0
					TOTAL	7

Fuente: Elaboración propia

4.4.3 Comparación de Modelos

La Tabla 4.1 muestra el resumen de los resultados de la evaluación de ambos modelos.

Modelo / Aspectos	**Propuesto**	**Ostadzadeh et. al (2007)**
Holístico	3	3
Heurístico	3	2
Crítico (Relevante)	4	2
Ético	3	0
TOTAL	13	7

Tabla 4.1 Resultados de la Comparación de Modelos
Fuente: Elaboración propia.

De la evaluación de los modelos se desprende que nuestro modelo cubre de mejor forma los aspectos heurísticos, críticos y éticos. El modelo que proponen Ostadzadeh et. al (2007) no cubre las cuestiones éticas; los aspectos heurísticos y críticos están en desarrollo. Y los aspectos holísticos son maduros.

Capítulo 5. Conclusiones y Trabajos Futuros

5.1 Conclusiones

Como resultado de este trabajo de investigación se diseñó e implementó un modelo heurístico, interpretativo y contextual aplicado al proceso de desarrollo de software.

Se propuso un modelo hermenéutico (interpretativo), holístico (totalidad), heurístico (innovación), crítico (mérito) y ético. El cual tiene como finalidad desarrollar software que ayude a transformar a las organizaciones, mostrándoles una ruta para que aprovechen todos los recursos con que cuentan o los que necesitan, poniéndose con ello en camino de operar y desarrollarse de forma sustentable.

El presente trabajo desarrolla un modelo para el desarrollo de software basado en la Ingeniería de Dominio. Parte fundamental de este enfoque es el desarrollo y evaluación de arquitecturas. En la Tesis se proponen una Metáfora y una Arquitectura Sistémica. Éstas son abstracciones que brindan la posibilidad de transmitir de significados a los distintos involucrados en el proceso de desarrollo de software. Sirven como base para el entendimiento, la negociación, el consenso y la comunicación, además de que facilitan la reutilización de experiencia y conocimiento. El proceso desarrollado para su definición y aplicación en el área de desarrollo de software, es replicable en otros ámbitos, como el educativo, el de ingeniería y cualquier otro en donde se requiera conceptualización y modelado.

Para diseñar este modelo se partió del concepto de arquitecturas de cómputo empresarial. Mediante el uso de metáforas y analogías, se cambia el enfoque actual del ingeniero de software (el cual también aplicable al líder de proyecto, al arquitecto, al analista y al diseñador de sistemas y software) de un enfoque rígido y utilitario (mercantilista y a corto plazo), por un enfoque heurístico, holístico y contextual; el cual contempla aspectos tecnológicos, políticos, legales, sociales y culturales.

Se cumplieron los objetivos específicos, ya que:

- Se formuló la problemática que enfrenta el área de desarrollo de software, confirmando que los problemas de comunicación son una de las principales causas de fallas en el proceso de desarrollo.
- Se implementó un modelo que coadyuva a solucionar tal problemática, tomando en cuenta a todos los interesados en el proceso de desarrollo de software, respetando su tradición, su cultura y sus valores.

- Se utilizó a la hermenéutica en conjunto con la heurística como recurso de indagación y comunicación entre los distintos entes involucrados en el proceso de desarrollo de software.

Todos ellos tendientes a brindarles a los usuarios software de calidad, en tiempo y a un precio accesible, en una sociedad que respete los valores fundamentales del hombre.

Nuestros productos y servicios deberán de considerar que:

- El diseño de soluciones tecnológicas debe partir de un requerimiento claro,
- tiene que existir una visión compartida entre el cliente y el proveedor del servicio, para lo cual hay que explorar los mecanismos de comunicación pertinentes,
- los diseños cubran las expectativas del cliente,
- el aspecto cultural es fundamental.

Se propuso y aplicó una rúbrica como método de evaluación de modelos de Ingeniería de Software. Esta rúbrica se aplicó a dos modelos de desarrollo, al modelo propuesto y al modelo que proponen Ostadzadeh et. al (2007). Los resultados de las rúbricas respectivas indican que nuestro modelo cubre de mejor forma los aspectos heurísticos, críticos y éticos. El modelo que proponen Ostadzadeh et. al (2007) no cubre las cuestiones éticas; los aspectos heurísticos y críticos están en desarrollo. Y los aspectos holísticos son los únicos que son maduros.

Una vez evaluados los cuatro aspectos propuestos para mejorar el proceso de desarrollo de software: Holística-hermenéutica (Totalidad): Visión (Contextual), Heurística (Innovación): Creatividad, Crítica (Relevancia): Liderazgo y Ética. Se concluye que el modelo propuesto es capaz de proporcionarnos una visión diacrónica (la evolución del sistema a través del tiempo), no sólo el panorama actual. Promueve un cambio provechoso para la organización, y los productos resultantes (software, sistemas de información, etc.) son relevantes para la misma.

Por lo tanto se corrobora nuestra hipótesis, que al usar un modelo heurístico, interpretativo y contextual se mejora el proceso de desarrollo de software.

5.2 Aportaciones

Las aportaciones del presente trabajo son de dos tipos: conceptuales y concretas.

Dentro de las aportaciones conceptuales se definió la relación entre las Ingenierías de Software y de Sistemas con las humanidades, las cuales tienen posibilidad de solucionar parte de la problemática del desarrollo de software.

5.2.1 Aportaciones Teóricas

Dando respuesta a las preguntas de investigación:

¿El paradigma sistemático aplicado al desarrollo de software es el más adecuado para un entorno complejo y en constante cambio, con participantes con intereses diferentes? No. El paradigma sistemático ha quedado rebasado por un entorno complejo y con crisis recurrentes.

¿Las fallas en la comunicación de los distintos participantes involucrados en un sistema, inciden en el proceso de desarrollo de software? Sí. Y estas se dan por falta de capacidad o por cuestiones éticas.

¿Los aspectos ético-culturales inciden en el proceso de desarrollo de software? Sí, aunque su cuantificación no está bien detallada.

¿Es posible modelarlos e incorporarlos en el proceso de desarrollo de software? Sí es posible modelarlos, diseñando el comportamiento que se espera de los distintos involucrados en el proceso de desarrollo.

¿Se puede implementar una herramienta que modele los aspectos ético-culturales y que mejore la comunicación de los participantes del sistema? Sí. Lo se realizó a través de la Arquitectura Sistémica.

5.2.2 Aportaciones Concretas

Nuestras aportaciones concretas son:

1. Un modelo heurístico, interpretativo y contextual aplicado en el proceso de desarrollo de software.
2. Una herramienta que facilita la comunicación entre los distintos involucrados en el proceso de desarrollo de software. A través de la Metáfora y la Arquitectura Sistémica.
3. Una aplicación del modelo interpretativo (hermenéutico) en el campo de las Ciencias de la Computación. Con la Heurística Interpretativa.
4. Una rúbrica de evaluación de modelos aplicada en la Ingeniería de Software. Esta rúbrica contempla los aspectos holísticos, heurísticos, críticos y éticos.

5.3 El Modelo Propuesto

El modelo propuesto para el desarrollo de software basado en Ingeniería de Dominio parte de la Metáfora de la Organización.

A partir de ésta se contrasta la situación actual con la deseada y con los elementos de la planeación. Este paso muestra parte de la problemática del desarrollo de software dentro de la organización.

Posteriormente se desarrollan las tareas de la Ingeniería de Dominio y de la Aplicación, en este paso es donde se diseña e implementa la Arquitectura de Software.

Se contextualizan lo aspectos éticos, legales y políticos aplicables al desarrollo referido, y se conforma la Arquitectura Sistémica.

Se implementa la fase, prototipo o modelo; se evalúa y con los resultados se retroalimenta y se reinicia el proceso de desarrollo.

Para guiarnos en cada uno de estos pasos se propuso una heurística interpretativa.

El modelo propuesto incorpora, además, los elementos que se proponen para evolucionar de la Ingeniería de Dominio a las Líneas de Productos de Software.

5.4 Trabajos Futuros

Aplicar el modelo propuesto usando alguna de las metodologías ágiles para el desarrollo de software o de las tradicionales como el Proceso Unificado de Modelado o el Rational Unified Process (RUP).

Implementar el modelo en alguna notación estándar como UML (Unified Model Language- lenguaje unificado de modelado) (OMG 2008a) o BPMN (Business Process Modeling Notation- Notación para el modelado de procesos de negocios) (OMG 2008c), bajo alguno de los enfoques de desarrollo emergentes como MDA (Model Driven Architecture- arquitectura dirigida por modelos) (OMG 2008), SOA (Service Oriented Architecture- arquitectura orientada a servicios) (OASIS 2008) o BPM (Business Process Management- administración de procesos de negocios) (OMG 2008b; Weske 2007; Smith y Fingar 2003).

5.5 Publicaciones Derivadas de Este Trabajo de Tesis

5.5.1 Revistas Arbitradas

Marín Solís, Ramón (2008). Transdisciplina Sistémica y Modelado de Sistemas de Información: La Cultura y el Software. *Revista internacional la nueva gestión organizacional.* Año 3 No. 6. México: Universidad Autónoma de Tlaxcala, Colegio de Estudios de Posgrado de la Ciudad de México,

Universidad Autónoma del Estado de Hidalgo y Universidad de Camaguey. ISSN 1870-2058. Indizada por LATINDEX.

Marín Solís, Ramón (2008). Heurística e investigación: más allá del método. *Nuevo Mundo- Estrategia Organizacional Bajo un Enfoque Cognoscitivo*. 1, (1). Noviembre 2008. México: Colegio de Estudios de Posgrado de la Ciudad de México. ISSN en trámite. pp. 121-125.

Flores Vasconcelos, David y Marín Solís, Ramón (2008). Cibern-ética: una opción transdisciplinar en los procesos de educación. *Nuevo Mundo- Estrategia Organizacional Bajo un Enfoque Cognoscitivo*. 1, (1). Noviembre 2008. México: Colegio de Estudios de Posgrado de la Ciudad de México. ISSN en trámite. pp. 115-120

5.5.2 Capítulos de Libros

Marín Solís, Ramón. (SF). Modelos empresariales integrales: las tecnologías de la información y su relación con la educación, la administración, las políticas públicas y el software. *La construcción del conocimiento: docente-alumno*. México: Colegio de Estudios de Posgrado de la Ciudad de México. Aceptado y en proceso para publicarse como capítulo.

Marín Solís, Ramón y Flores Vasconcelos, David (SF). La Hermenéutica Analógica en las Ciencias de la Computación: El ingeniero de software como hermeneuta en el Siglo XXI. *III Coloquio Internacional de Hermenéutica Analógica*. México: Instituto de Investigaciones Filológicas. UNAM. Aceptado y en proceso para publicarse como capítulo.

5.5.3 Memorias de Congresos

Flores Vasconcelos, David y Marín Solís, Ramón (2006). Telenoía y teleopoiésis como binomio estratégico para la eficiencia terminal en el posgrado. *1er Congreso internacional de innovación Educativa*. México: IPN. Julio 2006.

Flores Vasconcelos, David y Marín Solís, Ramón (2006). La ética del control: una opción transdisciplinar en Ingeniería de Sistemas. *1er Congreso internacional de innovación Educativa*. México: IPN. Julio 2006.

Marín Solís, Ramón. Mora Espinosa, Miguel Ángel. Gutiérrez Tornés, Agustín y Suárez, Guerra Sergio (2005). Sistemas Ético Profesionales en la Ingeniería: Hacia un Diseño Sistémico. *4° Congreso Internacional de Ingeniería Electromecánica y de Sistemas (CIIES)*. México: IPN. 17 Noviembre 2005. ISBN 970-36-0291-6

Marín Solís, Ramón (2005). Systems Engineering and Ethics: Towards a Systemic Design. *49th Meeting of the International Society for the Systems Sciences (ISSS)*. California: ISSS. Cancún, México. 4 Julio 2005. ISBN 0-9740735-4-7

Marín Solís, Ramón (2005). Information Systems in Post-modernity. *49th Meeting of the International Society for the Systems Sciences (ISSS)*. California: ISSS. Cancún, México. 1 Julio 2005. ISBN 0-9740735-4-7

Marín Solís, Ramón. Gutiérrez Tornés, Agustín y Mora Espinosa, Miguel Ángel. (2005). Rediseño al paradigma de la Ingeniería de Software: El Caso del Software Libre. *Congreso Nacional de Software Libre 2005*. México: CONSOL-IPN.

Marín Solís, Ramón. Orantes Jiménez, Sandra Dinora. Gutiérrez Tornés, Agustín y Mora Espinosa, Miguel Ángel. (2004). Rediseño al paradigma de la Ingeniería de Software: Problemática y propuesta. *Congreso Internacional de la decimoquinta reunión de otoño de Cómputo y Comunicaciones 2004*. Acapulco, Gro. México: IEEE-Sección México.

5.5.4 Ponencias en Congresos

Marín Solís, Ramón (2008). Administración de Procesos de Negocios con Software Libre: Filosofía y Arquitectura. *Congreso Nacional de Software Libre CONSOL 2008*, http://www.consol.org.mx/2008/ ; México, D.F: CONSOL-UACM. Resumen:
http://comas.consol.org.mx/2008/general/proposals/716

Marín Solís, Ramón y Flores Vasconcelos David (2006). La Hermenéutica Analógica en las Ciencias de la Computación. *III Coloquio Internacional de Hermenéutica Analógica.* México: Facultad de Filosofía y Letras. UNAM. 11 Octubre del 2006.

Marín Solís, Ramón. Bosco Hernández, Martha Diana. Gutiérrez Tornés, Agustín. Suárez Guerra, Sergio (2006). La Educación Sistémica y el Software Libre como Base para la Transformación de la Sociedad. *Congreso Nacional de Software Libre CONSOL 2006, http://www.consol.org.mx/2006/index.html*; México, D.F: CONSOL-IPN. 15 de agosto 2006.

REFERENCIAS

Ackoff, Russell L. (1972). Un *concepto de planeación de empresas*. México: Limusa. 1984. pp. 157

Ackoff, Russell L. (1974). *Rediseñando el futuro*. México: Limusa. 2001. pp.332

Ackoff, Russell L. (1989). *Cápsulas de Ackoff: Administración en pequeñas dosis*. México. Limusa. 2002. pp.203

Ackoff, Russell L. (1999). *El paradigma de Ackoff: Una administración sistémica*. México: Limusa. 2002. pp.367

Ackoff, Russell L. (1999a). On passing through 80'. En *Proceedings: Russell L. Ackoff and the advent of systems thinking conference*. Villanova, Paris, Francia: The College of Commerce and Finance. 1999. http://ackoff.villanova.edu/public_html/ackoff99.pdf (05 de Abril de 2003, 03:20:28 am.)

Ackoff, Russell L. (2001). *A brief guide to interactive planning and idealized design*. EUA: 31 de mayo 2001 http://www.sociate.com/texts/AckoffGuidetoIdealizedRedesign.pdf (18 enero 2004). pp.15

Ackoff, Russell L. (2003). *Redesigning society*. EUA: Stanford Business Books –Stanford University Press. 2003. pp. 184

ACM (2004). *ACM: Association for Computing Machinery, the world's first educational and scientific computing society*. http://www.acm.org/ (1/Ene/2005)

Agile Alliance (2001). *Manifesto for Agile Software Development*. 2001. http://www.agilemanifesto.org/ (24/nov/08)

Agile Alliance (2008). *Agile Alliance Home*. http://www.agilealliance.org/home (24/nov/08)

Ambler, Scott (2004). *Agile data home page: Bringing data professionals and application developers together*. http://www.agiledata.org/:EUA (12/05/2004)

Bass, Len. Clements, Paul y Kazman, Rick (2003). *Software architecture in practice*. EUA: Addison

Wesley. 2003, Second Edition. pp. 560

Beer, Stafford (1972). *Ciencia de la Dirección*. Argentina: Ed. Ateneo. 1972.

Beuchot, Mauricio (2003). La filosofía del hombre o antropología filosófica según la Hermenéutica Analógica. *VII Jornada de hermenéutica*. México, DF: UNAM Facultad de Filosofía y Letras. 2 de julio del 2003.

Beuchot, Mauricio. 2004. *Ética*. Torres Asociados. México. pp.174

Beuchot, M. (2005). *Perfiles esenciales de la hermenéutica* (4 ed.). México: Instituto de Investigaciones Filológicas, UNAM

Beuchot, Mauricio (2005a). *Tratado de Hermenéutica Analógica: Hacia un nuevo modelo de interpretación*. Itaca: México. Tercera Edición 2005. pp. 210

Boehm, Barry (2006). A view of 20th and 21st century software engineering. In *Proceeding of the 28th International Conference on Software Engineering* (Shanghai, China, May 20 - 28, 2006). ICSE '06. ACM Press, New York, NY, 12-29. DOI= http://doi.acm.org/10.1145/1134285.1134288

Charette, R. N. (2005). Why software fails [software failure]. *Spectrum, IEEE, 42*(9), 42-49.

Checkland, Peter y Holwell, Sue. (1998). *Information systems: Making sense of the field.* Inglaterra: John Wiley & Sons. 1998. pp. 262

Chesbrough, H. and Spohrer, J. 2006. A research manifesto for services science. *Commun. ACM* 49, 7 (Jul. 2006), 35-40. DOI= http://doi.acm.org/10.1145/1139922.1139945

Clements, Paul. (2003). Origins of software architecture study. *Essays on Software Architecture*. Software Engineering Institute. http://www.sei.cmu.edu/architecture/essays.html. Carnegie Mellon University. 2 octubre 2003. (12/05/2004)

Cole, M., & Avison, D. (2007). The potential of hermeneutics in information systems research. *Eur J Inf Syst, 16*(6), 820-833.

Congreso de la Unión (2003). *Ley Federal del Derecho de Autor*. http://www.cddhcu.gob.mx/leyinfo/pdf/122.pdf Cámara de Diputados: México. (04/11/2004)

Congreso de la Unión (2003a). *LEY Federal de Responsabilidades de los Servidores Públicos*. México: H. Congreso de la Unión. 2003. http://www.diputados.gob.mx/LeyesBiblio/pdf/115.pdf (1/Ene/09)

Congreso de la Unión (2004). *Ley Federal de la Propiedad Industrial*. http://www.cddhcu.gob.mx/leyinfo/pdf/50.pdf Cámara de Diputados: México. (04/11/2004)

Cortina, A. 1996. *El quehacer ético: guía para la educación moral*. Aula siglo XXI/ Santillana. Madrid. pp. 123

Donaldson, Scott y Siegel, Stanley (2001). *Successful software development 2nd edition*. EUA: Prentice Hall PTR. 2001. pp. 701

Dyba, T., Kitchenham, B. A., & Jorgensen, M. (2005). Evidence-based software engineering for practitioners. *Software, IEEE, 22*(1), 58-65.

El Emam, K., & Koru, A. G. (2008). A Replicated Survey of IT Software Project Failures. *Software, IEEE, 25*(5), 84-90.

Etxeberria, X. 2002. *Temas básicos de ética*. Desclée. Bilbao. pp. 207

Fowler, Martin (2003). *The new methodology*. EUA: http://www.martinfowler.com/articles/newMethodology.html (27/08/2003 12:03:59 a.m.)

Frakes, W. B., & Kyo, K. (2005). Software reuse research: status and future. *Software Engineering, IEEE Transactions on, 31*(7), 529-536.

García, Francisco José. Barras, Juan Antonio. Laguna, Miguel Ángel y Marqués, José Manuel. (2002). Líneas *de Productos, Componentes, Frameworks y Mecanos. Reporte Técnico DPTOLA-IT-2002-004*. Universidad de Salamanca.

http://diaweb.usal.es/diaweb20/archivos/10001127DPTOIA_IT_2002_004.pdf (8/Diciembre/2008)

IEEE. (1990). IEEE Std 610.12-1990, IEEE standard glossary of software engineering terminology. En *IEEE standards software engineering: Volume one, customer and terminology standards*. 1999 Edition. EUA: IEEE. 1999. pp.82

IEEE. (2000). *IEEE Std 1471-2000*, *IEEE Recommended practice for architectural description of software-intensive systems*. EUA: IEEE. 2000. pp.23

IEEE (2005). *Mission and Vision*. http://www.ieee.org/portal/index.jsp?pageID=corp_level1&path=corporate&file=vision.xml&xsl=generic.xsl (1/Enero/2005)

IEEE (2005a). *Quick Facts*. 1 Enero 2005. http://www.ieee.org/portal/index.jsp?pageID=corp_level1&path=about&file=quick_facts.xml&xsl=generic.xsl (5/Enero/2005)

IEEE (2005b) *IEEE Frequently Asked Questions (FAQ)*. .http://www.ieee.org/portal/index.jsp?pageID=corp_level1&path=about&file=FAQs.xml&xsl=generic.xsl (5/Enero/2005)

IEEE-CS (2004). *About the Computer Society*. http://www.computer.org/csinfo/ (1/Enero/2005)

IEEE-CS (2004) *SWEBOK. Guide to the Software Engineering Body of Knowledge. Versión 2004*. http://www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.pdf (16/Febrero/2005)

IEEE-CS/ACM. (1999). *Código de Ética y Práctica Profesional 5.2*. http://www.acm.org/serving/se/code_s.html (20/ago/2004)

IEEE S2ESC (2004). *Minutes of Meeting # 106*. http://standards.computer.org/sesc/s2esc_excom/minutes/2004-02/Minutes-Onsite-MBExCom-2004Feb.htm 26 y 27 Febrero 2004 (26/ago/04)

IEEE S2ESC (2004ª). *Minutes of Meeting # 111*.

http://standards.computer.org/sesc/s2esc_excom/minutes/2004-07/Minutes-Telecon-ExCom-2004Jul.htm 6 Julio 2004. (28/ago/04)

IEEE S2ESC. (2004b). *Background.* http://standards.computer.org/sesc/s2esc_geninfo/S2ESC-Background.doc (28/ago/04)

IEEE S2ESC. (2008) Software and Systems Engineering Standards Committee (S2ESC) *Executive Committee Information* http://standards.computer.org/sesc/s2esc_excom/excomndx.htm (25/nov/08)

ISO/IEC (2007) Software Engineering Metamodel for Development Methodologies. ISO/IEC 24744, International Organization for Standardization, Ginebra.

ISO/IEC (2008) ISO/IEC 12207:2008 Systems and software engineering -- Software life cycle processes, International Organization for Standardization, Ginebra.

ISO/IEC (2008a) ISO/IEC 15288:2008 Systems and software engineering -- System life cycle processes, International Organization for Standardization, Ginebra.

(ISO/IEC JTC1/SC7 2008*) Introduction and Mandate* http://142.137.17.56/Labo_Recherche/Lrgl/sc7/Introduction%20and%20Mandate.html (24/nov/08)

Jackson, Michael C. (2003). *Systems thinking: Creative holism for managers*. Inglaterra: John Wiley & Sons. 2003. pp. 352

Jasperson, Jon. Carte, Traci. Saunders, Carol S., Butler, Brian S. et al. (2002). Review: Power and information technology research: A metatriangulation review. *MIS Quarterly*, 26(4), 397-459. Retrieved November 24, 2008, from ABI/INFORM Global database. (Document ID: 275117551).

Kitchenham, B. (1996). *DESMET: A method for evaluating Software Engineering methods and tools*. http://www.osel.co.uk/desmet.pdf (1/dic/08)

Kitchenham, B., Al-Khilidar, H., Babar, M. A., Berry, M., Cox, K., Keung, J., et al. (2006).

Evaluating guidelines for empirical software engineering studies. Paper presented at the Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering.

Kruchten, P. (1995). The 4+1 View Model of Architecture. *IEEE Softw.*, 12(6), 42-50.

Maglio, P. P., Srinivasan, S., Kreulen, J. T., and Spohrer, J. 2006. Service systems, service scientists, SSME, and innovation. *Commun. ACM 49, 7 (Jul. 2006)*, 81-85. DOI= http://doi.acm.org/10.1145/1139922.1139955

Marín Solís, Ramón. (1995). *Propuesta para la integración del sistema red de control escolar institucional para el Instituto Politécnico Nacional.* Tesis de Licenciatura. México, DF: IPN-UPIICSA. 1995

Marín, Solís Ramón (2005). *Diseño de un Modelo de Arquitectura Sistémica para las Empresas de Consultoría en Tecnologías de la Información*. México: Sección de Estudios de Posgrado e Investigación de la ESIME Zacatenco. IPN. Tesis de Maestría en Ciencias, posgrado en Ingeniería de Sistemas. 2005. Pp. 175

McGovern, James; Ambler, Scott W.; Stevens, Michael E.; Linn, James; Sharan, Vikas; Jo, Elias K. (2004). *A practical guide to enterprise architecture*. EUA: Prentice Hall –Professional Technical Reference. 2004. pp. 306

McLaughlin, Laurianne. (2003). An eye on India: Outsourcing debate continues. *IEEE Software. May/June 2003 (Vol. 20, No. 3). E*UA: IEEE Computer Society

Mora Espinosa, Miguel Ángel. (2001). *Apuntes del curso propedéutico de la maestría en ciencias de la arquitectura*. México: Universidad Autónoma de Tabasco. Febrero. 2001.

Mora Espinosa, Miguel Ángel. (2003). La ética del control, una opción transdisciplinar en Ingeniería de Sistemas. *IV Seminario internacional en Ingeniería de Sistemas*. México: UNAM. 2004.

Mora Espinosa, Miguel Ángel. (2004). *Suma sistémica. Claves heurísticas para el rediseño conceptual de la Ingeniería de Sistemas en escenarios de crisis*. México: UNAM. Tesis de Doctorado. 2004.

Morgan, Gareth. (1996). *Imágenes de la organización*. México: Alfaomega. 1998. pp.408

Naur, Peter. Randell, Brian (Eds.) (1969). Software Engineering. *Report on a Conference sponsored by*

the NATO Science Committee. Garmisch, Germany, 7th to 11th October 1968, Brussels, Scientific Affairs Division, NATO, January 1969, 231 p. Disponible también en http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF (20/ene/07)

Naveda, Fernando y Seidman, Stephen B. (2006) *IEEE Computer Society Real-World Software Engineering Problems: A Self-Study Guide for Today's Software Professionals*. Hoboken, New Jersey:Wiley-Interscience. 2006

OASIS (2008). *Reference Architecture for Service Oriented Architecture Version 1.0*. OASIS (Organization for the Advancement of Structured Information Standards). http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra.html (1/Dic/2008)

Oktaba, H., Garcia, F., Piattini, M., Ruiz, F., Pino, F. J., & Alquicira, C. (2007). Software Process Improvement: The Competisoft Project. *Computer, 40*(10), 21-28.

OMG (2008). *OMG Model Driven Architecture*. Object Management Group. 2008. http://www.omg.org/mda/ (1/Dic/2008)

OMG (2008a). *OMG Unified Modeling language*. Object Management Group. 2008. http://www.omg.org/spec/UML/index.htm (1/Dic/2008)

OMG (2008b). *Catalog of OMG Business Strategy, Business Rules and Business Process Management Specifications*. Object Management Group. 2008. http://www.omg.org/technology/documents/br_pm_spec_catalog.htm#BPMN (1/Dic/2008)

OMG (2008c). Business Process Modeling Notation (BPMN) 1.1 Object Management Group. 2008. http://www.omg.org/spec/BPMN/1.1/ (1/Dic/2008)

Ostadzadeh, S., Aliee, F., y Ostadzadeh, A. (2007). A Method for Consistent Modeling of Zachman Framework Cells. *Advances and Innovations in Systems, Computing Sciences and Software Engineering* (pp. 375-380)

Principia Cybernetica Web (2003). Holism. *Web Dictionary of Cybernetics and Systems* http://pespmc1.vub.ac.be/ASC/HOLISM.html (22/06/2003)

PROSOFT. Programa para el Desarrollo de la Industria de Software (2004). *Estudio del nivel de madurez y capacidad de procesos de la industria de tecnologías de información*. México, D.F.: Secretaría de Economía. 2004. pp.64. http://www.software.net.mx/NR/rdonlyres/C1C2DA20-F67C-4101-9D22-42D8F3757874/708/Estudio_evaluacion2.pdf (09/10/2004)

Rhodes, Donna H. (2002). Systems engineering: An essential discipline for the 21st century. *Software Engineering Notes vol. 27 no. 5. September 2002.* EUA: IEEE Computer Society *2002.* p. 40

Rust, Roland y Miu, Carol. 2006. What academic research tells us about service. *Commun. ACM 49, 7 (Jul. 2006)*, 49-54. DOI= http://doi.acm.org/10.1145/1139922.1139948

Sánchez Vázquez, Adolfo (2003). Dimensión político-moral del compromiso intelectual. *Clase Magistral: Ética y política*. México: UNAM. 30 de octubre 2003.

Sánchez Vázquez, Adolfo (2003a). *Filosofía de la praxis*. México: Siglo XXI editores. 1ª Ed. 2003. Ed. anterior 1967. pp. 532.

Smith, Howard y Fingar, Peter (2003). *Business Process Management: the Third Wave*. 2003, Meghan-Kiffer Press. ISBN 0-929652-33-9

SEI (2007). *Domain Engineering.* Software Engineering Institute (SEI), Carnegie Mellon University. http://www.sei.cmu.edu/domain-engineering/domain_eng.html (2/Dic/2008)

SEI (2008) *About SEI*. http://www.sei.cmu.edu/about/ (24/nov/08)

SEI (2008a). *Software Product Line Acquisition: A Companion to A Framework for Software Product Line Practice Version 3.0*. Software Engineering Institute (SEI), Carnegie Mellon University. http://www.sei.cmu.edu/productlines/companion/pl_approach.html (8/Dic/2008)

Sommerville, Ian (2005). *Ingeniería del software.* Madrid: Pearson Educación. 7ª edición. pp. 687

Soni, D., Nord, R. L., & Hofmeister, C. (1995). Software architecture in industrial applications. Paper presented at the *Proceedings of the 17th international conference on Software engineering.*

Sparx Systems (2005). *UML tools for software development and modeling –Enterprise Architect UML*

modeling tool. http://www.sparxsystems.com.au/ (1/Nov/2008)

UMAI (1983). *Código de ética del ingeniero mexicano*. Publicado por Araceli Solano: http://www.cec.uchile.cl/~leherrer/docencia/etica.htm (5/Ene/2005)

Varela Fregoso, Guadalupe (1995). *Ética*. México: Instituto Politécnico Nacional. Segunda reimpresión 2001. pp.204

Weske, Mathias (2007). Business *Process Management: Concepts, Languages, Architectures*. Berlin: Springer-Verlag. 2007. pp. 368