01167

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA
DIVISIÓN DE ESTUDIOS DE POSGRADO
MAESTRÍA EN PLANEACIÓN

GUÍA DE ANÁLISIS ORGANIZACIONAL BAJO UN ENFOQUE SISTEMICO PARA EL DISEÑO DE SOFTWARE ORIENTADO A OBJETOS

TESIS

Que para obtener el grado de
Maestro en Ingeniería
Campo Disciplinario Planeación
Presenta:
Ing. Xavier Rojel Martínez

Directora de tesis: Dra. Patricia E. Balderas Cañas

Ciudad Universitaria **Mayo 2004**

Con cariño dedico este trabajo a mi familia:

A la profesora Teresita Martínez Pérez (mi madre) por haberme enseñado a leer y escribir, al profesor Aurelio Javier Rojel Ramírez (mi padre) por haberme inculcado el amor al estudio.

A Brenda por su amor brindado durante los años de estudio de la maestría.

A mis hermanos.

Mi mas sincero agradecimiento a la Dra. Patricia Balderas Cañas por el infinito apoyo que me brindó durante toda mi estancia en el posgrado, y por su atinada dirección en la realización de este trabajo. No hay palabras para agradecer la amistad y consejos de una mujer que da tanto, por el sólo hecho de difundir el conocimiento.

Mi agradecimiento a todos mis profesores del posgrado.

Gracias al CONACYT por la beca que me otorgó.

Gracias a la DGEP por el apoyo que me brindó por medio de la beca otorgada.

Gracias a la UNAM que una vez más me abrió sus puertas.

RESUMEN

Hace más de tres décadas se acuñó el término de crisis del software el cual indica entre otras cosas, que el software es caro, poco fiable y escaso. Estos elementos conforman una problemática que ha costado grandes cantidades de recursos a las empresas en todo el mundo. La ingeniería del software, surgió a raíz de la agudización de la crisis del software y a través de los años ha evolucionado, para combatir a la crisis del software, que se ha visto acrecentada con los desarrollos de software de gran tamaño. La evolución de la ingeniería del software ha llevado a nuevas formas de analizar, diseñar, desarrollar y mantener programas computacionales, dentro de las que destaca la orientación a objetos.

Existen en el mundo diversas metodologías, técnicas, guías, herramientas, etc., englobadas dentro de la ingeniería del software y con una orientación a objetos, las cuales han combatido a la crisis del software y han atenuado sus efectos, pero no han podido erradicarla, debido principalmente a que en general, tienen deficiencias en sus procesos de análisis de los sistemas organizacionales. Debido a esto, se generan diseños de software deficientes, con errores y alejados de la realidad que se vive en el sistema.

Este trabajo es una guía de análisis de sistemas organizacionales, en donde se requiere diseñar software de gran tamaño, basada en el enfoque sistémico, elemento de la ingeniería de sistemas. La guía comienza estableciendo el marco teórico necesario para el ingeniero en computación o el analista de sistemas, que esté realizando el análisis de un sistema, después ubica a la empresa en tiempo, lugar y sector, y la descompone en módulos o subsistemas mostrando los elementos y relaciones a diferentes niveles de detalle, utilizando elementos como el de cadena de valor y mapas conceptuales. Por último, la guía hace un estudio de los actores del sistema. Como parte de esta guía también se propone el detectar y corregir problemas, a fin de evitar repetirlos en los programas computacionales.

El objetivo de la creación de esta guía es establecer un puente entre la ingeniería de sistemas y la ingeniería del software, a fin de que la primera aporte los elementos necesarios para que la segunda genere diseños orientados a objetos congruentes con el sistema. La guía se aplicó en un caso de desarrollo de software para una empresa comercializadora de refacciones y para una planta de tratamiento de aguas residuales, obteniéndose en ambos casos una comprensión completa de los sistemas, que redundó en el ahorro de costos en el caso de la empresa comercializadora y en la elaboración de programas computacionales para modelación matemática de procesos, en el caso de la planta.

INDICE

1 INTRODUCCIÓN

1.1 Antecedentes

La mayoría de las empresas en nuestro país cuentan con programas informáticos que son de diferentes tamaños e infraestructuras, tenemos desde los que son instalados en una sola computadora hasta los que utilizan redes de comunicación a nivel nacional e internacional. Además puedo afirmar que el uso de la Internet ha tomado un lugar preponderante en la vida empresarial. El uso de computadoras se ha vuelto algo cotidiano y esencial en la vida de cualquier organización. Pero existe un gran problema dentro de esas organizaciones usuarias de programas informáticos y es el elevado costo de diseño, construcción y mantenimiento de dichos programas. Este costo puede ser medido desde distintos puntos de vista, que van desde los de tiempo, dinero y calidad de los programas. Esto no es un problema nuevo, tiene ya varias décadas.

> *"En 1968 en una conferencia sobre software, patrocinada por la OTAN se asumió el término de ingeniería del software y crisis del software. Con estos términos se quería expresar que el software era caro, poco fiable y escaso"*[1].

A raíz de la crisis del software nació la Ingeniería del Software, como una rama de la ingeniería que contrarrestaría los efectos de dicha crisis. "La ingeniería de software es el área de la computación que tiene que ver con las técnicas de construcción de sistemas de software de gran tamaño"[2], es este tipo de software el que se estudia en este trabajo.

De 1968 a la fecha las cosas no han cambiado mucho, con respecto a la crisis del software, excepto tal vez por el hecho de que el software ya no es tan escaso. Se puede apreciar que el problema no es reciente y ha sido analizado ampliamente, debido a esto encontraremos diferentes formas, métodos, técnicas, herramientas, etc. que han tratado de hacer que esta crisis del software se vea reducida; la ingeniería del software engloba la mayoría de esas formas, métodos, técnicas, herramientas etc. y se ha visto estudiada y mejorada a través de los años, pero no ha podido abatir la crisis del software.

En la revista Byte del mes de octubre de 1987 Brand Cox dijo: "Existe una bala de plata. Es un arma tremendamente potente, impulsada por vastas fuerzas económicas a la que nuevos

[1] Joyanes Luis(1996), p.5

obstáculos técnicos sólo pueden resistir brevemente. La bala de plata es un cambio cultural en lugar de un cambio tecnológico. Es un nuevo paradigma; una revolución industrial basada en partes reutilizables e intercambiables que modificarán el universo del software, de igual modo que la revolución industrial cambió la forma de fabricación"[3], esa bala de plata es la programación orientada a objetos.

Esa bala de plata se vino a colocar como una de las herramientas técnicas más importantes de los últimos tiempos en cuanto a computación se refiere; se mejoró en mucho el software creado, se redujeron los problemas como fallas por pérdida de datos, bloqueos, interferencias, lentitud, etc. pero las organizaciones siguen teniendo problemas referentes a relaciones entre áreas, intercambio de información, métodos de trabajo bajo ambientes computacionales, errores de interpretación de datos, etc., esto quiere decir que el estudio de las organizaciones enfocado a la construcción de sistemas computacionales, todavía se encuentra en formas incipientes, debido a que no se le ha dado la atención necesaria.

Por otro lado, el estudio de las organizaciones ha crecido considerablemente, se han podido mejorar formas de trabajo, se han solucionado problemas, se han establecido métodos y teorías que hacen que una organización trabaje de forma adecuada. Una de las disciplinas que se ha encargado de esto es la ingeniería de sistemas y en forma particular el enfoque sistémico, el cual se ha dado a la tarea entre otras cosas, de estudiar y mejorar las partes de una organización y sus interrelaciones.

1.2 Motivación del estudio

La ingeniería en computación es una de las industrias que más ha evolucionado en los últimos tiempos, alguna vez se le comparó con la industria automotriz, diciendo que si esta última hubiera tenido en algún momento los mismos recursos que se le han destinado a la primera para investigación y desarrollo durante las últimas décadas, actualmente los autos recorrerían las carreteras a velocidades impresionantes, alimentadas por unas cuantas cucharadas de combustible. Esto no es una comparación hecha a la ligera, las computadoras de hace cuatro décadas ocupaban espacios del tamaño de un dormitorio, y procesaban en horas cantidades de información similares a las que hoy en día cualquier computadora de bolsillo procesa en fracciones de segundo.

[2] Cohoon-Davison (2000), p. 28-29
[3] Joyanes Luis (1996), p.7

La vorágine del crecimiento de la computación se ha dado de manera exponencial, esto ha tenido consecuencias en todas las sociedades del mundo; es menester centrarme en lo que corresponde a la sociedad mexicana, la cual se encuentra teniendo necesidades y problemas (que expongo a lo largo de este trabajo) con el desarrollo de software computacional de gran tamaño, todo esto combinado con nuevas formas de trabajo que se han desarrollado durante los dos últimos lustros en las empresas, en específico la creación de organizaciones de consultoría en computación dentro del nuevo modelo de trabajo empresarial llamado *outsourcing*[4], dicho modelo nos ha llevado a la creación de numerosas empresas de consultoría computacional, de diferentes tamaños y perspectivas, que utilizan diversos métodos de trabajo, algunos loables e interesantes y otros del todo pragmáticos y alejados de la ingeniería del software.

El campo de la construcción de sistemas computacionales de gran tamaño, enfrenta a las consultorías y en general a los ingenieros en computación a diversos problemas cuya causa no tiene que ver con las herramientas computacionales o los equipos que se utilizan hoy en día, que dicho sea de paso, son sumamente poderosos, estos problemas tienen su origen en la forma en que se analizan a las organizaciones. Hoy en día una gran cantidad de universidades en todo el mundo dedicadas a la investigación en computación, están dirigiendo la mirada a otras ramas del conocimiento que se centran en el análisis de organizaciones, que si bien no son elementos del todo relacionados con la informática, en la práctica ya se ha demostrado que son elementos complementarios ideales para el desarrollo de software de gran tamaño.

La Maestría en Ingeniería de Sistemas campo disciplinario Planeación, es por si sola una herramienta poderosa para el análisis de organizaciones, y para el Ingeniero en Computación se vuelve una guía fundamental para introducirse y analizar empresas, con miras a implementar software de gran tamaño. La División de Estudios de Posgrado de la Facultad de Ingeniería en la UNAM, forma "profesionales con alto nivel académico, capaces de satisfacer las necesidades tecnológicas y de investigación de la era moderna en diversas áreas de la ingeniería"[5], por todo esto, el presente trabajo es una aplicación de los conocimientos y experiencia adquirida en la Maestría, aunada a lo mejor del conocimiento en ingeniería de software vigente hoy en día, con el fin de de generar una guía de análisis

[4] Outsourcing es la contratación de empresas externas especializadas para realizar algún proceso de la organización, en lugar de utilizar personal del organigrama. Cabe aclarar que este fenómeno no es privativo de la computación, es una nueva forma de trabajo a nivel general dentro de las empresas en México, con esto se evitan la contratación en forma definitiva de empleados por medio de contratación de despachos especializados, la especialización de estos despachos es una ventaja para la empresa que los contrata.

organizacional para el diseño de programas orientados a objetos bajo un enfoque sistémico, soportada fuertemente por la teoría, pero a la vez sencilla y ágil de aplicar, que responda a las necesidades de la sociedad mexicana en el ámbito de desarrollo de software computacional de gran tamaño, teniendo siempre un enfoque pragmático y holístico.

1.3 Formulación de la problemática

La problemática se plantea en el mapa conceptual mostrado en la *figura 1.1* en donde se explica de manera general cómo las organizaciones, se ven en la necesidad de utilizar programas computacionales, para poder lidiar con las grandes cantidades de información que generan y usan, pero los procesamientos y el análisis de información muchas veces no son los correctos, esto generado de un estudio deficiente de las necesidades del usuario, lo que da como resultado la generación de problemas que más adelante se explican. Además de que de manera inherente, los programas computacionales también tienen problemas que son el resultado de fallas técnicas, esto como resultado de un bajo dominio de las herramientas de desarrollo de sistemas como lenguajes de programación, manejadores de bases de datos, compiladores, etc.

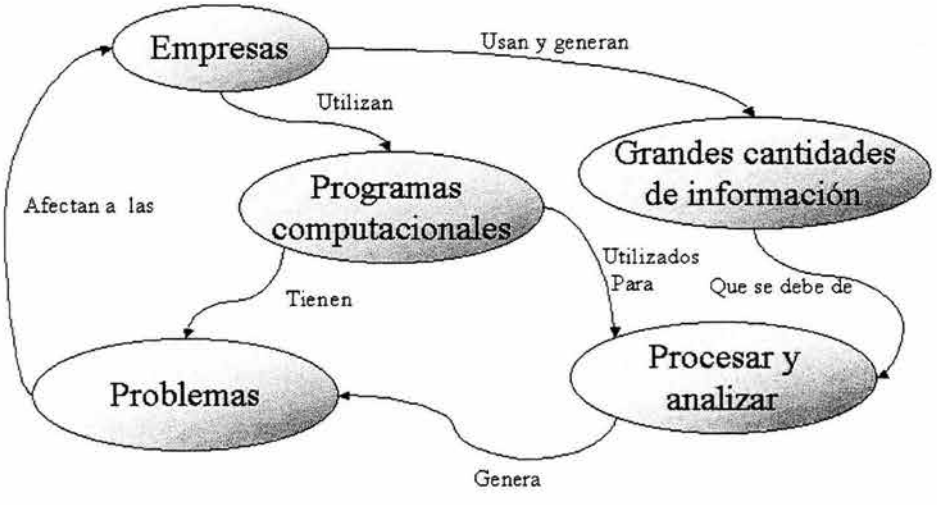


Figura 1.1 Formulación de la problemática

[5] Tomado de folletos de información de la División de Estudios de Posgrado de la Facultad de Ingeniería UNAM.

En el mapa conceptual, se habla de problemas de una manera general, pero es necesario ser un poco más puntual en cuanto a los problemas más frecuentes y generales que se presentan en las empresas que utilizan programas computacionales, a continuación menciono sólo algunos de los problemas que son el resultado de análisis deficientes de los requerimientos de los usuarios:

1. Los costos (en tiempo y dinero principalmente) de construcción son demasiado elevados.

2. El costo de mantenimiento, se torna la mayoría de las veces más alto que el costo de construcción.

3. Los programas se convierten por si solos en un elemento al cual hay que asignar personal, tiempo, dinero, creando muchas veces grandes áreas de sistemas; otra opción es contratar consultorías que ofrecen sus servicios, pero se crean dependencias funcionales sumamente peligrosas para la empresa, trayendo consigo un alto costo para la misma.

4. Los programas no cumplen cabalmente con las expectativas de la empresa.

5. El número de fallas al inicio, es considerablemente alto, llega un periodo de estabilización, pero por lo regular siempre sucede algo inesperado, y el programa termina siendo desechado, y una vez que se empieza la construcción de un nuevo sistema se comienza como si nunca hubiera habido un programa de computadora en la empresa.

6. Nunca se alcanza una estabilidad de uso de los programas computacionales, siempre se está trabajando en nuevos desarrollos.

En la *figura 1.2* observamos los aspectos que un sistema computacional correctamente diseñado y construido debe cubrir, y qué se requiere para lograr esto. Para poder plantear la problemática considero que se deben de tomar en cuenta estos aspectos que hacen que un sistema computacional trabaje correctamente, así como las actividades qué se requieren realizar para poder obtener un sistema computacional que responda a las necesidades de los usuarios.

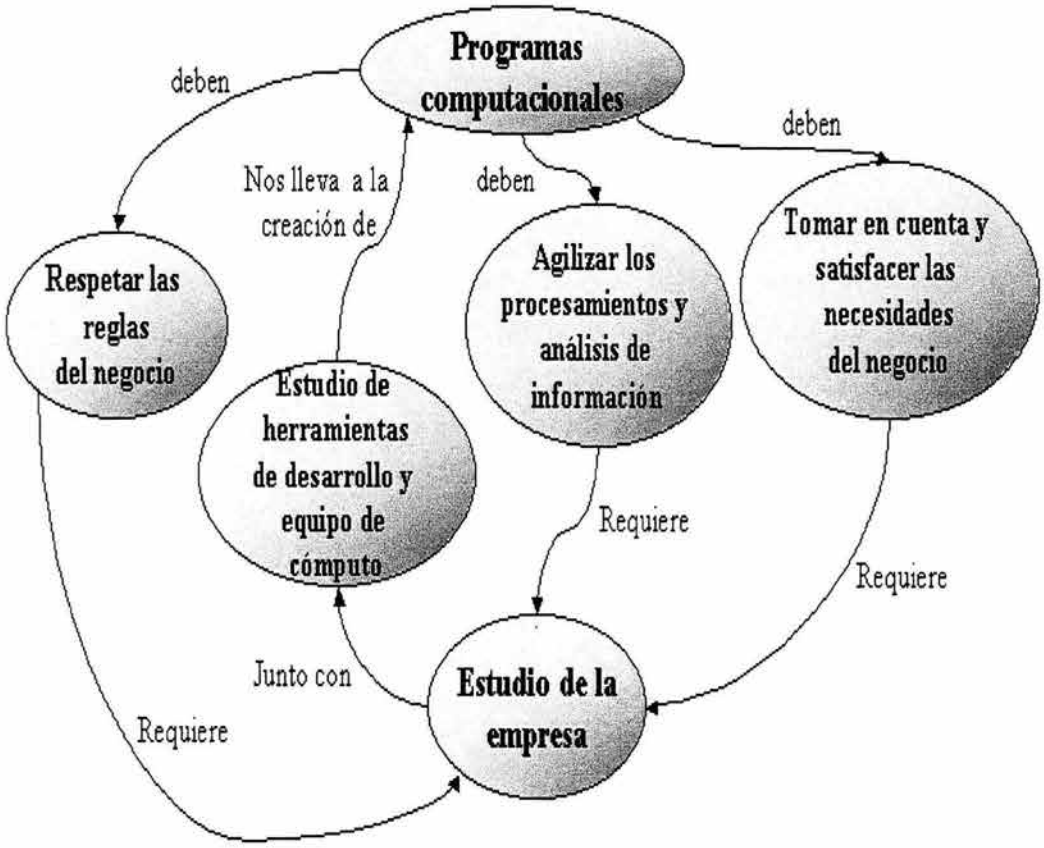


Figura 1.2 Programas computacionales y el estudio de la empresa

En la *figura 1.3* observamos los aspectos que de no cumplirse generan problemas a la empresa, estos aspectos tienen un estrecha relación con el estudio de la organización, lo cual podemos ver en la figura 1.2; no es difícil deducir de estas figuras que el problema por resolver tiene que ver con la forma en que se está analizando a la empresa, ya que este análisis tiene relación con las herramientas de desarrollo y equipo de cómputo a utilizar, con las necesidades y reglas del negocio, así como con la forma en que se procesa y se analiza la información.

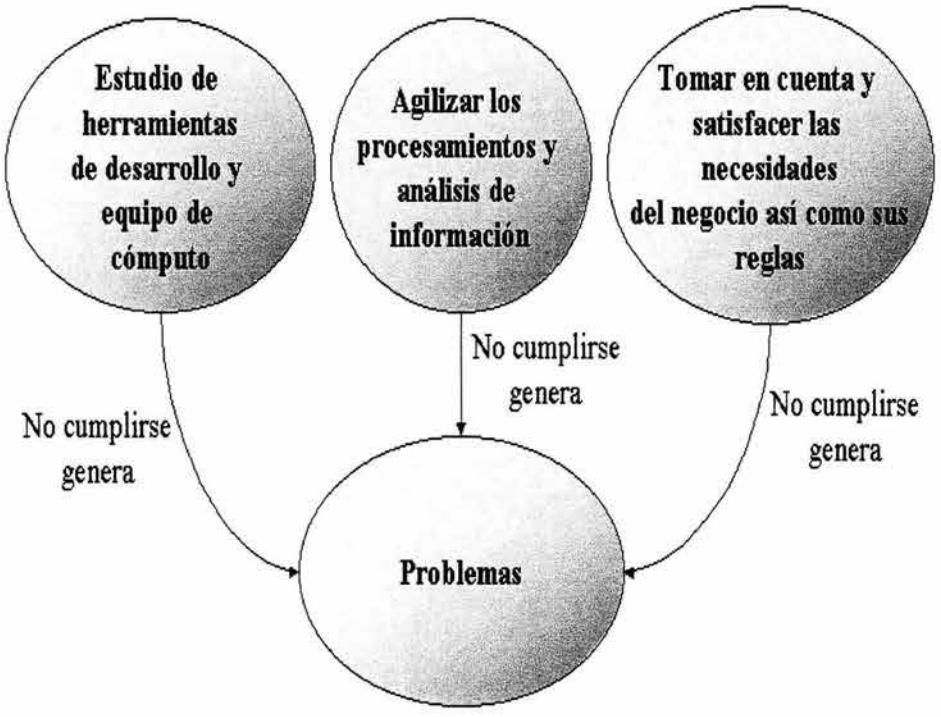


Figura 1.3 Generadores de problemas

Por todo lo anterior puedo decir que la problemática por resolver es el combatir el deficiente estudio que se hace de las organizaciones y que va encaminado a dirigir los esfuerzos y recursos para el diseño y construcción de software para la empresa. En otras palabras el problema es establecer un puente entre aspectos netamente empresariales como los administrativos, contables, financieros, económicos, humanos etc. y sus relaciones que se dan en el ambiente empresarial, con los aspectos técnicos de la computación como son de hardware (computadoras, servidores, redes de comunicación, etc.) y de software (programas de desarrollo, programas de soporte, etc.). Y no sólo eso, además, el puente debe de convertirse en un punto de reunión entre empresarios e ingenieros en computación en donde se hable un lenguaje común, a fin de poder construir y diseñar sistemas computacionales que por si mismos justifiquen su construcción, y que puedan integrarse como una parte del sistema organizacional, mejorando el desempeño del mismo en lugar de degradarlo.

1.4 Objetivo

Generar una guía de análisis organizacional bajo un enfoque sistémico para el diseño de software orientado a objetos, que sirva de puente entre la ingeniería de sistemas y la ingeniería de software. La ingeniería de sistemas tiene que ver con la forma en que se concibe a una organización y sus relaciones internas. La ingeniería de software atiende al cómo se construyen los sistemas computacionales para las organizaciones. Respecto a la ingeniería de sistemas, la guía de análisis deberá tomar como punto de partida y eje general al enfoque sistémico. Por lo que concierne a la ingeniería de software tendrá como punto de partida y eje general a los métodos de programación orientados a objetos y sus distintas variantes. Todo lo anterior busca generar una guía para atender las necesidades específicas de las empresas mexicanas que utilizan software de gran tamaño.

1.5 Contenido

En este capítulo se explicaron los antecedentes de la problemática de la crisis del software que se vive actualmente en el ámbito de desarrollo de sistemas computacionales de gran tamaño en las empresas mexicanas. Doy a conocer mis motivaciones como ingeniero en computación y estudiante de la maestría en sistemas que me llevaron al desarrollo de una herramienta para combatir los problemas inherentes al desarrollo de software. Además, planteo la problemática presente en la construcción de software, vía algunos mapas conceptuales y presento el objetivo que persigue este trabajo.

En el capítulo dos explico la crisis del software cuyo estudio se da dentro de la ingeniería del software. Diserto acerca de la diferencia entre software de gran tamaño o industrial y software no industrial o personal, debido a que es el primero del que me ocupo en este trabajo. Además, expongo conceptos básicos de la ingeniería de sistemas y el enfoque sistémico, centrándome en dos puntos principales: mapas conceptuales y análisis de stakeholders.

En el capítulo tres expongo conceptos básicos del análisis y programación orientado a objetos. Explico la razón por la cual el desarrollo de software es una tarea compleja, que da como resultado altos costos de desarrollo. Explico además la evolución de las herramientas de desarrollo desde sus inicios hasta su orientación a objetos. Analizo también una serie de metodologías clásicas de análisis y diseño orientado a objetos, mostrando los esfuerzos de estas para combatir la crisis del software y presento el Lenguaje de Modelado Unificado (UML), una herramienta muy de moda actualmente en el ámbito de desarrollo de software y

al parecer un próximo estándar, con el paso del tiempo. El capítulo termina con una explicación de un nuevo concepto de metodologías que se han desarrollado durante el último lustro, y que están posicionándose en las empresas donde se desarrolla software.

Finalmente en el capítulo cuatro presento la guía de análisis organizacional que es el resultado de este estudio. Inicio con la propuesta de una serie de antecedentes de conocimientos que propongo debe tener el consultor computacional. Además presento los pasos de esta guía, explicándolos todos ellos con un ejemplo real de un proyecto realizado para una empresa comercializadora de refacciones automotrices de la Ciudad de México.

2 INGENIERÍA DEL SOFTWARE Y ENFOQUE SISTÉMICO

2.1 Introducción

Han habido a lo largo de las últimas cuatro décadas diversos estudios encaminados a resolver los problemas que enfrenta el diseño, construcción y vida del software; para poder desarrollar una guía que atienda a las necesidades de las empresas en México fue necesario estudiar y analizar una serie de elementos teóricos tanto de la ciencia de la computación y del análisis de sistemas que han sido discutidos ampliamente por diversos autores y de los cuales tomé elementos que considero esenciales para la construcción de la guía que en este trabajo presento.

2.2 Crisis del software

Durante la década de los sesenta se dio un auge importante en la construcción de software, muchas empresas llevaban al menos una década construyendo programas computacionales para satisfacer diversas necesidades, el software en estos años principalmente se enfocaba a resolver problemas gubernamentales, militares o de empresas muy grandes, con el tiempo las empresas se dieron cuenta que el software que estaban utilizando era caro, escaso y poco fiable, elementos que dieron origen al término crisis del software.

La crisis del software a grandes rasgos, es una serie de problemas que enfrentan los ingenieros en computación que se encargan de construir programas computacionales, y cuyos principales síntomas son los altos costos y largos periodos de construcción de esos programas, y no obstante todo esto, una vez que han sido construidos, continúan los problemas, manifestándose en fallas continuas de los programas, lo que implica el uso de mas recursos para poder adecuarlos correctamente, todo esto hace que el software sea caro y poco fiable; hoy en día la escasez del software puede ponerse en duda, ya que muchas empresas cuentan con software, pero lo que no está a discusión ni puede cuestionarse es que el software que cubra cabalmente las necesidades del usuario y que sea fiable y barato, sigue siendo escaso.

Seguramente el lector de este trabajo puede haber sido testigo de esa crisis del software, en donde esa función de testigo puede llevarse a cabo desde dos puntos de vista, desde el punto de vista del constructor del programa computacional o como la persona de la empresa, fábrica, organización etc. que utiliza dicho programa. Pondré de ejemplo cualquier organización (me centraré principalmente en las empresas) que haya necesitado el uso de algún programa computacional para poder hacer frente a las cargas excesivas de información.

Sin temor a equivocarme, puedo decir que la historia se desarrolló en la mayoría de los casos de la siguiente manera: las personas que necesitaban un programa de computación (usuarios) contactaron a un experto (o aun un grupo de expertos) que pudiera desarrollar un sistema computacional. Dicho experto(s) podría(n) ser parte de la empresa o agente(s) externo(s). El usuario explicó los problemas que tiene con las cargas tan grandes de trabajo y el experto en computación tomó nota de todos esos detalles. Este tipo de pláticas se desarrollaron por cierto tiempo, casi nunca más allá de un mes, posteriormente, el experto, se presentó ante el usuario con un sistema que a primera vista resultaba impresionante por su rapidez y estética, pero que en el primer día de uso dio problemas de distintos tipos como pérdida de información, dificultad de manejo y lentitud, una vez que ya se habían introducido cantidades considerables de datos.

Obviamente los primeros errores se compusieron, pero al parecer sin explicación alguna aparecieron más, llegando a un punto en el cual el usuario empezó a dudar de si el programa que se le ofreció era la panacea que solucionaría todos sus problemas, cuando realmente le estaba dando más problemas de los que tenía antes del uso de este.

La historia continúa casi de igual manera, hasta que, algún tiempo después, el programa en computación aparentemente no tenía errores; sólo algunos que para el usuario le eran naturales y explicables, como al término del año, con un cambio en el mes, en la compra nuevo equipo, etc. Esto es, el usuario ya ve como naturales los errores que tiempo atrás representaban un serio dolor de cabeza. Después de todo esto, la historia no tiene un final feliz, ya que de alguna forma se piensa y se trabaja en el uso de otro programa de computación más potente con nuevos elementos que faciliten el trabajo, comenzando la historia una vez más, con problemas similares y otros nuevos, después de uno o dos años de funcionamiento del programa anterior.

El ejemplo planteado en el párrafo anterior, en la práctica significa grandes inversiones de recursos monetarios, humanos y de tiempo, lo que implica la pérdida de recursos para las empresas.

2.3 Ingeniería del software, sus ventajas y limitaciones

Además de la definición inicial del capítulo uno, "la ingeniería de software es el área de la computación que tiene que ver con las técnicas de construcción de sistemas de software de gran tamaño", a continuación presento otras definiciones de Ingeniería del Software.

Definición 1

"Ingeniería del software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable que sea fiable y trabaje en máquinas reales"[6]

Esta definición fue dada por el señor Bauer uno de los principales organizadores de la conferencia de software patrocinada por la OTAN en 1968 que mencioné al inicio del capítulo 1.

Definición 2

"La aplicación de un enfoque sistemático disciplinado y cuantificable en el desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería del software"[7]

Esta definición es la utilizada por la IEEE (The Institute of electrical and electronics engineers, inc.).

Definición 3

"La ingeniería del software es una disciplina o área de la Informática o Ciencias de la Computación, que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo"[8]

[6] Pressman Roger S. (1998), p. xxii

[7] Ibídem

[8] Ibídem, p. xxi

Para establecer una adecuada panorámica de lo que es la ingeniería del software ser requiere estudiar otros términos relacionados. Del diccionario en línea de La Real Academia de la Lengua Española[9] obtuve las siguientes definiciones:

*"**Software.***
(Voz inglesa.).
1. *Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora. "*

*"**Ingeniero, ra.***
(De ingenio, máquina o artificio).
1. Persona que profesa la ingeniería o alguna de sus ramas. La forma en masculino para designar el femenino (Silvia es ingeniero).
2. Hombre que discurre con ingenio las trazas y modos de conseguir o ejecutar algo."

*"**Ingeniería.***
1. *Estudio y aplicación, por especialistas, de las diversas ramas de la tecnología.*
2. *Actividad profesional del ingeniero."*

De todas las definiciones anteriores existen elementos que me interesa rescatar a fin de poder entender cabalmente qué es la ingeniería del software, y así comprender sus ventajas y sus limitaciones. Como primera instancia quiero destacar la utilización de la ciencia por especialistas, lo que implica que la ingeniería del software no es algo empírico o casual; se debe de basar en la aplicación de metodologías y por consecuencia métodos[10], para resolver problemas que en el presente trabajo tienen que ver con desarrollos de software de gran tamaño, todo lo anterior no implica que al igual que otras ramas de la ingeniería no tenga un sustento en el ingenio del ingeniero, que a mi sentir es una cualidad básica de la ingeniería.

Si la ingeniería del software es una ciencia y se ha desarrollado hasta la actualidad como tal, ofreciendo a los ingenieros una nutrida cantidad de conocimientos y metodologías para el diseño, desarrollo y mantenimiento, de grandes sistemas de software, es aquí en donde surge la gran interrogante: ¿qué es lo que ha pasado que no permite que se llegue a un estado en las empresas en donde el crear software ya no tenga nada que ver con el término

[9] Diccionario de la Real Academia de la Lengua Española en línea (2004) http://buscon.rae.es/diccionario/drae.htm

[10] En este trabajo una metodología se asume como una colección o conjunto de métodos

de crisis del software?. En primera instancia, diversos autores (Graham, Booch etc.) señalan dos problemáticas que se explican a continuación:

- El sentido puramente teórico de muchas de estas metodologías, que si bien en las universidades se exponen y se aprecian como herramientas poderosas para el desarrollo de software, en la práctica son difíciles de utilizar, por diversos motivos entre los cuales destaca la complejidad que hace que su aplicación en el campo de trabajo consuma demasiado tiempo, que es uno de los recursos que en la práctica el ingeniero en computación no tiene.

- El exagerado pragmatismo de otras metodologías, las cuales al aplicarse en desarrollos de gran envergadura, se muestran como soluciones a corto plazo que desde el inicio de su aplicación dejan de ser consistentes por su casi nulo rigor académico y terminan por desecharse para pasar a utilizar al cien por ciento el sentido común y el ingenio del ingeniero, todo esto sin ningún soporte científico.

Es obvio que en más de tres décadas de trabajo, se debieron de haber generado metodologías adecuadas, con un equilibrio entre la teoría académica y el pragmatismo que las empresa requieren, pero es aquí donde existe otra interrogante, si las metodologías son adecuadas ¿qué les hace falta para resolver la crisis del software?. A mi parecer es que todas ellas dan por hecho que el ingeniero conoce el sistema para el cual se va a generar el software, por lo cual empiezan a trabajar y a involucrar a los elementos del sistema y sus relaciones, además dan por hecho que los sistemas en su mayoría son ideales, esto es, libres de problemas, transformaciones, complejidades etc., esto en la teoría puede funcionar, pero en la realidad, los sistemas son complejos y sufren entropía; además los elementos no están bien definidos y menos sus relaciones. Todo lo anterior entre muchos otros problemas inherentes a cualquier sistema.

Además de los problemas antes mencionados, es esencial tomar en cuenta el propio mercado de trabajo, el cual, en el ámbito del desarrollo de software es por si sólo sumamente complejo, la intensa competencia motiva el surgimiento de nuevas compañías en la arena computacional, las cuales ofrecen nuevas herramientas para desarrollo de software mucho más potentes que otras que nacieron hace sólo algunos meses atrás. Aunado a la rápida evolución de las herramientas existe por otro lado, la competencia de los desarrolladores que las utilizan, los cuales en la mayoría de los casos proponen como valor agregado a sus desarrollos el abatimiento de tiempos de trabajo, lo que implica frecuentemente, pobres análisis de los sistemas sobre los cuales se va a trabajar.

De todo lo anterior, puedo decir que la computación es una de las ramas del conocimiento en la cual la frase *renovarse o morir* es una premisa básica, el ingeniero en computación es un profesionista que debe de estar en constante evolución y preparación para poder responder a los mercados.

Como mencioné al inicio de este trabajo, hay que voltear a mirar a otras ramas del conocimiento, en específico al análisis de sistemas, para complementar a la ingeniería del software a fin de brindarle al ingeniero en computación herramientas que le permitan entender los sistemas organizacionales. Hasta entonces, que apliquen las metodologías adecuadas teniendo un sólido entendimiento del sistema, con todos sus elementos, relaciones, problemas y complejidades.

2.4 Software Personal Vs. Industrial

Existen dos principales tipos de software que podemos encontrar a nuestro alrededor. El software que está dedicado a satisfacer necesidades específicas de un individuo o de organizaciones pequeñas, que llamaré software personal, utilizado por ejemplo para:

- Controlar una biblioteca familiar
- Llevar los gastos de un hogar
- Registrar las ventas de una pequeña tienda de ropa
- Guardar datos de una investigación de un científico
- Editar música
- El control de un robot
- El control de citas de un médico
- Etc.

Por otro lado, encontramos el software cuya principal misión es satisfacer necesidades de grandes organizaciones, de los que menciono algunos casos de utilización para:

- Controlar las ventas de una cadena de tiendas de autoservicio
- La expedición de licencias de manejo en un estado del país
- El control de tráfico aéreo
- La recaudación fiscal
- El control de movimientos bancarios
- El control de ensamblado de autos

- El control de envíos internacionales
- Etc.

Es necesario dividir de esta forma los diferentes tipos de software que podemos encontrar debido a que el software que ocupa y preocupa al ingeniero de software es el que debe satisfacer las necesidades de grandes organizaciones. El primer tipo de software se deja de lado no por menosprecio a su importancia, o por que su construcción no tenga mérito, sino por que existen elementos que hacen que el segundo necesite de otro tipo de visión para su diseño y construcción, es este último al que Booch llama "software de dimensión industrial"[11], y que en adelante llamaré software industrial.

Existen diversos elementos que hacen que estos dos tipos de software difieran en la forma de conceptuarlos y construirlos; antes de examinar cada uno de estos elementos que afectan y repercuten en la forma de diseñar y dirigir los proyectos de software, debemos de tener en cuenta que en la forma en que se relacionan con los dos tipos de software existen excepciones, es decir, que si tienen un comportamiento regular con uno de los dos tipos de software, no quiere decir que siempre sea así. A continuación defino los elementos y la forma de relación entre los dos tipos de software y así como algunas excepciones.

- Usuarios Finales

Son las personas que utilizan los sistemas. En el software personal, por lo regular son individuos aislados o pequeñas organizaciones, pensemos en el médico que cuenta con un software para el control de sus pacientes, o en los encargados del taller mecánico que cuentan con un sistema para el control de sus clientes, y en el caso del software industrial pensemos en una empresa paraestatal con dos mil empleados, repartidos en por lo menos diez gerencias, o en una cadena de tiendas de autoservicio.

- Costo monetario de construcción

Es la inversión en dinero que se hace para construir el software. El los sistemas personales, la cantidad de dinero invertido por lo regular es baja, sigamos con el ejemplo del software de control de citas del médico, el cual puede ser realizado a un costo muy bajo por ejemplo por un estudiante de computación, aquí podemos ejemplificar una excepción con algún software

[11] Booch Grady (1996) , p. 4

para investigaciones científicas, que si bien lo ocupan un número muy reducido de personas, sus costos en dinero pueden llegar a ser muy altos. En el software industrial las inversiones monetarias que se tienen que realizar, por lo regular, siempre son altas, así, existen múltiples ejemplos de software que ha costado millones de dólares[12].

- Tiempo de construcción

Es el tiempo del que se dispone para construir el software. La construcción de software personal, por lo regular no guarda una carrera contra el tiempo, esto se debe a que las tareas que va a realizar el software pueden hacerse con facilidad por unas cuantas personas. Sin embargo, la construcción del software industrial, por lo regular, siempre tiene carreras contra el tiempo, debido a que su utilización genera ahorro de dinero, al acelerar y hacer eficientes los procesos de las organizaciones, esto trae de la mano el poder hacer más y de mejor las tareas en las empresas, en la mayoría de los casos.

- Conocimiento asociado

Es el conocimiento no computacional implicado para realizar la construcción de estos sistemas, en el caso del software personal, el conocimiento generalmente no es multidisciplinario, por ejemplo un músico y un ingeniero en computación el primero explicando y el segundo diseñando y desarrollando, podrán construir un software que satisfaga las necesidades del primero, en cambio, imaginemos el software para administrar una cadena de tiendas de autoservicios. A primera vista podemos pensar en la participación de los expertos de las áreas de bodegas, ventas, recursos humanos, contabilidad, finanzas, etc., es decir, estamos hablando de un gran número de personas involucradas expertas en diversas disciplinas

- Vida del software

Es el tiempo que el software será utilizado. El software personal, por su costo en dinero, puede ser sustituido por otro con facilidad mientras que el software industrial, no puede ser sustituido rápidamente, pensemos en el director de una empresa que proponga sustituir el software adquirido el año pasado, que tuvo un costo de varios millones de dólares, seguramente sería muy cuestionado por muchas personas, antes de que toda la empresa pensara en esto como una opción.

[12] Las cotizaciones de desarrollos de gran tamaño hechas por consultorías importantes en nuestro país, por lo regular se hacen en dólares.

- Grupos de desarrollo

Es el número de personas que construirán el software. El software personal generalmente es construido por programadores aislados o por pequeñas empresas constructoras de software; mientras que el software industrial, la mayoría de las veces requiere de grupos de desarrollo de software de al menos un par de docenas de programadores.

- Utilidad del software

Es la importancia que el usuario o la organización dan al uso del software. Las actividades realizadas por el software personal por lo regular pueden ser realizadas de manera relativamente fácil por unas cuantas personas, de esto se deriva que la utilidad asignada al software es baja, mientras que las tareas realizadas por el software industrial, en muchos casos se vuelven complejas de realizar aún contando con grandes cantidades de personal, por ejemplo el procesamiento de un censo de población de cualquier país.

En la tabla 2.1 presento el resumen de la comparación de los dos tipos de software atendiendo a los elementos antes mencionados:

Elemento del software	**Software personal**	**Software industrial**
Usuarios finales	Individuos aislados u organizaciones pequeñas	Grandes organizaciones
Costo de construcción	Bajo	Alto
Tiempo de construcción	Sin importancia	Esencial y escaso
Conocimiento asociado	Específico	Multidisciplinarlo
Vida del software	Corta	Larga
Grupos de desarrollo	Pequeños	Grandes
Utilidad	Baja	Alta

Tabla 2.1 Comparación de elementos del software personal vs. software industrial

Me permito hacer la analogía entre la ingeniería en computación y la ingeniería civil, para explicar el porqué se dejará de lado el software personal y nos centraremos en el software industrial. Cuando el hombre sólo necesita construir una pequeña casa, sabemos que no es necesario la utilización del conocimiento de la ingeniería civil para realizar esta tarea, no necesitamos un estudio de suelo, no necesitamos expertos en cimentaciones, en concretos, no necesitamos calcular la resistencia del acero a utilizar, esto mismo pasa con el software personal, no necesitamos utilizar la mayoría del conocimiento computacional que se ha

generado en los últimos cuarenta años, teniendo un poco de creatividad, inteligencia y cuidado, seguramente ni la casa se caerá, ni el software fallará y podrán satisfacer las necesidades de quién los requiere. Por otra parte, cuando el hombre necesitó construir canales de riego, pirámides, puentes, edificios, etc., se vio en la necesidad de utilizar el conocimiento de la ingeniería civil para llevar a buen término estas tareas, lo mismo pasa con el software industrial, el cual debe de utilizar algoritmos sofisticados, diseño de base de datos, estructuras de datos, técnicas de programación, etc., y en general mucho del conocimiento que se ha acumulado en las últimas décadas, para poder llevar a buen termino cualquier construcción de software de este tipo.

Si hablamos de software industrial, cuya principal característica es su relación con organizaciones de gran tamaño, es necesario conocer cómo estudiar a esas organizaciones que son sistemas; es por eso que a continuación se presentan elementos de ingeniería de sistemas que más adelante, en el planteamiento de la guía, serán utilizados.

2.5 Enfoque sistémico

En las siguientes páginas se explican algunos elementos de la ingeniería de sistemas y principalmente las bases conceptuales del enfoque sistémico, que son el eje de la guía que en este trabajo se presenta. En el capítulo cuatro algunos de estos se retoman como parte de la guía.

Defino en primer lugar a un sistema como un conjunto de elementos relacionados entre si.

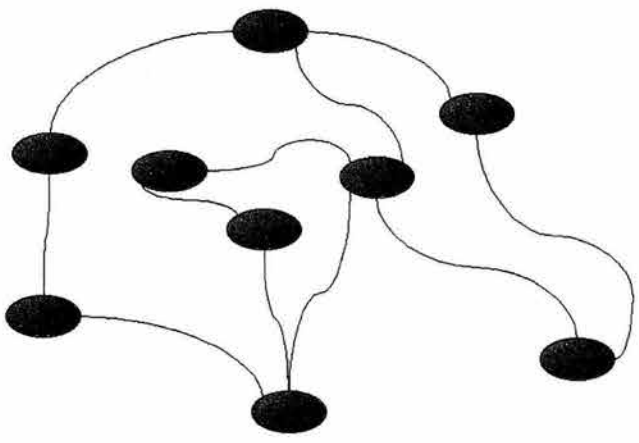


Figura 2.1 Representación abstracta de un sistema.

Respecto al enfoque sistémico presento una definición mostrada en las notas de clase del profesor Andrés Romo.

> *"El enfoque sistémico nos brinda herramientas para construir un marco general de referencia que nos simplifica el entendimiento de la masa compleja de cambio permanente que caracteriza a toda actividad humana."*[13]

Para hablar de un enfoque sistémico me refiero a las condiciones establecidas por Ackoff, las cuales explicaré con ejemplos basados en empresas para un mejor entendimiento, dejando un poco de lado la construcción de software y centrándome en la organización.

1.- Un sistema tiene una o más funciones definitorias.

Una empresa como sistema, tiene por objetivo central el generar ganancias, pero puede tener objetivos intermedios para alcanzar ese objetivo central. Por ejemplo, una asociación deportiva tiene como objetivo central promover la práctica de algún deporte.

2.- Cada parte en un subconjunto puede afectar el comportamiento o propiedades del todo.

Si en el departamento de compras (que puede verse como un subconjunto de una empresa) existe un comprador, que realiza de pésima forma su trabajo, eso podría traer como consecuencia que la organización empiece a tener problemas en el área de bodegas como resultado de compras masivas de mercancía que es difícil de vender. También, el departamento de finanzas, empezaría a notar reducciones en las ganancias, debido a que se están generando inventarios con excedentes invendibles, lo que repercute directamente sobre las ganancias de la empresa. De esta forma vemos cómo un elemento de un subconjunto está afectando al todo; cabe aclarar que la afectación también puede ser en forma positiva.

3.- Existe un subconjunto que es suficiente para llevar a cabo los objetivos del todo. Cada una de las partes es necesaria pero insuficiente para cumplir el objetivo por si misma.

Siguiendo con el ejemplo de la empresa, ya se sabe que el objetivo de la empresa es generar ganancias para el empresario; la empresa puede tener diversos departamentos como el de contabilidad, el de mercadotecnia, ventas, inventarios, etc. Se pueden quitar algunos departamentos y el objetivo de la empresa aún se sigue cumpliendo, por ejemplo,

[13] Romo Andrés, *Notas de clase de Evaluación de proyectos DEPFI UNAM*, p. I-8

de entrada podríamos pensar en quitar el departamento de mercadotecnia, que aunque es parte importante de la empresa, esta puede funcionar sin él. Un ejemplo claro de lo anterior es que muchas empresas no tienen ese tipo de departamentos y el objetivo se sigue cumpliendo con el trabajo de los demás departamentos.

4.- El comportamiento o propiedades propias o sobre el sistema de cada parte, depende del comportamiento o propiedades de por lo menos otra parte del sistema.

Fácilmente se puede dar un ejemplo pensando en departamentos que se relacionan fuertemente en una empresa. Cuántas veces no hemos oído que el departamento de bodegas se queja del departamento de recepción, que hace que se cargue el trabajo a determinadas horas por los malos procedimientos de recepción. O también, podemos pensar en los gerentes de ventas que se quejan de no saber qué vender, porque el departamento de bodega no informa correctamente de las existencias.

5.- El efecto sobre el sistema de un subconjunto de partes, depende del comportamiento de por lo menos otro subconjunto.

Pensemos por ejemplo en dos de las áreas más importantes de cualquier empresa, la de producción conformada por el departamento de bodegas, el departamento de troquelado, el departamento de acabados, etc., y el área de administración y finanzas, compuesta por el departamento de contabilidad, recursos humanos, finanzas, administración, etc. y en las pugnas que día a día se establecen entre las dos áreas y la dependencia que tiene una sobre la otra para poder conducir a la empresa hacia el logro de sus objetivo.

Vale la pena detenernos, en este momento y empezar a intuir, la conveniencia de tener un pensamiento sistémico, y esto como primera aproximación es fácil de contestar, debido a que uno de los principales problemas a los que nos hemos enfrentado múltiples consultores computacionales es que nuestro trabajo, ya sea por nuestra propia planeación o por pedimento de los empresarios, comienza siempre por un departamento de la organización, y cuando creemos que los problemas de ese departamento están solucionados, nos damos cuenta que no es así. Buscamos una razón, porque ya hemos invertido una gran cantidad de recursos y lo mejor de nuestros conocimientos, pero la respuesta es sencilla, si el departamento existiera sólo dentro de la organización seguramente nuestras soluciones harían que funcionara de manera adecuada, pero son las relaciones con las demás áreas de la empresa, las que hacen que nuestro trabajo se estropee, y la historia puede tornarse todavía más pesimista; una vez que ya sabemos que los problemas vienen de fuera del

departamento, nos avocamos a solucionar los problemas de los demás departamentos, pero una vez más vistos como entidades sin relación. Por eso cuando queremos que la empresa en general marche de manera adecuada utilizando un sistema computacional, tenemos una serie de problemas que a pesar de nuestros esfuerzos no se han resuelto con la automatización, palabra que defino a continuación:

Automatización: Es la implantación de sistemas computacionales con el fin de apoyar el trabajo diario de la empresa.

Para el objetivo de este trabajo no ahondaremos en otros tipos de automatizaciones, tales como las hechas en plantas industriales realizadas con maquinaria o con robots especializados, me centraré solamente en el ámbito de programas de computadora, las más de las veces administrativos.

Ya se sabe lo que es automatizar, rápidamente podemos dilucidar que una automatización siempre será caótica si existen problemas en la empresa. Esto significa que se van a automatizar problemas cuando no se tiene una visión sistémica que auxilie en la difícil tarea de analizar para entender a la empresa, a fin de resolver los problemas que se presentan en ésta, y generar relaciones eficientes entre las diferentes áreas de la organización vía herramientas computacionales.

Debemos entender de ahora en adelante que el implantar software en las organizaciones contribuye al mejoramiento del sistema llamado empresa, por lo cual, hablaré de forma indistinta del análisis de sistemas como una vía de mejoramiento de los sistemas o de implantación de software, ya que en la práctica van de la mano y cuando no sucede esto, los desarrollos de software y su implantación en las organizaciones se vuelve un proceso caótico que contribuye al deterioro de la organización.

2.5.1 Viejas tendencias del análisis de sistemas

La primeras teorías de administración nos mostraban a las empresas como cajas cerradas, como maquinaria susceptible de mejoras, pero no tomaban en cuenta la relación de ésta con su medio ambiente, esto no quiere decir que dichas teorías estuvieran equivocadas, si no más bien pienso yo que correspondían al momento histórico en el cual nacieron y florecieron, pero ahora, en el momento en el cual casi todo un continente usa una misma moneda llamada Euro, y en donde ellos han formado el bloque económico más importante a nivel mundial, conocido como la Unión Europea, el enfoque ha cambiado. Esto implica,

simplemente una caída de fronteras que hasta el momento son económicas, no sabemos hasta dónde pueda llegar este tipo de efectos globalizadores. Esto debe de ponernos alertas, debido a que es un indicador de que algo está pasando en el mundo y que está afectando el desarrollo de los países, y por ende a sus empresas. México es signo claro de este efecto globalizador, al pertenecer a un Tratado de Libre Comercio en donde los movimientos económicos que se generan en otro país vienen a repercutir en el nuestro. En más alto grado quien no ha oído hablar del efecto Tequila, el efecto Dragón, el efecto Tango, que no obstante de ser fenómenos que se generan en distintas partes del planeta, han dado como resultado repercusiones de tipo económico en países geográficamente alejados.

Esto nos hace pensar que es importante ver a las empresas desde un punto de vista sistémico, es decir como parte de un todo, que ese todo en primera instancia podría ser la economía a nivel mundial, como por ejemplo cuando hablamos de empresas trasnacionales, o la economía a nivel nacional cuando hablamos de empresas de importancia dentro del país. Por esto se establecen niveles ambientales, que a mi juicio son los marcos de referencia en donde empieza el estudio que se pueda hacer hacia una empresa determinada para poder implantar un sistema computacional.

A continuación se mencionan los niveles ambientales definidos por Cleland:

> *"Interno: Los elementos que están dentro de la jurisdicción oficial de la organización.*
>
> *Operativo: Incluye a los clientes, proveedores y grupos de interés con los cuales la empresa tiene relación directa.*
>
> *Ambiente general: Conformado con el contexto nacional y global que define las condiciones sociales, políticas, regulatorias, económicas y tecnológicas"*[14]

De aquí podemos empezar a inferir los distintos tipos de estudios que deben de hacerse para el análisis de la empresa, esto quiere decir que dependiendo del tipo de empresa y su contexto en el ámbito económico debemos de empezar a poner atención en el tipo de ambientes que deben de estudiarse. A fin de poder mejorar el desempeño de la organización vía la implantación de software.

Este tipo de estudios deben de hacerse tomando en cuenta las realidades de la empresa, y tomando en cuenta la cantidad de recursos con que cuente esta, a fin de poder evaluar

[14] Romo Andrés, Ob. cit., p. I-10

cuales son los filtros adecuados con respecto a los tipos de estudios, esto en otras palabras quiere decir que, una empresa que sea líder en el mercado en una de las delegaciones del D. F. es claro que los estudios de ambiente global a nivel mundial no tendría mucho caso para su desarrollo, mientras que un estudio de proveedores y sus competidores tendría mucho que ver con su crecimiento y la buena implantación de un sistema computacional.

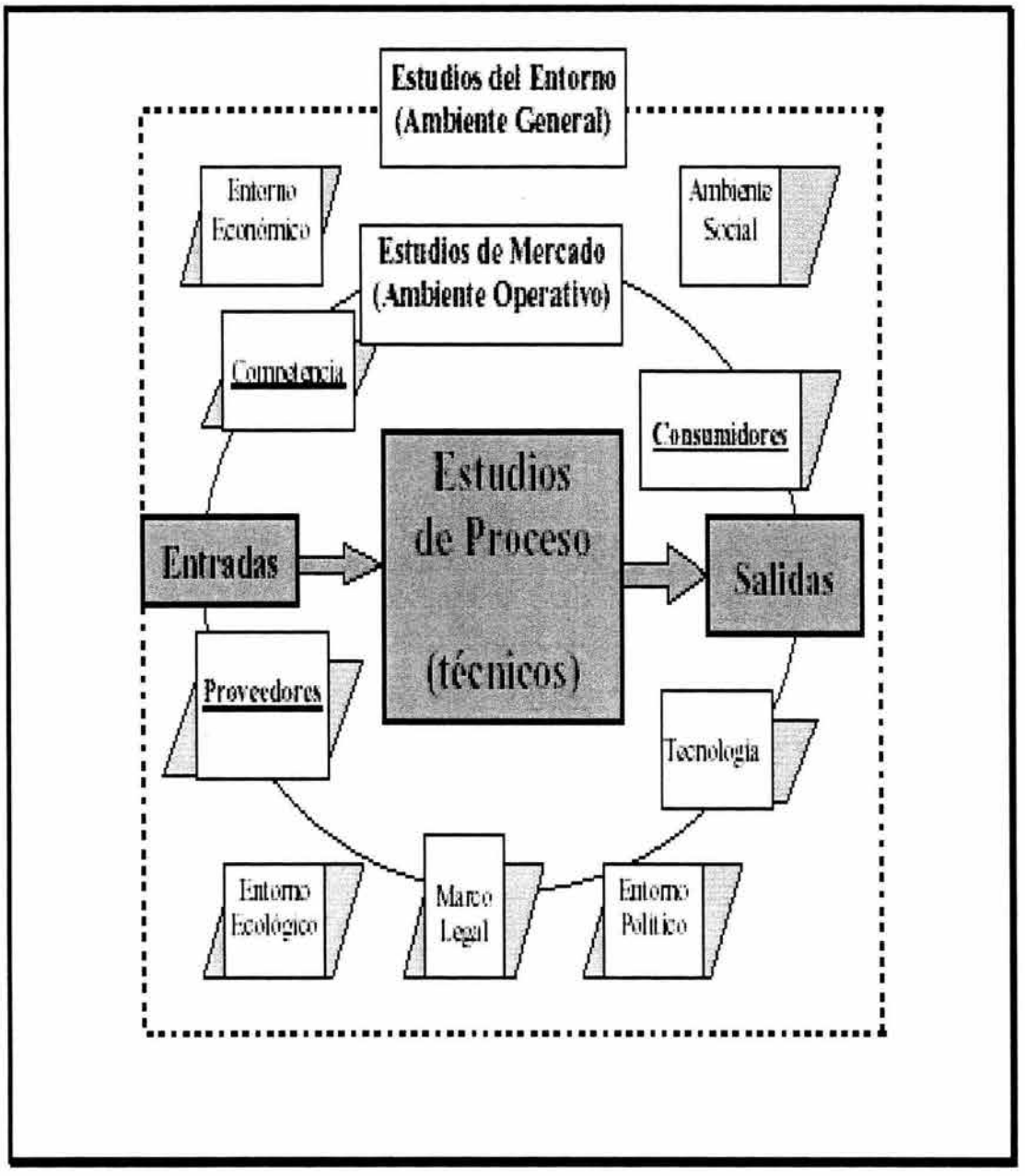


Figura 2.2 Estudios recomendados para el análisis y mejora de una empresa.[15]

[15] Adaptación de una ilustración de Romo Andrés (2000), p. I-9

2.5.2 Mapas conceptuales y sistemas suaves

Antes de comenzar a discernir esta parte del trabajo, es prudente dar una definición encontrada en un diccionario de mapa y de concepto:

> MAPA: "Representación convencional de toda o parte de la superficie esférica terrestre mediante su proyección en un plano a escala reducida. En el mapa se señalan los accidentes físicos, poblaciones etc. a base de símbolos o signos convencionales".[16]
>
> CONCEPTO: Simple representación mental de un objeto del que nada se afirma o se niega.[17]

Por lo anterior puedo decir que:

> MAPA CONCEPTUAL: Es una representación convencional por medio de símbolos o signos, de una visualización mental de una parte o de todo un sistema.

Con esto presento en la figura 2.3 un primer mapa conceptual, llamado de tipo caja negra hecho para una empresa de venta de refacciones automotrices de la Ciudad de México.

[16] Danae (1981)
[17] Ibidem

Figura 2.3 Primer mapa conceptual

Analicemos cómo este mapa conceptual engloba los conocimientos adquiridos a lo largo de este capítulo.

Primero. Da una primera visión sistémica de la empresa.

Segundo. Hace ver a la empresa enmarcada en los diferentes ambientes interno, operativo y ambiente general.

Tercero. Muestra los actores más importantes del sistema general e indica algunas primeras relaciones entre estos actores.

Cuarto. Utiliza el filtro de estudios de proyecto, ya que por ejemplo, sólo enmarcó a la empresa en un ambiente nacional, no es tan importante ver a la empresa en un ambiente internacional, debido a que su importancia está enmarcada a una parte del D. F.

Analicemos con un poco más de detalle el mapa conceptual aquí presentado para ver que en primer lugar engloba al sistema (la empresa) en su ambiente más general que para este caso es la ciudad de México, lo que implica que nuestro sistema es un subsistema de un sistema más grande, y como ya vimos nuestro subsistema puede afectar y ser afectado por las demás partes que conformen el sistema principal. En segundo término, se presenta el ambiente operacional, en donde se muestran los principales actores sobre los cuales se debe de hacer un estudio a fin de conocer los efectos del sistema sobre los otros actores y los efectos de la acción de los otros actores sobre el sistema. Por último se presenta el ambiente interno, en el cual se puntualizan los principales elementos, que vienen siendo las áreas de la empresa. También se muestran algunas salidas y entradas que sirven de relación entre los diferentes ambientes.

Hay que recalcar que la participación del cliente[18] en el desarrollo de este trabajo es primordial, este mapa puede ser incremental, esto es, puede irse mejorando hasta un nivel que presente la información adecuada, siempre cuidando de no cargar demasiado este tipo de gráficos. Algunos actores pueden desecharse y otros nuevos pueden incrementarse, todo a fin de que la "primera fotografía de la empresa" de verdad muestre la realidad del sistema en todos sus ambientes.

Ya he planteado una primera aproximación a los mapas conceptuales, pero como todo mapa, puede irse perfeccionando, y desagregándose, por esto es el momento de explicar otro tipo de mapas que nos ayudarán a un mejor entendimiento de la organización que estamos estudiando. A continuación comienzo a explicar otro tipo de mapas conceptuales propuestos por Peter Checkland.

[18] Hago alusión al cliente de manera indistinta por referirme a uno de los actores de la empresa que trabajarán de manera conjunta con el consultor

En la práctica de la ingeniería y durante la formación profesional se aprende a modelar sistemas, como por ejemplo los hidráulicos, mecánicos, térmicos, etc. Qué ingeniero no recuerda los problemas sobre la aplicación de fuerzas en ciertas direcciones, que se aplican en determinado punto de una viga, cuando las características de la viga son dadas, y de lo que se solicitaba calcular la fuerza ejercida en determinado punto. El procedimiento aprendido partía de la elaboración de un diagrama representando a la viga y a las fuerzas que muestro en la figura 2.4.

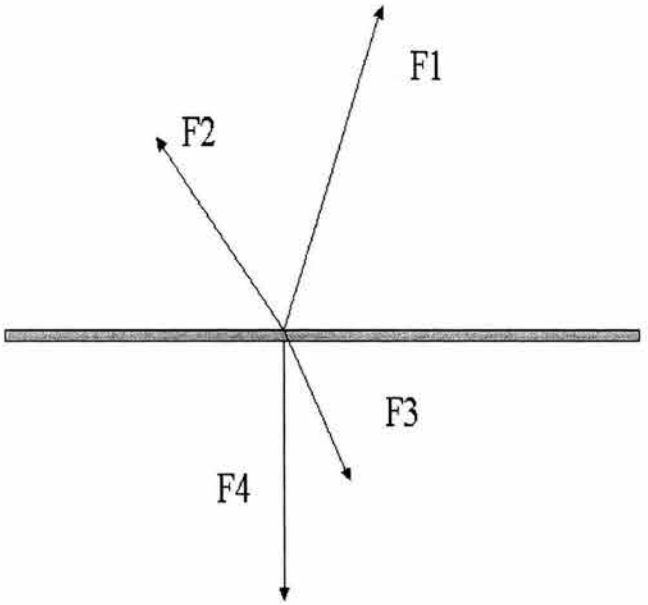


Fig. 2.4 Representación grafica de un conjunto de fuerzas aplicado a una viga

A continuación, se realizaban cálculos sencillos. La configuración anterior era una representación abstracta de la naturaleza. Se habían quitado los elementos innecesarios y se tenían sólo los elementos necesarios para el estudio.

Años después de egresados de la formación profesional los ingenieros de software se enfrentan a problemas de otra índole muy distinta al matemático o físico, como problemas administrativos, de desempeño de personal, de organización de departamentos, etc. Esto lleva a recordar con nostalgia los problemas escolares que no tienen comparación con los de resolver problemas de baja productividad en una empresa con cientos de trabajadores combinados con la implantación de software de gran tamaño.

Los problemas del campo profesional necesitan de una herramienta que ayude a simplificar un mundo tan complejo como el empresarial, donde surgen preguntas sobre cómo se puede modelar su funcionamiento para implantar un sistema computacional de gran tamaño que considere las pugnas entre los departamentos de ventas y mercadotecnia por ejemplo. El objetivo anterior puede parecer un poco ilógico, y hasta difícil de creer que se pueda lograr, pero para los ingenieros en computación nos resulta un poco familiar este problema y lo resolvemos parcialmente con la utilización del modelo entidad–relación para diseñar bases de datos, sin embargo esta herramienta es bastante limitada y tiene una desventaja, no se puede probar su efectividad hasta que un programa en computadora está funcionando en la empresa. Para dar a conocer la herramienta que nos ayudará a modelar este mundo real tan complejo, primero presentaré otro concepto.

SISTEMA SUAVE: Llamaré sistema suave a todo sistema que implique relaciones entre personas, que tenga que ver con elementos que no se pueden medir, tales como sentimientos, pensamientos, acciones etc.

Una forma sencilla de explicar cómo representar ese tipo de sistemas suaves es por medio de analizar la figura 2.5 que representa un modelo de cualquier religión. Este modelo fue desarrollado por Peter Checkland profesor y jefe del Departamento de Sistemas de la Universidad de Lancaster, Inglaterra en 1979.

Dicho modelo conceptual ejemplifica la forma en que se hacen estos mapas. Espero que el lector se sorprenda al ver la forma tan ingeniosa de representar algo que a primera vista se mira imposible. Hablar de religión implica un sin numero de elementos que podrían mencionarse.

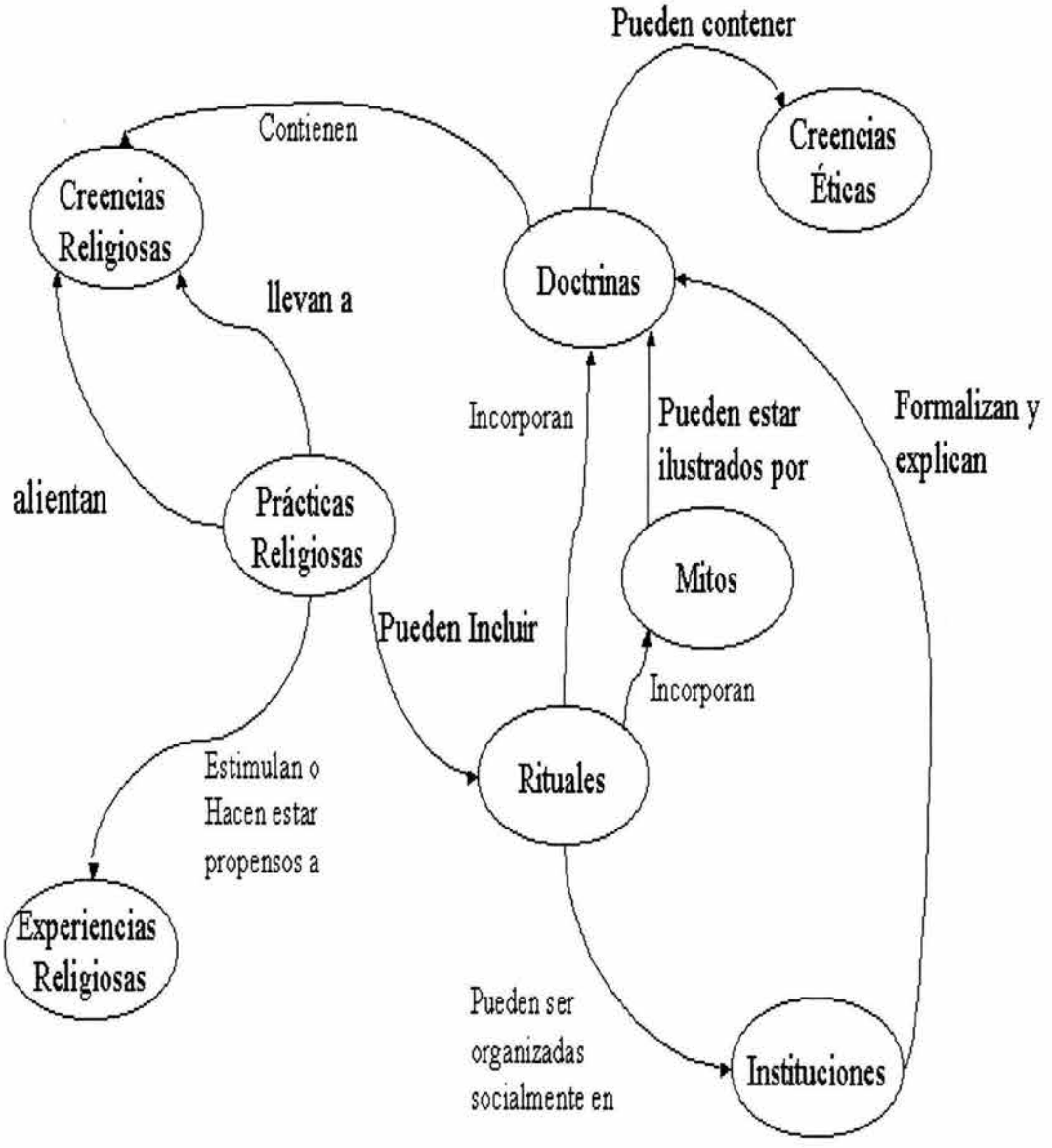


Fig. 2.5 Modelo conceptual de cualquier religión[19]

Aquí pediría al lector que tomara su tiempo, que en caso de que le haya sorprendido tanto como a mi este gráfico, lo disfrute y lo analice, que piense en la forma en que en una imagen está explicado un tema que ha sido discusión de muchos hombres por miles de años.

[19] Ilustración tomada de Checkland (1979)

Para el lector que esté inconforme con esta representación y que le pudiera parecer incongruente lo invito a que trate de encontrar qué elementos le faltan al modelo y qué le sobra y que piense por qué no esta de acuerdo con esta representación.

Para los dos tipos de lectores, lo que les puedo decir, es que en un modelo se implican elementos de un sistema complejo, no matemático, no físico ni económico, si no más bien filosófico. Que, para un estudio desde el punto de vista de sistemas nos da elementos que conforman al sistema llamado religión.

No es tema de este trabajo entrar en disertaciones filosóficas si es correcto representar de esta forma elementos socioculturales. Quizás los lectores estudiosos del ramo humanístico se podrán molestar con este tipo de representaciones, en cierto sentido tienen razón de hacerlo, este tipo de modelos pueden despreciar muchos elementos de un sistema suave, pero para la ingeniería que necesita sentar las bases para realizar estudios, financieros, administrativos, económicos, poblacionales, etc., encaminados a construir software industrial, es una excelente herramienta.

Es menester explicar mi entusiasmo hacia esta herramienta, y se debe a que la comparo con el modelado matemático que hacíamos en épocas de universidad, y que nos ayudaba a resolver complejos problemas físicos y matemáticos.

Es importante destacar algunas características del mapa conceptual que coinciden con algunos elementos de los modelos de sistemas duros (hidráulicos, mecánicos, etc.). Las características a destacar del mapa conceptual son importantes porque:

- De manera gráfica expone el sistema
- Descompone al sistema en sus partes
- Crea un punto de partida susceptible de mejoras
- Símbolos representan elementos
- Se representan las relaciones
- Por medio de símbolos se pueden llegar a ideas completas.
- Es un medio de estudio
- Puede ser tomado como un lenguaje para la solución de problemas.

El estado del arte de la creación de mapas conceptuales para sistemas suaves relacionados con la implantación de software es incipiente. La literatura es escasa, pero con lo que hay es suficiente para usar este lenguaje como herramienta en la difícil tarea de analizar empresas

para conocer a la organización además de detectar problemas y resolverlos todo esto para poder implantar programas computacionales que trabajen de forma óptima en la empresa.

Por lo anterior, no existen consensos bien establecidos de acuerdo a como deben de generarse estas representaciones. Pero aquí a forma de receta daré las pautas básicas[20] para la generación de estos modelos.

a) Definir la simbología a utilizar, Propongo usar óvalos para representar entidades (entendiendo como entidad una cosa, un pensamiento, un objeto etc.) y flechas para representar relaciones.

b) Utilizar entre cinco y nueve relaciones, casi siempre representadas por un verbo en infinitivo.

c) El mapa conceptual debe en general ser claro

d) Los mapas conceptuales pueden tener niveles de desagregación, esto es, que una entidad puede descomponerse después y dar lugar a su propio mapa conceptual.

Teniendo una visión pragmática incluyo algunos mapas conceptuales que desarrollados para una empresa de venta de refacciones automotrices de la Ciudad de México, de la cual hice un primer mapa conceptual de tipo caja negra en páginas anteriores y donde el objetivo de estos estudios fue la implantación de un software que controlara totalmente los procesos de la empresa.

En la figura 2.6 muestro el mapa conceptual de la empresa realizado a primer nivel. Ahí se engloban las áreas más importantes de la empresa y sus relaciones.

Como primer acercamiento al conocimiento detallado de la empresa, esta figura fue ideal, ya que me permitió identificar, junto con el cliente en donde había fallas operativas, y qué relaciones entre departamentos debían de mejorarse.

[20] Basadas en el artículo Checkland(1979)

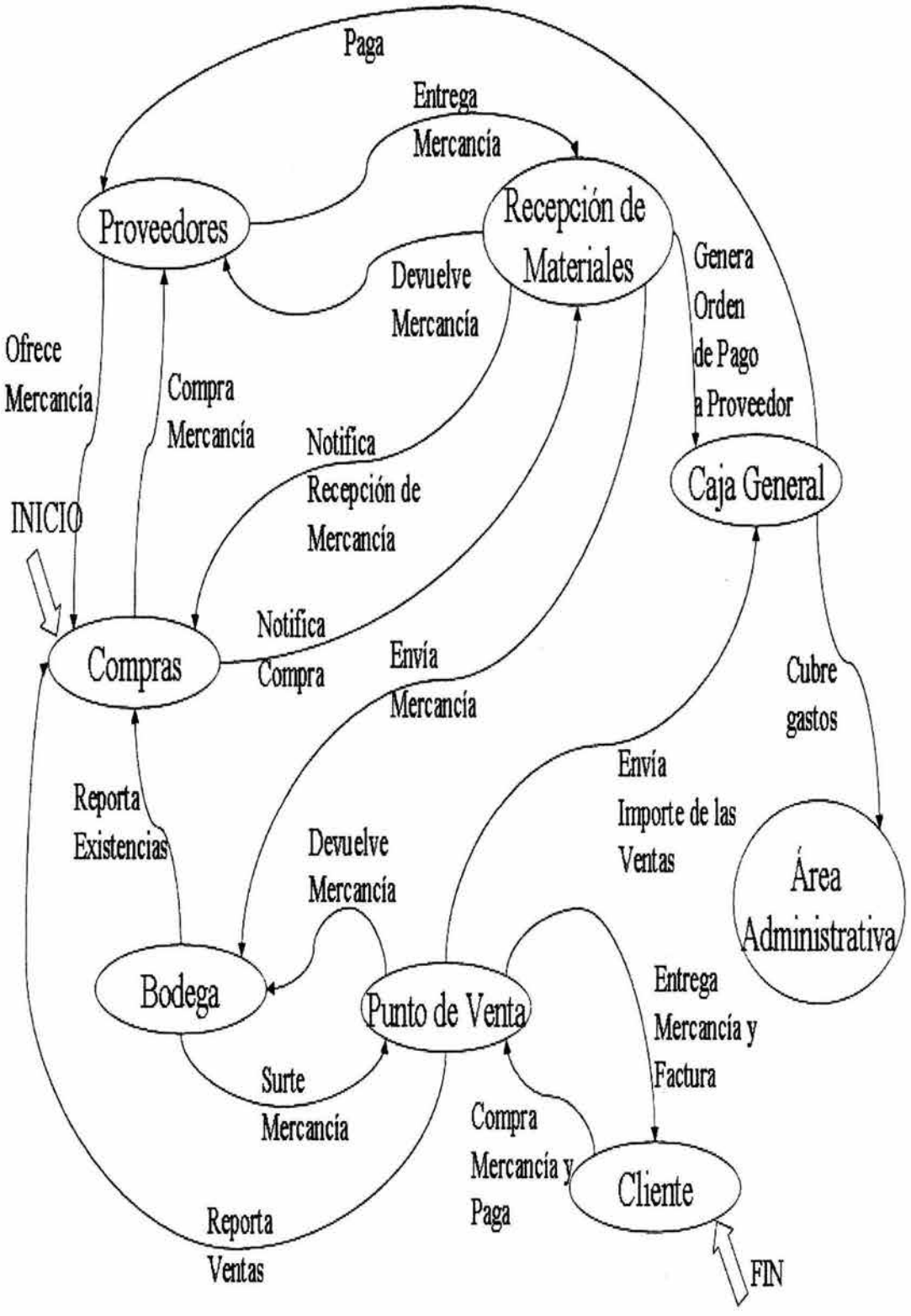


Figura 2.6 Mapa conceptual a primer nivel de una empresa comercializadora de refacciones automotrices

Para poder hacer un estudio completo de la empresa no fue suficiente con este mapa, fue necesario crear mapas conceptuales de las entidades del mapa a primer nivel a fin de comprender de manera formal su funcionamiento. En la figura 2.7, se muestra el mapa conceptual, a segundo nivel, del área de punto de venta.

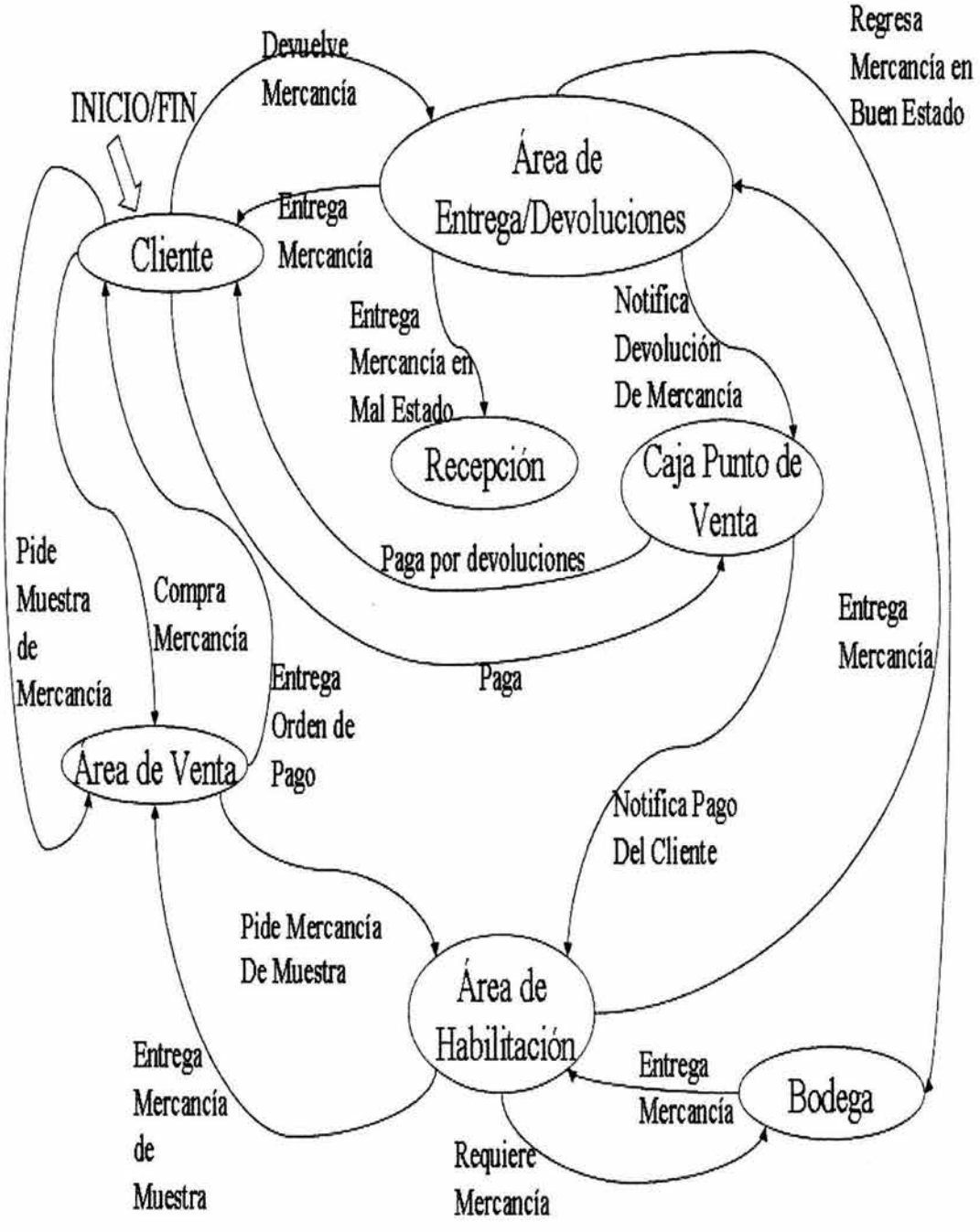


Figura 2.7. Mapa conceptual a segundo nivel del área de punto de venta de una empresa comercializadora de refacciones automotrices.

Con estos mapas, el estudio del sistema llamado empresa se facilita debido a que se tiene una herramienta que descompone al sistema en sus partes, muestra sus elementos y sus interrelaciones y permite avanzar en diferentes niveles. Esto es lo esencial para poder tener un enfoque sistémico de una organización; a fin de entender su lógica de funcionamiento, y estar en posibilidad de reducir los problemas al momento de desarrollar software para una empresa.

2.5.3 Análisis de stakeholders

Después de elaborar algunos mapas conceptuales de la empresa, herramienta que facilita el estudio que se esté realizando, se requiere considerar a un elemento muy importante que es el ser humano y su relación con el sistema. Esto es, estudiar cómo afectan y cómo son afectados por su entorno. Si se habla de la empresa, entonces se estudia al trabajador y cómo su desempeño afecta la consecución de los objetivos de la empresa.

Del mapa conceptual a primer nivel de la empresa, ahora hay que obtener un mapa de empleados para poder proseguir con el análisis, esto se hace colocando los nombres de los empleados y el número de ellos en lugar de las áreas en las entidades, el resultado se puede apreciar en la figura 2.8. Más adelante se explica por qué se le denomina diagrama de stakeholders.

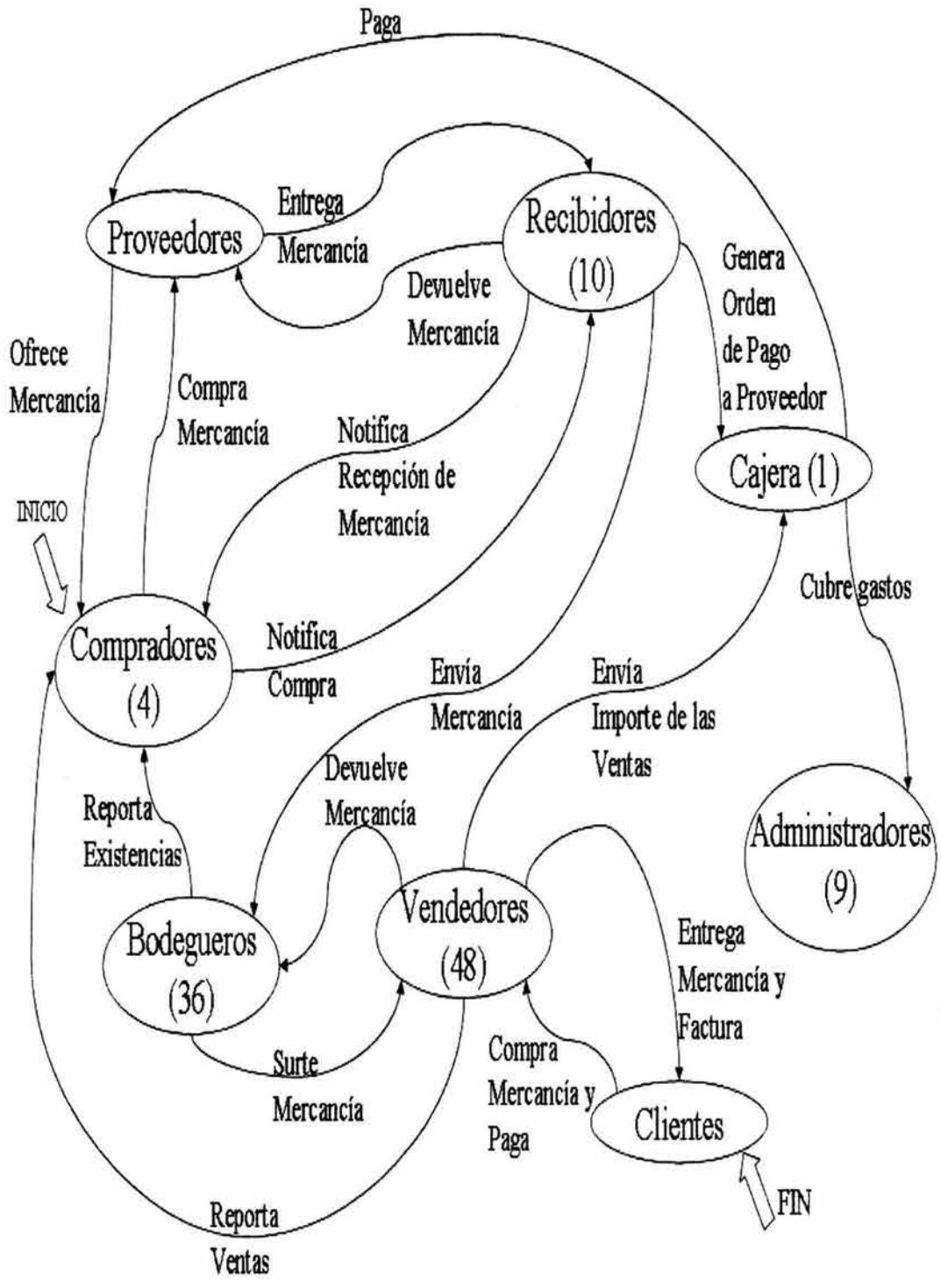


Figura 2.8. Mapa conceptual transformado en un diagrama de empleados.

En la figura 2.8 se puede apreciar el mismo mapa conceptual de primer nivel que realice anteriormente, pero ahora tiene otra utilidad ya a que las entidades contienen las denominaciones o nombres que se les da a los empleados que trabajan en ellas. Por

ejemplo donde existía la entidad de bodegas, ahora encontramos la palabra bodegueros. Y un número que indica el número de personas que existe en esa determinada área.

La palabra stakeholders, esta es una acepción norteamericana para definir a "toda persona o grupo que influye o es influida por los fines y el desempeño de las organizaciones".[21]

Existen dos tipos de stakeholders que se muestran en la siguiente figura:

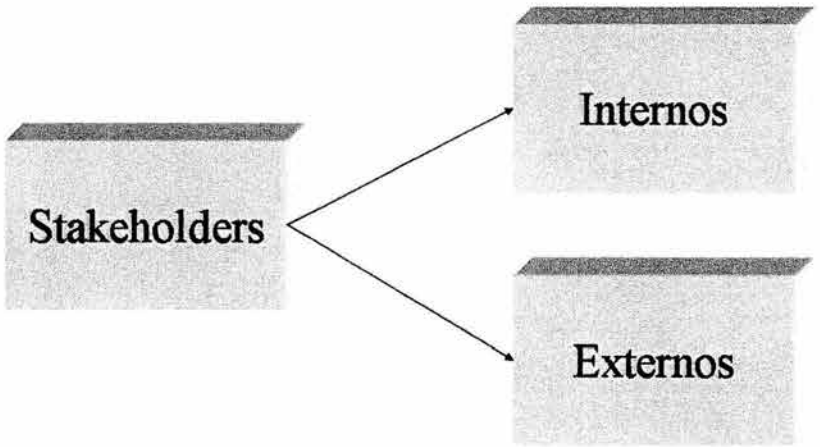


Figura 2.9. Tipos de stakeholders

Los stakeholders internos pertenecen directamente al sistema y los externos son elementos de un ambiente de nivel más alto que interactúan con los stakeholders internos.

Ejemplo de esto serían los bodegueros, los compradores, los vendedores, etc., que son empleados de la empresa, así como el cliente, el administrador, el contador, el jefe de recursos humanos etc. que son partícipes del sistema, todos ellos serían los stakeholders internos. En cambio, los proveedores, los inspectores de hacienda, los clientes, los competidores, son elementos externos al sistema pro tanto stakeholders externos.

[21] Aguilar (2000)

Bajo el concepto de stakeholder, se pueden realizar distintos estudios de las personas que participan en una organización, una de estos estudios es el de red de poder que presento en el siguiente punto.

2.5.4 Red de poder

Una red de poder es un diagrama que nos muestra a los elementos principales del sistema y la forma de relacionarse con los demás. Con esta herramienta podemos detectar problemas sobre las relaciones humanas que se dan dentro del sistema y que de manera obvia afectan el desarrollo de la empresa. Un ejemplo de este tipo de gráficas es la que presento en la figura 2.10.

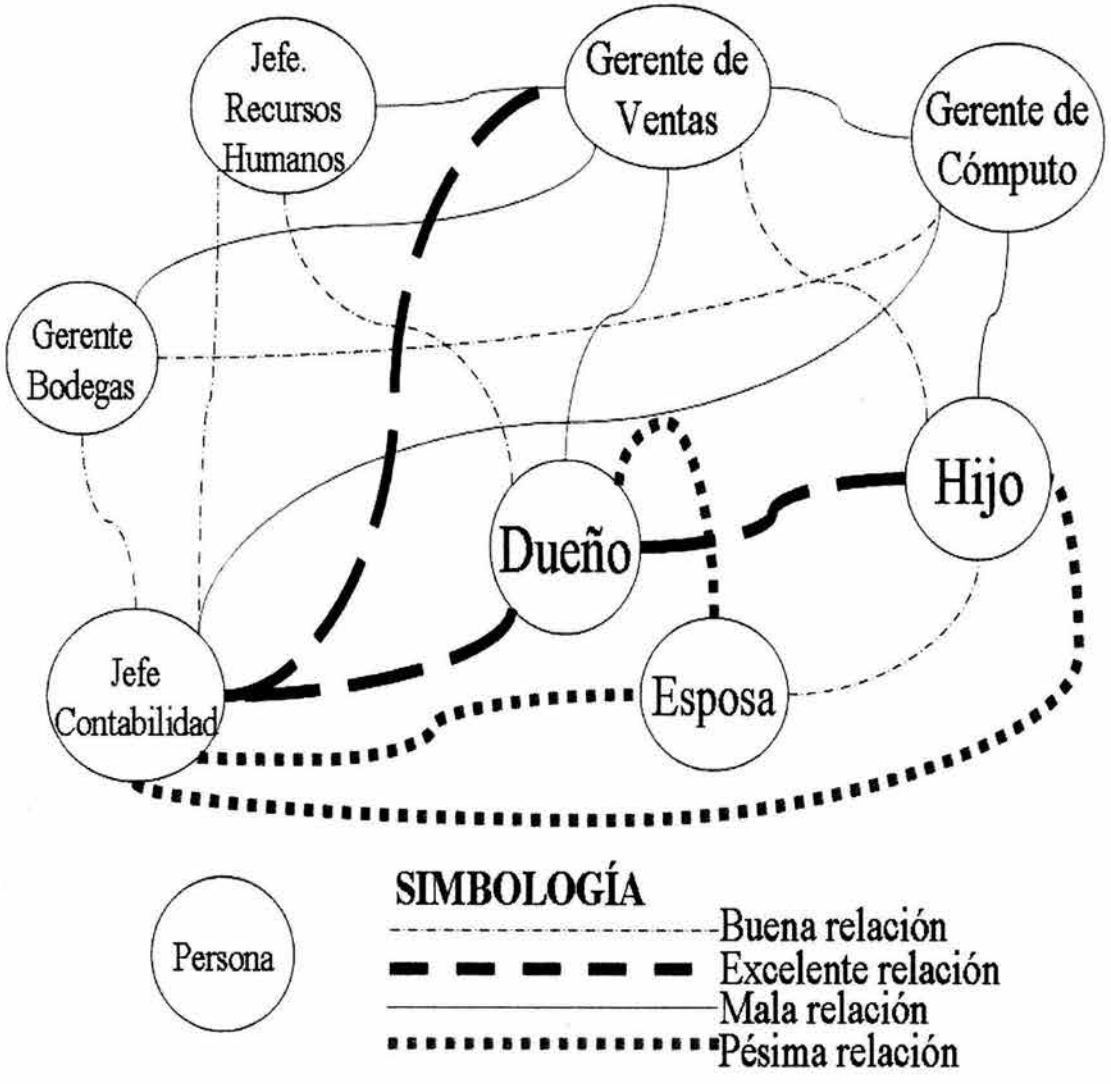


Figura 2.10. Red de poder.

En la figura 2.10 se pueden apreciar los principales actores en el ámbito de la empresa. Es posible identificar a los stakeholders internos y a los externos y nos podemos dar cuenta de quiénes son las personas que podrían en dado caso, apoyar el desarrollo del proyecto y quiénes podrían interferir con las tareas. En este ejemplo se puede ver que el dueño guarda estrecha relación con el jefe de contabilidad, y que es la única relación fuerte del dueño con algún elemento de la empresa exceptuando su hijo. Es aquí donde se confirma de manera definitiva el papel del tomador de decisiones y se tienen elementos más objetivos que indican hacia donde se deben dirigir los esfuerzos para buscar un apoyo o como vimos anteriormente un padrinazgo del proyecto.

A lo largo de este capítulo se han discutido las dos bases conceptuales sobre las que se construirá la guía que se presenta en este trabajo, por una parte, se ha explicado lo que es la ingeniería del software y su relación con la crisis del software, además se explicó hacia que tipo de desarrollos de software se dirige este estudio. También se han explicado algunos de los elementos de la ingeniería de sistemas y principalmente del enfoque sistémico, como los mapas conceptuales y el análisis de stakeholders.

En el siguiente capítulo se mostrarán diversas metodologías y herramientas que son elementos de la ingeniería del software y que tienen una orientación a objetos, que han sido elaboradas para atenuar los efectos de la crisis del software.

3 ELEMENTOS DEL ANÁLISIS ORIENTADO A OBJETOS

3.1 Antecedentes

En los dos primeros capítulos he hablado de manera general de la orientación a objetos, como parte de la ingeniería de software; este capítulo está dedicado a analizar con más detalle los elementos conceptuales básicos de dicha teoría, que están presentes en la guía que se desarrolla en el siguiente capítulo.

> *"La idea de que el mundo podía verse en términos de objetos o procesos fue una innovación de los griegos, y en el siglo XVII se encuentra a Descartes observando que los humanos aplican de forma natural una visión orientada a objetos del mundo"*[22]

> *"Aunque Ten Dyke y Kunz afirman que los diseñadores del misil Minuteman utilizaron técnicas rudimentarias de orientación a objetos ya en 1957, la historia de la programación orientada a objetos comienza realmente con el desarrollo del lenguaje de simulación de sucesos discretos, Simula, en Noruega en 1967, y continúa con el desarrollo de un lenguaje que casi toma como fetiche el concepto de objeto, Smalltalk, en los años 70".*[23]

Durante las últimas tres décadas se han desarrollado un gran número de estudios encaminados a combatir la crisis del software, tomando como base los lenguajes orientados a objetos, los cuales han revolucionado la forma de construir software en todo el mundo, hace aproximadamente tres décadas se comenzaron a utilizar en otros países, y su utilización de manera generalizada en México se dio alrededor de la última década, durante la cual, tanto las universidades como las empresas se han preocupado por difundir y utilizar esta forma de programación, que vino a sustituir a la programación estructurada que hace tres lustros dominaba los ámbitos académico y empresarial en nuestro país.

Si bien la programación orientada a objetos no ha sido explotada totalmente en el ambiente empresarial, es necesario estudiar más a fondo está rama de la computación, que al parecer estará vigente por muchos años más, una muestra de ello es la evolución de los sistemas típicos cliente servidor implantados en intranets que se desarrollaron con una filosofía

[22] Booch Grady(1996)., p.42.
[23] Graham Ian (1996), p. 3.

orientada a objetos, que se han sustituido por los sistemas que utilizan la Internet, y que vienen a constituir el comercio electrónico muy de moda en estos días, y que también han utilizado la orientación a objetos. Puedo afirmar que, sistemas operativos, infraestructuras, herramientas, etc. seguramente cambiarán, pero la orientación a objetos para construir software seguirá vigente por bastantes años aún.

El porque el diseño de los sistemas orientados a objetos estará vigente todavía por mucho tiempo tiene varias razones, las cuales presento en las siguientes páginas de este capítulo, mostrando las bondades y ventajas de esta forma de construir sistemas computacionales de tamaño industrial.

3.2 Complejidad de software

Booch afirma que el software de dimensión industrial tiene una complejidad esencial, que sobrepasa la capacidad intelectual humana, y con el término de esencial, quiere decir que esa complejidad puede ser dominada más nunca eliminada, además explica que dicha complejidad se genera debido a cuatro elementos que se explican a continuación.

3.2.1 La complejidad del dominio del problema.

- Existe un alto número de requisitos o requerimientos involucrados.
- Se presentan requisitos no funcionales como facilidad de uso, rendimiento, costos, fiabilidad etc.
- Los usuarios no pueden expresar sus requerimientos a los desarrolladores de software, esto a debido principalmente a dos factores, el primero es que a veces ellos mismos no tienen una idea clara de sus requerimientos, el segundo es que no hablan un lenguaje común con los ingenieros en computación.
- Las formas tan imprácticas que existen de expresar los requerimientos.

3.2.2 La dificultad de gestionar el proceso de desarrollo.

- Un solo individuo no puede entender completamente el sistema involucrado con estos desarrollos de software, lo que implica la utilización de grupos de trabajo.

- Entre mayor sea el número de individuos relacionados con el desarrollo de software mayores serán los problemas de comunicación y administración del proyecto, además hay que tomar en cuenta que actualmente muchos de estos grupos de trabajo se encuentran dispersos geográficamente.

3.2.3 La flexibilidad que se puede alcanzar a través del software

El software por ser flexible puede expresar cualquier tipo de abstracción, por lo que seduce al programador a construir la infraestructura básica de desarrollo. Para aclarar este punto pensemos en un ingeniero civil que va a construir un edificio, sería ilógico que tratara de tener dentro de su compañía a una acería para construir vigas de acero a la medida para el edificio, lo lógico es que contrate a otra empresa que le abastezca del material necesario, esto es algo que el ingeniero en computación en múltiples ocasiones no prevé, casi por regla, los ingenieros y las consultorías comienzan todos los desarrollos desde cero al llegar con un cliente nuevo.

3.2.4 Los problemas de caracterizar el comportamiento de sistemas discretos.

- Un sistema discreto implica miles de variables.
- El sistema puede verse como un sistema de estados.
- La transición entre estados no siempre es determinista.
- Un evento externo puede corromper el estado del sistema, debido a que los desarrolladores olvidaron considerar ciertas interacciones.
- Debido a que no se pueden probar completamente estos sistemas, por limitaciones en las herramientas y por propia capacidad intelectual humana, es suficiente tener un grado de confianza adecuado.

Booch basado en los puntos anteriores dijo:

"La sugerencia que se plantea aquí es estudiar en primer lugar cómo se organizan los sistemas complejos en otras disciplinas"[24]

[24] Booch Grady, (1996) , p. 9

El software como hemos analizado en los párrafos anteriores tienen una complejidad inherente, la cual no puede ser eliminada, sólo puede ser controlada. En el siguiente apartado analizo cómo han evolucionado las topologías de las herramientas de programación con las cuales se generan los desarrollos de software hasta el diseño orientado a objetos.

3.3 Evolución de los lenguajes de programación

La parte de la historia de la computación que concierne a la programación orientada a objetos tiene un gran número de eventos relevantes que pueden ser citados y desarrollados, pero aquí sólo presento una breve descripción de los elementos que a mi juicio son los más importantes para poder comprender las tareas que a lo largo del tiempo se han realizado para abatir la crisis del software.

La evolución de los lenguajes de programación la presento como estructura principal sobre la cual descansa la evolución de la forma de programar, que hasta nuestros tiempos se ha trastocado en programación orientada a objetos, atendiendo a las necesidades de distintos tipos de organizaciones.

3.3.1 Primera generación.

Algunos ejemplos de lenguajes de esta generación son:

- FORTRAN I
- ALGOL 58
- Flowmatic
- IPL V

Estos lenguajes aparecieron en el periodo comprendido entre los años 1954-1958 y tenían una fuerte orientación matemática. Eran utilizados principalmente para aplicaciones científicas, recibían como entradas expresiones matemáticas que eran interpretadas. Por estos tiempos, los procesamientos eran hechos en forma de lotes (batch), existía poco software y la mayoría se construía a la medida.

La filosofía sobre la cual se basaban los lenguajes de primera generación era la existencia de subprogramas que podían acceder a estructuras globales de datos disponibles para

todos los subprogramas. En este tipo de lenguajes el subprograma es la unidad básica. En la figura 3.1 muestro la topología de los lenguajes de primera generación.

En los desarrollos realizados con este tipo de lenguajes frecuentemente aparecía la entropía, manifestándose en forma de errores en múltiples acoplamientos de subprogramas, flujos de control no adecuados, etc. mucho de esto debido a la topología base de los lenguajes.

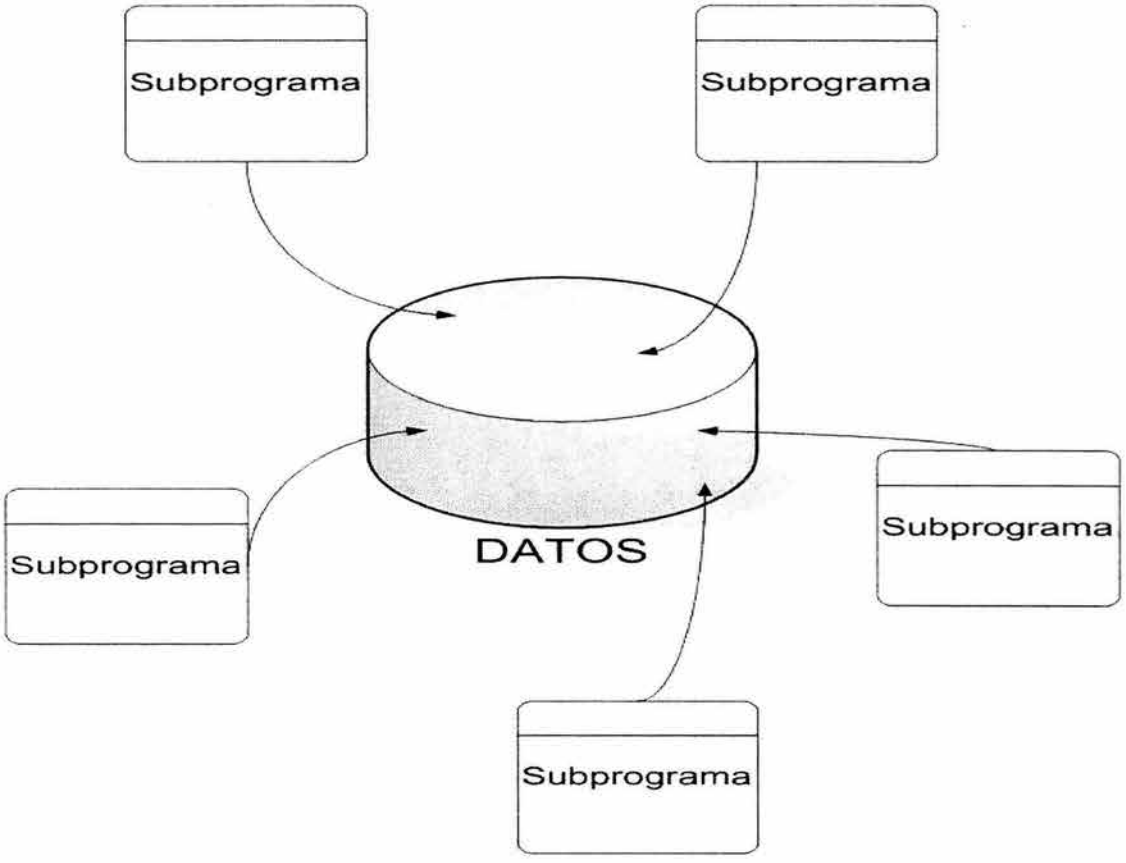


Figura 3.1 Topología de los lenguajes de primera generación.

3.3.2 Segunda generación

Durante el periodo que comprende a los años 1959-1961 aparecen lenguajes que hacían abstracciones algorítmicas, es decir, se le podía decir a la máquina que hacer, *lee tarjetas, procesa, emite resultados* etc. Se podían resolver un gran número de problemas pero esto implicó que se originara la abstracción de datos, esto debido al gran número de tipos de datos que se manipulaban. La abstracción de datos es por ejemplo la creación del tipo de

dato coche, basado en otros tipos de datos primitivos. A continuación presento algunos ejemplos de los lenguajes de esta generación:

- FORTRAN II
- ALGOL 60
- COBOL
- Lisp

En estos lenguajes al subprograma se le veía como una abstracción, como un elemento que simulaba la realidad, a esto se le llamo una abstracción procedimental, es aquí cuando nace la programación estructurada, basada en el paso de parámetros hacia los procedimientos y el diseño estructurado. La utilización de bloques comenzaba a ser una realidad. La topología de estos lenguajes la presento en la figura 3.2.

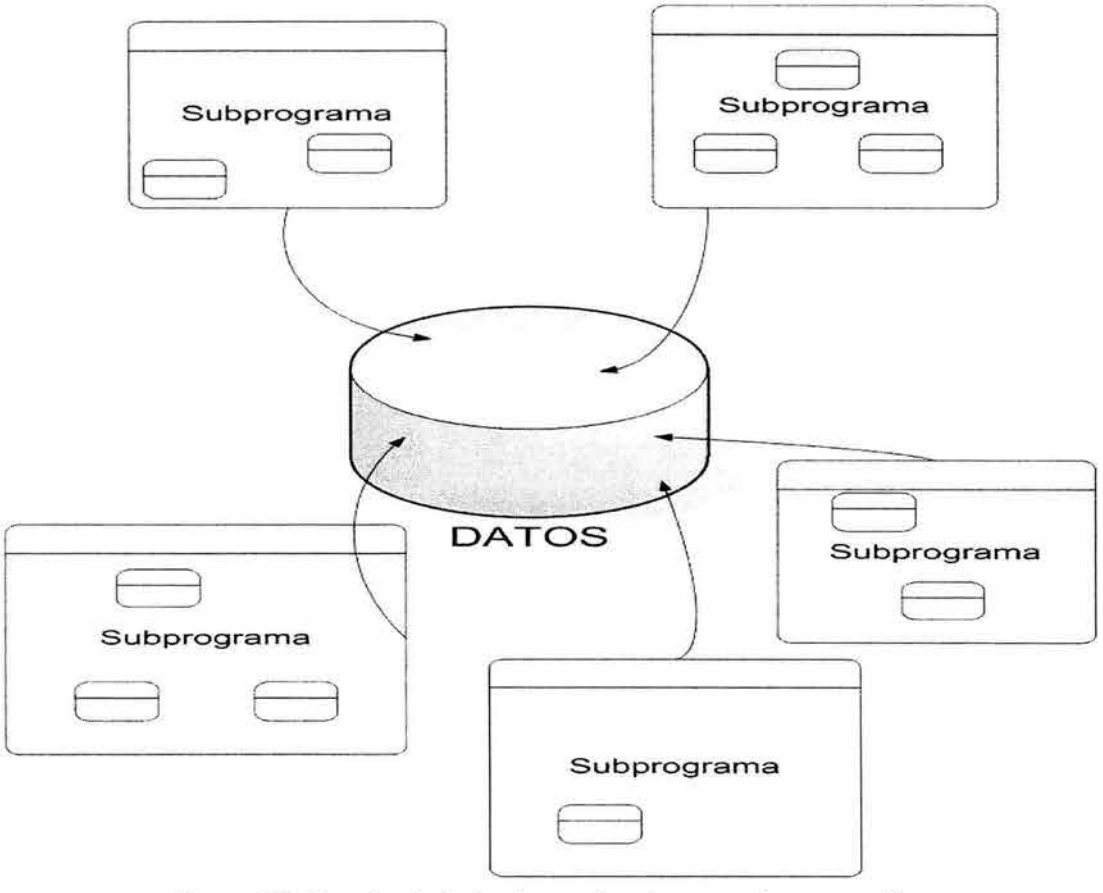


Figura 3.2. Topología de los lenguajes de segunda generación.

Estos programas podían resolver una gran gama de problemas, pero al enfrentarse a desarrollos industriales, se generaban fallas importantes.

3.3.3 Tercera generación

Durante el periodo que comprende los años 1962-1970, surgieron lenguajes como:

- PL/1
- ALGOL 68
- Pascal
- Simula

Estos lenguajes agrupaban subprogramas y datos en estructuras llamadas módulos, las cuales eran compiladas de forma independiente, la comunicación entre los módulos se daba por medio del paso de parámetros, que a decir verdad no era del todo correcta, lo que daba como resultado la presencia de errores sólo detectados en tiempo de ejecución, debido a que no se llevaba a cabo una rigurosa comprobación de tipos de datos, además de que también la abstracción de datos permitía inconsistencias. En la figura 3.3 presento la topología de los lenguajes de tercera generación, la cual no tiene gran variación con respecto a la segunda.

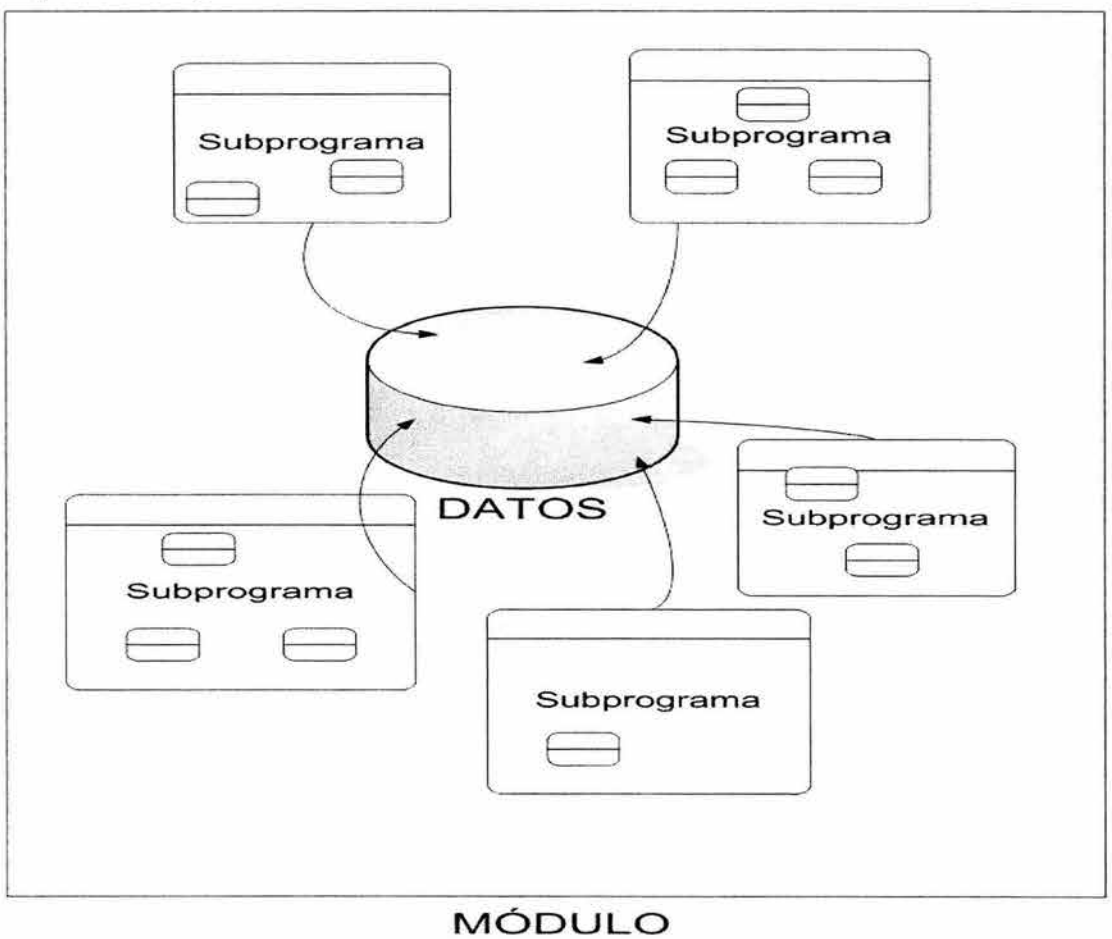


Figura 3.3. Topología de los lenguajes de tercera generación.

3.3.4 Lenguajes orientados a objetos

El periodo de los años setenta fue sumamente prolífico para la creación de lenguajes de programación, se generaron cientos de lenguajes, la mayoría de ellos no tuvieron trascendencia, pero es en este periodo en el cual se da la creación de los lenguajes basados en objetos[25] y orientados a objetos[26], el primero de ellos fue el Simula. En la figura 3.4 se muestra la topología de este tipo de lenguajes, en donde el elemento básico es la clase, que es una colección lógica de clases y objetos en lugar de subprogramas (más adelante se ahonda en estos términos), esta topología es la utilizada para desarrollos a baja escala, es decir sistemas computacionales pequeños, debido a que la utilización de esta topología presenta problemas en los desarrollos de tamaño industrial, debido a que el nivel de abstracción que se puede alcanzar es limitado.

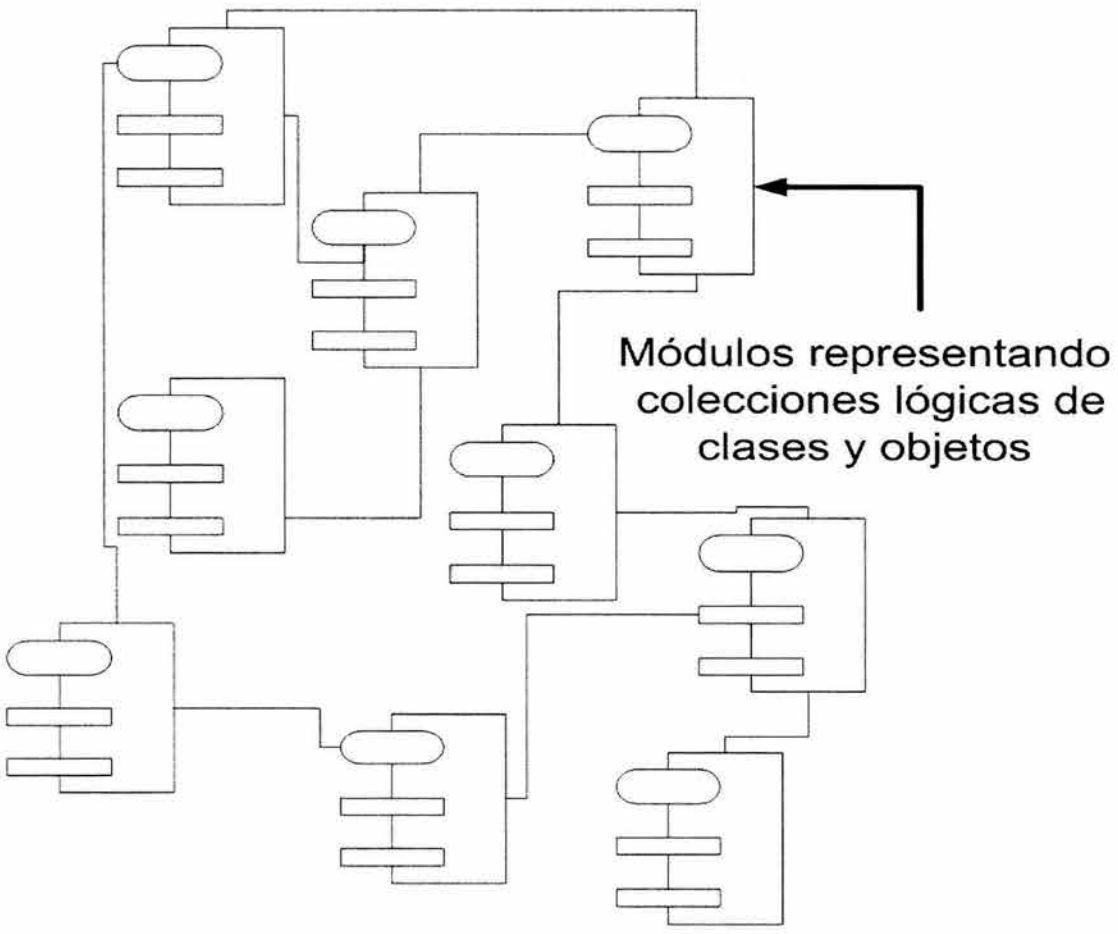


Figura 3.4 Topologías de lenguajes orientados a objetos y basados en objetos para desarrollos pequeños

[25] No soportan la herencia (más adelante se detalla este concepto).

Para los desarrollos a gran escala, "existen agrupaciones de abstracciones que se construyen en capas, una sobre otra"[27], cuya topología se muestra en la figura 3.5.

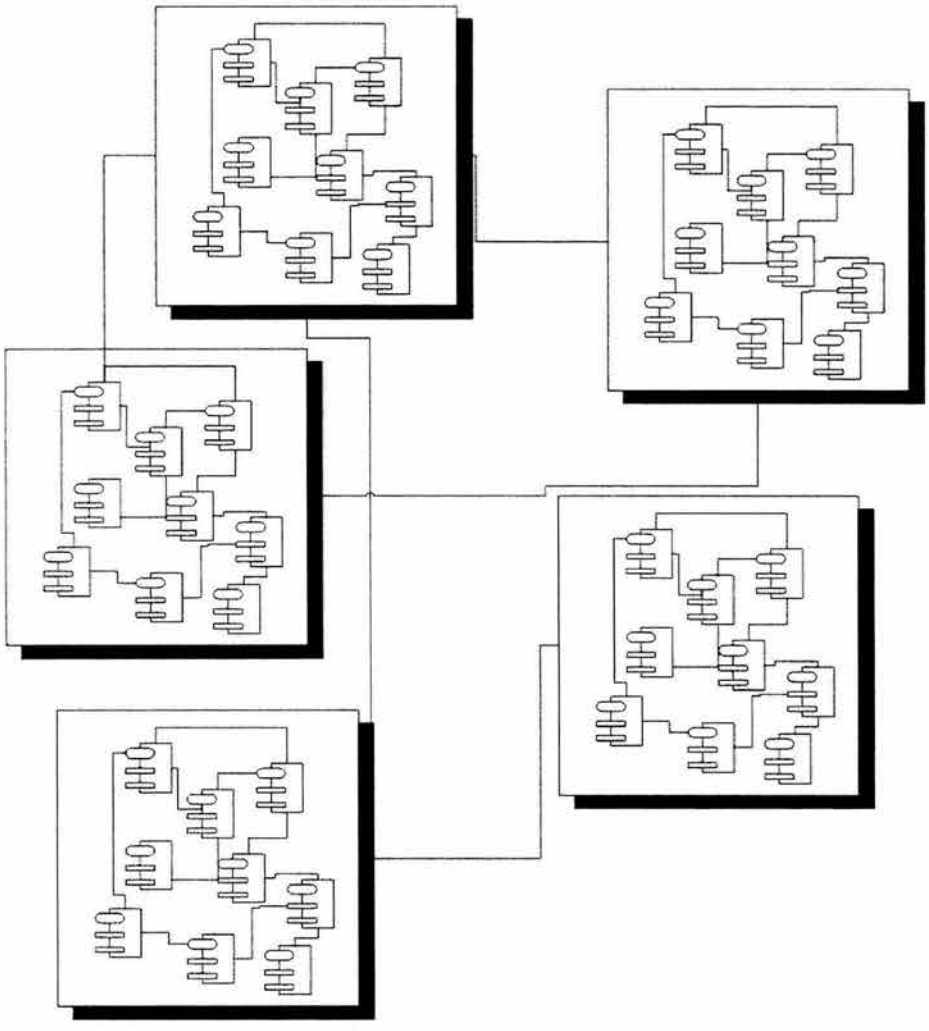


Figura 3.5 Topologías de lenguajes orientados a objetos y basados en objetos para grandes desarrollos.

En nuestro país se utilizan en las empresas y se enseñan en las universidades, diversos lenguajes orientados a objetos y basados en objetos, a continuación menciono los más importantes en ambos ámbitos:

- C ++
- Delphi 7 (Versión avanzada de Pascal)
- Power Builder

[26] Soportan objetos encapsulados, asociados a una clase, la cual puede heredar atributos a superclases (más adelante se detalla este concepto).
[27] Booch Grady (1996), p 38

- Visual Basic
- Visual C++

3.4 Conceptos

"Por desgracia la programación orientada a objetos significa cosas distintas para personas distintas. Tal como Rentsch predijo acertadamente «Mi impresión es que la programación orientada a objetos va a ser en los ochenta lo que la programación estructurada en los setenta. Todo el mundo va a estar a favor de ella. Todos los fabricantes van a promocionar sus productos afirmando que la soportan. Todos los administradores hablarán bien de ella. Todos los programadores la practicarán (de forma diferente). Y nadie va a saber exactamente qué es». Las predicciones de Rents siguen vigentes"[28] hasta nuestros días, en el campo laboral, sucede exactamente lo que acabo de citar. La única forma de abatir esta falta de compresión acerca de lo que es la programación orientada a objetos, en primera instancia es teniendo presentes los conceptos básicos que la conforman.

Cohoon y Davidson señalan que "algo es un objeto si tiene:

- Un nombre.
- Propiedades asociadas al mismo.
- Mensajes que puede entender."[29]

Para Grahan los objetos son:

"Las unidades básicas de construcción, para conceptualización, diseño o programación, son instancias organizadas en clases con características comunes. Estas características comprenden los atributos y procedimientos, denominados operaciones o métodos"[30].

Para poder comprender las definiciones planteadas en los párrafos anteriores, sigamos el siguiente ejemplo.

Tomemos un objeto del mundo real, supongamos un coche. El coche pertenece o es miembro (es una instancia) de una clase mucho más grande de objetos, los cuales llamaremos medios de transporte. A los medios de transporte se les puede asociar atributos genéricos, los cuales aplican para cada uno de los elementos de esta clase,

[28] Booch Grady (1996), p.31
[29] Cohoon.Davidson (2000), p.38.
[30] Grahan Ian (1996), p. 11.

es decir en específico aplican para el coche, o para un barco, una motocicleta, un jet, etc. Ejemplos de estos atributos son rapidez, peso, dimensiones, color, etc.

Una vez definida la clase, los atributos pueden reutilizarse al crear nuevas instancias de la clase, como por ejemplo un objeto cuatrimoto, el cual hereda los atributos de la clase medios de transporte.

Todo objeto de la clase medios de transporte, puede manipularse de varias maneras, puede moverse físicamente de lugar, puede modificarse su estructura y su forma (se puede pintar de otro color). Cada una de estas operaciones (servicios o métodos) modificarán uno o más atributos del objeto. Suponiendo que el atributo potencia, es un dato compuesto definido por:

Potencia = Velocidad de arranque + rapidez de desplazamiento + velocidad en los cambios

El método cambiar potencia, debe de modificar cualquiera de los dos atributos, por ejemplo, si al programar un juego en computadora de carreras de coches, necesitamos presentar al usuario dos tipos de coches, tenemos el mismo objeto coche, y cambiando los atributos del objeto vía el método cambiar potencia, podríamos tener un Ferrari y un Porche, cambiando sus atributos, sabiendo que estos dos tipos de coches en la vida real tienen características distintas.

Hay que tener presente que el método cambiar potencia, debe conocer los atributos que cada objeto tiene. Todas las operaciones válidas por ejemplo (comprar, vender, desplazar, cambiar potencia) de la clase medios de trasporte se presentan en la definición del objeto y son heredadas por todas las instancias de esta clase, todo lo anterior se presenta en la figura 3.6., en donde se puede visualizar que el objeto hereda todos sus atributos y operaciones de la clase, además de que al objeto se le pueden añadir nuevos atributos y métodos para generar objetos más particulares.

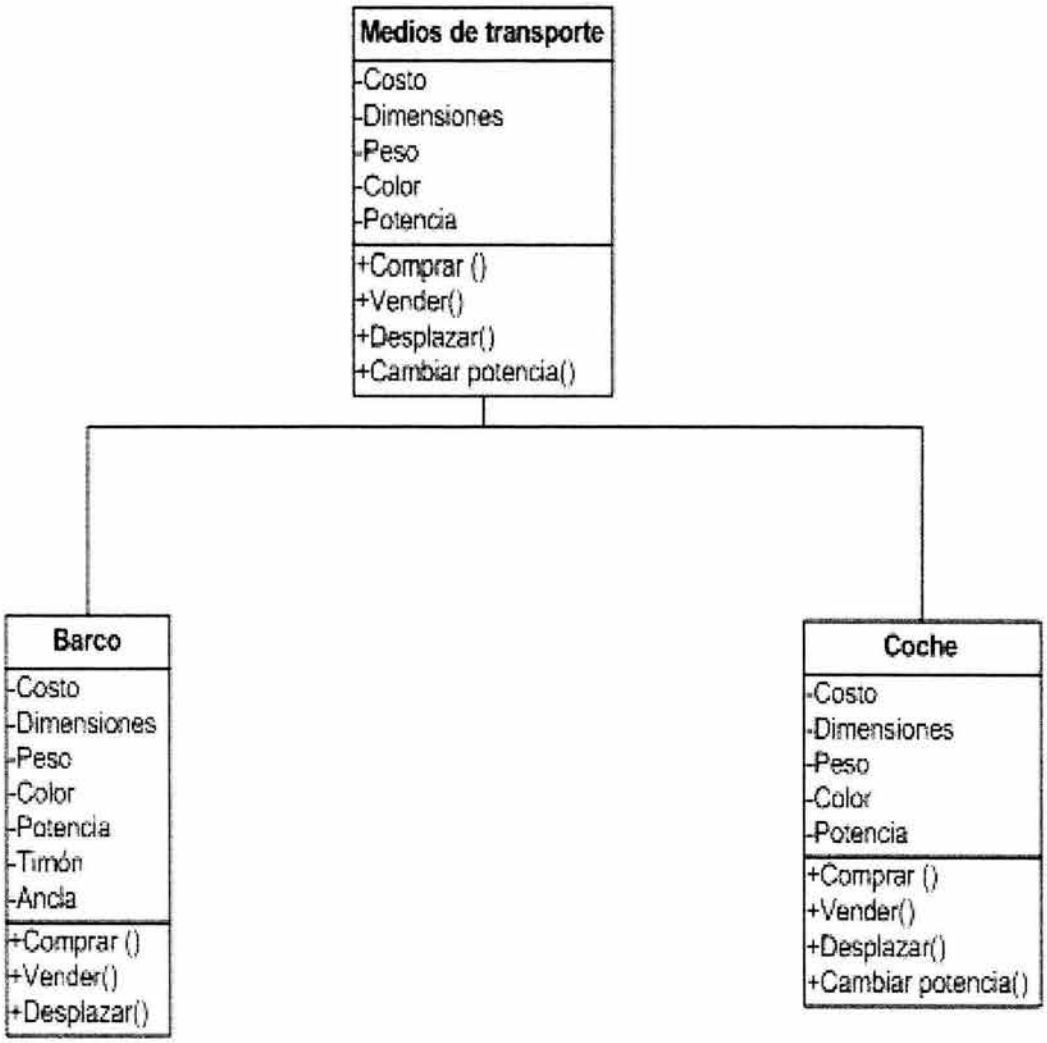


Figura 3.6 Herencia de operaciones de clase a objeto

Es fácil de intuir con lo anterior el por qué la orientación a objetos presenta múltiples problemas a la hora de programar, y esto principalmente se da por no respetar las bases teóricas, es común en un curso de programación orientada a objetos que los alumnos se sientan incómodos con estos términos debido a que no tiene un acotación bien definida, y comienzan a generar cuestionamientos diversos. A continuación presento algunas preguntas que me llamaron la atención en un curso que impartí en una universidad:

- ¿La nada o los sentimientos pueden ser modelados como objetos?
- ¿Cuál es la clase base de la cual debe de comenzar mi desarrollo?
- ¿Todo objeto tiene una clase base?

Preguntas de este tipo para ingenieros en computación expertos a primera instancia parecen demasiados obvias y hasta tontas, las cuales se desprecian al momento de contestarlas, pero

no se han dado cuenta que es aquí en donde comienzan los problemas que después se trasladarán a los lugares de trabajo, debido a que no son preguntas que se generan a la ligera, sino que son el resultado de problemas con algunos conceptos de la programación orientada a objetos que principalmente tienen que ver con el concepto de abstracción. Analizado este punto continuaré con el ejemplo del objeto coche que me ha servido hasta el momento para explicar de manera superficial algunos conceptos relacionados con la orientación a objetos.

El objeto coche (y todos los objetos en general) encapsula datos (los valores de los atributos que definen coche), operaciones (las acciones que se aplican para cambiar los atributos de coche, otros objetos (pueden definirse objetos compuestos), constantes (se fijan valores), e información diversa. El encapsulamiento significa que toda esta información se encuentra empaquetada bajo un nombre y puede reutilizarse como una especificación o componente de programa.

A continuación defino de manera más formal a los conceptos de la programación orientada a objetos, que descansan bajo el marco conceptual llamado modelo de objetos.

Dentro del modelo de objetos se encuentran cuatro elementos fundamentales, lo que implica que si un modelo carece de alguno de ellos no es orientado a objetos, de cada uno de ellos se da la definición que se encuentra en el libro de Booch *Análisis y diseño orientado a objetos con aplicaciones*:

- Abstracción.
    - "Una abstracción denota las características esenciales de un objeto que los distinguen de todos los demás tipos de objetos y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador"
    - Nos sirve para combatir la complejidad.
    - Sirve para enfatizar u obviar detalles de un objeto
    - La abstracción debe de capturar el comportamiento esencial mínimo de un objeto sin dejar lugar a las "sorpresas" (ejemplo de una sorpresa es que un coche vuele).
    - La abstracción se centra en el comportamiento observable de un objeto.

- Encapsulamiento
    - "El encapsulamiento es el proceso de almacenar en un mismo compartimento los elementos de una abstracción que constituyen su estructura y su

comportamiento; sirve para separar el interfaz contractual de una abstracción y su implantación"

- El encapsulamiento se centra en la implementación que da lugar al comportamiento de un objeto.
- El encapsulamiento oculta detalles de la implementación de los métodos de los objetos.
- Con el encapsulamiento el usuario, sólo ve su interfaz contractual y no se preocupa por como se dio la implantación, ejemplo de esto son los usuarios de un juego de carreras de coches, que saben que para un determinado modelo de coche, pueden escoger, potencia, colores, aditamentos, etc., pero no saben ni se imaginan que tanto se programó de manera interna para lograr todas estas funciones.

- Modularidad
  - "La modularidad es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados"
  - Modularidad es fragmentar un programa en componentes individuales, para reducir complejidad.
  - La cohesión significa el agrupar abstracciones que guarden relación lógica.
  - Se busca que los módulos estén débilmente acoplados para disminuir las dependencias entre ellos.

- Jerarquía
  - "La jerarquía es una clasificación u ordenación de abstracciones".
  - Una subclase, es el resultado de aumentar o redefinir el comportamiento de una superclase. Esto se puede visualizar en la Figura 3.6 en donde la clase barco, aumento el comportamiento de la clase medios de transporte.
  - Jerarquía es conocida también como herencia, la cual genera relaciones del tipo «es un», por ejemplo, barco es un medio de trasporte, o coche es un medio de transporte.
  - La herencia nos permite trabajar con la especialización, es decir las subclases son más especializadas que las superclases de donde se generan.

Existen además tres secundarios:

- Tipos (tipificación)

- "Los tipos son la puesta en vigor de la clase de los objetos, de modo que los objetos de tipos distintos no pueden intercambiarse o, como mucho pueden intercambiarse sólo de formas muy restringidas".
- El objetivo de la tipificación es la congruencia. Por ejemplo se debe de sumar números con números, más no se pueden sumar números con triángulos.

- Concurrencia
  - "La concurrencia es la propiedad que distingue un objeto activo de uno que no está activo".
  - "La concurrencia se centra en la abstracción de procesos y la sincronización".

- Persistencia.
  - "La persistencia es la propiedad de un objeto por la que su existencia trasciende el tiempo (es decir, el objeto continúa existiendo después de que su creador deja de existir) y/o el espacio (es decir, la posición del objeto varía con respecto al espacio de direcciones en el que fue creado)".
  - Para lograr este concepto se pueden utilizar bases de datos orientadas a objetos, para almacenar el estado y la clase de los objetos.

3.5 Programación, diseño y análisis orientado a objetos

Hasta ahora he presentado una panorámica general de la evolución de la programación orientada a objetos, la cual va de la mano del diseño y análisis orientado a objetos, ahora presento las definiciones formales dadas por Booch[31]. Estas definiciones se basan en los conceptos que se presentaron en las secciones anteriores de este capítulo.

> "La programación orientada a objetos es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase y cuyas clases son, todas ellas, miembros de una jerarquía de clases unidas mediante relaciones de herencia".

> "El diseño orientado a objetos es un método de diseño que abarca el proceso de descomposición orientada a objetos y una notación para describir los modelos lógico y físico, así suma los modelos estático y dinámico del sistema que se diseña".

[31] Booch Grady (1996)

> "El análisis orientado a objetos es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema".

En la figura 3.7 muestro de manera gráfica la relación que se da entre el análisis, el diseño y la programación, si bien en la figura no se menciona el término de orientación a objetos, es porque esta relación aplica no sólo para la programación orientada a objetos sino para cualquier otra técnica como la estructurada.

El analista de sistemas, se encarga de analizar el sistema a automatizar

Transmite los resultados de su análisis a los diseñadores computacionales

Los diseñadores, generan el diseño de las bases de datos, los módulos, los procesos etc.

El diseño es trasmitido a los programadores, los cuales se encargan de implementarlo

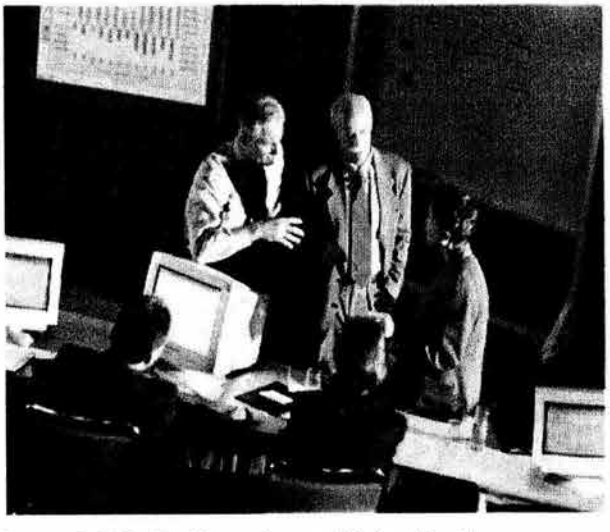


Figura 3.7 Relación entre análisis, diseño y programación.

La programación orientada a objetos es el ámbito de trabajo del programador, un buen programador debe de entender cabalmente la definición anterior, la cual engloba los principios de la programación orientada a objetos.

La utilización de la programación orientada a objetos para obtener software de tamaño industrial, implica como lo he enfatizado a lo largo de todo este trabajo la necesidad de contar con elementos de análisis y diseño adecuados para controlar la complejidad que engloba el software de gran tamaño.

Un programador para poder comenzar con su trabajo, debe de contar con un diseño sobre el cual basar sus acciones. El diseño orientado a objetos es el resultado de un análisis orientado a objetos.

La ingeniería del software ha tratado de establecer guías, métodos, metodologías etc., para abatir los costos altos, la poca fiabilidad y la escasez del software elementos generadores de la crisis del software. En el capítulo uno planteé la necesidad de mejorar el análisis y el diseño del software, en este capítulo muestro que el software en nuestro tiempo tiene una orientación a objetos en la mayoría de los casos, por lo que cualquier análisis y diseño que se tenga que realizar, lo indicando es hacerlo cuidando los aspectos de la orientación a objetos.

Así mismo, el capítulo dos plantee la premisa de que en las últimas tres décadas se ha trabajado arduamente en la elaboración de metodologías que satisfagan las necesidades de los clientes y que pueden combatir la crisis del software, esto ha dado como resultado múltiples metodologías encaminadas al desarrollo de software de gran tamaño. También, en ese capítulo mencione que seguramente muchas de ellas son trabajos loables que han permitido avanzar hacia mejores implantaciones de software de gran tamaño, pero hoy en día la crisis del software sigue presente, por lo cual, creo necesario el analizar qué pasa con todas esas metodologías, a fin de que la guía que presento, contenga elementos valiosos que puedan ayudar a todos esos trabajos, sin duplicar esfuerzos, y evitar las carencias que las metodologías usadas actualmente en las empresas tienen.

Por lo anterior presento una breve descripción de algunas de las metodologías de análisis orientado a objetos más difundidas en los ámbitos laborales y académicos en nuestro país. – Donde se enfatiza principalmente en cada una de las metodologías sus procesos de análisis de sistemas. Si se desea profundizar más sobre dichas metodologías, el lector puede consultar las referencias que se indican para cada caso.

3.6 Metodologías de análisis orientado a objetos

3.6.1 Metodología de Booch

Esta metodología abarca un "micro" y un "macro" proceso de desarrollo. En el nivel micro define un conjunto de tareas de análisis que se reaplican en cada etapa en el macro proceso. Por esto se mantiene un enfoque evolutivo. La metodología de Booch está soportada por una variedad de herramientas computacionales.

En el macro proceso de desarrollo las tareas fundamentales son:

- Establecer los requerimientos (conceptualización)
- Analizar para desarrollar un modelo ideal
- Diseñar de la arquitectura
- Implementar el diseño
- Mantener el software después de la liberación.

Cuando se refiere a la primera tarea, menciona que el propósito de esta parte de su metodología es "establecer los requisitos esenciales para el sistema"[32], además de que "la conceptualización es, por su propia naturaleza, una actividad intensamente creativa, y por tanto no debería de verse encadenada por reglas de desarrollo rígidas"[33].

Con respecto al análisis menciona que "como dice Mellor «el propósito del análisis es proporcionar una descripción de un problema. La descripción debe de ser completa, consistente, legible y revisable por las partes interesadas» ... el propósito del análisis es proporcionar un modelo del comportamiento del sistema"[34]

Aquí difiero en parte de lo planteado por Booch. Estoy de acuerdo que el análisis es un proceso creativo, pero el no encaminarlo o dirigirlo, en el campo laboral, lleva a los líderes de proyecto junto con los programadores por premuras de tiempo, presiones externas, problemas etc., a extremos de ser demasiados creativos o totalmente faltos de creatividad.

Cabe enfatizar que la metodología nos dice que se debe de realizar un análisis del sistema, y nos presenta los aspectos a analizar como los puntos funcionales principales del sistema,

[32] Booch Grady (1996), p. 285
[33] Ibídem, p. 286
[34] Ibídem, p. 287

construcción de escenarios, etc., pero no nos dice cómo analizarlos, esto en la práctica puede volverse un caos, ya que este punto hace que cada analista genere un análisis de forma particular, atendiendo a sus conocimientos, necesidades, y objetivos, lo que lleva a una seudo anarquía computacional, responsable en parte de la crisis del software.

Con respecto al micro proceso de desarrollo, los pasos que lo componen son los siguientes:

- Identificar clases y objetos
  - Proponer objetos candidatos
  - Conducir el análisis de comportamiento
  - Identificar escenarios relevantes
  - Definir atributos y operaciones para cada clase

- Identificar la semántica de clases y objetos
  - Seleccionar y analizar escenarios
  - Asignar responsabilidades para alcanzar el comportamiento deseado
  - Dividir las responsabilidades para equilibrar el comportamiento
  - Seleccionar un objeto y enumerar sus papeles y responsabilidades
  - Definir operaciones para satisfacer las responsabilidades
  - Buscar colaboraciones entre objetos

- Identificar relaciones entre clases y objetos
  - Definir las dependencias que existen entre objetos
  - Describir el papel de cada objeto participante
  - Validar los escenarios para la revisión completa

- Realizar una serie de refinamientos
  - Producir diagramas apropiados para el trabajo realizado en los puntos anteriores
  - Definir jerarquías de clase apropiadas
  - Crear agrupamientos basados en clases comunes

- Implementar clases y objetos (en el contexto del Análisis Orientado a Objetos esto implica complementar el modelo de análisis)

De la misma forma que en el macro proceso, en el micro proceso Booch plantea lo que se debe de hacer más no dice el cómo hacerlo.

Esta metodología utiliza una gran cantidad de gráficos para generar presentar su análisis y diseño, esta cualidad en la práctica la mayoría de las veces se vuelve una desventaja.

3.6.2 Metodología de Coad y Yourdon[35]

Esta metodología se considera, con frecuencia, como uno de los métodos del análisis orientado a objetos más sencillos de aprender. La notación del modelado es relativamente simple y las reglas para desarrollar el modelo de análisis son evidentes:

- Identificar objetos usando el criterio de "qué buscar"
- Definir una estructura de generalización-especificación
- Definir una estructura de todo-parte
- Identificar temas (representaciones de componentes de subsistemas)
- Definir atributos
- Definir servicios

En la metodología de Coad y Yourdon, en la fase de análisis se propone que se haga en cinco fases las cuales tienen un nombre particular:

- Temas: "El área problema se descompone en temas, que corresponden con la noción de niveles o capas en los diagramas de flujo de datos"[36], algo interesante que propone es que los temas deben de contener entre cinco y nueve objetos.
- Objetos. Se deben de identificar los objetos con detalle, en esta parte no mencionan de forma completa del cómo debe de hacerse esto.
- Estructuras. Se identifican las estructuras de herencia, esta parte es sumamente técnica y no ahondan en la forma en que se hace esto.

[35] Yourdon, E. and Coad, P. (1991)

[36] Grahan Ian (1996), p. 294

- Atributos. Se generan atributos detallados de los objetos.

- Servicios. Es la forma particular de nombrar a las operaciones, aquí se identifica las operaciones de los objetos.

El análisis en este método si bien es sencillo y tiene una notación de modelado bastante simple, resulta complejo para analistas no computacionales, que en la práctica no están involucrados con el análisis y diseño de sistemas computacionales, además de que cae en el problema de decir qué elementos se deben de analizar, pero no el cómo analizarlos.

3.6.3 Metodología de Jacobson

También llamada ISOO (Ingeniería del Software Orientada a Objetos), la metodología de Jacobson es una versión simplificada de Objectory, una metodología patentada, también desarrollada por Jacobson. Esta metodología se diferencia de otras por el énfasis en el caso de uso, que es una descripción o escenario que describe cómo el usuario interactúa con el producto o sistema, a continuación presento algunos de los puntos básicos de la metodología:

- Identifica los usuarios del sistema y sus responsabilidades globales, dentro de una tarea llamada casos de uso

- Construye un modelo de requisitos
  - Define los actores y sus responsabilidades
  - Identifica los casos de uso para cada actor
  - Prepara una visión inicial de los objetos del sistema y sus relaciones
  - Revisa el modelo usando los casos de uso como escenarios para determinar su validez

- Construye un modelo de análisis
  - Identifica objetos de interfaz usando información del tipo actor-interacción
  - Crea vistas estructurales de los objetos de interfaz
  - Representa el comportamiento del objeto
  - Separa subsistemas y modelos para cada uno
  - Revisa el modelo usando casos de uso como escenarios para determinar su validez

- El desarrollo es iterativo en los ciclos de análisis, construcción y pruebas

- El análisis de requerimientos está fuera de la metodología

Algo que hay que destacar de esta metodología es que a pesar de ser una de las metodologías más robustas en la actualidad y está rodeada de distintas herramientas para poder utilizarla, el que se base en un conjunto de requerimientos que no son obtenidos dentro de la misma, puede en algunos casos ser una debilidad, ya que en la práctica, el construir sistemas computacionales basándose en estudios deficientes de los sistemas tiene como consecuencia invariable, la crisis del software con sus elementos de altos costos, poca fiabilidad y escasez.

3.6.4 Métodología de Rumbaugh

Rumbaugh y sus colegas en la empresa General Electric desarrollaron la Técnica del Modelado de Objetos (OMT) para el análisis, diseño del sistema y diseño del nivel de objetos, cabe mencionar que esta metodología es considerada una de las más completas dentro del análisis orientado a objetos además de que se le ha asociado fuertemente con el lenguaje de programación C++, y es un estudio precursor de UML. Su notación es en extremo rica, esto hace que en varios de los casos en donde se aplica se vuelva complicada y detallada, para salvar este punto tiene a su favor que existen un buen número de herramientas computacionales para su implementación de forma automática.

La actividad de análisis dentro de esta metodología crea tres modelos: el modelo de objeto (una representación de objetos, clases, jerarquías y relaciones), el modelo dinámico (una representación del comportamiento del sistema y los objetos) y el modelo funcional (una representación de alto nivel de flujo de información a través del sistema). "El análisis presupone que existe una especificación de los requerimientos (o, al menos, no proporciona ninguna técnica con estos fines)"[37].

A continuación presento una descripción de las tareas a realizarse dentro de esta metodología:

- Desarrollar una declaración del ámbito del problema

[37] Graham Ian (1996), p 303

- Desarrollar un modelo de objetos
  - Identificar clases relevantes al problema
  - Definir atributos y asociaciones
  - Definir enlaces de objetos
  - Organizar las clases de objetos usando herencia

- Desarrollar un modelo dinámico
  - Preparar escenarios
  - Definir eventos y desarrollar una taza de eventos para cada escenario
  - Construir un diagrama de flujo de eventos
  - Desarrollar un diagrama de estados
  - Revisar el comportamiento para comprobar consistencia

- Desarrollar un modelo funcional para el sistema
  - Identificar entradas y salidas
  - Usar diagramas de flujo de datos para representar transformaciones del flujo
  - Desarrollar EP (Especificación del Proceso) para cada función
  - Especificar criterios de restricciones y optimización

Una vez que se ha llevado a cabo el análisis, le siguen de forma secuencial las etapas de diseño, construcción y mantenimiento.

3.6.5 Método de Wirfs-Brock

El método de Wirfs-Brock no hace una distinción clara entre las tareas de análisis y diseño. En su lugar, propone un proceso continuo que comienza con la valoración de una especificación del cliente y termina con el diseño:

- Evaluar la especificación del cliente
- Usar un análisis gramatical para extraer clases candidatas de la especificación
- Agrupar las clases en un intento de determinar superclases
- Definir responsabilidades para cada clase
- Asignar responsabilidades para cada clase
- Identificar relaciones entre clases
- Definir colaboraciones entre clases basándose en sus responsabilidades

- Construir representaciones jerárquicas de clases para mostrar relaciones de herencia
- Construir un gráfico de colaboraciones para el sistema

Un punto que hay que resaltar de esta metodología es que parte de una especificación de un requerimiento del cliente, cosa que en la práctica es sumamente peligroso, debido a que muchas veces los clientes no tienen una idea clara de sus necesidades.

3.6.6 Comparación de metodologías

Antes de realizar una comparación de metodologías, es necesario tomar en cuenta que las presentadas aquí son sólo una muestra de un gran número de metodologías que existen en el mundo, y que se enseñan en las universidades y se aplican en las empresas, y que han llegado hasta nuestro país, A continuación menciono sólo algunas otras metodologías[38]:

- Shlaer/Mellor
- Martin/Odell –Ptech
- Objetory
- OORASS (Análisis, Síntesis y Estructuración de Papeles Orientado a Objetos)
- Desfray
- OSA (Análisis de Sistemas Orientado a objetos)
- Systems Engineering Oriented Objects
- Texel
- BON – Nerson
- Fusion – Coleman

- FOA
- CGI
- Henderson-Sellers
- MOSES
- ADM3
- Berard
- Syntropy
- COOSD
- ORCA
- ALEX.OBJ
- MOOD
- OSDL
- OSMOSYS
- SYS-P-O

Aunque la terminología y etapas del proceso para cada uno de las metodologías de análisis orientado a objetos difieren, los procesos generales son en realidad muy similares. Para realizar un análisis orientado a objetos, el ingeniero de software debe ejecutar las siguientes etapas genéricas:

- Obtener los requisitos del cliente para el Sistema Orientado a Objetos

[38] En el libro de Ian Graham se pueden encontrar más metodologías, y la explicación de las que aquí presento.

- Identificar escenarios o casos de uso
- Construir un modelo de requisitos
- Seleccionar clases y objetos usando los requisitos básicos como guías
- Identificar atributos y operaciones para cada objeto del sistema
- Definir estructuras y jerarquías que organicen las clases
- Construir un modelo objeto-relación
- Construir un modelo objeto-comportamiento
- Revisar el modelo de análisis OO en relación a los casos de uso/escenarios

Un punto crítico en general de las diversas metodologías es la necesidad de la definición de los requerimientos del cliente, pero en la mayoría de los casos nos dice qué necesita más no el cómo obtenerlo, lo que da pie a un intenso proceso creativo del ingeniero en computación, que genera un sinnúmero de formas de trabajo, la mayoría de las veces incorrectas, con lo que se genera una especie de anarquía computacional al momento de definir requerimientos del cliente para construir software de gran tamaño. Como mencioné en páginas anteriores de este capítulo, el no encaminar los esfuerzos para obtener los requerimientos del cliente, crea confusión, que en la práctica profesional nos lleva a la crisis del software.

En el siguiente punto presento una forma de describir sistemas muy difundida en la actualidad, a la cual hay que poner atención, debido a que se está convirtiendo en un estándar en el ámbito del desarrollo de software.

3.7 UML (Unified Modeling Language)

3.7.1 Historia y conceptos

UML significa Lenguaje de Modelado Unificado, ha sido desarrollado por Grady Booch, James Rumbaugh, Ivar Jacobson y numerosas personas y organizaciones a partir de 1994, desde ese tiempo y hasta la fecha se encuentra en constante evolución y mejora, contiene además de los conceptos de las metodologías de Booch, Rumbaugh y Jacobson, conceptos

de otras personas dedicadas a la creación de metodologías orientadas a objetos de las cuales sólo menciono algunas: Peter Coad, Derek Coleman, Ward Cunningham, David Harel, Richard Helm, Ralph Johnson, Stephen Mellor, Bertrand Meyer, Jim Odell, Kenny Rubin, Sally Shlaer, John Vlissides, Paul Ward, Rebecca Wirfs-Brock y Ed Yourdon.

Diversas organizaciones como Microsoft, Hewlett-Packard, Oracle, Sterling Software MCI Systemhouse, Unisys, ICON Computing, IntelliCorp, i-Logix, IBM, ObjectTime, Platinum Technology, Ptech, Taskon, Reich Technologies, Softeam se asociaron con Rational Software Corporation[39] para dar como resultado nuevas versiones de UML, actualmente sigue en evolución, esta herramienta, la cual se está convirtiendo en un estándar en México, diversas organizaciones la han adoptado como una herramienta de trabajo. Razón por la cual presento las características principales de UML en este apartado.

UML puede ser utilizado con diversas metodologías, a lo largo del proceso de construcción de software de gran tamaño. Nos sirve para especificar, visualizar, construir y documentar aspectos de un sistema, sobre el cual se está trabajando para realizar software.

UML surge a partir de tres necesidades:

- La modelación de sistemas utilizando técnicas orientadas a objetos
- Soportar la complejidad de sistemas de gran tamaño
- Tener un lenguaje estándar, que pudiera ser utilizado por personas y/o máquinas.

UML está basado en cuatro principios básicos de modelado:

- "La elección de qué modelos crear tiene una profunda influencia sobre el cómo se acomete un problema y cómo se da forma a una solución"[40]
- "Todo modelo puede ser expresado a diferentes niveles de precisión"[41]
- "Los mejores modelos están ligados a la realidad"[42]

[39] Empresa en la que se desarrollo originalmente UML, comenzando en ella Booch y Rumbaugh, y uniéndoseles más tarde Jacobson
[40] Booch, Rumbaugh, Jacobson (1999), p. 7
[41] Ibidem, p.8

- "Un único modelo no es suficiente. Cualquier sistema no trivial se aborda mejor a través de un pequeño conjunto de modelos casi independientes"[43]

UML al ser un lenguaje de modelado, nos describe lo que se supone que hará un sistema de software, pero no nos dice como implementar dicho sistema, ni tampoco nos dice la forma en que se llegó a la conclusión de lo que hará el sistema. Se utiliza no solo en los sistemas de software, sino también en los sistemas de hardware, así como en los sistemas del mundo real. El Lenguaje Unificado de Modelado nos ofrece notaciones y diagramas estándar para modelar sistemas orientados a objetos, y explica la semántica de lo que estos diagramas y símbolos significan.

3.7.2 Características de UML

El UML es un lenguaje para construir modelos de un sistema; no le dice al ingeniero en computación la forma de realizar el análisis y diseño orientados a objetos ni le indica cuál proceso de desarrollo adoptar.

Con UML se puede describir un sistema bajo diferentes niveles de abstracción, con lo cual se puede reducir la complejidad, sin perder detalles necesarios para los ingenieros en computación y usuarios encargados del desarrollo de software de gran tamaño.

UML divide un sistema en diferentes vistas, representadas por diagramas especiales, los cuales en su totalidad representan la arquitectura del proyecto. Es necesario recalcar que UML no es un modelo estándar de desarrollo, sólo es un lenguaje de modelado; las metodologías que vimos en el punto anterior de este capítulo sí definen procesos concretos de desarrollo, UML recomienda usar los procesos de desarrollo definidos por otras metodologías.

3.7.3 Diagramas de UML

En todas las ramas de la ingeniería se construyen modelos, que son abstracciones del mundo real, esto se hace para comprender mejor el sistema computacional que se construirá. Los arquitectos utilizan y construyen planos de los edificios, los diseñadores de coches preparan modelos en sistemas CAD/CAM y como ya mencioné a lo largo de todo

[42] Ibidem
[43] Ibidem

este trabajo, los ingenieros en computación deben al igual que otros profesionistas construir y utilizar modelos de software.

UML utiliza los siguientes tipos de diagramas:

- Diagrama de clases. "Un diagrama de clases muestra un conjunto de clases, interfases y colaboraciones, así como sus relaciones. Estos diagramas son los diagramas más comunes en el modelado de sistemas orientados a objetos".

- Diagrama de Objetos. "Un diagrama de objetos muestra un conjunto de objetos y sus relaciones".

- Diagrama de casos. "Un diagrama de casos de uso muestra un conjunto de casos de uso y actores".

- Diagrama de interacción. "Un diagrama de interacción muestra una interacción, que consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos".

- Diagrama de secuencia. "Un diagrama de secuencia es un diagrama de interacción que resalta la ordenación temporal de los mensajes".

- Diagrama de colaboración. "Un diagrama de colaboración es un diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben mensajes".

- Diagrama de estados. "Un diagrama de estados muestra una máquina de estados, que consta de estados, transiciones, eventos y actividades".

- Diagrama de actividades. "Un diagrama de actividades es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema".

- Diagrama de despliegue. "Un diagrama de despliegue muestra la configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos".

Todos estos diagramas de UML se clasifican en dos grandes grupos: los estáticos o estructurales que nos muestran la estructura del sistema y los dinámicos o de comportamiento, que nos muestran el como se comporta el sistema. En la figura 3.8 muestro esta clasificación.

Figura 3.8 Clasificación de diagramas del UML

La práctica de crear diagramas para visualizar sistemas desde perspectivas o vistas diferentes es muy amplia. En UML existen cinco vistas complementarias que son las más importantes para visualizar, especificar, construir y documentar la arquitectura de los sistemas computacionales de gran tamaño las cuales menciono a continuación:

- Vista casos de uso: formada con los diagramas de casos de uso, colaboración, estados y actividades.

- Vista de diseño: formada con los diagramas de clases, objetos, colaboración, estados y actividades.

- Vista de procesos: formada con los diagramas de la vista de diseño. Recalcando las clases y objetos referentes a procesos.

- Vista de implementación: formada con los diagramas de componentes, colaboración, estados y actividades.

- Vista de despliegue: formada con los diagramas de despliegue, interacción, estados y actividades.

Podemos concluir que UML tiene un gran número de tipos de diagramas, en la práctica se pueden definir cuáles pueden ser utilizados un para determinado proyecto. Todos los diagramas mencionados son bidimensionales. UML también permite la definición de diagramas tridimensionales.

3.8 Metodologías rápidas o ágiles

Durante este capítulo he presentado diversos esfuerzos realizados durante las últimas cuatro décadas dentro del ámbito de la ingeniería del software a fin de combatir la crisis del software, y no obstante todo este esfuerzo se han desarrollado otro tipo de estudios, dentro de los cuales destaca el de las metodologías rápidas, las cuales tienen fundamentos distintos a los presentados anteriormente en este capítulo. A continuación explicaré a grandes rasgos algunos de los aspectos más importantes de este tipo de metodologías.

Las metodologías "clásicas" o "tradicionales" presentadas en puntos anteriores de este capítulo se centran en los procesos, como el de análisis, diseño, construcción, implementación, mantenimiento, etc., y todas las descomposiciones en subprocesos que pueden hacerse de los primeros, por otro lado, las metodologías rápidas se basan principalmente en el equipo de trabajo y en los productos a entregar al cliente.

Las metodologías rápidas dan una importancia alta al ser humano, a las interacciones entre desarrollador y cliente, y al desarrollo incremental del software.

En el año 2001, se llevo a cabo una reunión en Utah-EEUU, en donde nace el término "ágil" para la ingeniería del software, el cual fue acuñado por un conjunto de expertos en la

materia. Tiempo después de esa reunión se formó The Agile Alliance, una organización cuya misión era promover y enriquecer los conceptos de las metodologías rápidas, basando su filosofía en el manifiesto ágil[44] que presento a continuación.

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

VII. El software que funciona es la medida principal de progreso.

VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

X. La simplicidad es esencial.

XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

[44] Tomado de ISSI (2003)

XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Una de las metodologías ágiles más difundidas actualmente es la XP (Programación Extrema), de la cual explico su proceso de desarrollo en la figura 3.9.

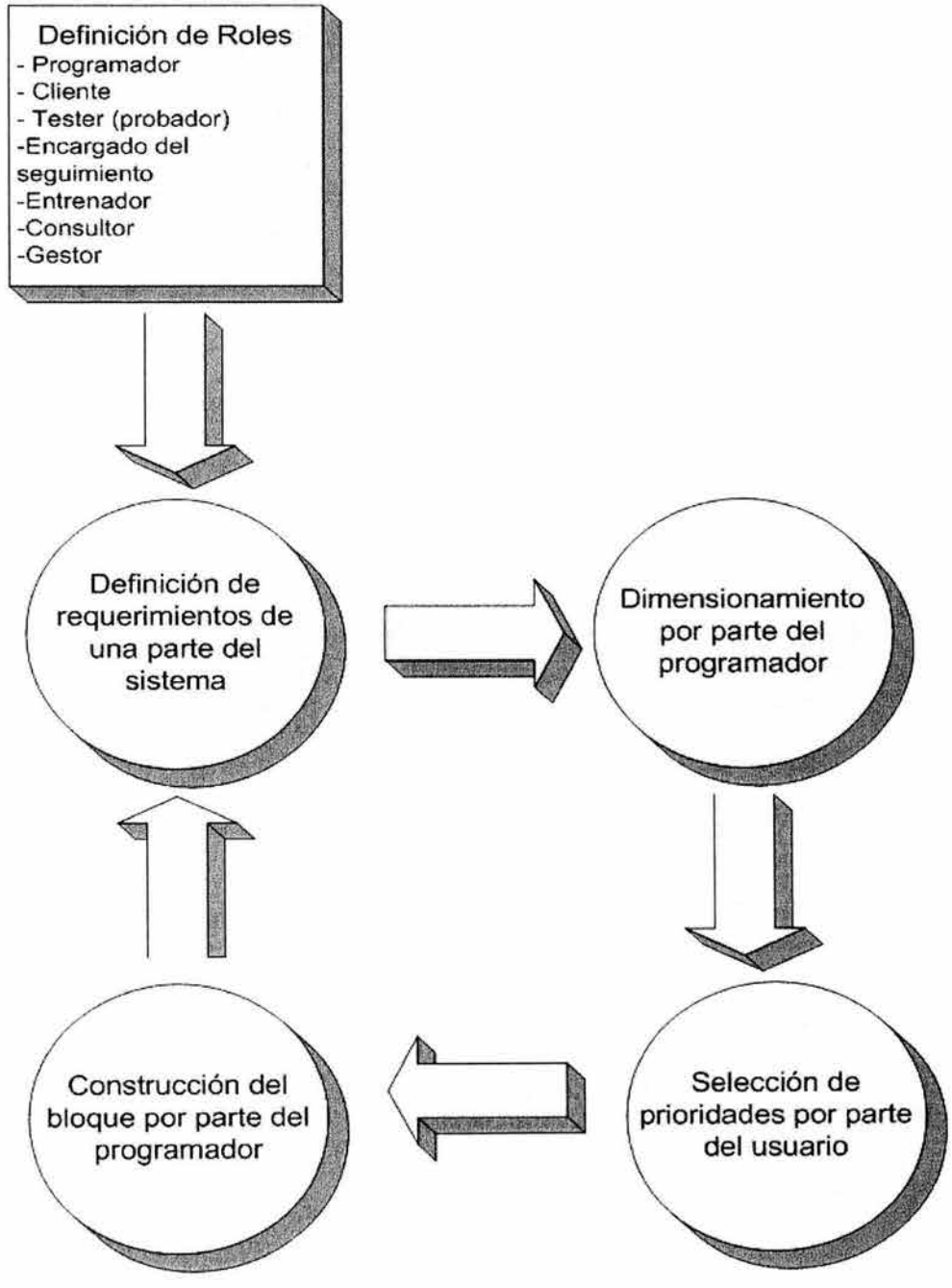


Figura 3.9 Proceso de desarrollo de la metodología XP (Programación Extrema)

Los autores de las metodologías rápidas proponen que sean utilizadas para desarrollos pequeños, y en este trabajo se ha planteado la premisa de hablar de software de gran tamaño, por lo que considero importante explicar que las menciono, debido a que las metodologías clásicas han descuidado elementos importantes del desarrollo de software como son los equipos de trabajo, la interacción con el cliente, etc., además de que las metodologías rápidas para cualquier desarrollador experto le sonarán muy familiares, debido a que los grandes proyectos de desarrollo de programas computacionales que se plantean llevarlos acabo bajo estándares rígidos y metodologías robustas, el proceso de desarrollo termina siendo semejante al de las metodologías ágiles.

En la tabla 3.1 presento una comparación entre las metodologías rápidas y las tradicionales.

Metodologías Ágiles	**Metodologías Tradicionales**
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Especialmente preparados para cambios durante el proyecto
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos
Poca documentación	Más documentación

Tabla 3.1. Comparación entre metodologías ágiles y tradicionales[45]

Durante los últimos años, en universidades y empresas, se ha trabajado en este tipo de metodologías, y en la actualidad existen diversos ejemplos, para concluir este capítulo, menciono sólo el nombre de algunas de ellas y el sitio Web en donde consultarlas.

- SCRUM (www.controlchaos.com)
- Crystal Methodologies (www.crystalmethodologies.org)

[45] Tomado de ISSI (2003)

- Dynamic Systems Development Method (DSDM) (www.dsdm.org)
- Adaptive Software Development (ASD) (www.adaptivesd.com)
- Feature -Driven Development (FDD)(www.featuredrivendevelopment.com)
- Lean Development10 (LD) (www.poppendieck.com)

A manera de resumen recordemos que en este capítulo se discutieron los elementos básicos de la programación orientada a objetos, diversas metodologías de análisis y diseño orientado a objetos, además presenté el lenguaje de modelado unificado y otro tipo de metodologías de desarrollo de software llamadas rápidas.

En el siguiente capítulo presento la guía objetivo de este trabajo.

4 GUÍA DE ANÁLISIS ORGANIZACIONAL BAJO UN ENFOQUE SISTÉMICO PARA EL DISEÑO DE SOFTWARE ORIENTADO A OBJETOS

4.1 Antecedentes

4.1.1 Visión del consultor

El ingeniero en computación encargado de desarrollar grandes sistemas de software para las empresas, es contratado en la actualidad vía el outsourcing, y llamado de diferentes maneras en las empresas, ingeniero en computación, ingeniero de software, ingeniero de sistemas, consultor, etc., y es este último nombre el más utilizado en muchas empresas en nuestro país; por lo cual, sabiendo que el ingeniero en computación es un consultor en las empresas, comienzo con el desarrollo de esta guía motivo de este trabajo, en donde hablaré de forma indistinta del ingeniero en computación y del consultor, que para este estudio son una misma persona.

El entrar en una empresa como consultor es el inicio de una relación comercial, que como todo tipo de relación, (social, afectiva, deportiva etc.) tiene que alimentarse; después de múltiples pláticas con varios de mis colegas que han estado en distintas empresas como consultores, coincidimos, que la mejor forma de alimentar una relación comercial es por medio del conocimiento sólido de consultor que debe de ser aplicado en beneficio de la empresa, además de una actitud positiva hacia el trabajo y hacia la organización. Existen otros elementos que de una u otra forma coadyuvan a reforzar la relación comercial, como creación de amistades personales con el empresario, compañerismo con los empleados de la organización, respeto, etc.

Debe de ser claro que el beneficio hacia el consultor es inherente cuando él está dedicando su tiempo y esfuerzo hacia el cumplimiento de las metas de la empresa. Para esto es importante que el consultor esté enterado de los precios que rigen el mercado con respecto a su trabajo con el objetivo de que la cantidad de dinero que él recibirá por prestar sus servicios sea la adecuada a fin de que permitan que se desarrolle como persona y como profesionista. Un

punto también importante es que el consultor cuente con los elementos necesarios para saber manejar un proyecto, con respecto a tiempos y costos.[46]

Existen tres conceptos básicos de la evaluación de proyectos que se deben de tener presentes cuando se está en el campo de trabajo.

Un proyecto es distribuidor de la riqueza. El realizar un proyecto debe de traer beneficios a todos los actores participantes, no necesariamente monetarios, hablo de riqueza en su definición más amplia, riqueza en conocimiento, en valores, en dinero, etc. Aquí estamos hablando de cliente y consultor. Debe de haber un Ganar- Ganar.

Un proyecto es transitorio. Debe de tener un inicio y un fin muy bien establecido. Esto debe de quedar bien asentado en el contrato de prestación de servicios, para evitar pugnas posteriores.

Un proyecto debe dar un producto o servicio específico. Esto evita muchos dolores de cabeza, al delimitar cabalmente lo que le se le va a entregar al cliente una vez que se termine el proyecto o durante su desarrollo.

La misión más importante de estos tres puntos es el evitar ambigüedades, las cuales siempre de una u otra forma traen problemas al consultor, y degradan la relación comercial que se ha establecido.

No es el propósito de este trabajo ahondar en temas de administración y evaluación de proyectos[47], pero sí es necesario mencionar que existen puntos importantes en estas materias que hay que tomar en cuenta, para poder llevar a un buen fin el desarrollo que se nos ha confiado, los puntos conceptuales tanto de administración como de evaluación de proyectos que considero esenciales y básicos para manejar un proyecto de desarrollo de software de gran tamaño son los que a continuación explico[48]:

Control del proyecto: El tener el control del proyecto implica para el consultor el poder informar al cliente de manera precisa y a tiempo, factores primordiales para reafirmar la

[46] Para esto existen cursos de evaluación y administración de proyectos

[47] Para mayor referencia de los temas de administración y evaluación de proyectos existen diversos trabajos, entre los cuales se encuentran las notas de clase del M.I. Andrés Romo [Romo(2000)] citadas en la bibliografía al final de este trabajo.

[48] Puntos basados de las notas de clase de la materia de Evaluación de Proyectos impartida por el M.I. Andrés Romo [Romo (2000)]

informar de esta manera, el cliente sabrá que la persona en la que está depositando no sólo su confianza sino sus recursos, está respondiendo de manera adecuada a la organización. Otro aspecto en este punto es que el consultor con un buen control del proyecto podrá organizar de manera adecuada la utilización de sus recursos.

Planificación y programación: Estos puntos deben de ser estudiados y desarrollados con gran precaución, debido a que un mal estudio, de estos temas, llevará al consultor a empezar a incumplir con los tiempos de entrega que se planteó al inicio del proyecto. Este incumplimiento es uno de los grandes problemas de los desarrollos de software de gran tamaño, que orilla a la crisis del software, por generar costos elevados en el proyecto. Para evitar lo anterior es más adecuado según mi experiencia, definir de manera realista, legal y competitivamente los tiempos de desarrollo. Es necesario que antes de la firma del contrato se puedan redefinir tiempos, actividades y costos a fin de poder cumplir con los compromisos que nos fijamos. El manejo de la técnica PERT[49] puede ser utilizada en este punto, su conocimiento puede ahorrarnos muchos problemas en el futuro, porque es una técnica para coordinar proyectos a gran escala, basada en redes, con la cual se evalúan los tiempos de desarrollo de un proyecto, además de poder monitorear el avance del mismo.

Padrinazgo: La parte del padrinazgo es uno de los puntos que yo llamo no escritos del trabajo del consultor. El tener un padrino, implica el tener a alguien que ayude y se comprometa con el desarrollo del proyecto, lo que en las metodologías rápidas llaman gestor. Para esto es esencial llegar a la persona correcta que tomará este papel. Siempre es adecuado buscar a esa persona que tiene el papel del "poder atrás del poder". Se puede llevar a un nivel superior el papel de padrinazgo, logrando que la mayoría de las personas de la organización funjan como padrinos, y así se estará en posibilidades de establecer una mejor cooperación por parte de los participantes de la organización punto que discutiré ampliamente más adelante y que es el estudio de stakeholders.

Manejo de riesgos: Con planificaciones efectivas se minimizan los riesgos inherentes al trabajo del consultor, estos riesgos son múltiples y toman diversas formas. Creo yo, en este punto se deben de estudiar los riesgos generados por competencias desleales, muy de moda en el mundo de la consultoría.

[49] Hillier,Hillier,Lieberman (2000), p. 243-296

Papel del cliente: La palabra asociación, es la que mejor define la relación que debe de haber entre el cliente y el consultor. Pero yo añado la de confianza, que refuerza a la asociación misma, y que es el medio de ganar relaciones comerciales a largo plazo.

Hasta aquí he explicado algunos de los puntos que deben de considerarse para una correcta puesta en marcha de un proyecto. Pero además de este tipo de consideraciones es importante tener conocimientos de otras materias que yo marcaría como primordiales para un desarrollo y consecución óptimo de cualquier trabajo a nivel empresarial, los cuales se presentan en el siguiente punto.

4.1.2 Conocimientos del consultor

El ingeniero en computación es un consultor empresarial, lo que a su vez lo convierte en términos más comunes, en consejero de organizaciones. Por esto, es importante que cuente con un conjunto de conocimientos previos que deben de estar sólidamente establecidos como su entorno intelectual; esto puede ser la diferencia entre un consultor especializado y un consultor común, de los cuales existen muchos en el mercado.

Es importante recalcar que en grandes proyectos el ingeniero en computación es parte de grandes equipos de trabajo, dentro de los cuales la mayoría de las veces se cuenta con expertos en distintas disciplinas como contables, administrativas, financieras, etc., pero esto de nada sirve si la persona que se va a encargar de realizar el desarrollo del sistema de software (el ingeniero en computación) no tiene un panorama sólido de estas u otras disciplinas, ya que se pierde mucho tiempo y recursos en hacer que este comprenda y pueda plasmar en programas computacionales lo que otros expertos le explican.

Mencioné a lo largo del capítulo uno y dos que una organización puede verse como un sistema, y cualquier sistema puede ser objeto de estudio, pero es necesario acotar este punto, el estudio aquí presentado esta encaminado a las organizaciones de tipo empresarial, las cuales son especializadas, y se rigen en forma general por principios de administración de empresas, contabilidad y finanzas, y requieren de grandes desarrollos de software como por ejemplo:

- Bancos
- Instituciones financieras
- Casas de bolsas
- Escuelas

- Comercios
- Industrias

No por esto esta guía no podría ser aplicada en otro tipo de organizaciones más especializadas, que si bien también se rigen por conceptos de administración, contabilidad y finanzas, su columna vertebral son conocimientos altamente especializados, y la mayoría de las veces no requieren sistemas de gran tamaño como se discutió ampliamente en el capítulo dos. De este tipo de organizaciones presento algunos ejemplos:

- Organizaciones de monitoreos volcánicos
- Plantas de tratamientos de aguas residuales[50]
- Centros de estadística
- Plantas de procesamientos automatizados

Basado en los párrafos anteriores puedo decir que los conocimientos básicos (pueden haber otros) de todo consultor computacional están englobados en tres grandes rubros:

Administrativos.- El objetivo del consultor es mejorar el desempeño de una empresa, ese objetivo, la administración lo ha perseguido desde hace muchas décadas. La literatura es extensa acerca del proceso administrativo y su aplicación en las empresas.

Contables[51].- Un lenguaje común para los empresarios es el contable, para esto es necesario que el ingeniero en computación conozca, el manejo total de una empresa desde el punto de vista contable, empezando desde la aplicación de las diversas pólizas, hasta la interpretación de los estados financieros.

Financieros.- Tener conocimientos administrativos y contables lleva al consultor a establecer un lenguaje común con los empresarios y otros consultores de otras disciplinas, además de conocer la forma en que se maneja una empresa. Un conocimiento financiero, además nos coloca en posibilidades de no solo comprender el manejo de la empresa, sino el de hacerla crecer, por medio de la aplicación coherente y continua de conocimientos financieros.

[50] En el anexo II presento como ejemplo un estudio realizado para una planta de tratamiento de aguas residuales, en el que se utilizan aspectos de la guía aquí propuesta.

[51] Véase Romero 2000, Contabilidad Superior, una referencia apropiada para ingenieros en computación con conocimientos básicos de contabilidad.

Computacionales.- Debe de ser claro que los conocimientos antes expuestos sólo mencionan los conocimientos adicionales a los técnicos que debe de tener cualquier ingeniero de sistemas computacionales como son de programación, bases de datos, compiladores, redes, etc.

4.2 Primeras actividades

Haciendo un recuento, he planteado que el consultor exitoso debe de tener ciertos conocimientos en materia de administración, contabilidad, y finanzas, que debe de tener un enfoque sistémico y debe de tener actitud de servicio. Claro está que esto no es una receta, por ejemplo para los ingenieros en computación, además de esto, es importante tener sólidas bases de programación de computadoras, manejo de proyectos, nuevas tecnologías, bases de datos, electrónica, etc.

Una vez que se está dentro de la empresa, que se ha ganado la confianza inicial del cliente y que el grupo de trabajo del consultor ha sido seleccionado para realizar un proyecto de mejora de su empresa vía la implantación de software de gran tamaño, es momento de empezar a demostrar lo que el consultor y su equipo saben, esto siempre es importante, ya que cada muestra de conocimiento, es un generador de confianza por parte del cliente hacia el equipo de trabajo.

Antes de empezar a hablar del primer día de trabajo dentro de una nueva empresa, hago una recomendación que se hace dentro de las nuevas técnicas de venta[52] enseñadas a vendedores exitosos de Estados Unidos, que consiste en documentarse acerca del giro de la empresa, por ejemplo, si se va a trabajar con una empresa dedicada a la venta de muebles, al menos hay que investigar un poco acerca de sus principales competidores, de las empresas que son casos de éxito, de los principales problemas que enfrentan etc. Así la primer visita a la empresa, ya como los ganadores del proyecto, debe de considerarse como el inicio del estudio y no como una visita social.

4.2.1 Reconocimiento de actores

Para la realización de cualquier proyecto de desarrollo de software se debe de conformar un equipo de trabajo, y un punto muy importante de esto es definir los nombres de los participantes del equipo, por ejemplo en el argot futbolístico se tiene al portero, los defensas,

[52] Miller,Herman (1989)

los medios, los delanteros, etc., los cuales no sólo tienen un nombre sino actividades bien establecidas; se tiene que hacer una actividad semejante con el equipo de desarrollo de software, a fin de conocer los nombres de los participantes del equipo. Con la práctica, y la experiencia, son actividades que se hacen de inmediato, pero hay que tener mucho cuidado a pesar de ser consultores expertos, se debe recordar que dentro del grupo de trabajo puede haber consultores novatos que no tengan experiencia, por lo cual es necesario dejar bien asentados los nombres de los participantes del equipo.

Esta actividad tiene como objetivo evitar ambigüedades en el trabajo, centrar las bases a fin de poder empezar nuestro desarrollo de software.

La razón de ser y existencia de un consultor computacional, se debe a los problemas en las empresas, y al trabajo que se genera dentro de ellas, si estos no existieran simplemente los consultores computacionales no existirían, por eso comienzo con la definición de cliente:

Cliente.- Es la persona que comisiona un estudio con el fin de lograr "saber algo" o "hacer algo" relacionado con el sistema en que está inmerso.

Consultor.- Es la persona a la cual el cliente le encarga que realice un estudio con el fin de lograr "saber algo" o "hacer algo" relacionado con el sistema en que está inmerso.

Tomador de decisiones.- Es un individuo participe de un sistema que puede alterar tanto partes como relaciones del sistema.

Propietario del problema.- Es aquella persona que se encuentra inmerso o tiene una discrepancia entre el "es" y el "debiera ser" de algo y que en nuestro estudio puede ser una parte o el todo. Esta persona

NOTA: Todos estos roles pueden ser adoptados por personas o grupos diferentes o por una sola persona o un grupo y pueden ser ejecutores de actividades en el sistema excepto el consultor que es externo al sistema.

Con la identificación del tomador de decisiones hay que recordar el concepto de padrinazgo mencionado anteriormente, para analizar lo importante y benéfico que sería para el desarrollo del proyecto que el tomador de decisiones se vuelva el padrino del proyecto.

Para entender los roles explicados, planteo a continuación un ejemplo:

Existe una empresa distribuidora de refacciones para tractores, el señor Juan Pérez es el dueño de la misma, pero después de 25 años de trabajo su nivel económico le permite pasar la mayor parte del tiempo viajando por el mundo en compañía de su esposa, en un periodo de estancia en México se dio cuenta que su empresa no marchaba del todo bien, y llamó a su Gerente de finanzas el Sr. Luis el cual ha trabajado en la empresa desde hace veinte años y es el brazo derecho del dueño. además de tener una estrecha relación de amistad personal con el dueño, cuando el dueño se va de viaje, todos en la empresa saben que el que toma las decisiones es él.

El dueño de la empresa le pide explicaciones, a su Gerente de finanzas y él le explica que algo está pasando, que a pesar de ser una empresa que se ha mantenido por muchos años, no tiene idea de que sucede con el área de mercadotecnia, que está conformada por las mismas dos personas que hace diez años hicieron que la empresa duplicara sus ventas. El Sr. Luis cree que lo que está pasando es que se han quedado rezagados con respecto a las nuevas tendencias empresariales, no tienen página en Internet, no están seguros de que esta sirva para algo, no saben si deben de diversificar sus inversiones, etc. por eso el dueño de la empresa preocupado, le indica que genere una partida dentro del presupuesto para contratar un despacho con gente joven que les pueda explicar que está pasando con su empresa, y en dos meses después de que regrese de otro viaje por Europa vuelven a platicar.

Rápidamente con este ejemplo podemos identificar quienes son los actores, el Sr. Juan Pérez es el cliente, el Sr. Luis es el tomador de decisiones, el despacho que se contratará pasarían a ser los consultores, el propietario del problema es el departamento de mercadotecnia.

Hasta este momento he introducido la palabra problema, la cual bajo un enfoque sistémico debe de estar bien definida, lo que haré en el siguiente punto.

4.2.2 Definición de problema

Conforme avanzo en el planteamiento de la guía, lo que hago es llamar a las cosas por su nombre, tratando de que todas las partes tengan un nombre y se sepa cual es el papel que jugarán (no es un pensamiento sistémico inconsciente, es totalmente consciente).

Al discutir acerca de los actores, planteé algo nuevo al definir al propietario del problema, y es esa diferencia entre el "es" y el "debiera ser". Para continuar con un pensamiento

sistémico, pasaré a definir qué es un problema, que tiene relación con el "es" y el "debiera ser".

Cuando un consultor platica con sus clientes, ellos le dirán todo lo que pasa con su problema, le hablarán de la preocupación y del sufrimiento que este les causa, le dirán acerca de las muchas horas que pasan trabajando tratando de solucionar problemas y de lo mucho que descuidan a sus familias. Todo eso está bien, el consultor debe de hacer las veces de Psicólogo, pero para que el consultor le demuestre a su cliente que sabe del tema debe de definirle problema como:

PROBLEMA.- Es la discrepancia entre el es" y el "debiera ser".

Esto quiere decir, que se deben de empezar a ver los problemas como algo mas objetivo y hasta en determinado punto de vista tangible, quitando sentimientos, ambigüedades, etc. Se busca modificarle sus ideas al cliente acerca de lo que son los problemas, a fin de que como parte del equipo ayude a solucionarlos. En la figura 4.1, se muestra la definición de problema de manera gráfica.

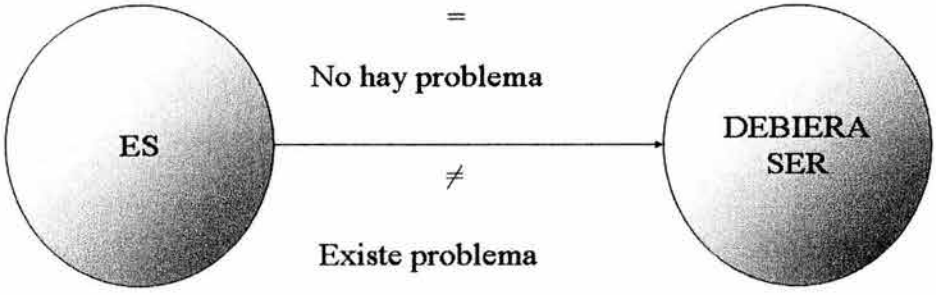


Figura 4.1 Definición de problema visto gráficamente

Con esta nueva definición de problema se puede comparar la forma de ver un problema de forma clásica y la forma de ver un problema a partir de la definición planteada

Forma clásica

Problema: "Ya no me alcanza para pagar la nómina algunas veces, eso me preocupa de sobremanera, ya que no quiero imaginarme el día en que tenga que retrasarle el pago a mis trabajadores.

Nueva visión del problema.-

Es: Cada fin de semana no se con seguridad si podré completar el dinero para el pago de la nómina

Debiera ser: Debería todos los fines de semana tener listo el dinero en la caja de la empresa para el pago de la nómina.

Como existe una diferencia entre el es y el debiera ser, el consultor computacional puede decirle al cliente con toda seguridad: "Señor, usted tiene un problema".

Según Checkland[53] los problemas pueden dividirse en dos tipos: los estructurados y los no estructurados. Un ejemplo de problema estructurado es: "transportar la máquina A de la bodega X a la bodega Y", y un ejemplo de problema no estructurado es: "en la fábrica existe descontento entre los obreros y no se qué hacer con esta situación". Hay que tener presentes esta separación de problemas, debido a que el ingeniero en computación puede enfrentarse a ambos al analizar un sistema.

4.2.3 El primer cuestionario

La forma en que se introduce un ingeniero en computación a una empresa, la mayoría de las veces, es caótica y desgastante, tanto para el consultor como para el cliente y en general las personas involucradas, debido a que típicamente, esta introducción se convierte en un conjunto de juntas de trabajo y visitas a las instalaciones en donde todo mundo le explica al ingeniero los problemas que tiene, y este toma notas hasta el cansancio, notas que

[53] Checkland(1993)

seguramente volverá a tomar en otro momento, con más detenimiento y con un nivel de detalle adecuado que no puede alcanzase en las primeras visitas.

Para evitar esa forma caótica de introducirse en las empresas, una alternativa es la aplicación del siguiente cuestionario, el cual le dará una idea al ingeniero en computación de las expectativas del cliente, los recursos, y los actores del sistema con el que se está involucrando, claro está que este cuestionario[54] conlleva conocimientos y definiciones planteadas en puntos anteriores, que el consultor debe de explicarle al cliente.

1.- ¿Cuáles son sus aspiraciones al contratar un consultor?

2.- ¿En su empresa, quiénes son los tomadores de decisiones y quiénes los propietarios del problema?

3.- ¿Cuáles son las razones para que los tomadores de decisiones y los propietarios del problema, llamen problema, al "problema".

4.- ¿Cuáles son los posibles nombres para el problema en el contexto del sistema?

5.- ¿Con qué recursos contamos para resolver el problema?

- Gente
- Experiencia
- Financieros
- Tiempo etc.

6.- ¿A lo que llamamos problema, estará resuelto cuando suceda, pase, se convierta o desaparezca qué?

El objetivo del cuestionario es tratar de encausar el pensamiento y los esfuerzos del cliente, en páginas anteriores ya hablé del papel del cliente dentro del proyecto, y es momento de empezar a descubrir que muchas de las soluciones a los problemas, de alguna manera las tiene el cliente sin saberlo. Esto de ninguna forma se convierte en un afán de que el cliente haga el trabajo del consultor, por el cual él está pagando, sino que hay que ser lógicos y

partir del supuesto siguiente: si alguien conoce la empresa, es el mismo cliente, y esto se acentúa cuando las empresas tienen en el mercado más de cinco años, es demasiado fantasioso el pensar que un consultor va a llegar a reinventar una empresa que se ha mantenido en el mercado por más de diez o veinte años, claro está que el consultor con sus conocimientos, aportará elementos importantes a la empresa, pero no hay que llegar con un sentimiento de querer volver a hacer todo.

Este tipo de cuestionarios reafirma la confianza que el cliente está depositando en el consultor. No son nuevos y muchas consultorías los hacen, pero creo que el principal problema de esos cuestionarios es que parecen hechos por la oficina del censo, preguntan un sin número de cosas que hacen que el cliente termine por fastidiarse y contesta cualquier cosa. En cambio este cuestionario a pesar de su reducido tamaño, conlleva de manera intrínseca todo lo relacionado al porqué un consultor está sentado con el cliente platicando.

Como experiencia personal, un cliente después de dos días de haberle dado el cuestionario, no se sentía seguro de contestarlo, y reafirmaba una y otra vez que ese cuestionario era muy difícil de resolverse, y de alguna manera lo es, ya que el hacer que el cliente empiece a tener un pensamiento sistémico no es tarea de un día. Por eso recomiendo que se le encamine al cliente con la resolución del cuestionario, pero solo encaminar, para que el cliente se de cuenta que el trabajo que está encomendando tiene mucho de fondo.

4.2.4 Ubicación de la empresa

Se debe de ubicar a la empresa en tiempo, lugar y sector. Esto para poder saber con qué se está trabajando, explicaré a continuación para que sirve este tipo de ubicaciones.

Tiempo: Da una idea clara de la importancia de la empresa, una empresa con muchos años en el mercado, (más de diez) quiere decir que algo dentro de su funcionamiento, esta siendo llevado de manera correcta dado que le ha permitido permanecer en un mercado en donde la muerte financiera de empresas es acontecer diario.

Lugar: Recordemos que ahora se está inmerso en un pensamiento sistémico, y que además existen niveles ambientales en donde se puede ubicar a una empresa, esto es, una empresa con una instalación única enmarcada en cualquiera de las delegaciones del D. F. Seguramente tendrá problemas diferentes con otra que tiene por ejemplo diversas sucursales distribuidas a lo largo de la región sureste del país.

[54] Tomado de Suárez Javier (2000)

Sector: Recordemos los tres sectores productivos.

Sector primario donde se producen bienes agrícolas, ganaderos y piscícolas.

Secundario se generan bienes industrializados.

Sector terciario o de servicios.

Algunos autores refieren el sector de extracción.

Todas estas actividades de ubicación no son otra cosa que una antesala para poder generar una conceptualización de la empresa desde un punto de vista sistémico.

El ubicar de esta forma a la empresa, posiciona al consultor en un marco de referencia que sabiéndolo utilizar va a ser una herramienta importante en la generación de soluciones.

Como parte de una primera ubicación de la empresa se debe de construir el primer mapa conceptual, que es de tipo de caja negra, el cual se explicó en el capítulo uno, en este punto reproduzco la figura 2.3 del mapa realizado para una empresa comercializadora de refacciones del distrito federal presentando la figura 4.2.

Figura 4.2 Mapa conceptual de tipo caja negra para un empresa comercializadora de refacciones

4.2.5 Planteamiento de objetivos

Alicia: ¿Qué camino debo tomar?
Gato: Eso depende del lugar hacia donde vayas.
Alicia: ¡No sé para dónde voy!
Gato: Entonces, ¡no importa cuál camino debas tomar!
Lewis Carroll,1872

El párrafo anterior sirve para pensar hacia donde se requiere encaminar el estudio que se realice para una empresa, y una primera respuesta lógica es hacia la resolución de los problemas de la empresa, o bien para cumplir con lo pactado en el contrato, otras tantas dirían hacia la obtención de un análisis de la empresa. Esto indica que dentro de la guía

también se deben de plantear objetivos para encausar de forma adecuada los esfuerzos de los consultores para llevar a buen término el proyecto de desarrollo de software.

No obstante que en el contrato se dejan asentados los trabajos que se van a realizar en la empresa, es necesario hacer un estudio más profundo de la forma en que se van a cumplir con esos compromisos asentados en el papel.

Existen diversas formas de plantear objetivos, unas se centran en las actividades, los indicadores, en determinados resultados etc. Pero lo que podemos encontrar en común es que existe un objetivo general que se encuentra en el nivel más alto, y que existen otros más específicos que ayudan a que se cumpla el objetivo más general.

Hay que tener presente que "todo proyecto debe de ser planteado, organizado, controlado y evaluado en función de sus objetivos.... los objetivos tienen una naturaleza jerárquica, parten de un ideal de una misión organizacional o sectorial a objetivos particulares de un proyecto y metas a seguir dentro del mismo. Cada uno de estos debe de ser suficiente para el cumplimiento del nivel superior de objetivos. Por otro lado, mientras se profundiza en la jerarquía se restringen los alcances y se hacen cada vez más específicos, hasta llegar a una actividad en particular... el paso más importante hacia una ejecución exitosa es desarrollar un conjunto estructurado de objetivos"[55].

También es fundamental establecer objetivos coherentes con la realidad de la empresa, que coadyuven a un desarrollo correcto del trabajo del consultor.

En esta parte de la guía, presento el uso de una herramienta, utilizada por el Banco Mundial en el desarrollo de proyectos a gran escala, para el planteamiento de los objetivos y la forma de cumplirlos que da lugar o que genera una estructura jerárquica.

A continuación explico a detalle lo que es el marco lógico.

> *"El marco lógico es una herramienta de apoyo para la toma de decisiones, para la planificación de los objetivos de un proyecto y consecuentemente para la planificación, evaluación y control de sus entradas y salidas. Los objetivos están organizados en forma piramidal partiendo del propósito central o motivo primario de un proyecto hacia sus metas específicas. Es un método fácil de entender y provee una estructura conceptual que lleva a una visión general de los aspectos más fundamentales en la concreción de un proyecto. Otra*

[55] Romo (2000), p. II-2

ventaja de este método, es el gran apoyo que otorga a la evaluación, porque las metas se delinean claramente y en forma explícita se identifica cómo se evalúa. Un tomador de decisiones puede hacer estimaciones realistas de los resultados y ubicar problemas que puedan ser encontrados.

Este procedimiento proviene directamente del enfoque sistémico y a su vez aprovecha elementos de administración por objetivos. Empezó a utilizarse internacionalmente en 1969 por la agencia bilateral de los Estados Unidos"[56].

El marco lógico permite:

"- Generar una visión general completa y concisa a la vez.
- Establecer vínculos directos con los distintos componentes en la formulación del proyecto. Facilitando con ello su elaboración.
- Brindar un marco para enfocar las dinámicas de grupo y tormentas de ideas en la determinación de fines, medios e indicadores.
- Simplificar la evaluación y el monitoreo."[57]

El marco lógico es una "representación matricial de los objetivos, escritos en forma jerárquica y definiendo los medios de verificación y suposiciones importantes para alcanzarlos"[58].

En la figura 4.3 se muestra la estructura del marco lógico.

[56] Ibidem p. II-3
[57] Ibidem p. II-4
[58] Ibidem

Descripción	**Indicadores**	**Medios de verificación**	**Suposiciones Importantes**
Meta Sectorial /empresarial			(Objetivo a superobjetivo)
Objetivo del proyecto			(Objetivo a meta sectorial
Salidas:			Salidas a objetivos
Actividades:			(Actividades a salidas)

Figura 4.3 Marco lógico

A continuación presento la explicación de sus partes:

"Salidas. Son los resultados deseables e indeseables de un proceso. Por ejemplo la basura de una industria de transformación es una salida del sistema, tanto como el producto principal.

Sector: es el sistema que engloba al proyecto o el macro- proyecto que le da origen. Por ejemplo la construcción de una presa, como parte de un programa agrario.

Indicadores. Son aquellas variables que objetiva y precisamente miden el cumplimiento de metas.

Medios de verificación: son los mecanismos específicos por medio de los cuales se miden los logros. Si se puede medir se puede administrar.

Los indicadores deben de ser orientados en función de Cantidad, Calidad y tiempo."[59]

La lógica para la creación del marco lógico es la siguiente:

"Esta organizado en forma matricial y sigue una lógica vertical horizontal. Verticalmente se aclara el por qué del proyecto, definiéndose los objetivos sectoriales y del proyecto; así como sus insumos o entradas y sus salidas esperadas. En forma horizontal se determinan los Qué. Qué es lo que se va a producir y qué evidencias hay para señalar su consecución. Así es como, horizontalmente enlista los indicadores los medios de verificación y las suposiciones.

Procedimiento.

1.- Se define la lógica vertical.

Se establece la cadena de objetivos jerárquicamente, partiendo de los objetivos generales de la empresa o sector, seguidos de los del propio proyecto y finalizando con las tareas o actividades que satisfacen enteramente a los señalados en el nivel anterior. Manteniendo el enfoque sistémico, al indicar las actividades se describen las entradas más relevantes.

La estructura conformada de esta manera, sigue los siguientes supuestos o hipótesis:

- *Si brindamos los siguientes insumos, entonces tendremos determinadas salidas;*
- *Si se alcanzan esas salidas, entonces se cumple el objetivo del proyecto;*
- *Si se cumple el objetivo de los proyectos, entonces se alcanzan las metas sectoriales.*

Cada nivel debe satisfacer las condiciones necesarias y suficientes para satisfacer al nivel superior. Es así como los insumos tienen que ser los estrictamente necesarios y suficientes para el o los productos (salidas). Los productos deben de ser necesarios y suficientes para alcanzar los objetivos, y estos a su vez, para las metas sectoriales.

2.- Se determina la lógica horizontal.

Para cada nivel se especifican: los indicadores que demuestran el nivel de avance con respecto a la meta deseada; los medios de verificación o mecanismos específicos para

[59] Ibidem

verificar el avance; y, las suposiciones que afectan el éxito de proyecto y los factores ambientales fuera del control del decisor.

En este paso, los responsables de la formulación deben de preguntarse lo que es importante y lo no lo que es fácil de medir. Después de definir el objetivo, se describen las condiciones que indiquen que se haya alcanzado, de acuerdo con los medios de verificación. Pueden existir múltiples indicadores, que señalen la consecución del objetivo. Es raro encontrar un indicador que por si solo, sea suficiente."

Con el fin de ilustrar la estructura y la lógica de creación en la figura 4.4 presento un marco lógico que realicé parta la misma empresa comercializadora de refacciones de la cual he trabajado en este capítulo. En donde se plantearon los objetivos a cumplir y en donde se planteó la forma de medir ese cumplimiento, en esa figura de pueden ver de manera clara los objetivos y la forma de alcanzarlos. Dentro de las actividades que se presentan en el nivel más bajo, se pueden ver algunas de que ya he explicado y otras que se explicaré más adelante.

Descripción	**Indicadores**	**Medios de Verificación**	**Suposiciones Importantes**
Meta empresarial: Organizar la empresa Para poder hacerla crecer	Aumento de ventas y disminución de gastos	Reportes de ventas y gastos	El dinero "salvado" por la reorganización de la empresa será invertido en el crecimiento de esta
Objetivo del proyecto: Detectar problemas de la empresa y corregirlos. Organizar la empresa Generar crecimiento empresarial.	Número de problemas solucionados. No. de departamentos reorganizados. Ganancias	Hojas de control de problemas detectados y soluciones. Reporte de ganancias.	Se tendrá acceso a toda la información de la empresa. El equipo de consultores, administrará la empresa y tomará decisiones.
Salidas del proyecto Organigrama. Manual de procedimientos Automatización de la empresa. Programas de mercadotecnia. Programas de capacitación a empleados.	Tareas realizadas por cada persona. Tiempo y errores al desempeñar un determinado proceso. No de reportes en papel. Reducción de la rotación del personal	Reporte de actividades de empleados. Reporte de metas alcanzadas. Reportes de anomalías o problemas de los departamentos Reporte de rotación de personal.	Se creará una cultura del uso de la información estadística.
Actividades: *Generación de mapas conceptuales de la empresa.* *Análisis de Stakeholders* *Identificación de problemas* *Solución de problemas.* *Reorganización por departamento.* *Reorganización de la empresa.* *Generación de manuales de procedimientos.* *Automatización de la empresa.*	Presupuesto 50,000 25,000 200,000 300,000 130,000 160,000 200,000 1,750,000 2,685,000	Reporte de avance del proyecto	Cada actividad será evaluada por la empresa. Consultores y capacitadores disponibles al costo estimado.

Figura 4.4 Marco lógico de un proyecto para una empresa comercializadora de auto partes

4.3 Módulos de la empresa

4.3.1 Cadena de valor

En secciones pasadas describí una situación típica del primer día de trabajo como consultores, y se podría pensar que el consultor ya se ha llevado toda una mañana en identificar a los actores del sistema y en contestar con el cliente el cuestionario de entrada, si esto todavía no se ha logrado, es válido empezar a dejarle tarea al cliente, recordemos que desde el punto de vista de ventas, la respuesta y el interés que un cliente ponga en el proyecto, puede ser medido por medio de tareas que hay que dejar que él realice, esto con un doble fin: involucrarlo en el proyecto tratando de crear una efectiva sinergia entre cliente – consultor, y el de medir el grado de compromiso que él tiene, si deja a un lado las tareas que se le asignaron, o evade juntas y en general muestra desinterés en el proyecto se deben de *prender focos rojos*, debido a que ese cliente, puede estar perdiendo la confianza, lo que podría significar una cancelación del proyecto.

Ya que el consultor se ha introducido en la empresa, y que se tiene un esbozo de un marco lógico para la realización de un sistema de software, es tiempo de que se le pida al cliente que muestre las instalaciones de su empresa, pero llevando en mente un objetivo, el de reconocer la cadena de valor y poder hacer un estudio de la misma. Esta será una primera aproximación hacia a la empresa, con el fin de que estudio para implantar software tenga un orden bien establecido, y no se empiece a generar una serie de notas sin razón, que por lo regular los consultores principiantes hacen.

El objetivo es generar la cadena de valor, para esto debemos de tener claro qué es y para qué sirve.

Una cadena de valor es una representación abstracta del flujo productivo que sigue una empresa.

Para poder entender de mejor forma esta definición observemos la cadena de valor hecha para la empresa distribuidora de refacciones automotrices que ya he estado analizando a lo largo de este capítulo.

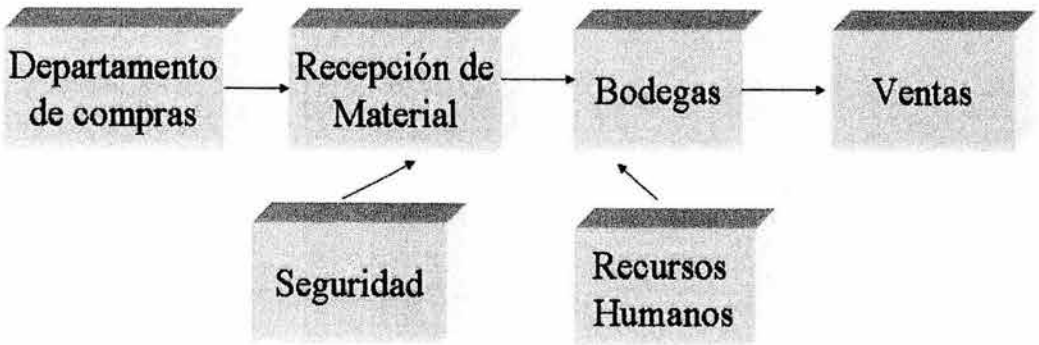


Figura 4.5. Cadena de valor de una empresa comercializadora de refacciones automotrices

Al visitar las instalaciones de la empresa hay que encaminar esta visita, el cliente por cortesía seguramente querrá mostrar la empresa en general, pero eso podría llevar muchos días; además de que hay que tener en cuenta que la visita debe de permitir, construir la cadena de valor que, como se aprecia en la figura 4.5, no debe de llevar demasiado tiempo. Por experiencia un cliente en su afán natural de querer dar pronta solución a sus problemas, guiará al consultor hacia el área que le esté provocando dolores de cabeza, y tratará de llenarlo de información, con mucha sutiliza hay que aceptar esa información, pero no hay que dejar que el cliente y el propio consultor queden enfrascados en esa área, ya que eso impide formar una visión general y sistémica de la empresa.

Siendo más explícito, veamos lo que he planteado hasta ahora como una primera visualización de una extensión de tierra, hecha desde un avión, se pueden ver a grandes rasgos los elementos que componen el sistema. Se puede ver que hay un río, pero no se sabe qué tipo de peces, se observa que existen grande coníferas, pero no se puede

distinguir la especie, y en general sólo se tiene una vista superficial del terreno, no hay que preocuparse, ya habrá tiempo para estudiar a fondo cada uno de los componentes de este vasto territorio.

Siguiendo con el ejemplo se puede apreciar lo importante que es conocer todo el panorama aunque de manera somera en un principio, supongamos que se determina que una determinada extensión de tierra, tiene problemas por falta de agua, de inmediato se podría decidir cavar pozos, tarea inútil si desde el avión se puede ver un río que se encuentra dentro del mismo territorio. Esto aunque parezca un poco ingenuo sucede muy a menudo en las empresas, un consultor podría preocuparse por mejorar por ejemplo el departamento de bodegas generando un gran sistema de software que manejara grandes espacios para poder apilar grandes cantidades de mercancía, y que se contratara personal especializado para manejar este sistema. Además, de que el sistema computacional interactuara con maquinaria especial para el transporte y manipulación de mercancía, y cuando se revisan los estados financieros de la empresa. El cliente y los consultores se dan cuenta que todas esas medidas lo único que generaron fueron gastos, en realidad la solución estaba en mejorar el departamento de compras, implementando políticas más eficientes de compra de mercancía, como por ejemplo verificando los efectos estaciónales sobre las mercancías, organizando y clasificando las mercancías con respecto a una clasificación ABC, todo lo anterior claro está soportado por un sofisticado sistema computacional.

Después de apreciar la importancia de la cadena de valor al generar un panorama general de la empresa, analizo la información que arroja la figura 4.5, que nos muestra una cadena de valor real elaborada para una empresa de comercialización de refacciones automotrices, junto con el mapa conceptual de la figura 4.2. De entrada, presenta las áreas que existen y como se da el flujo operacional en la empresa. El área de compras pacta la adquisición de mercancía con el proveedor, dicha mercancía es recibida por un área de recepción de mercancía, la cual manda a bodega la mercancía adquirida y de ahí se envía cuando se requiere al departamento de ventas. También existen otras áreas que son de apoyo a las actividades medulares de la empresa.

Del ejemplo anterior se infiere una concepción esquemática formal de una cadena de valor, misma que presento en la figura 4.6.

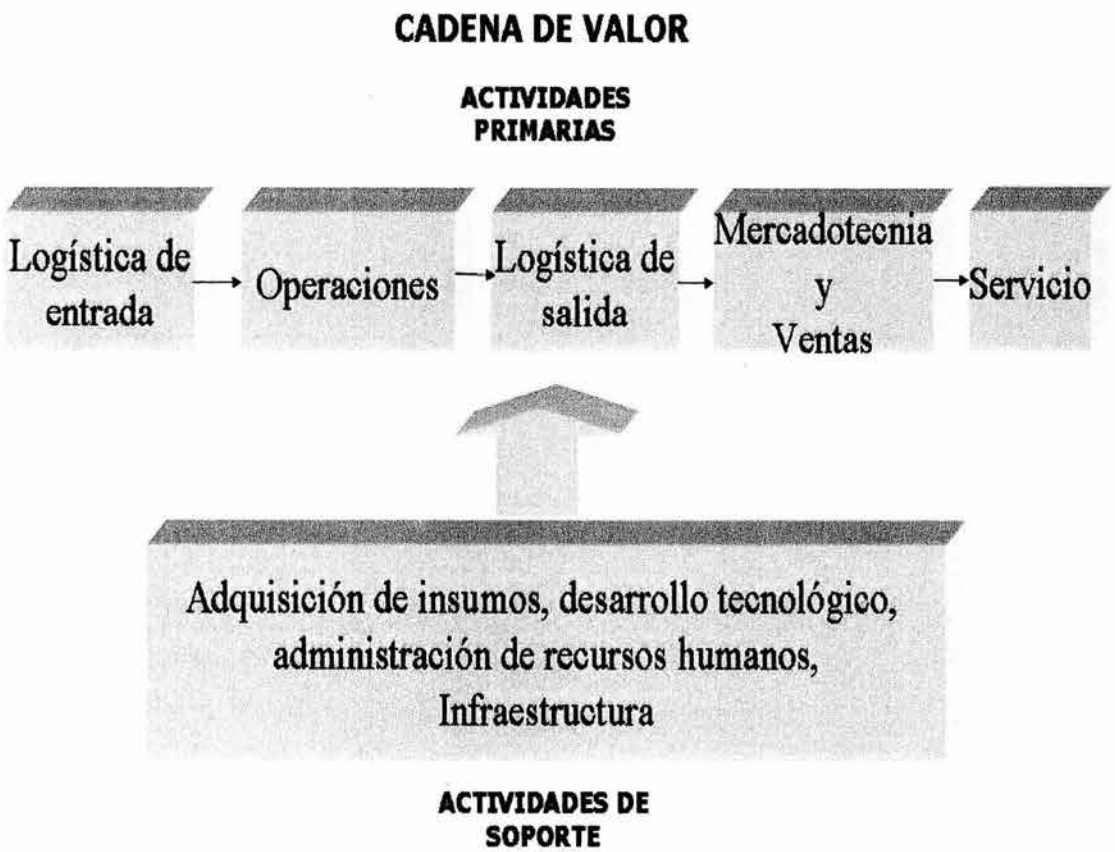


Figura 4.6 Representación conceptual formal de una cadena de valor

En esa figura 4.6 se aprecian las áreas fundamentales que sostienen a una empresa. Se distinguen las actividades principales de la empresa, que generan los productos o servicios directamente, y las de soporte. Por ejemplo para el caso de la empresa comercializadora de refacciones, las actividades primarias son las que están encaminadas a adquirir y vender refacciones, mientras que existen otro tipo de actividades que apoyan este trabajo, como las actividades financieras, contables, administrativas, de recursos humanos etc.

Hasta aquí ya se tienen identificadas las principales áreas de la empresa, el consultor ya tiene como tarea, construir la cadena de valor para poder mostrársela al cliente y que él valide que en realidad así está funcionando su empresa.

Hay que tomar en cuenta que las metodologías de análisis orientado a objetos en algunos de sus procesos, necesitaban la determinación de los módulos de un sistema, esta es una primera aproximación, hay que tomar en cuenta que el proceso de análisis puede llevarse a diferentes niveles, suponiendo que las áreas de la empresa están bien planteadas por el consultor y validadas por el cliente, ya se está en posibilidad de determinar las áreas a automatizar por parte del consultor, todo esto avalado por un marco lógico, que guíe el proceso de desarrollo.

4.4 Determinación de problemas en la empresa

Si en la empresa existen problemas funcionales, y el análisis para la implantación de software no considera la identificación y la solución de los mismos, seguramente se automatizarán problemas, lo que llevará al consultor a encontrarse en el ámbito de la crisis del software, ya discutida en los primeros dos capítulos.

La tarea de este trabajo es mejorar el desempeño de una empresa vía la automatización de la misma utilizando software, pero el desarrollo de cualquier sistema computacional siempre se verá afectado por la cantidad de problemas que en ella se den y por la magnitud de los mismos, el primer paso para resolver un problema es plantearlo, y determinar su naturaleza, sus causas y generar posibles soluciones, en este punto describiré algunas formas de determinar cuáles son los problemas que aquejan a la empresa.

4.4.1 Cuestionario de ajuste

El cuestionario de ajuste es una herramienta para conocer cuáles son los principales problemas dentro de la organización, cuál es el ánimo del empleado con respecto a la empresa, qué problemas aquejan al empleado, qué le molesta y qué le satisface. Cabe recordar que las modernas técnicas de administración enseñan que un empleado que trabaja contento con la empresa, es un empleado productivo, y yo añadiría, gustoso del cambio que siempre implica el uso de un nuevo sistema computacional. Muchos sistemas computacionales han sido desechados debido a que los empleados no han cooperado en la implantación de los mismos.

A continuación presento el modelo del cuestionario de ajuste.

El siguiente cuestionario está basado en un tipo de cuestionario llamado de ajuste, utilizado comúnmente por las áreas de recursos humanos. Este cuestionario idealmente debe ser aplicado a todo el personal de la empresa, no importando el puesto.

FECHA ____/____/_____ (dd/mm/aa)
NOMBRE_____
FECHA DE INGRESO ____/____/____ (dd/mm/aa)

ÁREA O DEPARTAMENTO EN DONDE LABORA

_____

PUESTO_____
NOMBRE DEL JEFE
INMEDIATO_____

- **¿CUÁNTAS PERSONAS ESTÁN A TU CARGO?**_____
  NOMBRE DE TRES DE ELLAS

  _____

- **¿QUÉ EXPECTATIVAS TENÍAS O QUÉ ESPERABAS AL INGRESAR A ESTA EMPRESA?**

  _____

- **¿EN QUÉ PORCENTAJE SE HAN CUBIERTO ESAS EXPECTATIVAS? (EXPLICA)** _____

- **¿QUÉ METAS TIENES CON RESPECTO A LA EMPRESA?**

  _____

- **TE DESCRIBIERON LAS FUNCIONES O ACTIVIDADES DE TU PUESTO AL INGRESAR A LA EMPRESA?**_____

- **¿LO QUE TE DESCRIBIERON, SE RELACIONA CON LO QUE REALIZAS?**

  _____

- **¿RECIBISTE CAPACITACIÓN AL INGRESAR A LA EMPRESA?**

  _____

- **¿QUÉ TAN SATISFECHO ESTÁS CON TU TRABAJO?**

  _____

- **¿EN QUÉ PORCENTAJE DOMINAS TU TRABAJO? (EXPLICA)**

  _____

- **¿TE INTERESARÍA TRABAJAR EN OTRA ÁREA DE LA EMPRESA?**

  _____

- **¿ESTÁS DE ACUERDO CON TU SUELDO Y PRESTACIONES?**_____

  _____

- **¿QUÉ SUGIERES PARA MEJORAR TU TRABAJO?**

  _____

- **¿QUÉ SUGIERES PARA MEJORAR LA EMPRESA?**

  _____

A CONTINUACIÓN ENUMERA LOS TRES PROBLEMAS MÁS FRECUENTES DE TU ÁREA DE TRABAJO (EN CASO DE QUE EXISTAN), PARA CADA PROBLEMA MENCIONA EL NOMBRE DEL PROBLEMA, SU UBICACIÓN, EL TIEMPO EN EL QUE SE HA VENIDO DANDO Y LA GRAVEDAD DEL MISMO, Y MENCIONA SI TIENES ALGUNA SUGERENCIA PARA CORREGIR EL PROBLEMA. (EXPLICA A DETALLE CADA PUNTO)

PROBLEMA 1

- **NOMBRE**_____

- **LUGAR DONDE SE LOCALIZA EL PROBLEMA**

  _____

- **¿DESDE CUÁNDO SE VIENE DANDO?**

  _____

- **¿QUÉ TAN GRAVE ES?**

  _____

- **FORMA DE SOLUCIONARLO**_____

PROBLEMA 2

- **NOMBRE**_____
- **LUGAR DONDE SE LOCALIZA EL PROBLEMA**

  _____
- **¿DESDE CUÁNDO SE VIENE DANDO?**

  _____
- **¿QUÉ TAN GRAVE ES?**

  _____
- **FORMA DE SOLUCIONARLO**_____

PROBLEMA 3

- **NOMBRE**_____
- **LUGAR DONDE SE LOCALIZA EL PROBLEMA**

  _____
- **¿DESDE CUÁNDO SE VIENE DANDO?**

  _____
- **¿QUÉ TAN GRAVE ES?**

  _____
- **FORMA DE SOLUCIONARLO**_____

COMENTARIOS AL CUESTIONARIO_____

_____

_____

Con el cuestionario anterior se pueden evitar automatizaciones de problemas, hay que recordar que los métodos de análisis orientado a objetos, en la mayoría de los casos necesitan el conjunto de requerimientos del cliente para poder iniciar con sus etapas, pero qué puede garantizar que estos requerimientos no estén mal planteados, debido a que existen problemas funcionales dentro de la organización, mismos automatizaremos. Esta automatización es un error grave y generador, en muchos casos, de la crisis del software.

4.4.2 Identificación de problemas por medio de soluciones

Al parecer es algo más natural para el pensamiento humano el determinar la solución antes que el problema mismo, un estudiante en la escuela dice, si administrara mejor mi tiempo seguramente mis calificaciones mejorarían, ahí plantea ya la solución, pero el verdadero problema no lo está planteando, tal vez el verdadero problema es que no conoce métodos de administración de tiempo, o que tenga demasiadas actividades además de la escuela, etc.

Dentro de una empresa se puede aplicar este método para poder analizar las propuestas que dan los stakeholders. A continuación presento un ejemplo real de la empresa comercializadora de refacciones automotrices.

Ejemplos:
"Si me pagaran más, haría mi trabajo mejor"
"Yo creo que si el Ing. XXXX no nos gritara, muchos de nosotros no nos iríamos"

Se toma la lluvia de soluciones y se trata de identificar el problema

Aquí el empleado identificó las soluciones a los problemas que lo afectaban y que por ende afectaban a la empresa. De ahí se pudo determinar que la empresa tenía serios problemas con su planta ejecutiva que requería de capacitación porque no sabía como tratar al empleado, esto quiere decir que en ese departamento hasta en ese momento la generación de un software no solucionaría nada. Además se determinó que los sueldos eran en extremo bajos, pero esto se debía a que las finanzas de la empresa eran mal llevadas, no por que la empresa no fuera rentable, sino porque nunca se sabía a ciencia cierta cuánto dinero había para pagar a los empleados, esto si generó la necesidad de tener un software que pudiera llevar de manera precisa y oportuna la contabilidad y la nómina de la empresa.

He presentado hasta este momento un par de formas para encontrar problemas, quiero recalcar, que la forma de hacer esto puede tomar múltiples variantes, y seguramente en otras disciplinas existen otros métodos, los cuales son bienvenidos en esta guía, la cual indica que debe haber un punto en el cual se deben de tomar en cuenta los problemas de la empresa antes de pasar a realizar diseños y desarrollar software, y más aún si se trata de desarrollos a gran nivel.

4.5 Solucionar problemas antes de la implementación

La implantación de grandes sistemas se software que se hace para motivar un alto desempeño organizacional, en la práctica se ha visto desligada de aspectos organizacionales fundamentales como los administrativos, los de recursos humanos, etc., muchas veces los consultores no se explican el motivo por el cual los sistemas de software no tienen un buen desempeño dentro de las organizaciones si a todas luces se sabe que el sistema es correcto, se ha realizado con las últimas tecnologías, soporta grandes cantidades de información, y técnicamente es un buen trabajo.

Los sistemas de software al ser implantados en un sistema (organización), pasan a ser un elemento del mismo, el cual se verá afectado por el desempeño de otras partes, debido a esto, aunque el sistema sea correcto, otras partes pueden afectar el funcionamiento del sistema en forma positiva o negativa.

En el punto anterior presenté una forma de detectar problemas dentro de las organizaciones, parte fundamental de la solución de los mismos, ya que un problema que no está bien establecido, seguramente no se le podrá dar solución, o las soluciones que se le apliquen no resolverán la esencia del problema. Antes de pasar a discutir una técnica para resolver problemas, en este momento quiero hacer una aclaración que muchos ingenieros en computación podrían esperar, y es la delimitación de los ámbitos del trabajo de un consultor computacional. Tradicionalmente, para un ingeniero en computación, su labor es la de generar sistemas computacionales y nada más, esto es correcto, pero después de cuatro décadas de luchar contra la crisis del software desde distintos ángulos y no haber podido erradicarla, es necesario tener una mente abierta al cambio, y reconocer que la implantación de software es más que una base de datos con programas alrededor. Lo que pasa en las organizaciones termina por afectar los trabajos del ingeniero en computación, es por ello, que a pesar de que tradicionalmente la tarea del consultor computacional no sea la de resolver problemas de la empresa, este debe de saber que si no los resuelve o pide que la organización los resuelva, estos repercutirán en el desempeño de su software.

A continuación presento la metodología KT[60] utilizada para solucionar problemas dentro de una organización, creada por Kepner y Tregoe consultores empresariales con una larga trayectoria en E.U., la cual, en primera instancia identifica plenamente el problema, para después generar posibles soluciones a fin de tomar una decisión, y por último, se ubican los problemas potenciales que habrá de cuidar para evitar desviaciones del mismo tipo. Explicaré la aplicación de la metodología KT utilizando un problema real presentado en la empresa comercializadora de refacciones automotrices

En el área de bodegas de la empresa, se contaba con un sistema obsoleto para el manejo de inventarios, el cual generaba demasiadas fallas, si bien el sistema no contaba con las validaciones adecuadas, con una correcta utilización podría generar salidas de información confiables y podría servir como base para una migración a otra nueva herramienta más poderosa, desarrollada con tecnología de punta. Como el dueño de la empresa desconfiaba que la nueva herramienta que se le proponía fuera a ser una solución a sus problemas, se utilizó la metodología KT, para resolver el problema y demostrar que no era el sistema actual o cualquier otro sistema el origen de la problemática del área.

El problema de esta área era una alta rotación de personal. Siempre se tenían aprendices, lo que genera que se realizaban con lentitud y errores los trabajos. De cada 10 personas que se contrataban, aproximadamente 8 se iban en un transcurso no mayor de un mes.

Este problema no solo afectaba a esa parte de la empresa, sino que generaba retrasos en la empresa en general. Se tenían además elevados costos en cuestión de publicidad para contratación de nuevo personal. Los empleados que no abandonaban la empresa, trabajaban con desgano, no utilizaban de forma correcta el sistema computacional, además de que siempre aprovechaban la oportunidad de sustraer mercancía de la empresa.

La información presentada a continuación en la tabla 4.1, se obtuvo por medio de entrevistas con los empleados, y la aplicación de cuestionarios que expliqué en puntos anteriores. Se plantea la identificación del problema en lugar, tiempo y extensión, se enuncia la desviación y se hace una comparación lógica, se buscan distingos entre la desviación y su comparación lógica más cercana, y de ahí se deduce el problema en sí. Dicha metodología lo que hace es determinar de manera objetiva características de los problemas y comparaciones, le quita la parte de subjetividad que el ser humano siempre le agrega a los problemas.

[60] Kepner, Tregoe (1989)

Enunciado de la desviación: *Maltrato hacia los empleados.*

	Problemas de especificación	Desviación del desempeño	Comparación lógica más cercana.	¿Qué es lo que distingue a ...	¿Sugiere algo el distingo?
Identificación	¿En que consiste el problema?	Los empleados trabajan descontentos, y en la primera oportunidad dejan la empresa.	PUDIERA SER que es peligroso su trabajo, pero NO ES.	Los empleados mencionan que la empresa además del tener mucho trabajo, son maltratados psicológicamente.	Nada
Lugar	¿En donde se da el problema?	Es observado en las áreas de bodegas y mostradores.	PUDIERA SER, pero NO ES en el área de compras.	Las dos áreas son controladas por un mismo gerente.	El gerente de estas dos áreas podría ser el causante de esta situación.
Tiempo	¿Desde cuando se da este problema?	Siempre se ha dado. (es importante recalcar que la mayoría de empleados no tiene más de año y medio)	PUDIERA SER que se empezara a dar desde alguna fecha, pero NO ES.	El gerente de las dos áreas, lleva trabajando más de cuatro años.	El gerente de estas dos áreas lleva mucho más tiempo que la mayoría de los empleados.
Extensión	¿Cuantas personas son afectadas por este problema? ¿Cuántas áreas de la empresa son afectadas?	De las dos áreas, aproximadamente de cada 10 personas, 8 refieren un gran resentimiento hacia la empresa. Las dos referidas en la identificación de tiempo.	PUDIERA SER que solo unas cuantas personas sean las afectadas, pero NO ES. PUDIERA SER que sean más que las dos áreas mencionadas, pero NOES.	El gerente de las dos áreas, no tiene injerencia en otras áreas.	El gerente de las dos áreas, no tiene relación con otras áreas.

Tabla 4.1 Utilización de la metodología KT

Con base en esta tabla, es inmediata la deducción de que es el gerente de estas áreas el responsable de la gran rotación de personal que se da en estas áreas, debido al maltrato que le da a los empleados.

Para verificar la hipótesis, se procedió a entrevistar de manera aislada a varios de los empleados, con lo cual se comprobó que la hipótesis es verdadera, ya que generándoles un clima de confianza, motivó a que dijeran quién es la persona que los maltrata y referenciaran los maltratos, con lo cual mencionaron que lo que "mas les molestaba es que les gritara frente a sus compañeros, y siempre los regañe".

Debido a que es necesario tomar una decisión con respecto a este problema, se procedió a identificar las posibles alternativas con respecto al problema. Se aplicó otra metodología de los mismos autores (Kepner-Tregoe), a fin de poder tomar la mejor decisión, cuyos puntos centrales son los siguientes.

Se fija un objetivo obligatorio y otros objetivos deseados que deben de cubrir las alternativas de solución propuestas.

El objetivo obligatorio es:

Disminuir la rotación actual de personal de las áreas en cuestión de 10 a 8 (de diez que se contrataban, ocho se iban en menos de un mes 10-8) a 10 - 2.

Por objetivos deseados se tienen:

Aumento de la productividad.
Utilización correcta de los sistemas computacionales
Generación de motivación de los empleados hacia el trabajo.

Como alternativas se tienen:
Capacitar al gerente en cuestiones de trato al personal
Contratar un nuevo gerente.

En la tabla 4.2 muestro en una tabla el análisis de las alternativas con respecto al objetivo principal.

Objetivo obligatorio	Capacitación del actual gerente	Contratación del nuevo gerente
Reducción de la rotación de personal	El gerente tiene actitudes muy negativas para con la empresa por lo tanto NO PASA	PASA

Tabla 4.2 Selección de alternativas para un objetivo

Para el problema del maltrato, podemos determinar rápidamente, que se debe de contratar un nuevo gerente, ya que la actitud del gerente actual, es totalmente negativa hacia los empleados.

Una vez que se llegó a esta conclusión, el dueño tuvo confianza en la nueva herramienta, que se le proponía, debido a que la rotación de personal bajó, lo que dio como resultado que en cuestión de dos semanas la mayoría de los empleados usaba el sistema actual de forma correcta, el cual podría en ese momento a ser utilizado como base para una migración a otro sistema nuevo. Además como valor agregado, el equipo de desarrollo del nuevo sistema ganó la confianza del personal de bodegas y punto de venta, lo que ayudó en el levantamiento de los requerimientos que se estaba haciendo, para el desarrollo del nuevo sistema computacional.

Dentro de la metodología no solo se propone la solución de los problemas sino la identificación de cualquier problema potencial, esto es una mentalidad proactiva en lugar de reactiva.

Un problema potencial que se podría presentar es que el nuevo gerente vuelva a realizar las mismas prácticas insanas con respecto al personal.

Por lo tanto, hay que identificar las causas de los problemas. A continuación enumero algunas acciones que debe de llevar a cabo para evitar que las causas del problema se vuelvan a presentar.

- El nuevo gerente debe de evitar alzar la voz a los empleados y realizar castigos económicos que afecten los sueldos de los empleados.

- El nuevo gerente debe de conocer totalmente cuáles son sus funciones y las de sus empleados.

- El nuevo gerente no debe de crear círculos de poder con algunos de los empleados.

Para esto las acciones preventivas que se deben de llevar a cabo son las siguientes:

- Pedir reportes mensuales de los descuentos en nómina de los empleados de las dos áreas en cuestión para verificar que el sueldo de los empleados no sea afectado.

- Entrevistar a los empleados de manera personal, para poder crear un clima de confianza, que haga que externen los problemas que los aquejan.

- En caso de renuncias de empleados, pedir al área de personal, que les realice una entrevista de salida, con el fin de determinar los motivos de su renuncia.

- Pedir reportes mensuales al nuevo gerente acerca de los avances con respecto a los objetivos fijados por la empresa.

Se debe de tomar en cuenta que el problema expuesto en este documento es uno de los más difíciles de ubicar y de solucionar, ya que implica manejo de personal, por lo que también es necesario tomar en cuenta que cualquier decisión tomada traerá consecuencias. La consecuencia más clara es el dejar a la empresa sin la experiencia del gerente que será despedido, por lo cual se recomendó que una buena táctica a realizar, es hablar con el gerente, pedir que modifique su actitud, al mismo tiempo contratar un nuevo subgerente, que en determinado momento asuma las funciones de gerente, una vez que conozca el funcionamiento de la empresa.

Con esta metodología se pueden identificar, corregir y prevenir problemas de la empresa y tener una visión clara de las acciones que se deben de tomar en cuenta cuando se está dirigiendo una empresa o se está actuando como consultor.

4.6 Mapas conceptuales de la empresa

Una vez que se tiene una panorámica de la empresa en general, es tiempo de pasar a niveles de detalle más bajos, con el fin de obtener productos que después se puedan transformar en clases, objetos, relaciones etc. elementos de la programación orientada a objetos.

Retomando lo explicado hasta este momento, se ha ubicado la empresa en su nivel más alto, se han nombrado los roles del equipo de trabajo, y se han identificado y corregido problemas que pudieran afectar el desarrollo de software, como paso siguiente hay que comenzar a realizar mapas conceptuales de la empresa, la explicación de estos la realicé en el capítulo dos, por lo cual sólo retomaré algunos elementos básicos de esa explicación.

Después de ubicada la empresa con el mapa conceptual de tip de caja negra en donde se pueden apreciar el ambiente y los factores externos que se relacionan con el sistema que se está trabajando, se procede a construir el mapa conceptual de la organización en su conjunto. En la figura 4.7 se presenta el mapa conceptual de la empresa en general.

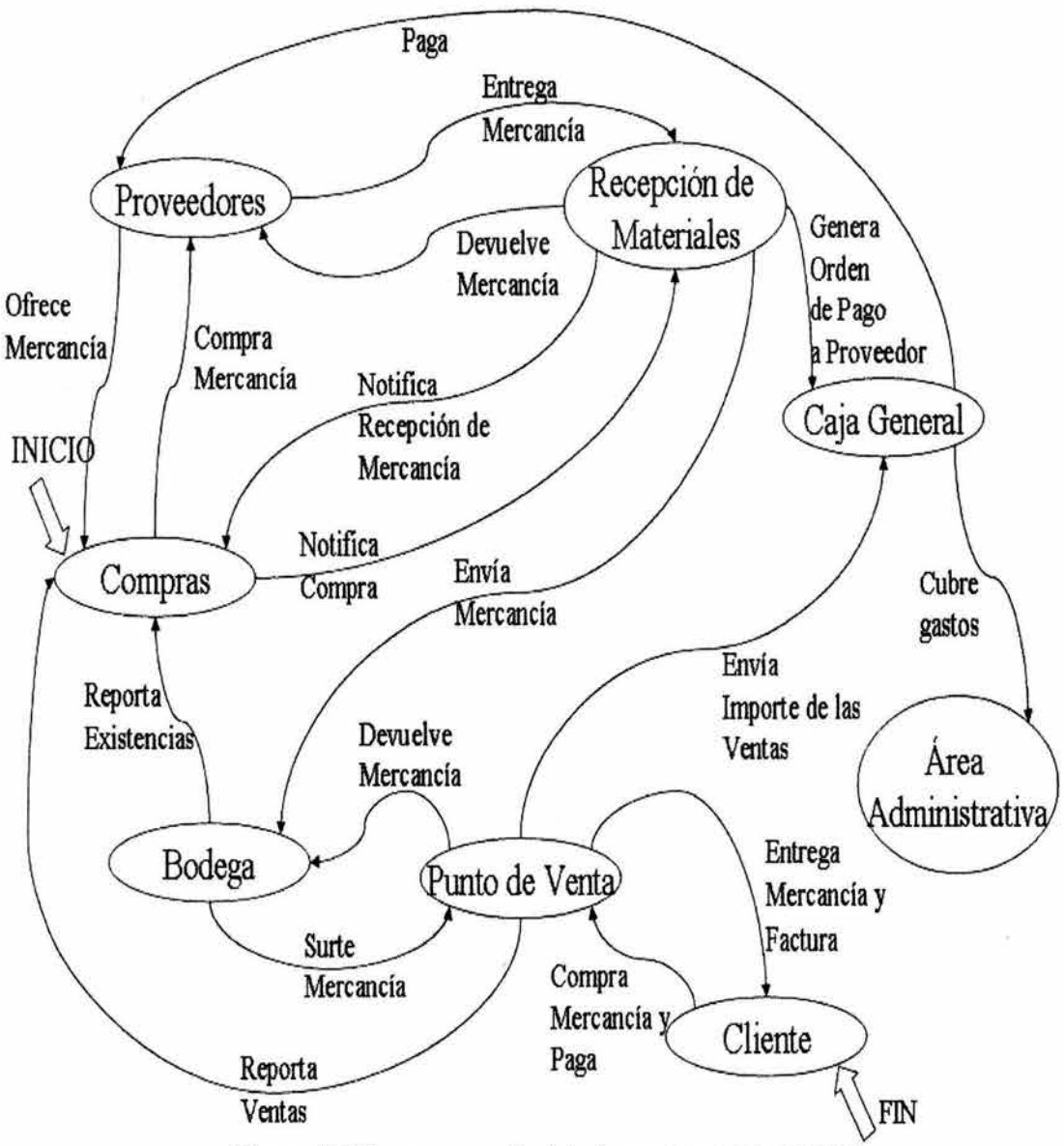


Figura 4.7 Mapa conceptual de la empresa en general.

Una vez que se tiene este tipo de mapas, se descompone cada uno de los elementos principales de este primer mapa en sus respectivos componentes, para analizar los elementos de los subsistemas que componen el sistema principal, De la figura 4.7 a la figura 4.12 presento algunos de los mapas conceptuales de los subsistemas que son áreas de la empresa comercializadora de refacciones automotrices que se ha estado analizando en este trabajo.

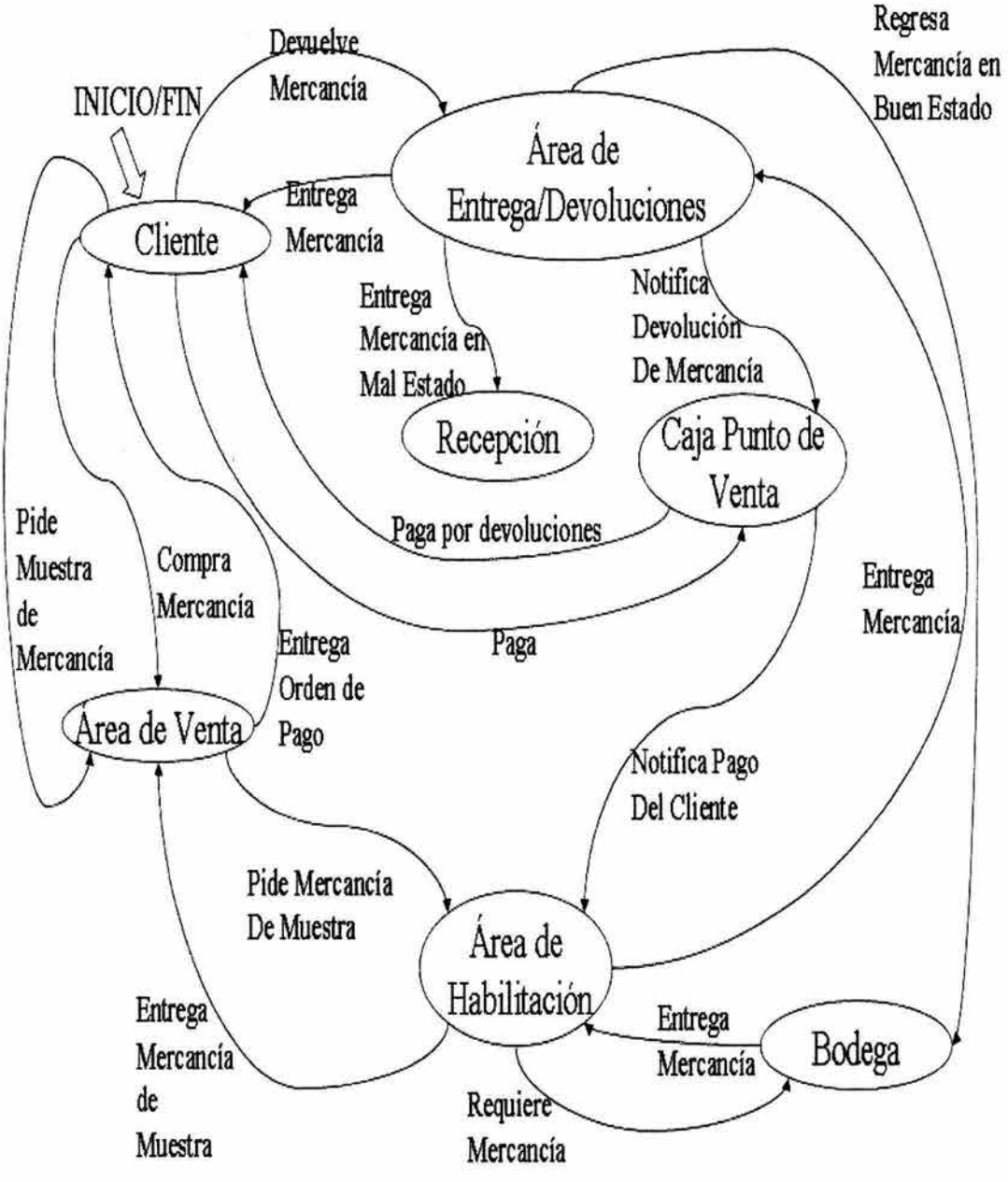


Figura 4.8 Mapa conceptual del área de punto de venta.

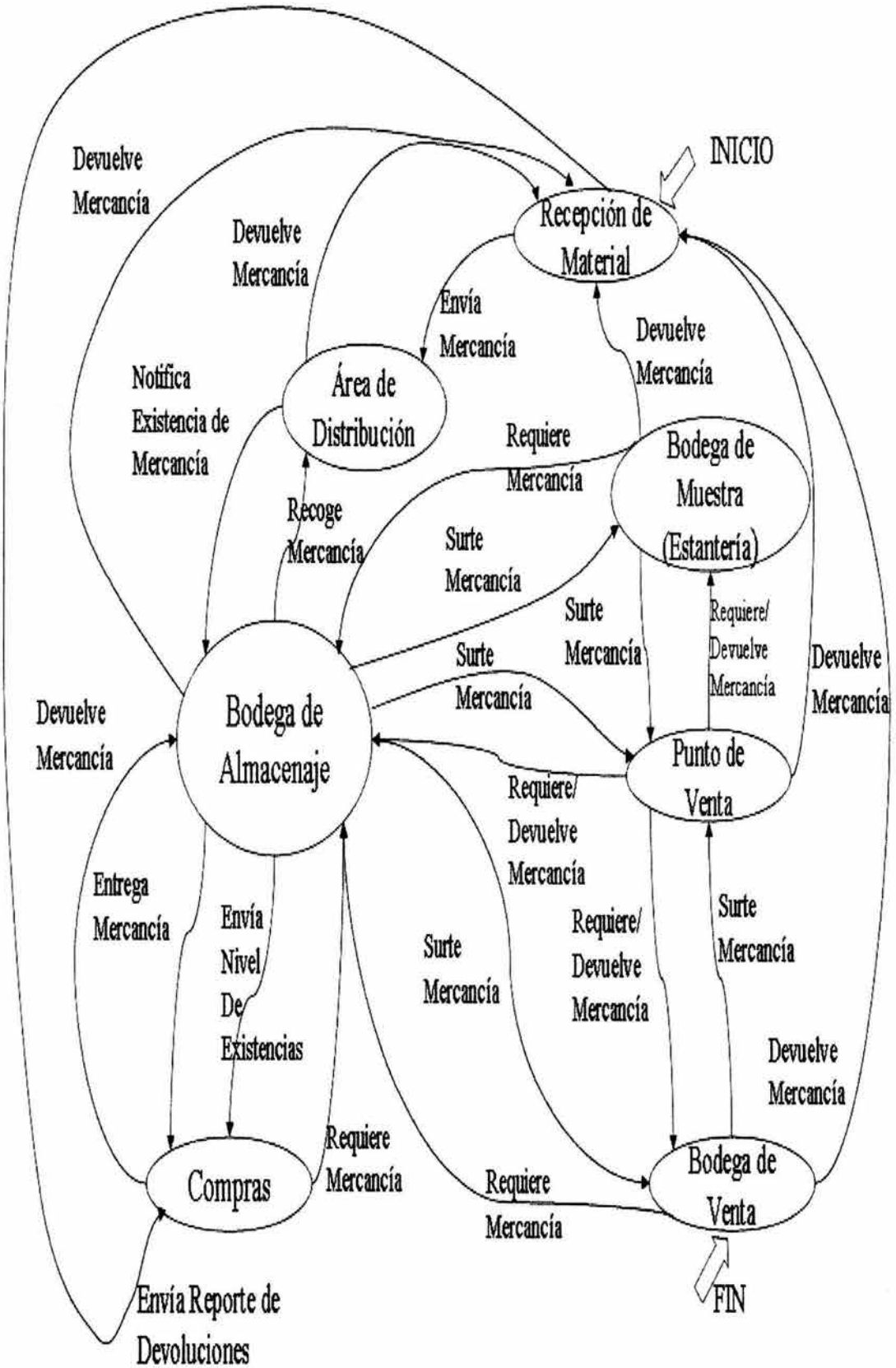


Figura 4.9 Mapa conceptual de área de bodegas

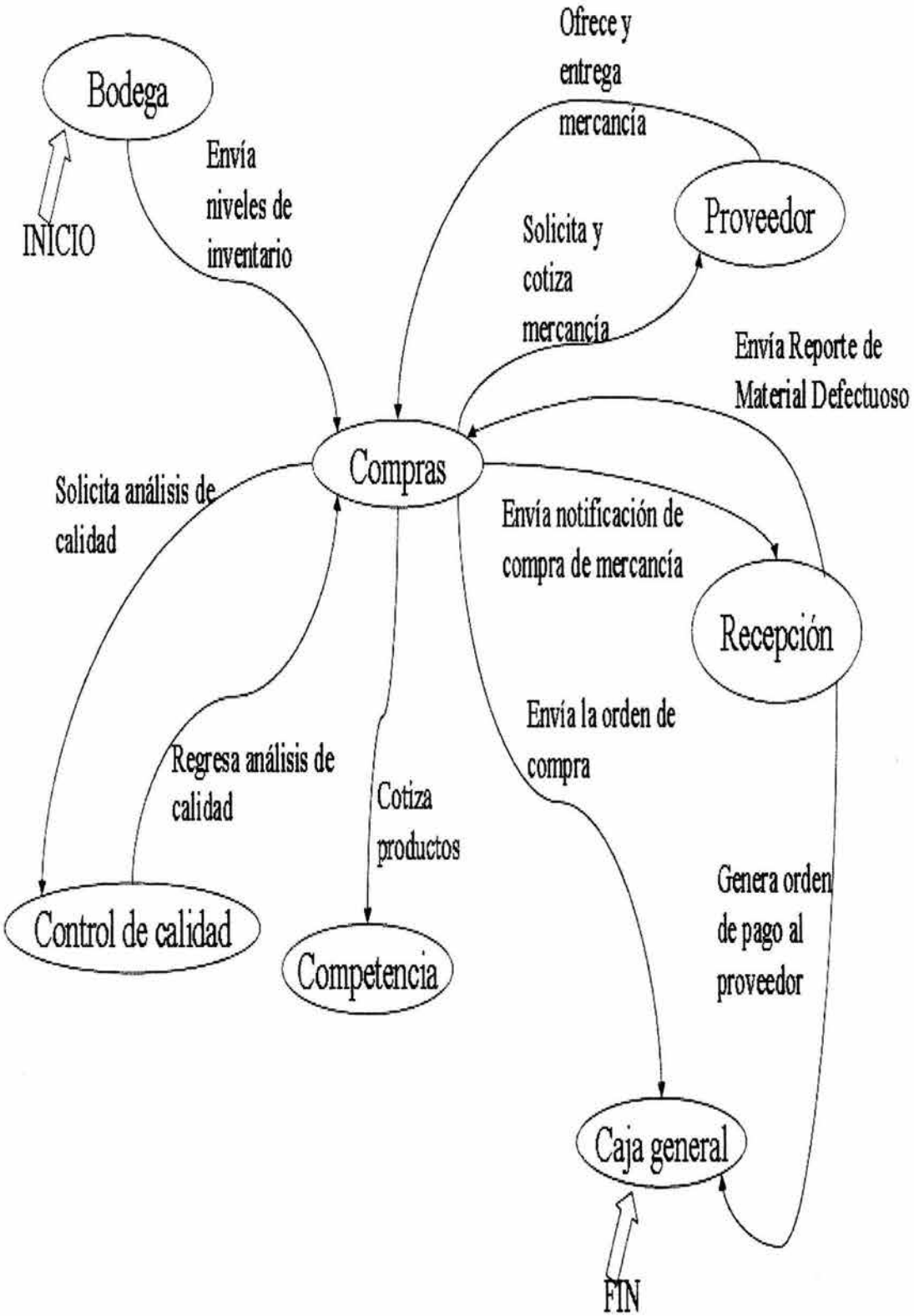


Figura 4.10 Mapa conceptual del área de compras

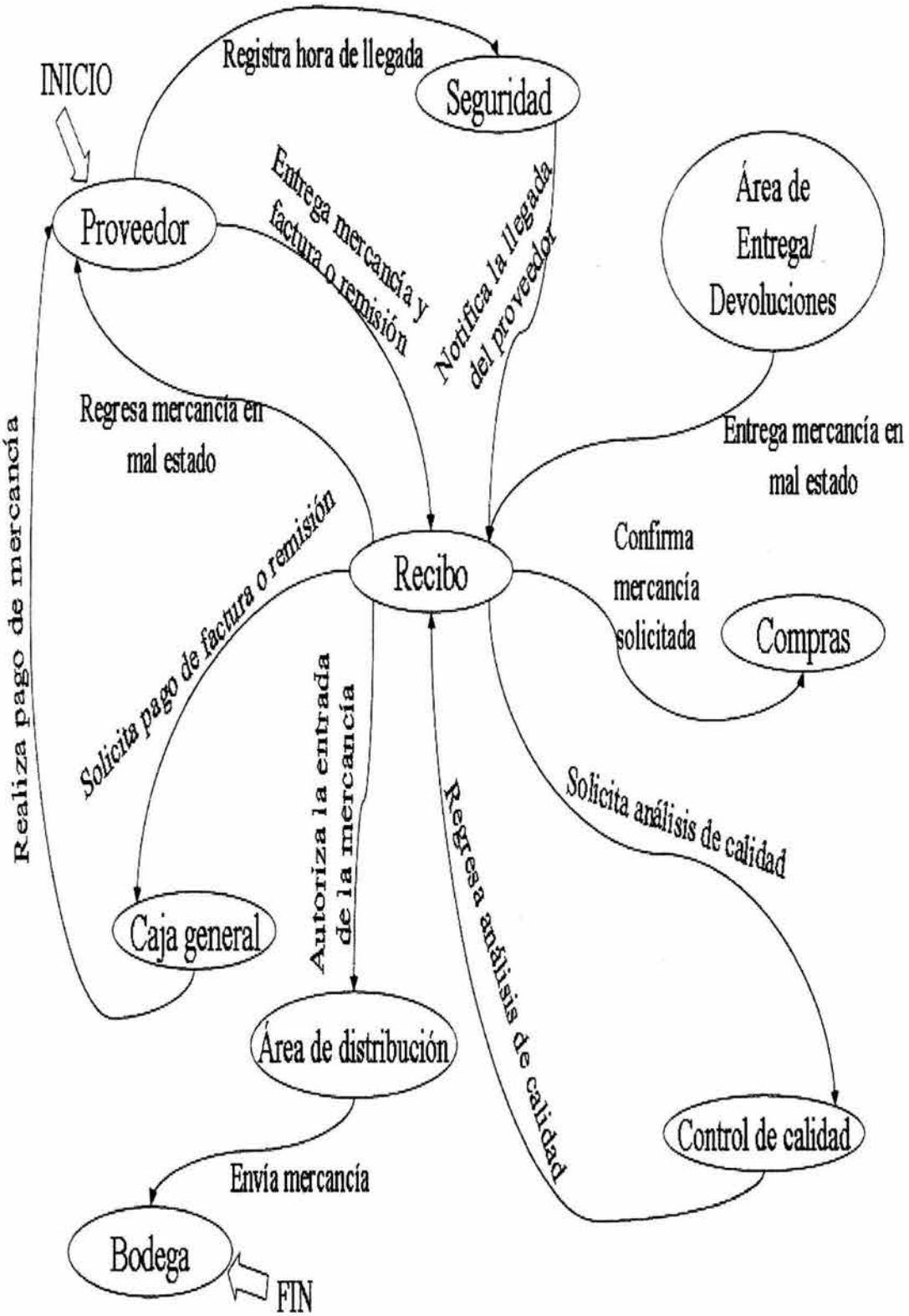


Figura 4.11 Mapa conceptual del área de recibo de mercancía

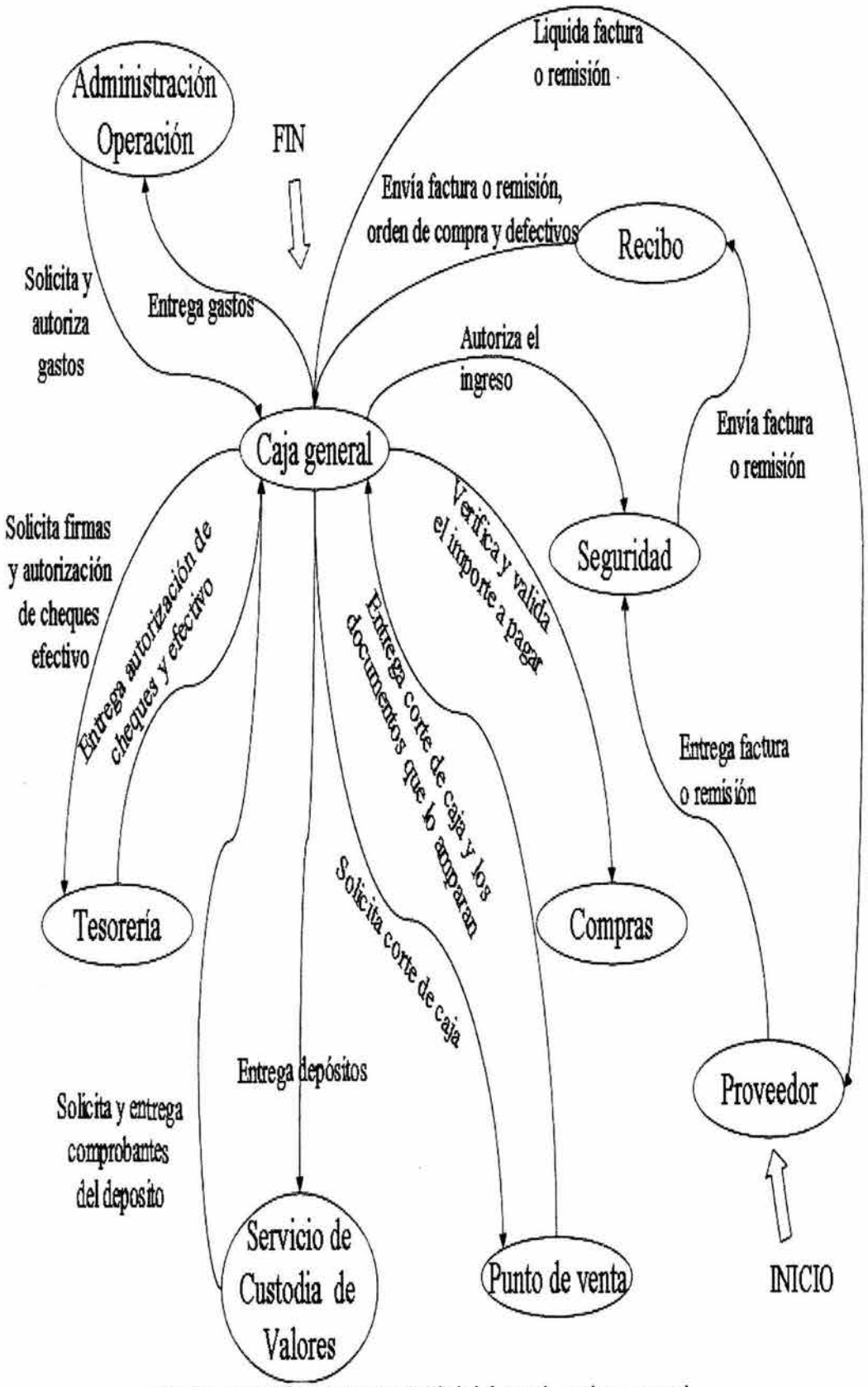


Figura 4.12 Mapa conceptual del área de caja general

Este trabajo de descomposición del sistema en sistemas, debe de ser validado por los stakeholders de la empresa, porque hay que recordar que las personas que verdaderamente conocen la empresa son ellos.

Se pueden aumentar niveles de desagregación hasta el punto que se tengan elementos simples, los cuales puedan ser representados como clases, objetos y relaciones dentro de la programación orientada a objetos.

4.7 Análisis de actores del sistema

Como expliqué en el capítulo dos, los primeros mapas de actores del sistema o stakeholders se hacen con la conversión de los mapas conceptuales obtenidos en el punto anterior y modificados de tal forma que, en lugar de las áreas, se presenten las personas que trabajan en esas áreas con sus nombres respectivos. En la figura 4.13 presento este tipo de mapas de stakeholders.

Se puede continuar realizando este tipo de mapas a diferentes niveles de desagregación, con el fin de llegar a estructuras que el equipo de programación identifique como viables para comenzar los trabajos de escritura de código.

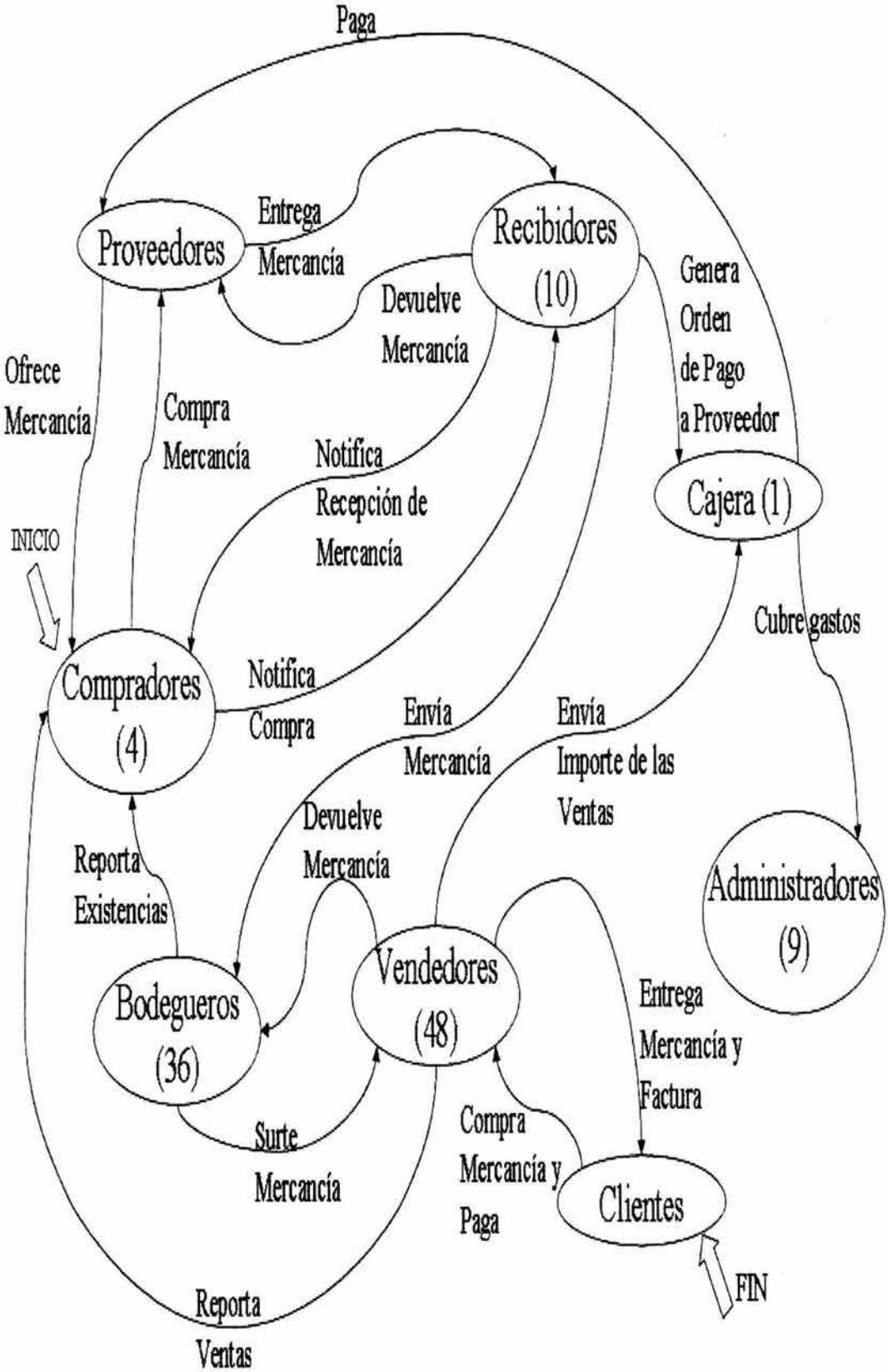


Figura 4.13 Mapa conceptual del stakeholders de la empresa en general

Dentro del análisis de stakeholder que presenté en el capítulo dos mencioné que se puede realizar otro tipo de análisis llamado red de poder, con el fin de identificar el tipo de relación que se da entre los actores del sistema, recordando que debido a que el software que se implante en el sistema, pasa a ser un elemento del mismo, es necesario saber cómo se dan las relaciones entre las personas que se encuentran inmersos en el sistema, esto con el fin de reforzar algunos conceptos como el de padrinazgo que ya se explicó en este trabajo, además de poder detectar puntos clave en donde relaciones adecuadas entre los usuarios motiven el buen uso del sistema computacional, así como su desarrollo y también por el contrario, detectar malas relaciones que puedan afectar la implantación del software que se desarrolle.

En la figura 4,14 presento la red de poder para la empresa que he estado utilizando en este trabajo.

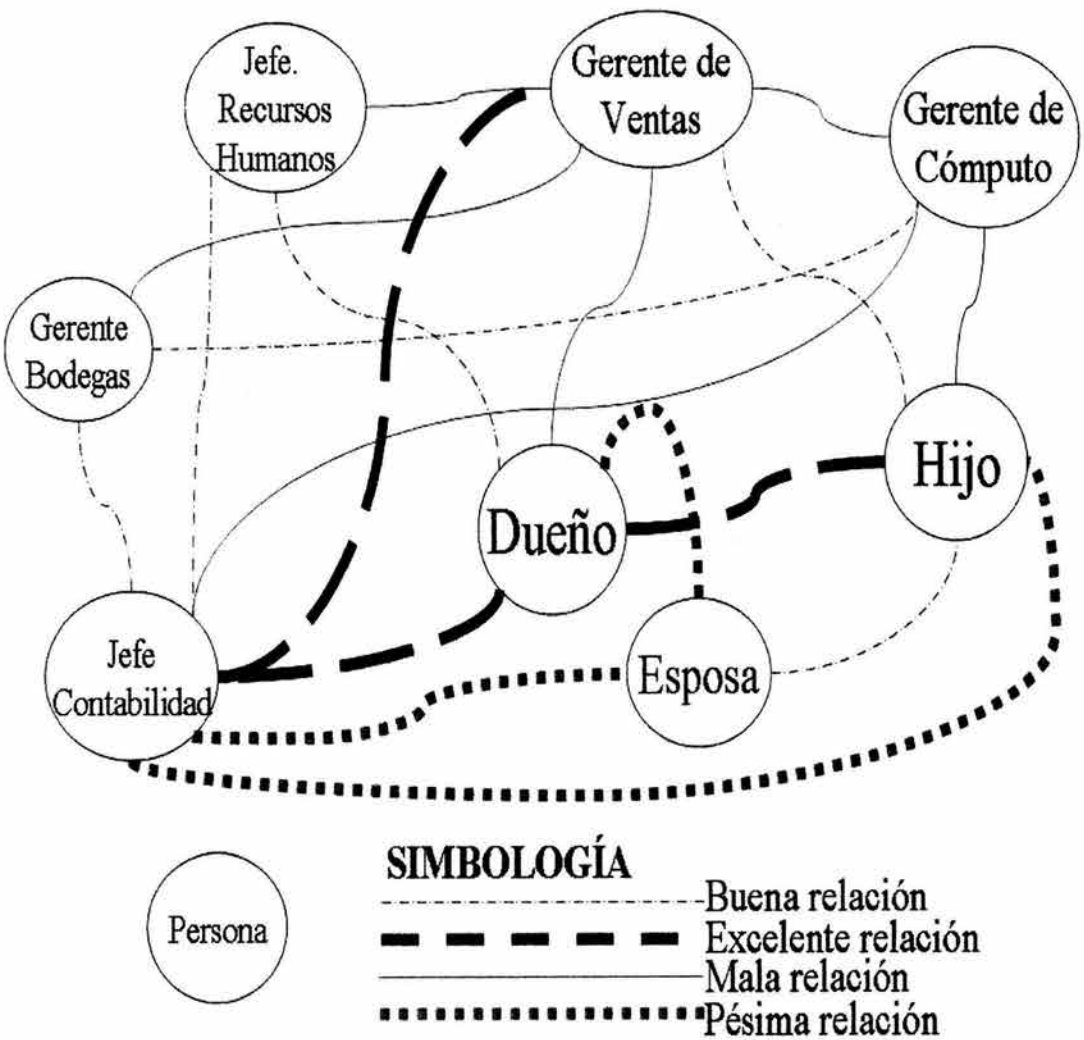


Figura 4.14 Red de poder

4.8 Esquemas de la guía de análisis

Como he mencionado a lo largo de este trabajo, la guía de análisis que presento debe de ayudar al consultor computacional a realizar de manera rápida sus actividades, por lo cual, el presentarla de manera esquemática, permite al grupo de desarrollo de software después de haber leído y conocido sus elementos a detalle, tener una referencia rápida de la guía.

De la figura 4.15 a la 4.22 presento de manera esquemática la guía.

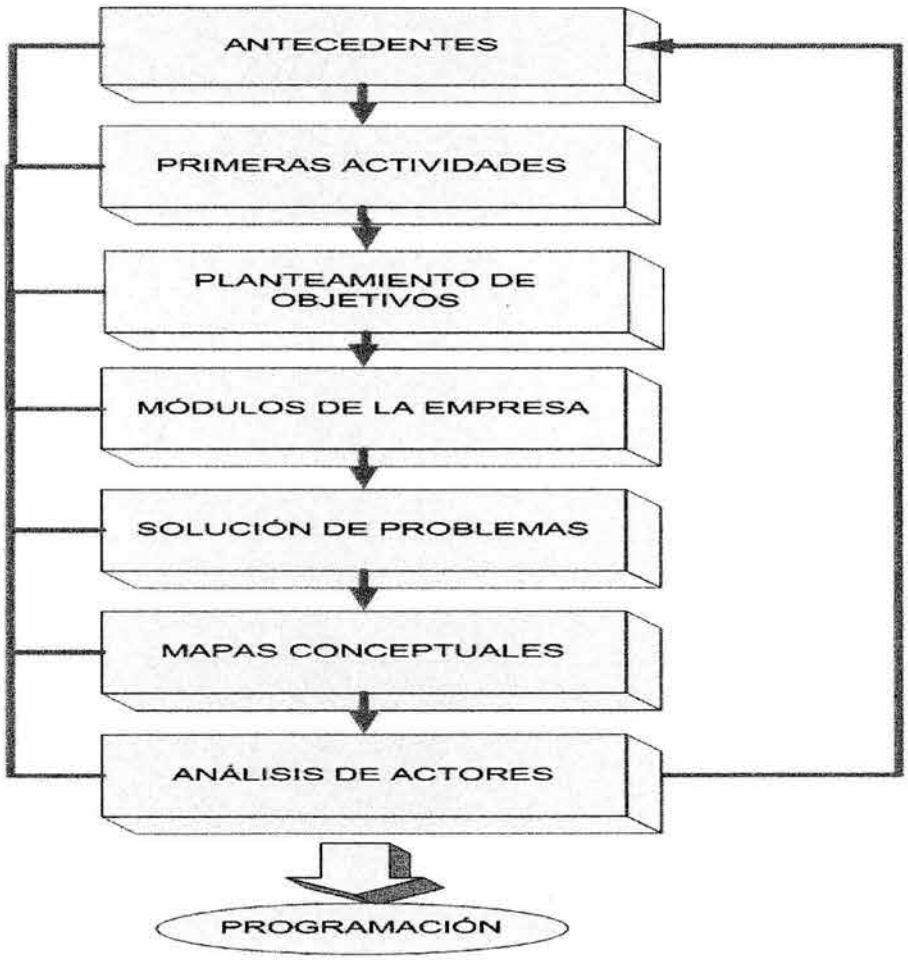


Figura 4.15 Guía de análisis organizacional

En la figura 4.15 se observa el flujo de la guía, por medio de flechas dirigidas, pero existen también líneas sin dirección que indican que desde cualquier bloque podemos pasar a cualquier otro dependiendo de nuestras necesidades.

A continuación presento por separado cada uno de los bloques de la guía, explicando de manera gráfica sus elementos principales.

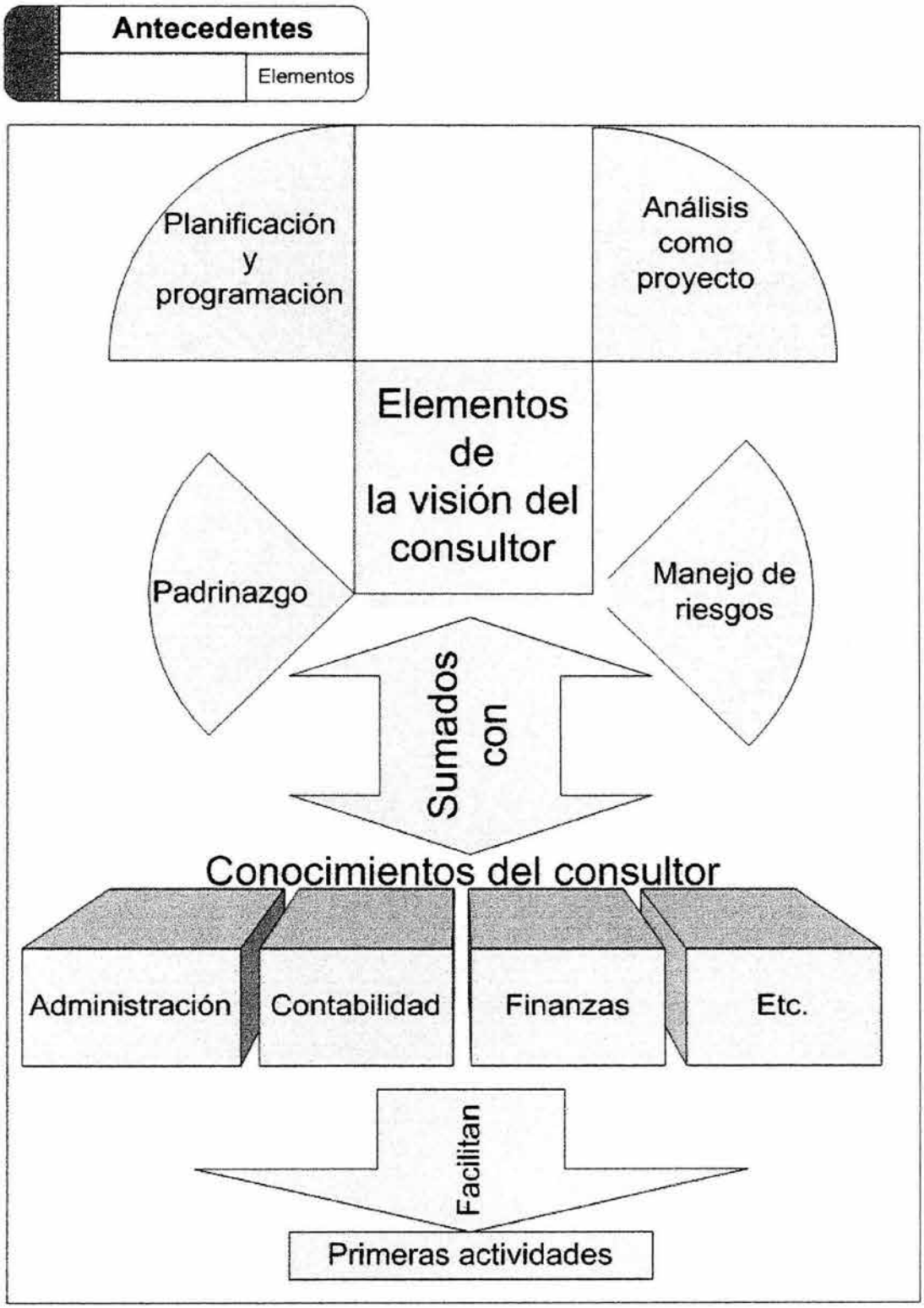
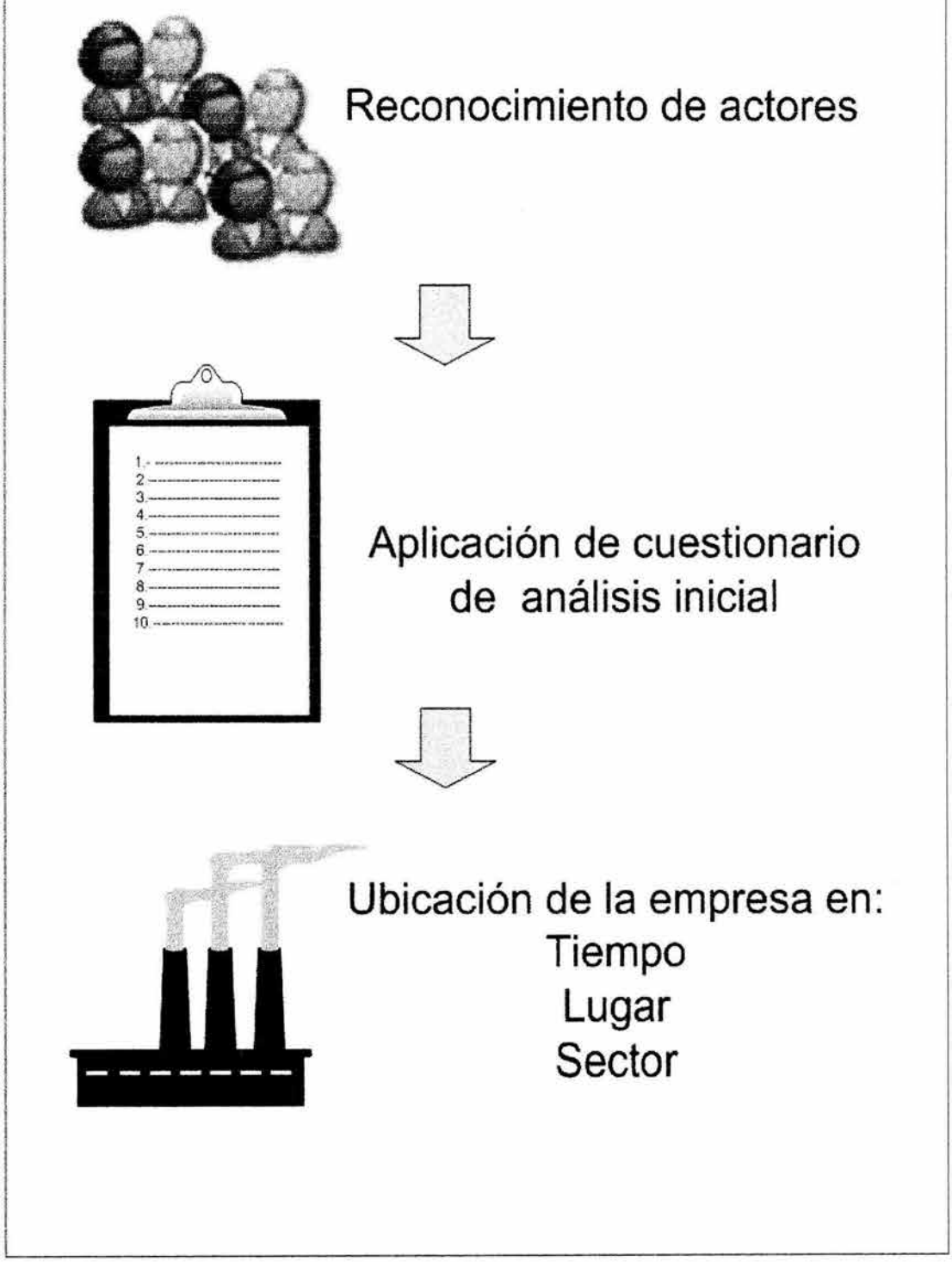


Figura 4.16 Visión y conocimientos del consultor

Primeras Actividades

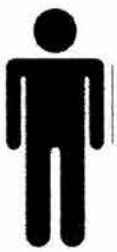
	Elementos



4.17 Actividades iniciales dentro de la empresa

Planteamiento de objetivos

	Elementos



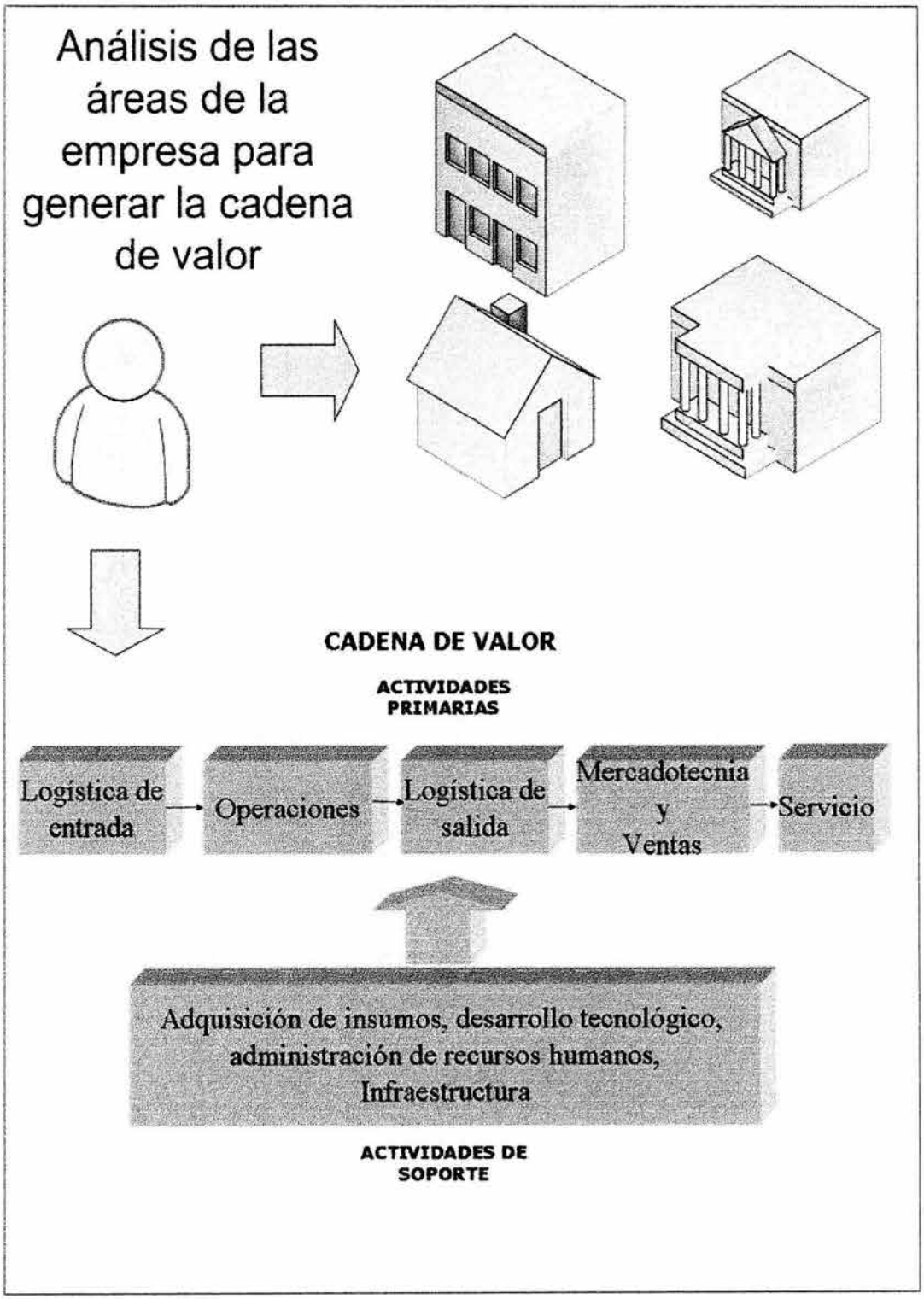
Hacia a dónde vamos

Descripción	**Indicadores**	**Medios de verificación**	**Suposiciones importantes**
Meta Sectorial /empresarial			(Objetivo a superobjetivo)
Objetivo del proyecto			(Objetivo a meta sectorial
Salidas:			Salidas a objetivos
Actividades:			(Actividades a salidas)

Uso del marco lógico

4.18 Planteamiento de objetivos vía el marco lógico

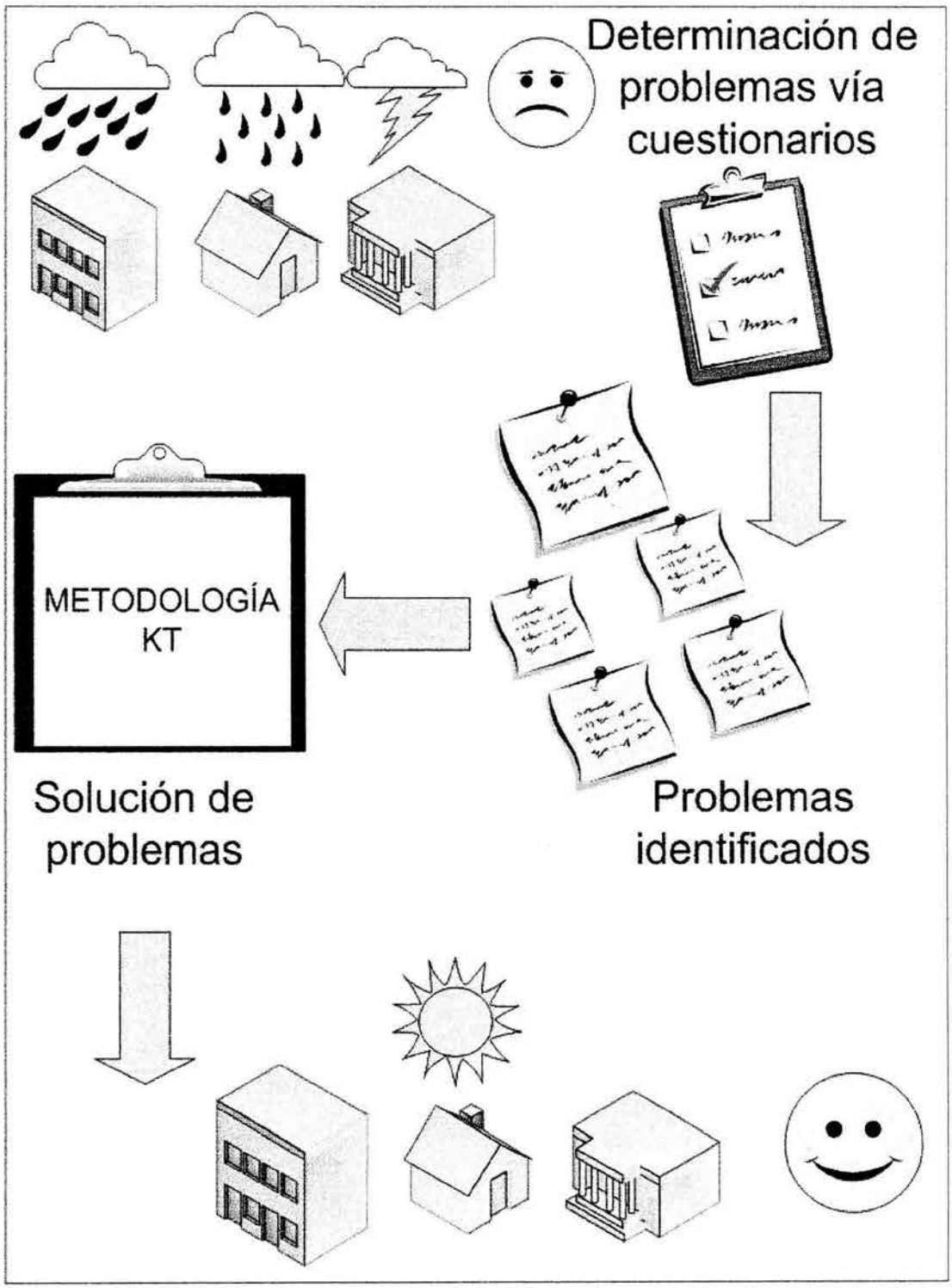
Módulos de la empresa	
	Elementos



4.19 Utilización de la cadena de valor para determinar los módulos de la empresa

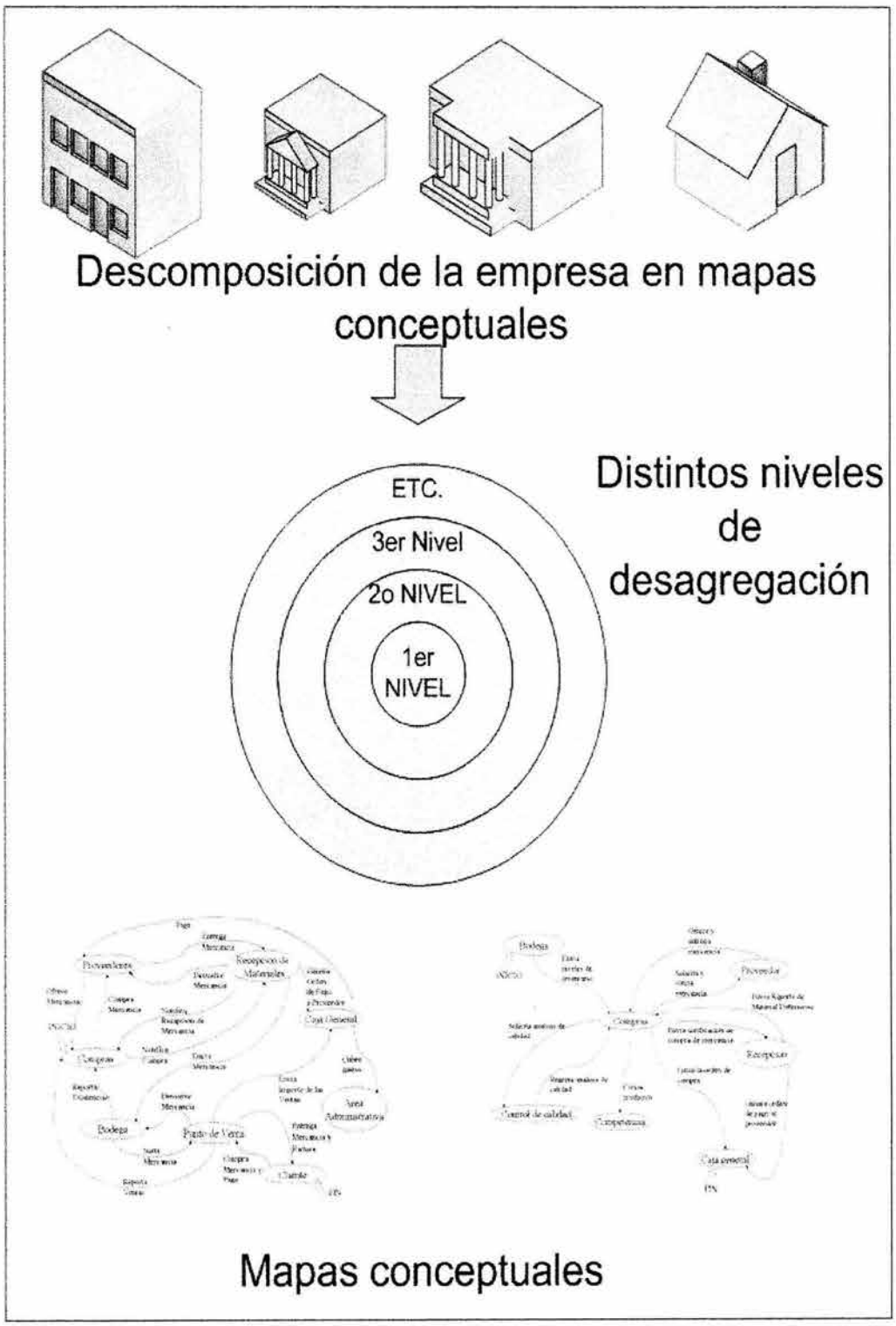
Determinación de problemas en la empresa

Elementos



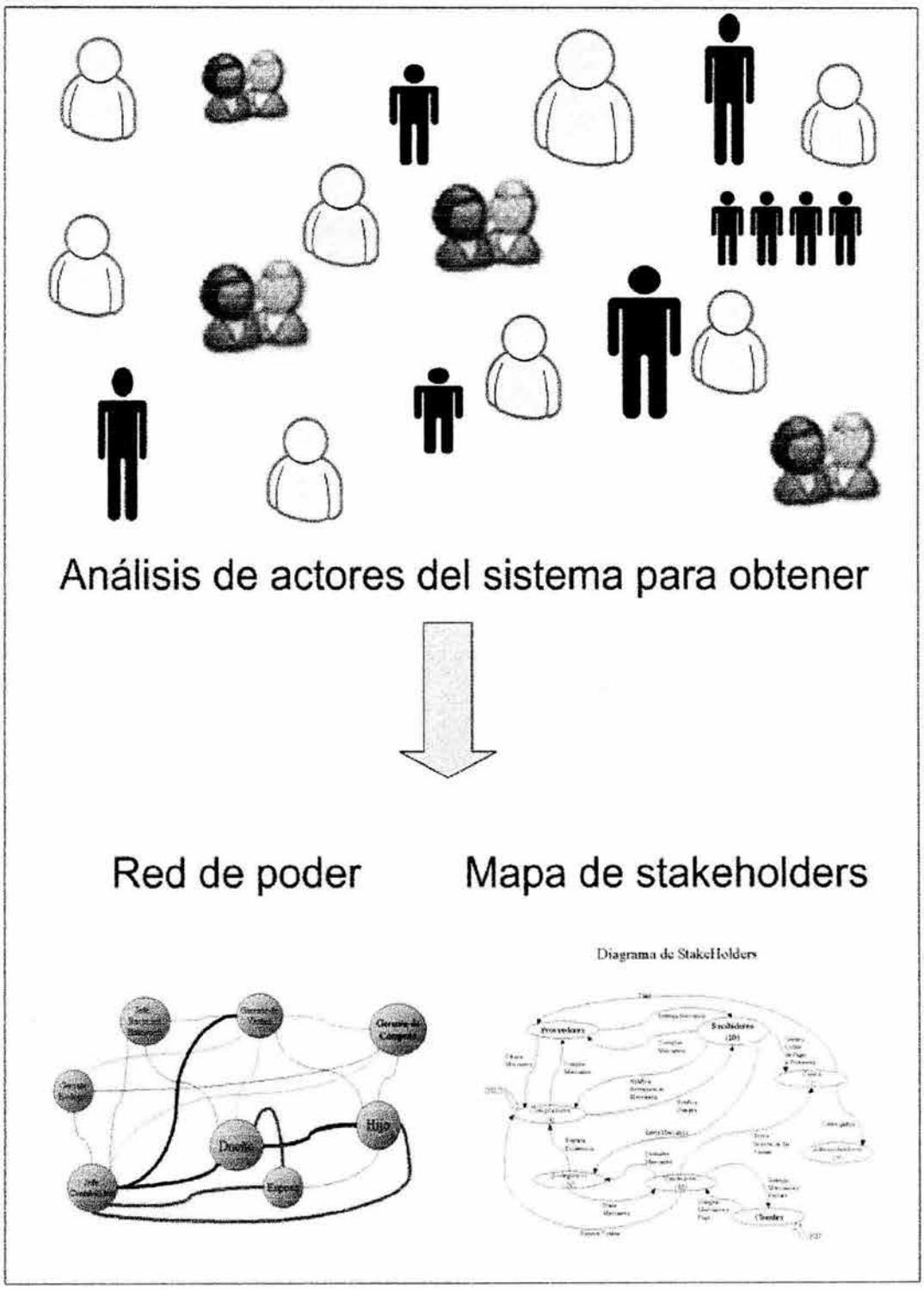
4.20 Identificación y solución de problemas

Mapas conceptuales	
	Elementos



4.21 Generación de mapas conceptuales a distintos niveles de desagregación

Análisis de actores del sistema	
	Elementos



4.22 Análisis de actores del sistema

4.9 Combinación de la Guía de análisis con la metodología XP

En el campo de trabajo por lo regular se utilizan combinaciones de distintas metodologías, el diseñar una guía que no pueda interactuar con otras herramientas utilizadas por los profesionistas del software, limitaría su campo de acción, la guía que en este trabajo he presentado tiene una orientación hacia la sencillez y la adaptabilidad, cosa que muestro al combinar esta guía con los elementos de la metodología rápida llamada XP, que expliqué en el capítulo tres, la combinación de ambas herramientas la muestro en la figura 4.23.

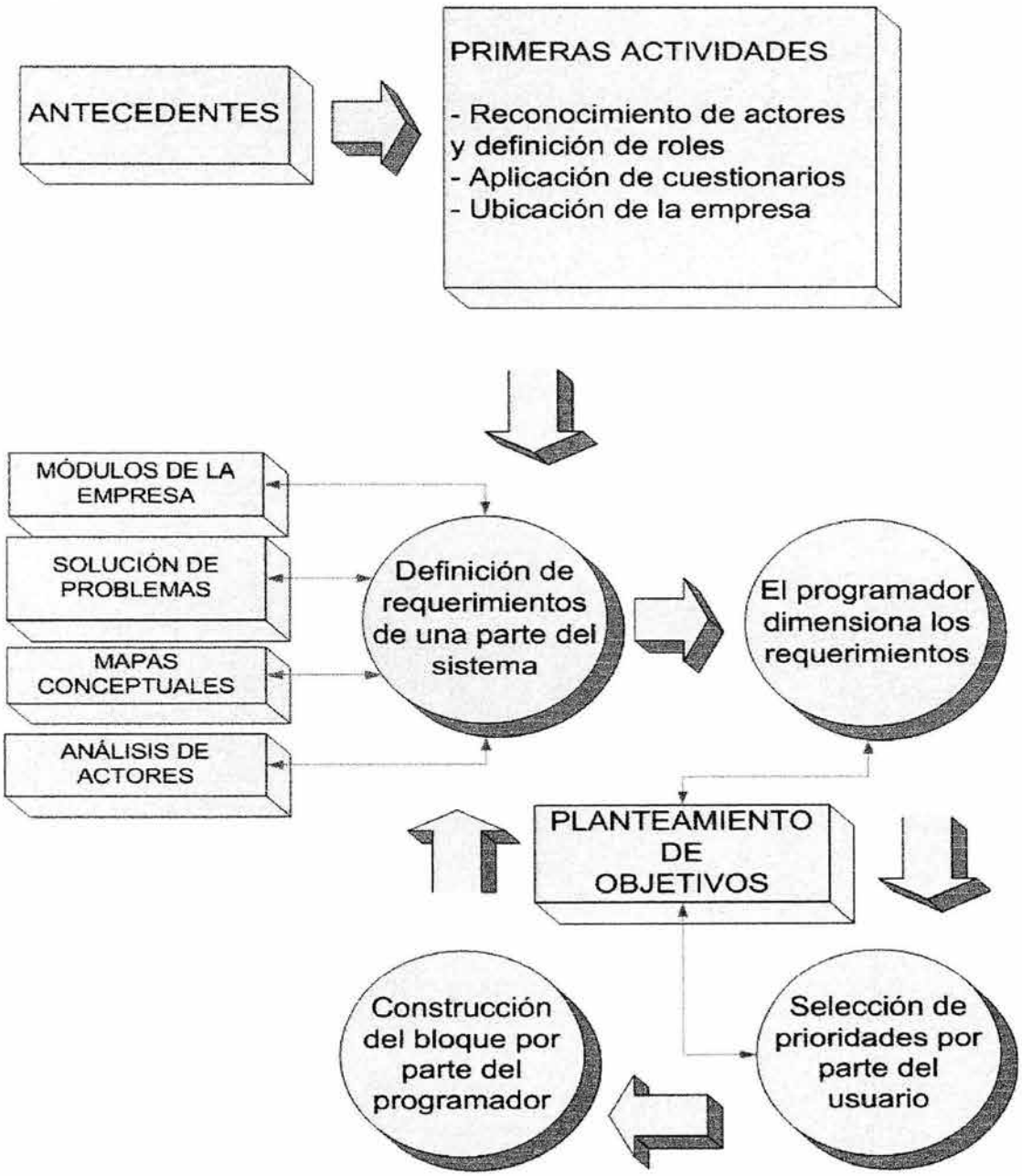


Figura 4.23 Combinación de la guía de análisis con la metodología XP.

La combinación de las dos herramientas, muestra que la estructura de la guía, puede ser fusionada, a fin de mejorar otras herramientas y/o mejorarse ella misma, es este el sentido la guía se presenta como una herramienta flexible, que puede combinarse con otros productos existentes en el mercado de trabajo.

4.10 Inicio del diseño y la programación orientada a objetos

Una vez que se tiene analizado el sistema, que se conocen los módulos principales que pueden ser diseñados, que de cada módulo se conocen los elementos a detalle, tanto funcionales como humanos, y que se han detectado y solucionado problemas que afectarían el desarrollo de un sistema computacional, se puede comenzar con las fases de diseño y construcción del software, porque la guía presentada, ha dado como resultado un análisis del sistema, el cual es la base para poder implementar mejoras o simplemente automatizarlo, vía la implantación de software.

El siguiente paso es elegir la metodología de diseño y construcción adecuada, recuérdese que en el capítulo tres se estudian algunos ejemplos. Tomando en cuenta, que estas metodologías pueden ser utilizadas en forma conjunta con UML la herramienta que al parecer podría pasar a ser un estándar, hay que considerar esta asociación, para complementar los elementos de notación de las metodologías vía un lenguaje común de notación como UML.

5 CONCLUSIONES

El desarrollo de software de tamaño industrial es un elemento clave para el desarrollo de las empresas mexicanas las cuales requieren de sistemas computacionales cada vez más complejos, que atiendan sus necesidades, en una era en donde la Internet y en general la computación, se está convirtiendo no sólo en una herramienta para el trabajo, sino en un modo de vida.

La demanda de software de calidad requiere de profesionistas en computación comprometidos con el desarrollo del país, que puedan lidiar con la crisis del software, problema inherente al desarrollo de sistemas computacionales, descubierto ya hace más de tres décadas.

Si bien la complejidad del software es un elemento que no puede eliminarse por completo, y que genera que la crisis del software se presente, esta complejidad puede ser atenuada por medio del uso correcto de metodologías, guías, herramientas, lenguajes de programación etc., elementos de la ingeniería del software que han evolucionado a lo largo de varias décadas y han dado como resultado nuevas formas de trabajo, entre las cuales destaca la orientación a objetos, la cual lleva vigente en el mercado laboral al menos dos décadas en países como Estados Unidos, y por lo menos una década en México, y ha contribuido a atenuar la crisis del software, pero todavía requiere de atención para mejorar algunos de sus procesos, principalmente los de análisis de sistemas.

Diversos autores de metodologías de análisis y diseño orientadas a objetos proponían desde hace décadas que la solución a los problemas de sus metodologías no se encontraría en los campos de la computación, sino en otras disciplinas del conocimiento. La utilización del enfoque sistémico fue correcto para desarrollar la guía que en este trabajo presenté, debido a que los diversos elementos que la conforman permiten analizar desde distintas perspectivas y de forma detallada cualquier sistema, y dan como resultado información fundamental del sistema, necesaria para realizar diseños orientados a objetos.

Crear un puente entre la ingeniería de sistemas y la ingeniería de software, fue una premisa básica y acertada, ya que el estudio de la ingeniería de software con sus metodologías y herramientas como UML, indicó que ya se tienen los elementos esenciales para la construcción de software de tamaño industrial, pero el diseño que se basa en análisis deficientes, no genera resultados alentadores a la hora de implantar el software, por tanto era necesario crear una herramienta que guiara los esfuerzos de los ingenieros en

computación al momento de analizar organizaciones, hacia la entrega de elementos que permitieran a las metodologías y herramientas orientadas a objetos crear diseños correctos de sistemas computaciones que satisfagan las expectativas de los usuarios.

La guía presentada ha sido implementada con éxito, obteniendo satisfacción del cliente, mejora en los procesos productivos y desarrollo ágil de software, en algunos proyectos como el desarrollado en el último capítulo, lo que indica que su diseño fue adecuado, ya que al ser una guía de análisis permite que el consultor computacional utilice otros conocimientos acompañados de su creatividad, pero siempre estableciendo límites bien definidos de los estudios que se deben de realizar con el fin de obtener estudios correctos de las organizaciones. Como toda herramienta, la única forma de probar su efectividad es utilizándola, sólo su uso exhaustivo podrá detectar puntos de mejora.

La existencia de otro tipo de herramientas de modelado y diseño orientado a objetos, hacen que la guía tenga una limitante, la de no llegar hasta ese nivel de entregar productos en los formatos de otras herramientas como UML que parece convertirse en un estándar a nivel mundial. Esto me motiva a continuar trabajando en esta línea de estudio para superar esta limitante.

Las perspectivas de mejoramiento de esta guía pueden visualizase hacia el interior de la misma y hacia el exterior. Hacia el interior, se pueden mejorar sus procesos de análisis, se pueden enriquecer sus técnicas de análisis como por ejemplo introduciendo técnicas para hallar problemas y resolverlos basadas en heurística; se pueden implementar enfoques de planeación como procesos de mejora o análisis funcional; además sería conveniente que dentro de la misma guía se utilizara de manera formal alguna herramienta de control de proyectos como PERT. Hacia el exterior sería adecuado que los productos que esta entregue a otras metodologías de diseño orientado a objetos se presenten con estructura y simbología propia de la orientación a objetos como la de UML. Trabajos semejantes se están realizando actualmente en la Universidad de Lancaster Inglaterra dirigidos por Peter Checkland.

6 BIBLIOGRAFIA

- Ackoff R,.L (1987), Planificación de la empresa del futuro, México: Limusa
- Ackoff R,.L (1994), The democratic corporation, New. York: Oxford University Press.
- Booch Grady, (1996), Análisis y diseño orientado a objetos con aplicaciones, México: Adison Wesley Iberoamerica S.A. de C.V., p.p. 638
- Booch, Rumbaugh, Jacobson (1999), El lenguaje unificado de modelado, España: Addison Wesley Iberoamaericana, p.p. 432
- Cohoon – Davison (2000), Programación y diseño en C++, España: Ed. McGraw Hill, p.p. 1021.
- Checkland (1979), Techniques in soft system practice part:1, systems diagrams: some tentative guidelines, Journal of Applied Systems Analysis. No. 6
- Checkland(1993), Pensamiento de sistemas práctica de sistemas, México: Noriega editores, p.p. 367
- Checkland, Scholes (1994), La metodología de los sistemas suaves en acción, México: Noriega editores.
- Churchman(1973), El enfoque de sistemas, México: Diana, p.p. 270
- Cleland (1975) Systems Análisis and Project Management, Tokyo: Mc Graw Hill Kogalisha, p.p. 398
- Danae (1981), Enciclopedia básica Danae, Barcelona: Ediciones Danae
- Date(2001), Sistemas de bases de datos, México: Perarson Educación, 2001
- Graham Ian (1996), Métodos orientados a objetos, E.U.A.: Addison-Wesley, p.p.610
- Hillier,Hillier,Lieberman (2000), Introduction to Management Science, E.U.A.: Mc.Graw-Hill, p.p. 720
- Joyanes Luis (1996), Programación Orientada a objetos, México: McGraw-Hill
- Kepner,Tregoe (1989), El nuevo directivo racional, México: Mc Graw hill
- Miller,Heiman (1989), La venta conceptual, España: Folio, p.p. 261
- Pressman Roger S. (1998), Ingeniería del software un enfoque práctico, España: McGraw-Hill, p.p. 581
- Romero, Javier (2000), Contabilidad superior, México: McGraw-Hill, p.p. 783
- Wilson (1990), Systems: concepts, methodologies and aplications, UK: Wiley, p.p. 391

- Yourdon, E., Coad, P. (1991), Object-Oriented Analysis, E.U.A.: Yourdon Press

Otras Fuentes de Información

- Aguilar Mauricio (2000), El paradigma de los Stakeholders, artículo difundido en DEPFI UNAM por su autor
- Romo, Andrés (2000), Notas de clase del curso de Evaluación de proyectos, DEPFI UNAM
- Suárez, Javier (2000), Notas de clase del curso de Enfoque de sistemas, DEPFI UNAM
- Diccionario de la Real Academia de la Lengua Española (2004) , en línea, http://buscon.rae.es/diccionario/drae.htm
- Lars Mathiassen, Andreas Munk-Madsen (2003), Soft systems in software design, en línea, http://www.cs.auc.dk/~larsm/Dr_Techn/Volume_II/12.pdf
- Nicola Gigson (2003), Enterprise Resource Planning, en línea, http://www.computer.org/proceedings/hicss/0001/00017/00017041.PDF
- Fahad Al-Humaidan, B.N. Rossiter (2003), A taxonomy and evaluation for systems analysis methodologies in a workflow context. , en línea, http://www.cs.ncl.ac.uk/research/trs/papers/751.pdf
- Charles Lane (2003), Methods for transitioning from Soft Systems Methodology Model to Object Oriented Análisis, en línea, , en línea, http://www.dodccrp.org/1999CCRTS/pdf_files/track_6/092galvi.pdf
- Ian Sommerville (2003), Systems Engineering for Software Engineers, en línea, http://www.comp.lancs.ac.uk/computing/resources/ser/syseng/SysEng.pdf
- Introduction to software Engineering (2003), en línea, http://193.61.148.143/se99/
- Jeremy Rose and Mary Melrum (2003), en línea, Requirements generation for web site developments using SSM and the ICDT model, en línea, http://www.cs.auc.dk/%7Ejeremy/pdf%20files/BIT%201999.pdf
- ISSI (2003) Metodologías ágiles en el desarrollo de software, en línea, http://64.233.167.104/search?q=cache:abursINw_i0J:www.willydev.net/descargas/prev/TodoAgil.Pdf+Metodolog%C3%ACas+%C3%A1giles+en+el+desarrollo+de+software&hl=en
- SCRUM(2004), en línea, www.controlchaos.com
- Crystal Methodologies (2004), en línea, www.crystalmethodologies.org
- Dynamic Systems Development Method (DSDM) (2004), en línea, www.dsdm.org
- Adaptive Software Development (ASD) (2004), en línea, www.adaptivesd.com
- Feature -Driven Development (FDD) (2004), en línea, www.featuredrivendevelopment.com
- Lean Development10 (LD) (2004), en línea, www.poppendieck.com

ANEXO I

I.1 Aplicación de la guía para desarrollo de software especializado

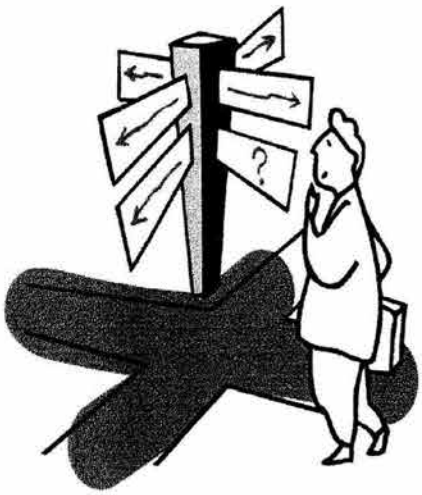
Si bien la herramienta presentada en este trabajo es ideal para desarrollo de software de gran tamaño, también se probó como una guía para desarrollar software especializado de simulación de procesos de tratamientos de aguas residuales[61].

SEMINARIO DE ENFOQUE SISTÉMICO[62]

Una primera actividad del análisis fue la presentación de la guía al grupo de trabajo en donde se les explicó en qué consistía la guía de manera general con una serie de láminas que a continuación muestro:

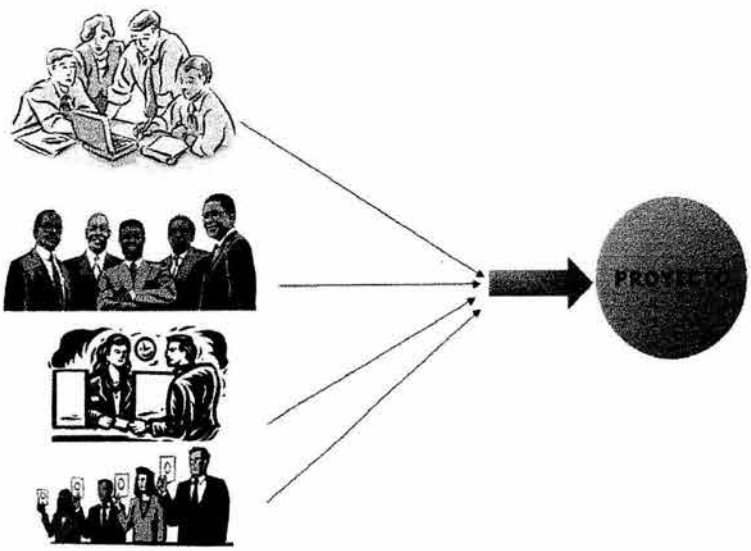
[61] Participación en el Proyecto #41596-Y de Investigación Básica, SEP-CONACYT, 2002-2003
[62] Actividad del Proyecto #41596-Y (citado)

La incertidumbre en el proyecto



Grupos diversos, con conocimientos diversos, puede dar como resultado incertidumbre

El enfoque sistémico como herramienta de cohesión



El enfoque sistémico orienta los esfuerzos, genera sinergia

Algunas definiciones

Cliente:Persona que comisiona un estudio con el fin de lograr "saber algo" o "hacer algo" relacionado con el sistema con el que está inmerso.

Tomador de decisiones: Individuo participe de un sistema, que puede alterar tanto partes como relaciones del sistema.

Algunas definiciones

Propietario del problema: Persona que se encuentra inmersa o tiene una discrepancia entre el "es" y el "debiera ser" de algo, que puede implicar parte o todo el sistema.

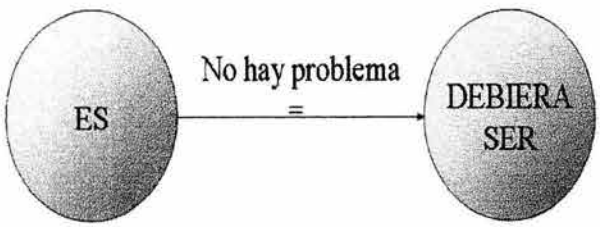
NOTA: Todos estos roles pueden ser adoptados por personas diferentes o por una sola persona.

Algunas definiciones

Problema: Es esa discrepancia entre el "es" y el "debiera ser".

Algunas definiciones

Consultor: Persona encargada de hacer que el propietario del problema deje de tener discrepancias entre el "es" y el "debiera ser", todo esto por encargo del cliente.



Algunas definiciones

Proyecto: Conjunto de actividades estructuradas para satisfacer una necesidad o resolver un problema en la forma de productos o servicios.

El proyecto surge por un problema sin resolver o por una necesidad insatisfecha.

Algunas definiciones

Conceptos básicos del proyecto:

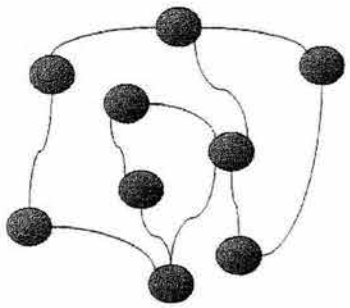
- Es distribuidor del riqueza.

- Es transitorio.

- Es específico.

Algunas definiciones

Sistema: Es un conjunto de elementos relacionados entre si.



Algunas definiciones

Para enfocar nuestro trabajo con un enfoque sistémico se debe de cumplir o satisfacer las siguientes condiciones propuestas por Ackoff.

I.- El sistema tiene una o más funciones definitorias.

II.- Cada parte en un subconjunto puede afectar el comportamiento o propiedades del todo.

III.- Existe un subconjunto que es suficiente para llevar a cabo los objetivos del todo. Cada una de las partes es necesaria pero insuficiente para cumplir el objetivo por si misma.

Algunas definiciones

... condiciones propuestas por Ackoff.

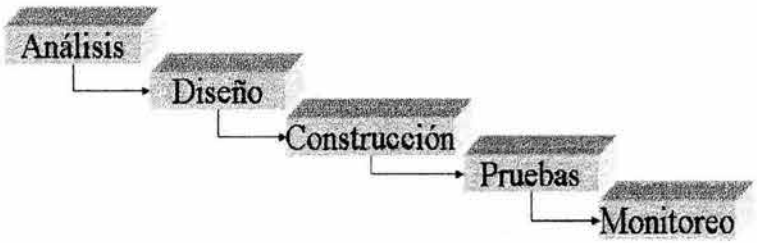
IV.- El comportamiento o propiedades propias o sobre el sistema de cada parte depende del comportamiento o propiedades del por lo menos otra parte del sistema.

V.- El efecto sobre el sistema de un subconjunto de partes, depende del comportamiento de por lo menos otro subconjunto.

Ciclo de vida del proyecto

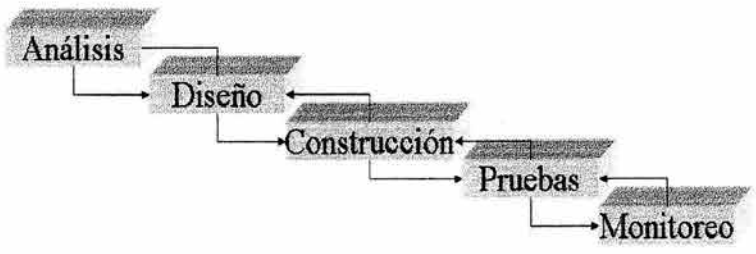
Un proyecto debe de ser flexible y adaptable. Esto debido a lo cambiante e impredecible del mundo actual.

Viejas tendencias del manejo de proyectos.



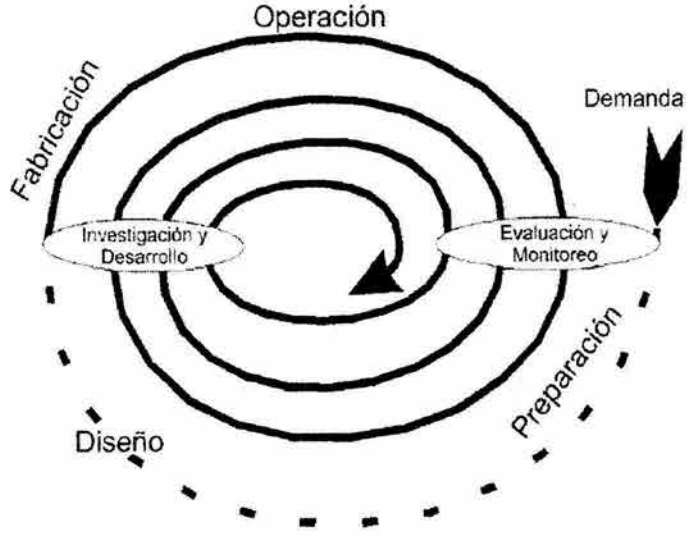
Ciclo de vida del proyecto

Una segunda panorámica con retornos:



Ciclo de vida del proyecto

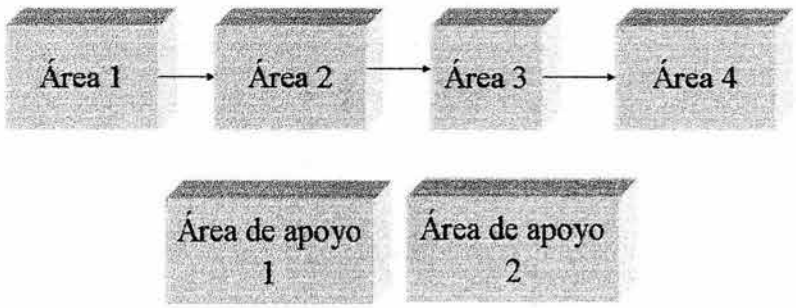
Metodologías de desarrollo de proyectos hoy en día:



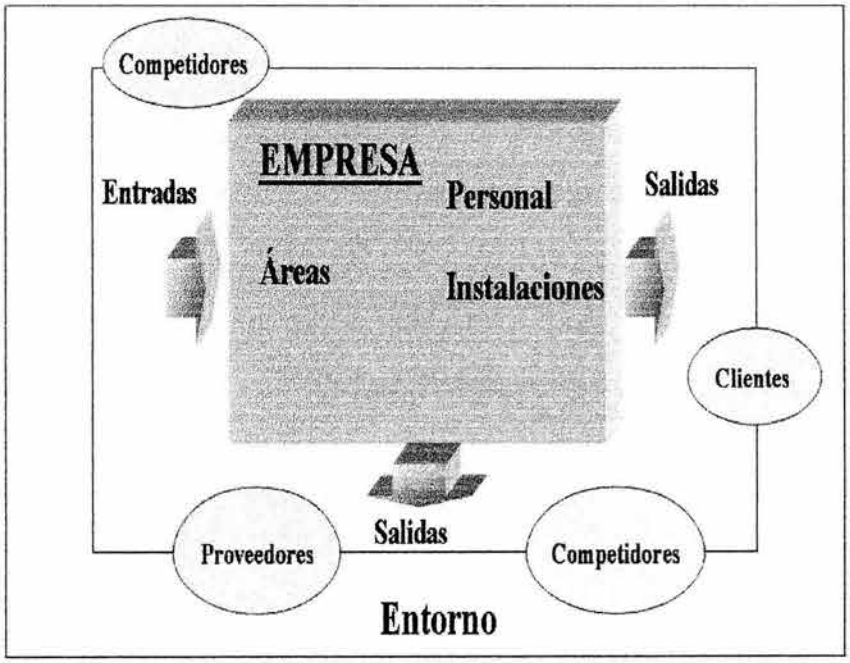
Identificación de la cadena de valor de la organización

Visitemos las instalaciones:

Objetivo: Obtener la cadena de valor.



Uso de mapas conceptuales Caja negra



Mapas conceptuales

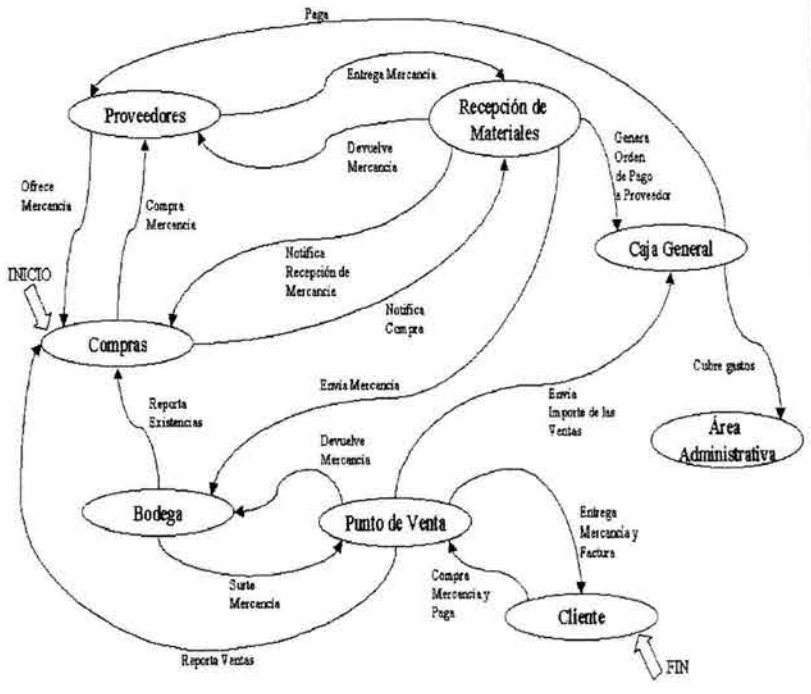
PRIMER NIVEL

Simbología

Entidad

Relación

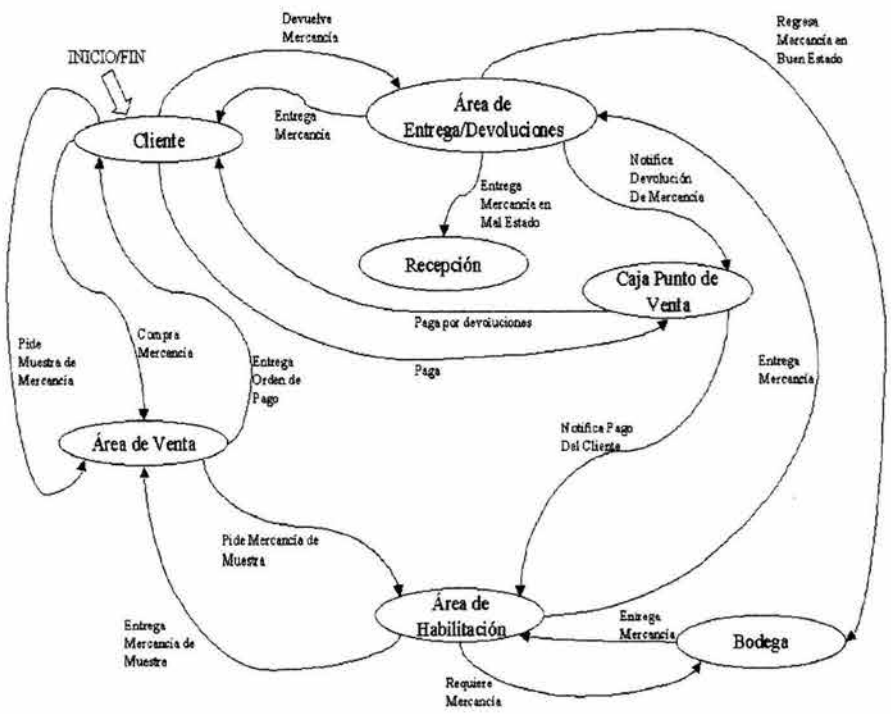
Diagrama conceptual de la empresa

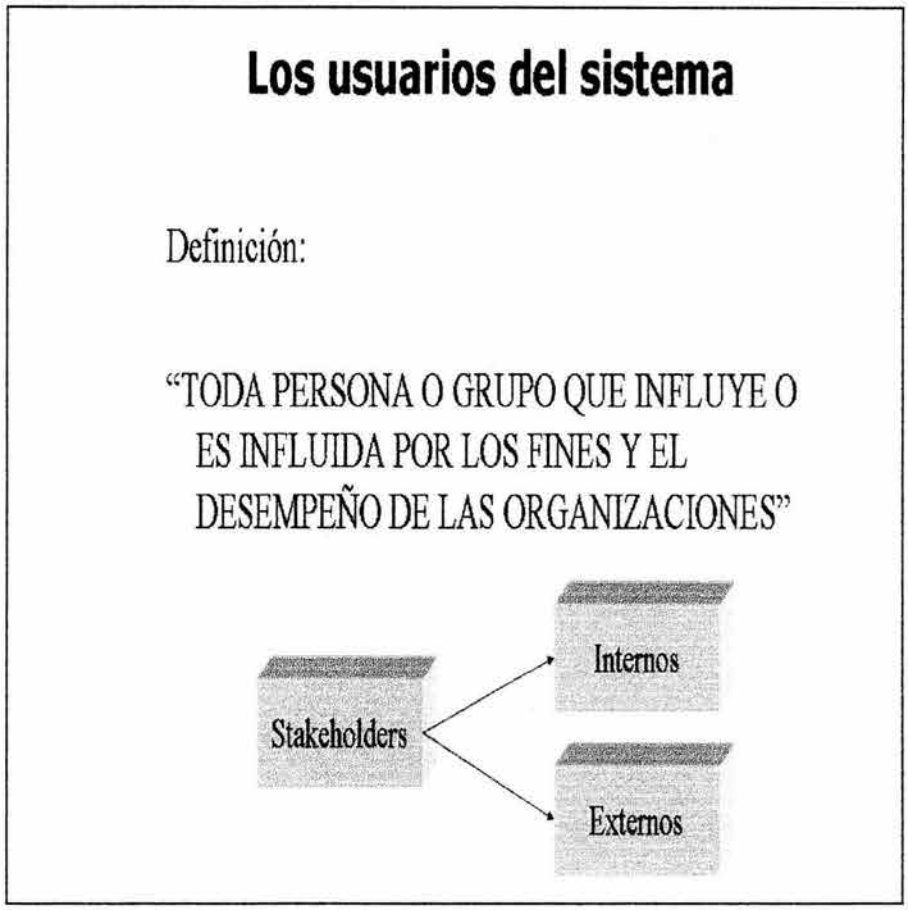


Mapas conceptuales

SEGUNDO NIVEL

Diagrama conceptual del Área de Punto de Venta

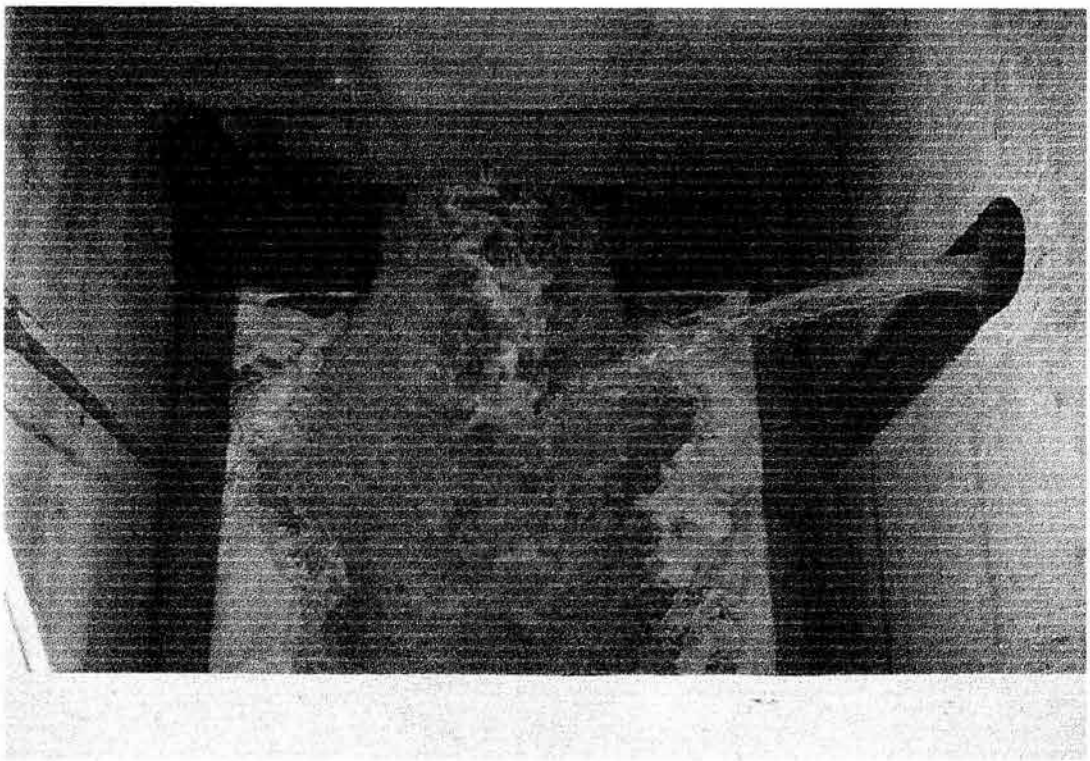




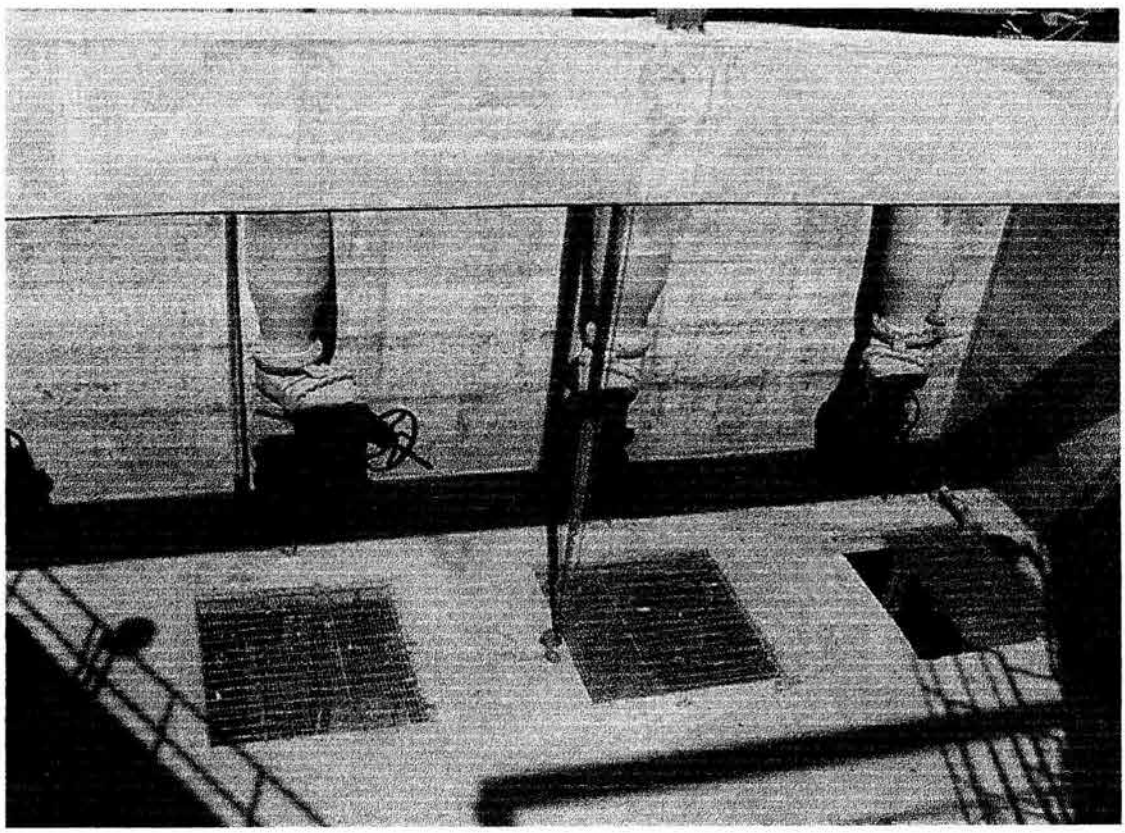
ANÁLISIS DE UNA PLANTA DE TRATAMIENTO DE AGUAS RESIDUALES[63]

Después de la presentación, el grupo de trabajo procedió a analizar el sistema, determinando los procesos más importantes. A continuación presento una serie de fotografías del sistema analizado.

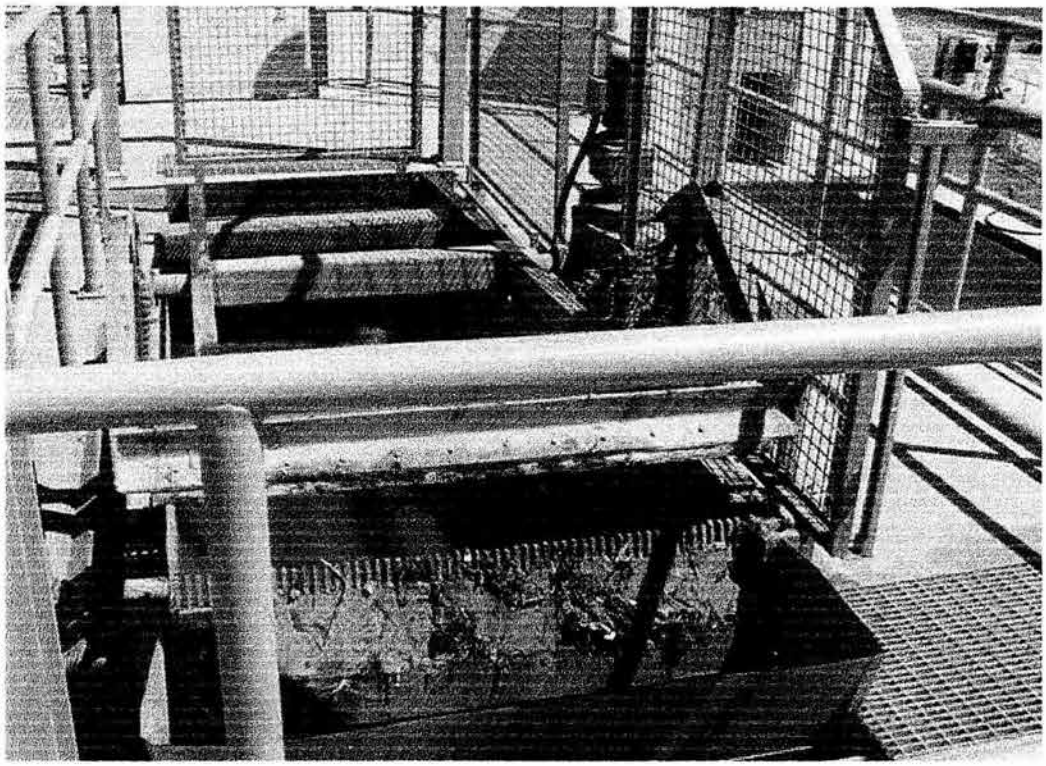
[63] Balderas P., Méndez J. y Rojel X., *Modeling some dynamics phenomena with Maple6® in a CAS-Based Math Class*, conferencia aceptada en Montreal Int'l Symposium on Technology and its Integration into Mathematics Education (TIME-2004). Ecole de technologie supérieure, Montreal Québec, Canadá, Julio 2004.



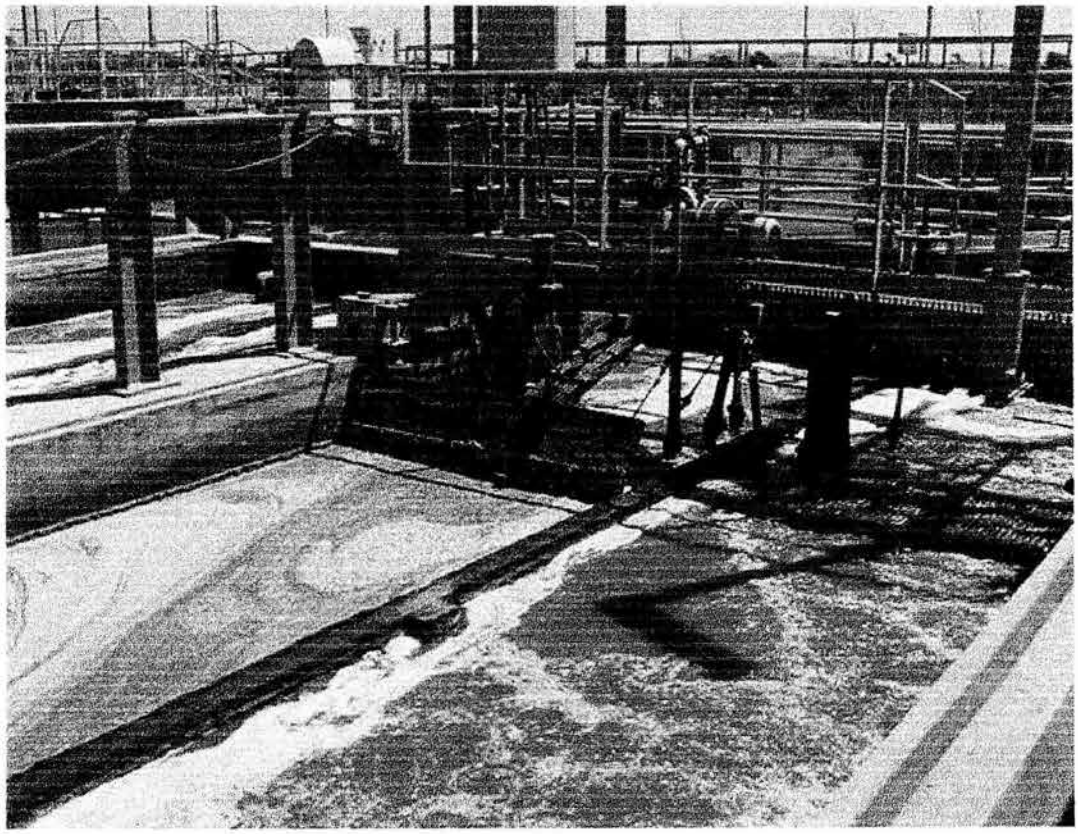
Recepción de agua residual



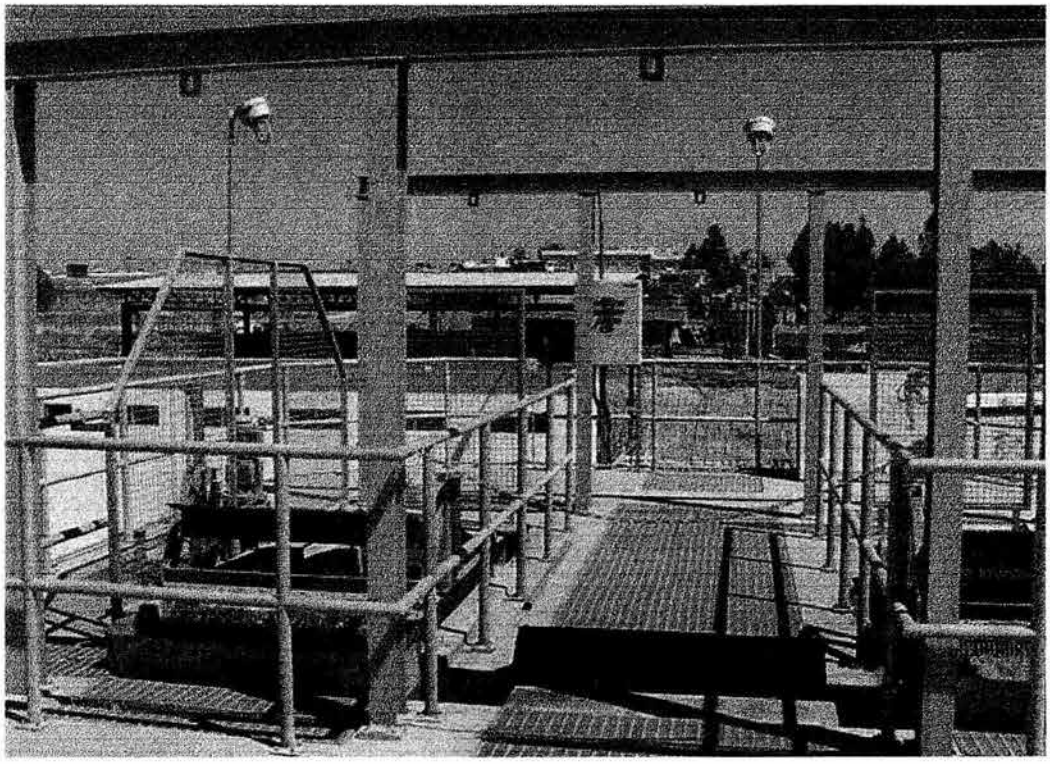
Estación de bombeo



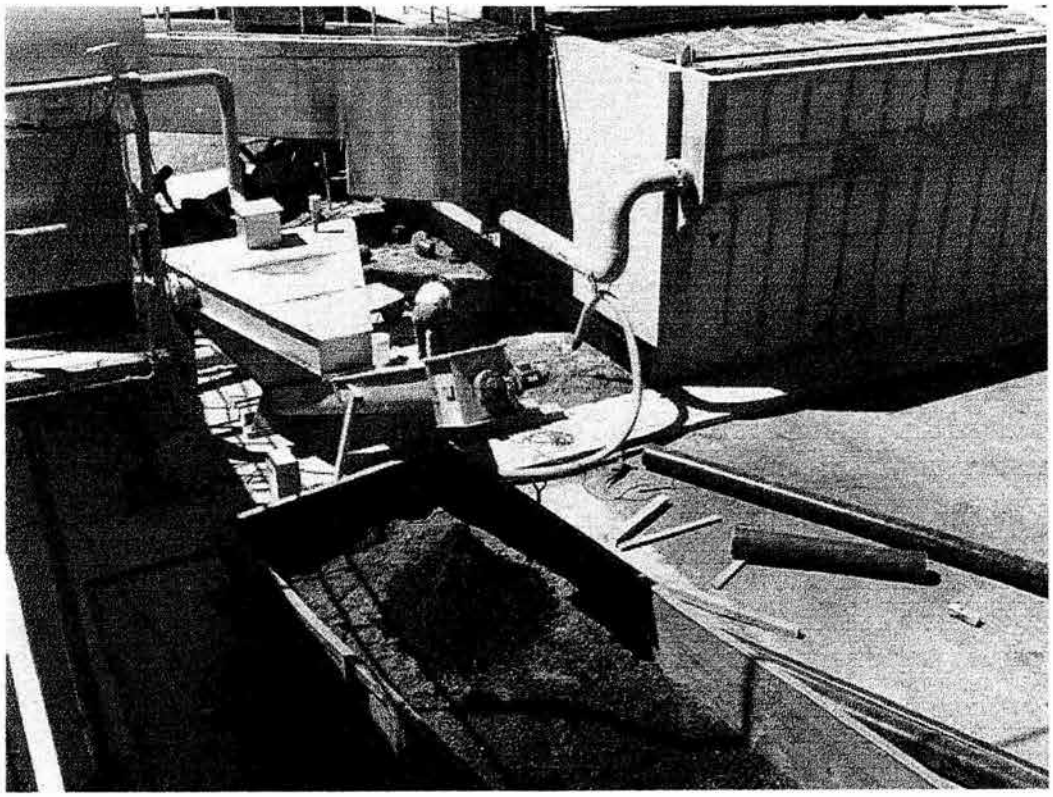
Cribado fino



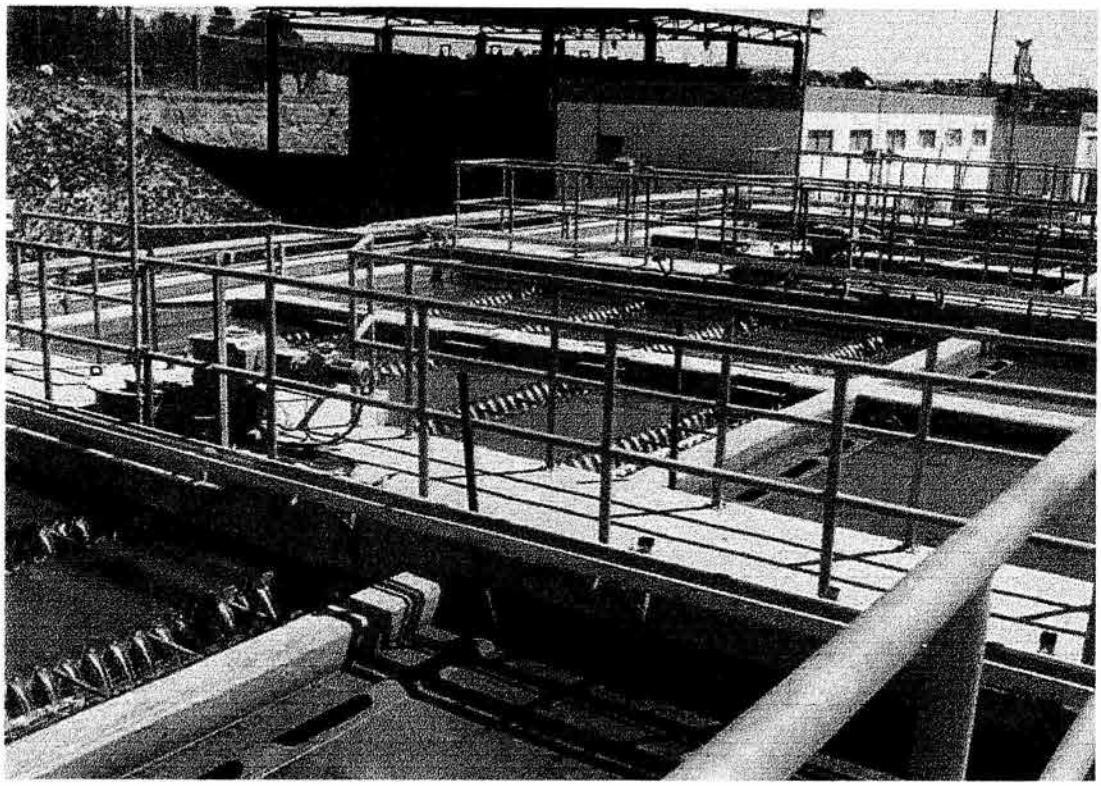
Remoción de grasas



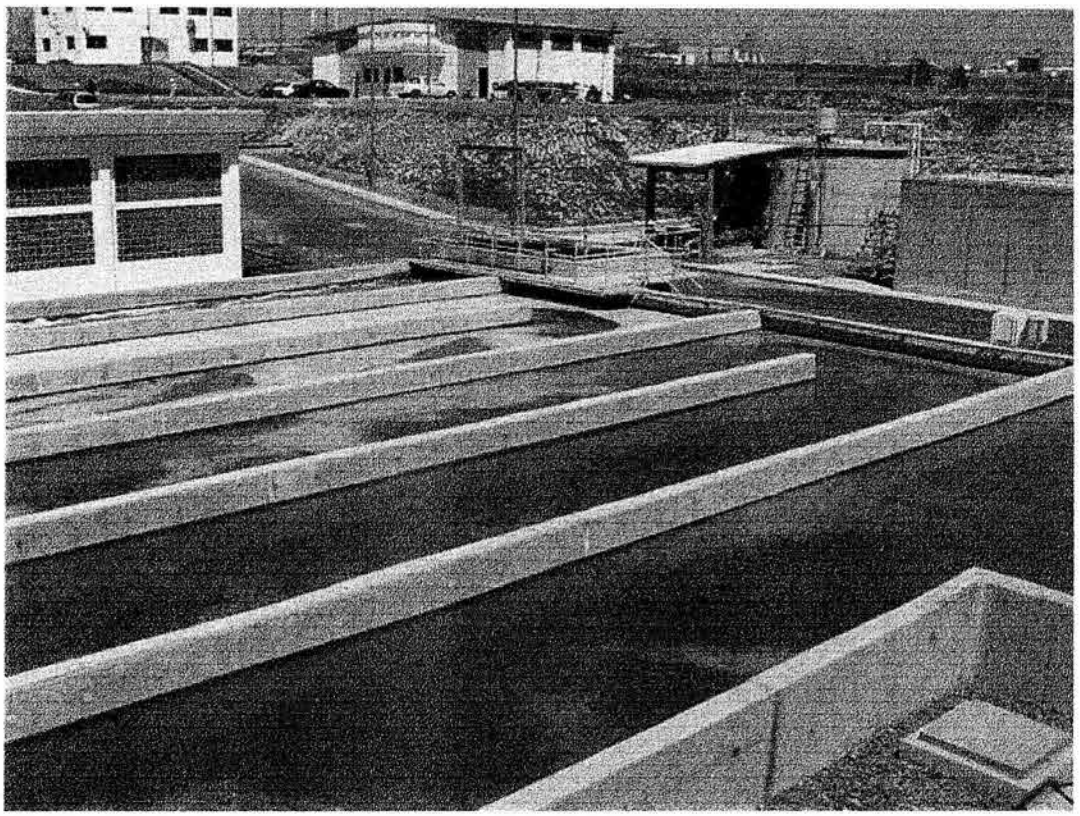
Pasillos de supervisión del tanque de remoción de grasas



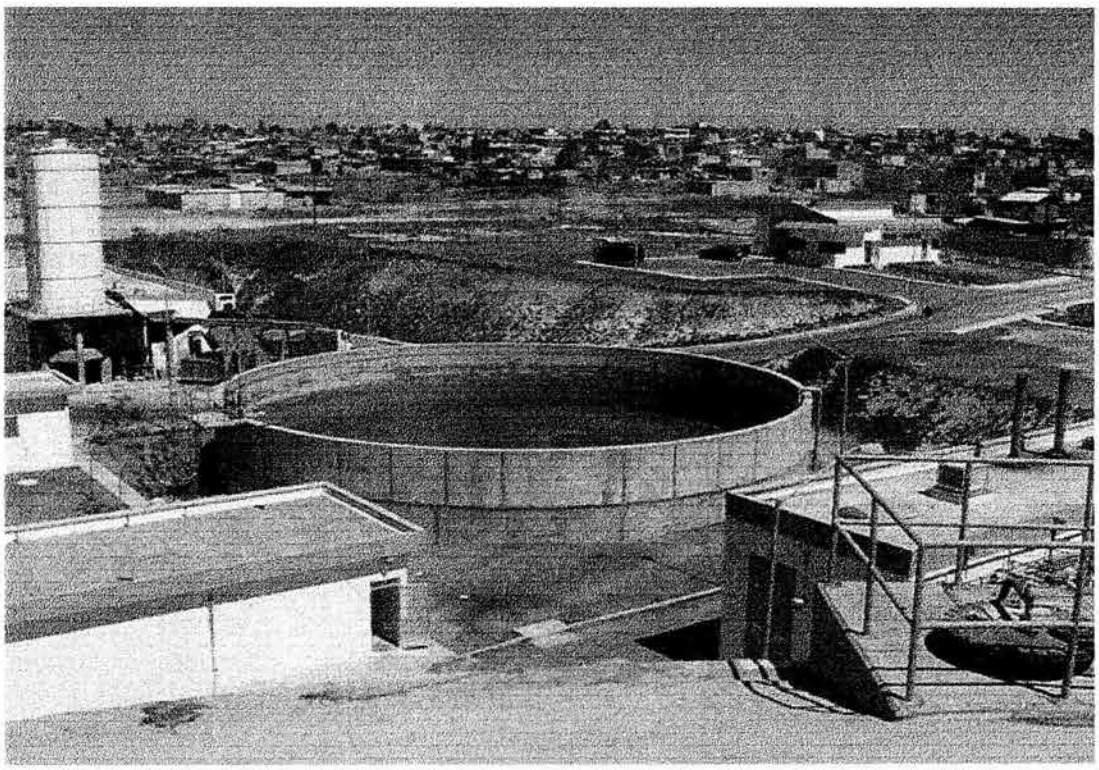
Remoción de grasas y plásticos



Remoción de arena



Cloración



Depósito de lodos

Después de varias visitas a esta planta, y la aplicación de los primeros pasos de la guía de análisis presentada en este trabajo, se elaboraron varios mapas conceptuales, de los que presento a continuación un ejemplo.

Procesos de tratamiento de aguas residuales (municipales) en una planta

Por último se seleccionaron tres procesos que se modelaron y se generaron sus respectivos programas en computadora en un paquete matemático llamado Maple6®. Los procesos fueron los tiempos de residencia de las aguas residuales en tanques, los procesos de apertura de válvulas y el tiempo de vaciado de un tanque contra la abertura de una válvula.

A continuación presento uno de los programas computacionales generado.

```
>#Facultad de Ingeniería, División de Estudios de
Posgrado, UNAM
>#Programa: TIEMPO DE VACIADO VS ABERTURA VÁLVULA
>
>Abertura_valvula:=proc(archivo,num_datos,ver_datos::nonne
gint)
>#El archivo recibe como parámetros
>#Parámetro 1 <<archivo>> Es el archivo del cual se
estraen los datos a procesar
>#Parámetro 2 <<num_datos>> Es el número de datos
contenidos en el archivo que se procesarán
>#Parámetro 3 <<ver_datos>> Indica si antes del
procesamiento de los datos se requiere que se muestren los
mismos
>
>    #Declaración de variables
>    local datos_importados,
>          cadena,
>          caudal,
>          contador,
>          #Variales que deberán de tener un nombre que
diga algo
>          f,
>          c,
>          x,
>          inverso_caudal,
>          tv,
>          tv_min,
>          tv_max:
>
>    datos_importados := Vector(1..num_datos):
>    caudal           := Array (1..num_datos):
>    inverso_caudal   := Array(1..num_datos):
>
>
>    cadena := "Archivo del que se toman ":
>    cadena := cat(cadena,num_datos," datos:"):
>
>    cadena := cat(cadena,archivo):
>    print(cadena) :
>
>    #Se procede a la importación de los datos NOTA:los
valores no deben de estar separados por comas
```

```
>
datos_importados:=ImportVector(archivo,format=rectangular)
;
>
>    #Bloque para visualizar los datos con los que se
trabajará
>    if ver_datos = 1 then:
>       for contador from 1 to num_datos do :
>          print(cat("Dato [",contador,"] -
>",datos_importados[contador])):
>       end do:
>    end if:
>
>    #Se pasan los datos importados a un arreglo para su
manipulación
>    caudal := datos_importados:
>
>    #Se declaran las librerías a utilizar para el cálculo
>    with(linalg):
>    with(stats):
>
>    #se procede a calcular el tiempo de vaciado
>
>       #a) se obtiene un vector con los inversos de los
valores del archivo
>
>       # se declara una función para el cálculo del
inverso
>       f := x -> x^[-1] * 293760.00:
>       for contador from 1 to num_datos do:
>          inverso_caudal[contador] := f(caudal[contador]):
>          print(inverso_caudal[contador]):
>       end do:
>       #b se obtienen algunos valores (recomendaría
explicar qué son)
>       tv_min:=min(inverso_caudal):
>end proc;
```

```
Abertura_valvula := proc(archivo, num_datos , ver_datos:nonnegint )
local datos_importados , cadena , caudal , contador , f, c, x, inverso_caudal , tv,
tv_min , tv_max;
    datos_importados := Vector( 1 .. num_datos );

    caudal := Array( 1 .. num_datos );
    inverso_caudal := Array( 1 .. num_datos );
    cadena := "Archivo del que se toman " ;
    cadena := cat( cadena , num_datos , " datos:" );
    cadena := cat( cadena , archivo );
    print( cadena );
    datos_importados := ImportVector ( archivo , format = rectangular );
    if ver_datos = 1 then for contador to num_datos do print(
            cat( "Dato [" , contador , "] ->" , datos_importados [ contador ] ))
        end do
    end if ;
    caudal := datos_importados ;
    with( linalg );
    with( stats );
    f := x → 293760.00×x^[ -1 ];
    for contador to num_datos do
        inverso_caudal [ contador ] := f( caudal [ contador ] ); end proc
        print( inverso_caudal [ contador ] )
    end do ;
    tv_min := min( inverso_caudal )
```