ESTADO ACTUAL Y TENDENCIAS DE LOS DEPARTAMENTOS DE SISTEMAS DE LAS EMPRESAS GRANDES DE CALI

(Nivel de utilización de técnicas estructuradas para el desarrollo de software).

GUILLERMO LONDOÑO ACOSTA

Físico, Universidad del Valle. Magister en Física, Instituto Venezolano de Investigaciones Científicas (IVIC), Caracas, Venezuela. Profesional Asociado a Investigación, Instituto Venezolano de Investigaciones Científicas (IVIC), Caracas, Venezuela. Asistente de Investigación, Universidad de Wisconsin, Milwaukee, U.S.A. Magister en Ingeniería de Sistemas, Universidad del Valle. Profesor Universitario ICESI en el área de ciencias físicas y tecnológicas y en el área de ingeniería de software.

I. INTRODUCCION

Este tema ha sido poco explorado y reconocido y sobre él es difícil formular hipótesis precisas o de cierta generalidad. Para satisfacer esta necesidad de conocimiento es indispensable llevar a cabo una investigación aplicada, exploratoria y descriptiva que nos dé una visión general y varias características fundamentales del medio, en cuanto al desarrollo de software.

Los resultados de la presente investigación podrán servir como retroalimentación a las entidades de educación superior, para el diseño de cursos, como también a las propias empresas como punto de referencia.

II. OBJETIVO GENERAL

Conocimiento del estado actual y principales tendencias de los departamentos de sistemas de las empresas grandes de Cali. Se hace énfasis en investigar si se utilizan o no técnicas estructuradas para el desarrollo de software.

OBJETIVOS ESPECIFICOS

Puesto que el problema consiste en una deficiencia de conocimiento, todos los

objetivos específicos están orientados a satisfacer estas necesidades.

Determinar:

1) Cuál es la estructura de los departamentos de sistemas.
2) Cuál es la estructura y función de los comités directivos de sistemas.
3) Cuál es el perfil profesional de las personas que trabajan en estos departamentos.
4) Cuál es el hardware y el software que poseen.
5) Cuál es el modelo de ciclo de vida que usan para el desarrollo de proyectos de sistemas.
6) Qué técnicas estructuradas usan.
7) Cuál es el factor de costo de sistemas, el cual definimos como la inversión hecha en hardware y software comprado, dividida entre los activos brutos de la empresa.
8) Cuál es la tendencia en el uso de lenguajes.
9) Si existe organización y métodos para sistemas.
10) Cuáles son las principales necesidades.

III.
MARCO TEORICO

Puesto que entre los principales objetivos está el determinar cuál es el modelo de ciclo de vida utilizado en el desarrollo de un proyecto de sistemas y cuáles son las principales metodologías estructuradas utilizadas en las diferentes fases del desarrollo, tomaré como marco teórico una recopilación bibliográfica sobre estos temas.

1. Modelos del ciclo de vida de un proyecto en la ingeniería del software

El ciclo de vida del software describe las etapas a las cuales éste es sometido desde su nacimiento hasta su muerte. Este conjunto de etapas recibe el nombre de modelo o paradigma del ciclo de vida del software.

El ciclo de vida sin planificación

Este modelo de ciclo de vida es el resultado de la falta de deseo de planificar y detallar lo que se hará antes de realmente hacerlo. En este modelo se empieza a codificar casi tan pronto como se inicia el proyecto. La filosofía básica es que es más fácil escribir los programas, depurarlos y lograr aproximadamente lo que el usuario quiere hacer en lugar de gastar mucho tiempo aplicando todo un análisis y diseño del sistema.

Los siguientes tres factores fomentan estas creencias:

- *Sicológicos:* Codificar satisface la necesidad de lograr que los programas funcionen. Podemos ver los frutos de nuestra labor en términos de líneas de código. Gastar la misma cantidad de tiempo detallando los requerimientos, definiendo la arquitectura del sistema completo, refinando algunas definiciones necesarias, o planeando nuestras actividades, no ofrece un ejemplo tan obvio de logro; en consecuencia, la tendencia es a evitar tales actividades.
- *Educación:* La educación que la mayoría de la gente recibe está relacionada con resolver problemas determinísticos. Los problemas a resolver están bien establecidos; ellos no requieren investigación o juicios. Ellos tienen una y solamente una respuesta y criterios completamente claros. Una persona que venga de esta clase de ambiente tiene poca motivación para tratar con cosas desconocidas como el análisis y diseño de sistemas.
- *Naturaleza del software:* La mayoría de las tareas lucen relativamente fáciles hasta que empezamos a trabajar sobre ellas. Una razón para explicar esto es la naturaleza lógica del software. Nadie realmente ve un programa de computador. Lo que vemos

son imágenes estáticas del programa. Vemos cintas magnéticas, diskettes y listados, pero los programas mismos nunca se ven. Esto tiene mucho que ver con la tarea de desarrollar software. Cuando examinamos una actividad física tal como la construcción de una casa, podemos, a priori, identificar muchas etapas de la construcción. Además podemos ver que si se hacen cambios en alguna etapa, probablemente las etapas subsecuentes se verán afectadas y el proyecto completo tendrá que cambiarse. La construcción de software, sin embargo, está realmente fuera de nuestra experiencia cotidiana. Por lo tanto, parece relativamente simple ajustarse a las necesidades del usuario sin análisis ni diseño, a causa de que no podemos identificar con sólo lo que está involucrado en las entrevistas, aquellos requerimientos.

El ciclo de vida sin planificación, está basado sobre la idea de que habrá muchos errores de código y por lo tanto, no se puede perder tiempo precioso en fases tan científicas como análisis y diseño.

Los siguientes tres modelos son planteados por Presmman [1]:

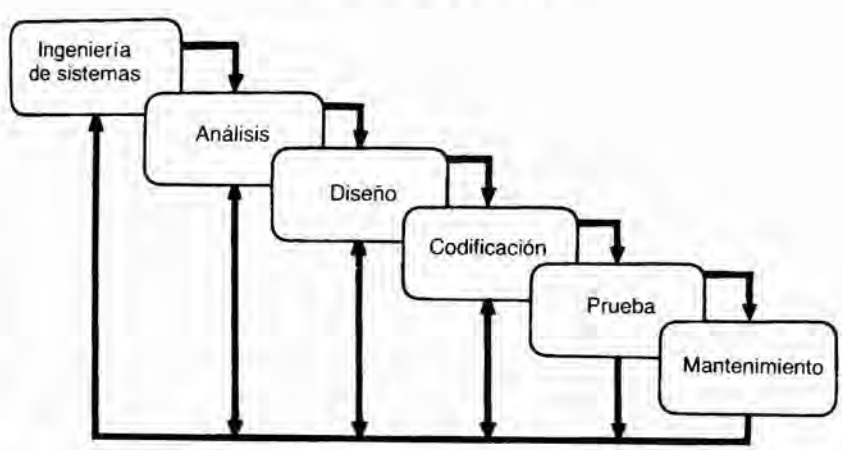
Ciclo de vida clásico

La figura 1 ilustra el modelo de ciclo de vida clásico llamado también el modelo en cascada. Este modelo exige un enfoque sistemático, secuencial, del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento. Este modelo abarca las siguientes actividades:

- *Ingeniería y análisis del sistema:* Define el papel de cada elemento de un sistema informático, asignando finalmente el papel que jugará el software. Esta visión del sistema es esencial cuando el software debe interrelacionarse con otros elementos tales como hardware, personas y bases de datos. Una vez que está asignado el ámbito del software, se asignan los recursos, se estiman los costos y se definen las tareas y planificación del trabajo.

- *Análisis de los requerimientos del software:* El proceso de recogida de los requerimientos se centra e intensifica especialmente en el software. Para comprender la naturaleza de los programas que hay que construir, el analista debe comprender la función,

Figura 1.
Ciclo de vida clásico



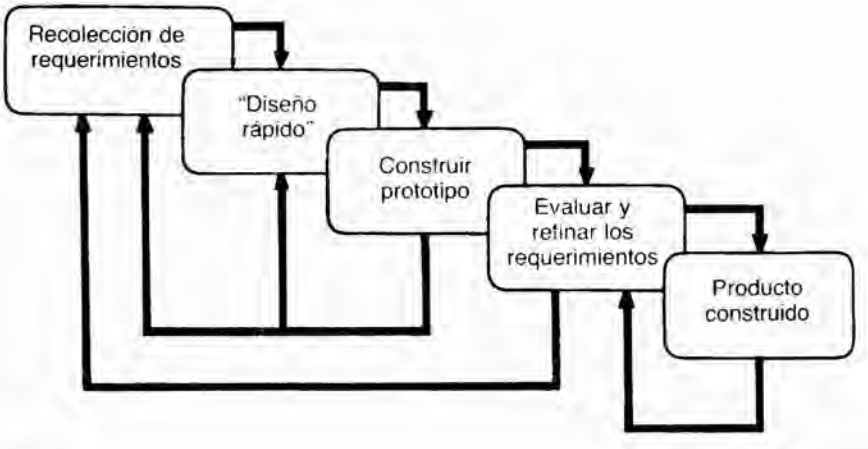
rendimiento e interfases requeridos. Los requerimientos tanto del sistema como del software se documentan y revisan con el usuario.

- *Diseño:* El diseño del software es realmente un proceso multipaso que se enfoca sobre tres aspectos: estructura de datos, arquitectura del software y detalle procedimental. El proceso de diseño traduce los requerimientos en una representación del software que pueda ser establecida de forma que se obtenga la calidad requerida antes de que comience la codificación. Como los requerimientos, el diseño se documenta y forma parte de la configuración del software.
- *Codificación:* El diseño debe traducirse en una forma legible para la máquina. El paso de la codificación ejecuta esta tarea. Si el diseño se ejecuta de una manera detallada, la codificación puede realizarse mecanicamente.
- *Prueba:* Una vez que se ha generado el código, comienza la prueba del programa. La prueba se enfoca sobre la lógica interna del software, asegurando que todas las sentencias se han probado, y sobre las funciones externas, esto es, realizando pruebas para asegurar que la entrada definida producirá los resultados que realmente se requieren.
- *Mantenimiento:* El software sufrirá indudablemente cambios después de que se entregue al usuario. Los cambios ocurrirán debido a que se han encontrado errores, debido a que el software debe adaptarse por cambios del entorno externo (por ejemplo, un cambio solicitado debido a que se tiene un nuevo sistema operativo o dispositivo periférico), o debido a que el usuario requiere aumentos funcionales o de rendimiento.

Construcción de prototipos

La secuencia de sucesos para el modelo de construcción de prototipos se muestra en la figura 2. Como en todos los modelos de desarrollo de software, la construcción de prototipos empieza con la recolección de los requerimientos. El analista y el usuario se reunen y definen los objetivos globales para el software, identifican todos los requerimientos conocidos y perfilan las áreas en donde será necesario una mayor definición.

Figura 2.
Construcción de prototipos.



Luego se produce un diseño rápido. El diseño rápido se enfoca sobre la representación de los aspectos del software, visibles al usuario (por ejemplo, métodos de entrada y formatos de salida). El diseño rápido conduce a la construcción de un prototipo. El prototipo es evaluado por el usuario y se utiliza para refinar los requerimientos del software a desarrollar. Se produce un proceso interactivo en el que el prototipo es afinado para que satisfaga las necesidades del usuario, al mismo tiempo que facilita al que lo desarrolla una mejor comprensión de lo que hay que hacer. Idealmente, el prototipo sirve como un mecanismo para identificar los requerimientos del software. Si se construye un prototipo que funciona, el realizador intenta hacer uso de fragmentos de programas existentes o aplica herramientas (por ejemplo, generadores de informes, gestores de ventanas, etc.) que faciliten la rápida generación de programas que funcionen.

Técnicas de cuarta generación

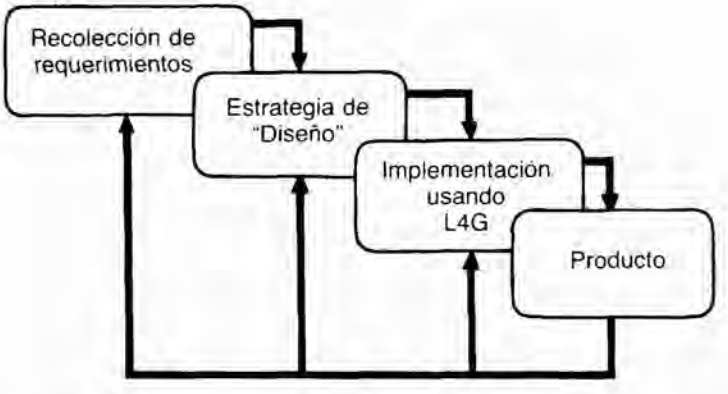
Este modelo abarca un amplio espectro de herramientas del software que tienen una cosa en común: todas facilitan al que desarrolla el software la especificación de algunas características a alto nivel. Luego, la herramienta genera automáticamente el código fuente basándose en la especificación del técnico. El modelo T4G para la ingeniería del software se orienta hacia la habilidad de especificar software a un nivel que sea más próximo al lenguaje natural o en una notación que proporcione funciones significativas.

Actualmente, un entorno para el desarrollo del software que soporte el modelo T4G incluye algunas o todas las siguientes herramientas: lenguajes no procedimentales para consulta a base de datos, generador de informes, manipulador de datos, interacción y definición de pantallas y generación de código; capacidades gráficas de alto nivel; y capacidad de hoja de cálculo.

El modelo T4G para la ingeniería del software se describe en la figura 3. Como en otros modelos, T4G empieza con el paso de recolección de requerimientos. Idealmente, el usuario debe describir los requerimientos y estos deben traducirse directamente en un prototipo operacional.

La implementación usando L4G facilita al que desarrolla el software, la descripción de los resultados deseados, los cuales se traducen automáticamente en

Figura 3.
Técnicas de cuarta generación



código fuente para producir dichos resultados. Obviamente, debe existir una estructura de datos con información relevante y debe estar rápidamente accesible al L4G.

Existen otros modelos como el incremental, el modelo en V y el modelo en espiral, los cuales no son utilizados en nuestro medio.

Metodologías estructuradas

Todas las aplicaciones del software pueden colectivamente llamarse procesamiento de datos. El software se construye para procesar datos; para transformar datos de una forma a otra; esto es, para aceptar entrada, manipularla de alguna forma y producir una salida. En este procesamiento de la información los aspectos que más nos interesan acerca de ella son: cómo fluye, cuál es su contenido y cuál es su estructura. Estos tres aspectos son conocidos como el dominio de la información. Para comprender completamente el dominio de la información, deben considerarse cada una de estas tres partes.

Dependiendo del aspecto del dominio de la información al cual se le dé más énfasis, nacen dos escuelas de metodologías de análisis y diseño estructurado a finales de la década de los 70: metodologías orientadas al flujo de la información y metodologías orientadas a la estructura de la información. Estas metodologías obviamente son fuertes en el aspecto de la información hacia el cual están enfocadas.

Las metodologías estructuradas se crearon para lograr que el desarrollo de software dejara de ser una habilidad personal y se convirtiera en una disciplina tipo ingeniería. Estas técnicas evolucionaron desde una metodología de programación a técnicas que incluyen metodologías de análisis, diseño y prueba como también conceptos sobre el manejo de proyectos y herramientas de documentación.

Hay dos versiones similares de análisis estructurado orientado al flujo de datos: De Marco - Yourdon[2] y Gane-Sarson.[3] Ambas representan una disciplina estructurada basada sobre los siguientes conceptos:

- Organización jerárquica descendente.
- Descomposición funcional.
- Herramientas gráficas de comunicación y documentación.

Existen dos metodologías (Jackson y Warnier-Orr) de análisis orientadas a la estructura de la información que tienen varias características en común:

- Todas asisten al analista en la identificación de los objetos de información clave (también llamados entidades o ítems) y operaciones (también llamadas acciones o procesos).
- Todas suponen que la estructura de la información es jerárquica.
- Todas requieren que la estructura de la información se represente usando secuencia, selección y repetición.
- Todas dan un conjunto de pasos para transformar una estructura de datos jerárquica en una estructura de programa.

Las metodologías de diseño estructurado emplean la descomposición funcional como su mecanismo básico de diseño. Las metodologías de diseño usadas más ampliamente son: el diseño estructurado de Yourdon - Constantine, la metodología de diseño de Jackson y la metodología de diseño de Warnier-Orr.

Todas estas metodologías se basan en el diseño, más informal, top-down o descendente. Este diseño usa un proceso paso a paso que empieza con la visión funcional más general de lo que se va a realizar, descompone este panorama general en subfunciones y luego repite el proceso hasta que todas las subfunciones son lo suficientemente pequeñas como para ser implementadas en lenguaje de computación.

El diseño estructurado de Yourdon está compuesto de estrategias de diseño,

ayudas para evaluación y técnicas de documentación. Esta técnica estructurada es el método de diseño top-down más varias guías tanto para sistematizar el proceso de diseño como para medir la calidad.

La metodología de diseño de Jackson también es un refinamiento de la metodología de diseño top-down. La principal diferencia con la metodología de diseño top-down o el diseño estructurado de Yourdon está en que la metodología de Jackson se guía por los datos, mientras que las otras se guían por los procesos. La metodología de Jackson empieza con el diseño de las estructuras de datos y la estructura del programa se deriva de la estructura de los datos.

La metodología de diseño de Warnier-Orr también se guía por los datos pero está enfocada hacia la salida del sistema y esto la hace diferente de los otros métodos de diseño estructurado. La filosofía de Warnier-Orr dice que la salida del sistema determina completamente la estructura de los datos y estos a su vez determinan la estructura del programa. El proceso de diseño de Warnier-Orr empieza con una descripción jerárquica de la salida del sistema. La estructura de entrada y la estructura del programa se derivan de la estructura de la salida.

El cuadro N° 1 reúne las principales metodologías estructuradas. Los cuadros 2 y 3 muestran un resumen de las principales metodologías de análisis y diseño.

Estas metodologías nacieron a finales de la década de los 70 y en esa época no existían lenguajes de programación de cuarta generación, ni estaban disponibles herramientas para construir prototipos que ayudaran en el desarrollo de sistemas. Los computadores personales no se habían popularizado y no existían herramientas que automatizaran estas metodologías. El desarrollo en estas áreas ha tenido impacto sobre la aceptación del análisis estructurado.

En la actualidad los desarrolladores de software tienen que tratar con sistemas de bases de datos y sistemas en tiempo real. Esto hace que el análisis estructurado moderno[4] emplee el diagrama entidad-relación,[5] para modelar los datos, y el diagrama de transición de estados para modelar los eventos. La integración de estos dos modelos con el modelo de procesos (diagrama de flujo de datos) conforma el análisis estructurado moderno.

IV
METODOLOGIA DE LA INVESTIGACION REALIZADA

Estamos interesados en empresas que tengan departamentos de sistemas bien conformados y con cierta experiencia. Estos departamentos son más o menos numerosos, con personas que conocen muy bien la organización, no son solamente técnicos sino también personas que administrativa y organizacionalmente apoyan a la empresa. Los departamentos de sistemas de este tipo son los que realmente llevan a cabo desarrollos de software.

Las empresas pequeñas y medianas, normalmente sistematizadas usando microcomputadores, con presupuestos bajos para la adquisición de los equipos y todo lo que tiene que ver con la sistematización, no desarrollan software sino que compran paquetes disponibles en el mercado que se adecúen más o menos bien a sus necesidades. El costo del paquete es mucho más bajo que los costos de desarrollo.

Para seleccionar la población de empresas a encuestar nos apoyamos en la guía para la investigación de mercados de Publicar S.A. Esta guía trae todas las empresas de Colombia que figuran en las cámaras de comercio, clasificadas de acuerdo con su actividad económica según el CIIU (Código Internacional Industrial Uniforme), clasificadas luego por ciudades y finalmente en orden alfabético.

Cuadro 1
Resumen de las metodologías estructuradas:

Metodología	Orientada a	Etapas del ciclo de vida cubiertas	Herramientas utilizadas
De Marco-Yourdon	Flujo de la información	Análisis	Diagramas de flujo de datos Diccionario de datos Lenguaje estructurado Tablas y árboles de decisión
Gane - Sarson	Flujo de la información	Análisis	Diagramas de flujo de datos Diccionario de datos Lenguaje estructurado Tablas y árboles de decisión
Yourdon - Constantine	Flujo de la información	Diseño Arquitectónico	Diagramas de flujo de datos Cartas de estructura
Jackson	Estructura de la información	Análisis Diseño Arquitectónico Diseño de programas	Diagrama de especificación Diagramas de Jackson Lenguaje estructurado
Warnier - Orr	Estructura de la información	Análisis Diseño Arquitectónico Diseño de programas	Diagrama de entidades Diagrama de ensamblamiento de líneas. Diagramas de Warnier - Orr

Cuadro 2

Comparación de Metodologías de Análisis Estructurado

De Marco - Yourdon	Gane - Sarson	Jackson	Warnier - Orr
1. Construir un modelo físico del sistema actual. 2. Construir un modelo lógico del sistema actual. 3. Construir un modelo lógico del sistema propuesto. 4. Crear una familia de nuevos modelos físicos. 5. Estimar costos y programación para cada modelo. 6. Seleccionar uno de los modelos. 7. Empaquetar la especificación en subsistemas. Nota: Todos los modelos se construyen usando diagramas de flujo de datos.	1. Construir un modelo lógico del sistema actual. 2. Construir un modelo lógico del sistema propuesto. 3. Diseñar una base de datos física. 4. Construir un modelo físico del sistema propuesto. 5. Empaquetar la especificación en subsistemas. Nota: Todos los modelos se construyen usando diagramas de flujo de datos.	1. Identificar las entidades y acciones del sistema. 2. Construir un modelo del mundo real ordenando las acciones en el tiempo y representándolas mediante diagramas de Jackson. 3. Seleccionar y relacionar las entidades y acciones mediante los datos. Especificar los detalles de los procesos usando lenguaje estructurado.	1. Examinar como se mueven los datos entre productores y consumidores de la información usando diagramas de entidades. 2. Establecer las funciones de aplicación usando diagramas de Warnier. 3. Modelar la estructura de la salida del sistema usando diagramas de Warnier-Orr.

Cuadro 3
Comparación de Metodologías de Diseño Estructurado

Yourdon - Constantine	Jackson	Warnier - Orr
1. Refinar diagramas de flujo de datos. 2. Establecer categoría del flujo de la información o de transacción. 3. Construir la carta de estructura del programa haciendo análisis de transformación y/o transacción. 4. Factorizar la estructura 5. Refinar la estructura usando heuristicas de diseño. 6. Desarrollar las descripciones de las interfases y la estructura global de datos.	1. Expandir el diagrama de especificación del sistema, mediante la conexión de procesos de funciones. 2. Especificar las ligaduras de tiempo impuestas en el sistema. 3. Desarrollo de programas: a) Hacer un diagrama de estructura jerárquico para los flujos de datos de entrada y salida. b) Formar con los diagramas anteriores una estructura del programa. c) Hacer una lista de las operaciones y asignar cada operación a un componente de la estructura del programa. d) Transcribir la estructura del programa en texto estructurado.	1. Representar cada salida del programa como una estructura jerárquica. 2. Definir todos los elementos de datos necesarios para producir la salida. 3. Definir todos los eventos que puedan afectar los elementos de datos. 4. Definir los archivos físicos para los datos de entrada. 5. Diseñar la lógica de procesos necesaria para producir la salida a partir de la entrada. 6. Añadir lógica de control y procedimiento para manipulación de archivos.

El código CIIU clasifica las empresas en nueve grupos de acuerdo con su actividad económica:

- Agricultura, caza, silvicultura y pesca.
- Explotación de minas y canteras.
- Industria manufacturera.
- Electricidad, gas y vapor.
- Construcción.
- Comercio, restaurantes y hoteles.
- Transporte, almacenamiento y comunicaciones.
- Establecimientos financieros, seguros, bienes inmuebles y servicios a compañías.
- Servicios comunales, sociales y personales.

Esta guía también trae la información clasificada de acuerdo con los activos brutos de la empresa, lo cual permite segmentar según el tamaño de éstas. El rango más alto de clasificación corresponde a activos brutos superiores a mil millones de pesos. Las empresas que aparecen en este rango son las que hemos definido como empresas grandes y son éstas las que constituyen nuestra población de estudio.

Esta guía no trae el valor exacto de los activos brutos de las empresas sino que las clasifica en diferentes rangos. Para poder calcular el factor de costos de sistemas, fue entonces necesario acudir a la Cámara de Comercio de Cali para obtener información sobre el valor exacto de los activos brutos de cada empresa.

Para capturar los datos utilizamos la entrevista como instrumento de recolección. Al diseñar la encuesta se debe tener en cuenta que las preguntas logren satisfacer todos los objetivos específicos.

V.
RESULTADOS

Se presentan los resultados sobre una muestra de catorce empresas encuestadas. En esta muestra tenemos nueve empresas manufactureras, cuatro de servicio y una de investigación agropecuaria.

La investigación continuará con el fin de cubrir varias empresas en cada actividad económica y lograr así resultados más precisos, como también correlaciones entre las variables estudiadas y la actividad económica, por ejemplo.

A continuación se presentan los resultados obtenidos hasta el momento:

- En el 86% de los casos existen comités de sistemas.
  Composición: Vicepresidentes, gerentes, directores y el jefe o gerente de sistemas.

  Funciones: Establecer prioridades y pautas de desarrollo, controlar el desarrollo de los proyectos, sugerir compra de hardware y software.

  Estos comités, en promedio, se reúnen cada mes.

  En el 25% de las empresas ya existe o se está conformando Organización y Métodos para sistemas.
- En el organigrama de la empresa, el departamento de sistemas depende de:

Administración y finanzas	65% de los casos.
Subgerencia general	21% de los casos.
Contraloría	14% de los casos.

- Fecha promedio de iniciación en sistemas: 1978 ± 3 años.
- Costo promedio de sistemas para empresas de manufactura: 3.1% ± 2%.
  Costo promedio de sistemas para empresas de servicio: 35% ± 15%.
- Perfil profesional del jefe de sistemas:

Con posgrado en sistemas	20%
Ingeniero de sistemas	36%
Otras profesiones	44%

- Origen de los ingenieros de sistemas de estos departamentos:

Universidad de Los Andes	26%
Universidad Industrial de Santander	26%
ICESI	26%
Universidad Javeriana	13%
Universidad INCA	7%
EAFIT	2%
Universidad Autónoma	2%
Universidad Distrital	2%

- Equipo utilizado:

IBM/34	6%
IBM/36	13%
IBM/38	3%
IBM/4361	6%
IBM/AS400	10%
Digital Vax	6%
Data General	3%
HP-3000	13%
HP-9000	6%
HP-RS	17%
Unisys	3%
Texas 1500	3%
Texas 990	3%
Compac	6%

- Número promedio de terminales: 35 ± 15.
  Número promedio de micros: 17 ± 12.

- El ciclo de vida con prototipos es el más utilizado y las razones que se dan para su uso son: 1) La presión a la cual se encuentra sometido el departamento de sistemas, por parte de los usuarios y 2) Dificultad en establecer explícitamente al principio todos los requerimientos.

  Se usan generalmente prototipos en papel o utilizando un procesador de palabras. Otros construyen programas típicos, para determinadas áreas de la empresa, denominados programas esqueleto.

  Ocho de las catorce empresas ya están usando herramientas de cuarta generación y por lo tanto, están empezando a usar el ciclo de vida asociado a estas herramientas.

- Aplicaciones desarrolladas por las empresas: 62% ± 20%

- En el 50% de las empresas el departamento de sistemas procesa, en promedio, el 24% de la información.

- Porcentaje de aplicaciones escritas en:

Cobol	33%
RPG	25%
DBase	4%
Basic	3%
Oracle	13%
Speedware	7%
Linc	7%
Accel	6%
Informex	2%

- El siguiente cuadro presenta los resultados en cuanto al uso de metodologías:

Metodología	No la conoce	No la usa	Poco uso	Uso frecuente	Uso extensivo
De-Marco	64%	29%	7%	–	–
Gane-Sarson	64%	–	14%	22%	–
Jackson	93%	7%	–	–	–
Warnier-Orr	93%	7%	–	–	–
Yourd-Const	93%	7%	–	–	–

- Razones dadas para el no uso de metodologías. Se da el porcentaje de empresas que dio esa razón:

1) Desconocimiento de las metodologías 24%
2) Al usar una metodología completa el costo en tiempo es muy alto y esto no es permitido por las directivas 18%
3) Poco entrenamiento en el uso de metodologías 15%
4) Se cree poco en los beneficios obtenidos por el uso de metodologías 12%
5) Son muy engorrosas 10%
6) Se tiene éxito sin el uso de metodologías 7%
7) No existe la infraestructura adecuada para usarlas 7%
8) Hace falta vender la idea . . . 7%.

- Principales necesidades manifestadas:

1) Establecer una metodología para todo el ciclo de vida del proyecto.
2) Pasar a cuarta generación.
3) Aumentar y mejorar el uso de prototipos.
4) Se necesitan técnicas para recoger la información.

VI. CONCLUSIONES

Estos departamentos de sistemas tienen una década de experiencia en cuanto al desarrollo de software y podríamos decir que están en su etapa de adolescencia. Esta es la causa de algunos de los resultados obtenidos, como por ejemplo:

- Muchos departamentos de sistemas nacieron como parte de administración y finanzas y continúan allí.
- Un alto porcentaje de los jefes de departamento no tienen formación académica en sistemas.
- Las personas que desarrollan software han tenido muy poco entrenamiento formal en las metodologías estructuradas. Cada individuo enfoca su tarea con la experiencia obtenida en trabajos anteriores. Algunas personas aplican un método ordenado y eficiente de desarrollo mediante prueba y error, pero muchos otros desarrollan malos hábitos que dan como resultado una calidad pobre.

Es interesante observar los resultados de Carma McClure[6] sobre las metodologías más utilizadas en los Estados Unidos:

Gane - Sarson	13.8%
De Marco	12.3%
Yourdon - Constantine	29.1%
Warnier - Orr	7.2%
Jackson	2.4%

- Las razones dadas para el no uso de metodologías y las principales necesidades manifestadas también apoyan la conclusión acerca de la juventud de estos departamentos de sistemas.
- Todavía quedan algunos departamentos que usan el modelo de ciclo de vida sin planificación.

  Se usa más el modelo con prototipos que el modelo de ciclo de vida clásico, debido a que este último exige al usuario establecer explícitamente al principio todos los requerimientos y luego tener paciencia; puesto que no se dispone de una versión funcionando del programa hasta las etapas finales del desarrollo del proyecto. Un error importante no detectado, al comienzo, puede ser fatal.

  La construcción de prototipos es un método eficiente, ya que exige una mayor comunicación usuario-analista, pero presenta el inconveniente de compromiso de implementación para obtener un prototipo que funcione rápidamente. Pueden utilizarse partes inapropiadas en la construcción y después de un cierto tiempo de familiarización con estas partes, se olvidan las razones por las que eran ina-

propiadas. La elección menos correcta forma entonces parte integral del sistema y esto puede aumentar mucho la fase de mantenimiento. Esto sucede en muchas empresas.

- El hecho de que el 35% de las aplicaciones se han desarrollado con herramientas de cuarta generación se debe a que cerca del 40% de las empresas encuestadas ya han dado el paso a cuarta generación y sus nuevas aplicaciones las están desarrollando con estas herramientas.

Sería muy interesante poder medir el costo promedio de sistemas en varias actividades económicas. Este factor puede convertirse en un punto de referencia para las empresas. Muchas veces se escucha decir a un jefe de sistemas, que no tiene un punto de referencia para medirse o para justificar la compra de hardware o software.

Por último, podemos concluir que este tipo de investigaciones también dan un marco referencial a las facultades de sistemas de las universidades de nuestra región.

VII. BIBLIOGRAFIA

1. Pressman R.S., *Ingeniería del software: Un enfoque práctico*, McGraw-Hill, Madrid, 1988.
2. DeMarco T., *Structured analysis and system specification*, Prentice Hall, 1979.
3. Gane T. and Sarson C., *Análisis estructurado de sistemas*, El Ateneo, Buenos Aires, 1990.
4. Yourdon E., *Modern structured analysis*, Prentice Hall, New York, 1989.
5. Chen P., *The entity-relationship model*, Toward a unifying view of data, ACM Trans. on Data Base Systems, Vol. 1, Nº 1, March 1976, pp. 9- 36.
6. McClure C., *Software news case 1987 survey*.
7. Jackson M., *System development*, Prentice-Hall, 1983.
8. Orr K., *Structured requirements definition*, Ken Orr & Associates, Inc., Topeka, KS, 1981.
9. Yourdon E. and Constantine L., *Structured design*, Yourdon Press, 1978.
10. Martin J. and McClure C., *Structured techniques: The basis for CASE*, Prentice-Hall, 1988.