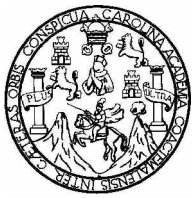


Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

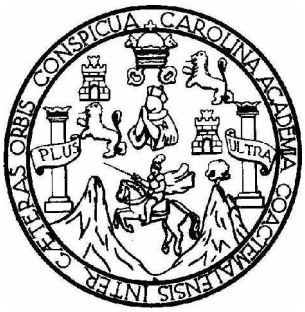
ANÁLISIS COMPARATIVO ENTRE EL PENSUM DE PROGRAMACIÓN DE COMPUTADORAS DE LA CARRERA DE BACHILLERES EN COMPUTACIÓN Y LAS NECESIDADES DE LAS EMPRESAS DESARROLLADORAS DE SOFTWARE EN EL ÁREA METROPOLITANA DE GUATEMALA

Erik Arnulfo Santizo Bardales

Asesorado por el Ing. Edgar Estuardo Santos Sutuj

Guatemala, julio de 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



FACULTAD DE INGENIERÍA

ANÁLISIS COMPARATIVO ENTRE EL PENSUM DE PROGRAMACIÓN DE COMPUTADORAS DE LA CARRERA DE BACHILLERES EN COMPUTACIÓN Y LAS NECESIDADES DE LAS EMPRESAS DESARROLLADORAS DE SOFTWARE EN EL ÁREA METROPOLITANA DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR:

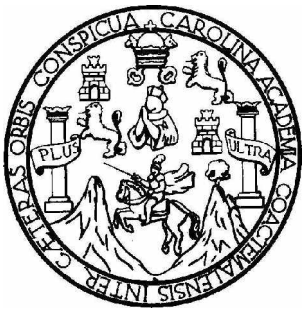
ERIK ARNULFO SANTIZO BARDALES
ASESORADO POR EL ING. EDGAR ESTUARDO SANTOS SUTUJ
AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, JULIO DE 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA



NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paíz Recinos
VOCAL I	Inga. Glenda Patrícia García Soria
VOCAL II	Inga. Alba Maritza Guerrero de López
VOCAL III	Ing. Miguel Angel Dávila Calderón
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paíz Recinos
EXAMINADOR	Ing. Freiry Javier Gramajo López
EXAMINADOR	Ing. Edgar René Ornelyz Hoil
EXAMINADOR	Ing. César Augusto Fernández Cáceres
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

Análisis comparativo entre el pensum de programación de computadoras de la carrera de bachilleres en computación y las necesidades de las empresas desarrolladoras de software en el área metropolitana de Guatemala,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas el 1 de septiembre de 2005.

Erik Arnulfo Santizo Bardales

ACTO QUE DEDICO A:

Dios, por ser la fuerza que me mantiene día a día en todos los proyectos que emprendo.

María Auxiliadora, por guiarme en el arduo camino de la vida.

San Juan Bosco, porque a través de sus enseñanzas aprendí a luchar por los sueños.

Mis padres, Arnulfo y Angela, por su cariño, apoyo, confianza y amor en todos los momentos de mi vida.

Mis hermanos, Gaby y Jorge Luis, porque sin su ayuda y cariño no sería la persona que soy ahora.

Mi esposa, Myrna Lisseth, por ser la luz y la alegría que ilumina y acompaña mi camino, animándome a alcanzar mis sueños más altos.

Mi familia, que me ha brindado su cariño y sabiduría durante mi vida.

Mis amigos, con los que he compartido los muy diversos momentos de la carrera, que de una u otra forma me ayudaron en mi crecimiento personal.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Abstracto Es un adjetivo que indica una cualidad con exclusión de sujeto.

Browser Un navegador Web o Web browser es una aplicación software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores Web de todo el mundo a través de Internet.

Código fuente El código fuente es un texto escrito, generalmente por una persona, que se utiliza como base para generar otro código con un compilador o intérprete para ser ejecutado por una computadora.

Código objeto Se llama código objeto en programación al código resultante de la compilación del código fuente, por lo general está codificado en código de máquina y distribuido en varios archivos resultantes de la compilación de cada archivo de código fuente.

Compilación Compilación es el proceso por el cual se traducen programas en código fuente a programas en código objeto.

Compilador	Un compilador acepta programas escritos en un lenguaje de alto nivel y los traduce a otro lenguaje, generando un programa equivalente independiente, que puede ejecutarse tantas veces como se quiera.
Hardware	Se denomina hardware o soporte físico al conjunto de elementos materiales que componen un ordenador.
Lenguaje de máquina	Lenguaje de máquina es el sistema de códigos directamente interpretable por un circuito micro-programable, como el microprocesador de un ordenador o el microcontrolador de un autómata.
Mnemónico	En Informática un 'Mnemónico es una palabra que sustituye a un código de operación -lenguaje de máquina- con lo cual resulta más fácil la programación, es de aquí de donde resulta el concepto de lenguaje ensamblador.
Ordenador	Una computadora -Hispanoamérica- u ordenador -España- es un dispositivo electrónico compuesto básicamente de un procesador, una memoria y los dispositivos de entrada/salida (E/S).
Programador	Un programador es una persona que se dedica a la programación o que escribe programas. También, puede decirse que es la persona que traduce algoritmos a un lenguaje que pueda entender una computadora.

Reglas sintácticas	Reglas que gobiernan la estructura de un lenguaje o herramienta de programación.
Reglas semánticas	Reglas que se ocupan de verificar que los componentes que se colocan en un lenguaje o herramienta de programación tengan un significado lógico.
Software	Es la parte lógica del ordenador, esto es, el conjunto de programas que puede ejecutar el hardware para la realización de las tareas de computación a las que se destina.
TI	Tecnologías de la información
Usuario	Es la persona, organización u otra entidad que depende de los servicios de un computador o sistema computacional para obtener un resultado deseado.
Web	La World Wide Web -del inglés, Telaraña Mundial- la Web o WWW, es un sistema de hipertexto que funciona sobre Internet. Para ver la información se utiliza una aplicación llamada navegador Web para extraer elementos de información -llamados "documentos" o "páginas Web"- de los servidores Web o "sitios" y mostrarlos en la pantalla del usuario.

RESUMEN

Análisis comparativo entre el pensum de programación de computadoras de la carrera de bachilleres en computación y las necesidades de las empresas desarrolladoras de software en el área metropolitana de Guatemala es un estudio comparativo entre las herramientas utilizadas por los bachilleres en computación y las empresas agremiadas a SOFEX.

Se presenta un listado de las herramientas utilizadas en las instituciones educativas, las herramientas utilizadas por las empresas y su relación entre ellas.

El estudio reveló una gran separación entre los dos sectores, ya que, su relación se limita a las herramientas, Visual Basic, la cual es de las más utilizadas por empresas de SOFEX, y, Delphi y Visual Fox, siendo estas de las menos utilizadas para el desarrollo de sistemas o programas.

Como resultado de este trabajo se presentan algunas recomendaciones para mejorar la relación entre los dos sectores involucrados en este trabajo.

OBJETIVOS

General

Presentar un estudio de las herramientas de programación impartidos a estudiantes de la carrera de bachillerato en computación como parte de su pensum de estudios, y determinar si estas herramientas ayudan a los nuevos bachilleres a desarrollarse como programadores en las empresas agremiadas a SOFEX.

Específicos

1. Investigar las herramientas impartidas, actualmente, a los bachilleres en computación como parte de su pensum.

2. Investigar las herramientas utilizadas, actualmente, por las empresas que conforman SOFEX en el desarrollo de software.

3. Determinar si las herramientas enseñadas a los alumnos de bachillerato, son las mismas que las utilizadas por SOFEX.

4. Utilizar estadística descriptiva para mostrar si las herramientas impartidas son las utilizadas en SOFEX.

5. Determinar si las herramientas impartidas se ajustan a las necesidades de SOFEX.

INTRODUCCIÓN

En el presente trabajo de investigación, se ha hecho un estudio comparativo de los programas o herramientas que aprenden los bachilleres en computación y aquellos que son utilizados por las empresas agremiadas a SOFEX, con el objetivo de verificar si el contenido del pensum de estudio de los bachilleres les provee las herramientas necesarias para desarrollarse como programadores.

Se ha escogido a la gremial de exportadores de software, SOFEX, por ser un grupo de empresas vanguardistas e innovadoras de Guatemala en este ámbito.

Se eligieron a los bachilleres en computación como muestra, debido a que son ellos los que reciben una educación estrictamente enfocada a la programación, a diferencia de los técnicos y peritos en computación que se especializan en hardware.

La información se recolectó por medio de encuestas telefónicas en ambos grupos y se analizó utilizando estadística descriptiva.

Los resultados muestran que los estudiantes se especializan en programas como: Pascal, Visual Basic y DELPHI; mientras que en las empresas los programadores necesitan manejar principalmente: Java, Visual Basic y .NET.

Los resultados anteriores indican que estos dos sectores coinciden, únicamente, en la herramienta de Visual Basic y que es necesario instruir a los nuevos programadores en el uso de herramientas más actuales.

Al final de este documento, se hacen algunas recomendaciones con el objetivo de graduar mejores programadores que puedan ser una parte importante dentro del desarrollo del país.

1. PROGRAMACIÓN DE COMPUTADORAS

1.1 Computadora

Una computadora u ordenador es un sistema digital con tecnología microelectrónica capaz de procesar información a partir de un grupo de instrucciones denominado programa. La estructura básica de una computadora incluye microprocesador (CPU), memoria y dispositivos de entrada / salida (E / S), junto a los buses que permiten la comunicación entre ellos.

La característica principal que la distingue de otros dispositivos similares, como una calculadora no programable, es que puede realizar tareas muy diversas cargando distintos programas en la memoria para que los ejecute el procesador.

1.2 ¿Cómo funcionan las computadoras?

La evolución de las computadoras ha llevado mucho tiempo, si pensamos desde el tiempo del ábaco, pero su desarrollo en los últimos 25 años ha sido increíblemente grande. La computadora se ha convertido en una herramienta de suma utilidad para las personas hoy en día, pero la mayoría realiza el mismo procedimiento, la enciende, trabaja y la apaga sin tomar en cuenta todas las actividades que realiza para llegar a este punto. Una descripción general de este proceso sería:
Un circuito de control llamado BIOS (basic, input, output, system = sistema básico de entrada y salida), inicia una inspección del sistema. Dicha secuencia se denomina en el argot computacional POST (POWER ON SELT TEST).

El propósito del examen es confirmar la existencia y buen funcionamiento de algunos componentes vitales de las computadoras: Microprocesador, memoria Ram, Bios, dispositivos básicos de comunicación (como el teclado y el Floppy) y señal de video. Cuando uno de estos elementos tiene algún problema (esta desconectado o en corto), el BIOS utiliza una serie de mensajes - según su fabricante - para informar que "x" elemento tiene una anomalía.

Pasado el POST, el Bios tiene como segunda misión la búsqueda de un Sistema de control que debe cargar en la memoria. Este es el Sistema operativo (Unix, Linux, Windows, etc.). Para hacerlo debe buscar en las unidades de disco existentes en el PC (Floppy, Disco duro, Unidades ópticas o CD-ROM, red, etc.). Si lo encuentra, lee sus instrucciones y deposita un conjunto básico de instrucciones en la memoria Ram para que desde allí el Sistema Operativo continúe con el control de la computadora. Si no se encuentra el sistema operativo, el BIOS lanza un mensaje anunciando que se necesita colocar en el PC un disquete de arranque.

Terminada la carga del sistema operativo, el PC puede trabajar con distintos programas. El sistema operativo coordina entonces con los programas, el control de la máquina para ejecutar tareas especificas. Ambos, sistema operativo y programas utilizan la memoria RAM como lugar de operaciones de datos, grabando y borrando en operaciones sucesivas la información resultante. Como elemento de verificación visual, el monitor del PC presenta en su pantalla los resultados de lo que ocurre en la memoria de la computadora.

Todo el movimiento generado (transmisión y ordenamiento de datos) es dirigido por el cerebro del sistema, el microprocesador. Este a su vez utiliza toda una red de subalternos (otros chips) para ordenar la transmisión de señales que se necesitan para que el PC funcione. Al fin cuando la ejecución de un trabajo es aprobada por el operador de la computadora, se ordena al programa que guarde los datos en su forma definitiva.

La acción de guardar se hace depositando la información en dispositivos como: disco duro, CD, cinta magnética, otro PC o un disquete. La grabación es seguida del retiro de la información procesada, de la memoria RAM, es decir que esta queda libre (vacía) para iniciar otro proceso con el mismo programa o con otro. Cuando la computadora se apaga, todo dato o señal eléctrica existente en la memoria RAM, se pierde.

1.3 Programas y algoritmos

Un algoritmo es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema. Un programa normalmente implementa (traduce a un lenguaje de programación concreto) un algoritmo.

Los programas suelen subdividirse en partes menores (módulos), de modo que la complejidad algorítmica de cada una de las partes sea menor que la del programa completo, lo cual ayuda al desarrollo del programa.

Según Niklaus Wirth un programa está formado por algoritmos y estructura de datos.

Se han propuesto diversas técnicas de programación, cuyo objetivo es mejorar tanto el proceso de creación de software como su mantenimiento. Entre ellas se pueden mencionar las programaciones lineal, estructurada, modular y orientada a objetos.

1.4 Compilación

El programa escrito en un lenguaje de programación (comprensible por el ser humano) no es inmediatamente ejecutado en una computadora. La opción más común es compilar el programa, aunque también puede ser ejecutado mediante un intérprete informático (HTML).

El código fuente del programa se debe someter a un proceso de transformación para convertirse en lenguaje máquina, interpretable por el procesador. A este proceso se le llama compilación.

Normalmente la creación de un programa ejecutable conlleva dos pasos. El primer paso se llama compilación (propiamente dicho) y traduce el código fuente escrito en un lenguaje de programación almacenado en un archivo a código en bajo nivel, (normalmente en código objeto no directamente al lenguaje máquina). El segundo paso se llama enlazado (del inglés link o linker) se junta el código de bajo nivel generado de todos los ficheros que se han mandado compilar y se añade el código de las funciones que hay el las bibliotecas del compilador para que el ejecutable pueda comunicarse con el sistemas operativo y traduce el código objeto a código máquina.

Estos dos pasos se pueden mandar hacer por separado, almacenando el resultado de la fase de compilación en archivos objetos, para enlazarlos posteriormente, o crear directamente el ejecutable con lo que la fase de compilación se almacena sólo temporalmente.

Un programa podría tener partes escritas en varios lenguajes (generalmente C, C++ y Asm), que se podrían compilar de forma independiente y enlazar juntas para formar un único ejecutable.

1.5 Programación e ingeniería del software

Existe una tendencia a identificar el proceso de creación de un programa informático con la programación, que es cierta cuando se trata de programas pequeños para uso personal, y que dista de la realidad cuando se trata de grandes proyectos.

El proceso de creación de software desde el punto de vista de la Ingeniería tiene los siguientes pasos:

- Reconocer la necesidad de un programa para solucionar un problema ó identificar la posibilidad de automatización de una tarea.
- Recoger los requisitos del programa. Debe quedar claro qué es lo que debe hacer el programa y para qué se necesita.

- Realizar el análisis de los requisitos del programa. Debe quedar claro cómo debe realizar el programa las cosas que debe hacer. Las pruebas que comprueben la validez del programa se pueden especificar en esta fase.
- Diseñar la arquitectura del programa. Se debe descomponer el programa en partes de complejidad abordable.
- Implementar el programa. Consiste en realizar un diseño detallado, especificando completamente todo el funcionamiento del programa, tras lo cual la codificación debería resultar inmediata.
- Instalar el programa. Consiste en poner el programa en funcionamiento junto con los componentes que pueda necesitar (bases de datos, redes de comunicaciones, etc.)

La Ingeniería del Software se centra en los pasos de planificación y diseño del programa, mientras que antiguamente (programación artesanal) la realización de un programa consistía únicamente en escribir el código.

2. DEFINICIÓN DE LENGUAJES

2.1 Definición

El lenguaje es la capacidad del ser humano para comunicarse mediante un sistema de signos o lengua para ello. No se debe confundir con lengua o idioma, que es la representación de dicha capacidad.

Los lenguajes son, explicados de una manera fácil, aunque reduciendo sus alcances e importancia para la formación de nuestro mundo, formas de representar cosas.
La mayoría de las veces el término se refiere a los lenguajes que los humanos utilizan para comunicarse, es decir las lenguas naturales, ya sea lenguaje hablado, lenguaje de signos o el empleado en la literatura. El lenguaje natural incluye todas las comunicaciones animales, incluyendo el lenguaje humano. En las matemáticas y en la informática, por ejemplo, los lenguajes artificiales son llamados lenguajes formales (incluyendo lenguajes de programación).

2.2 Lenguaje Natural

Se utiliza el término lenguaje natural para referirse principalmente al lenguaje humano. El término se utiliza en contraposición a los lenguajes formales, lenguajes artificiales como el Esperanto que han sido creados por el hombre con un fin explícito, como por ejemplo los lenguajes de programación, el lenguaje matemático o lógico.

La lengua natural evoluciona enmarcada por una cultura de hablantes nativos que utilizan dicha lengua con una finalidad comunicativa. De esta forma, se distingue entre lenguas tales como el chino mandarín, el español y el inglés, las cuales son lenguas naturales; y el esperanto que se le denomina lengua planificada.

2.3 Lenguaje de Programación

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un programa informático. Un lenguaje de programación permite a un programador especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos y, qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural.

Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, es decir, ser traducido al lenguaje de máquina, o ser interpretado para que pueda ser ejecutado por el ordenador.

2.3.1 Historia

Los lenguajes de programación se han desarrollado desde la aparición de la primera computadora, a continuación de destacan algunos de los lenguajes más importantes en la historia.

2.3.1.1 FORTRAN

El primer compilador de FORTRAN se desarrolló para una IBM 704 entre 1954 y 1957 por la empresa IBM, por un grupo liderado por John W. Backus.

En la época se consideró imprescindible que los programas escritos en FORTRAN corrieran a velocidad comparable a la del lenguaje ensamblador; de otra forma, nadie lo tomaría en cuenta.

El lenguaje ha sido ampliamente adoptado por la comunidad científica para escribir aplicaciones con cómputos intensivos. La inclusión en el lenguaje de la aritmética de números complejos amplió la gama de aplicaciones para las cuales el lenguaje se adapta especialmente y muchas técnicas de compilación de lenguajes han sido creadas para mejorar la calidad del código generado por los compiladores de Fortran.

2.3.1.2 LISP

El elemento fundamental en Lisp es la lista, en el sentido más amplio del término, pues tanto los datos como los programas son listas. De ahí viene su nombre, pues Lisp es un acrónimo de "LIStProcessing". Los lenguajes de este tipo se llaman "aplicativos" o "funcionales", porque se basan en la aplicación de funciones a sus datos.

En Lisp se distinguen dos tipos fundamentales de elementos:

Átomos: son datos elementales y pueden pertenecer a varios tipos: números, caracteres, cadenas de caracteres y símbolos.

Listas: son secuencias de átomos o de listas encerradas entre paréntesis. Además, existe una lista especial, "nil", que es la lista nula, que no tiene ningún elemento.

En Lisp, una función se expresa como una lista. Algunas de las funciones predefinidas de Lisp tienen símbolos familiares (+ para la suma, * para el producto), pero otras son más exóticas, especialmente dos que sirven precisamente para manipular listas, descomponiéndolas en sus componentes. Sus nombres ("car" y "cdr") son un poco extraños, reliquias de tiempos pasados y de la estructura de los ordenadores de segunda generación, "car" devuelve la cabeza de una lista y "cdr" su cola o resto.

Lisp sigue una filosofía de tratamiento no-destructivo de los parámetros, de modo que la mayoría de las funciones devuelven un lista resultado de efectuar alguna transformación sobre la que recibieron, pero sin alterar esta última.

Uno de los motivos por los que Lisp es especialmente adecuado para la IA es el hecho de que el código y los datos tengan el mismo tratamiento (como listas); esto hace especialmente sencillo escribir programas capaces de escribir otros programas según las circunstancias.

2.3.1.3 COBOL

El lenguaje COBOL, acrónimo de "Common Business Oriented Language" nació del deseo de crear un lenguaje de programación "universal", que pudiera ser usado en cualquier ordenador (en los años 1960 existían numerosos modelos de ordenadores incompatibles entre sí) y que estuviera orientado principalmente a los negocios, es decir, a la llamada informática de gestión.

En la creación de este lenguaje participó la comisión CODASYL, compuesta por fabricantes de ordenadores, usuarios y el departamento de defensa de Estados Unidos de América. El primer prototipo del lenguaje salió a la luz en 1960 denominándose COBOL-60. Gracias a la ayuda de los usuarios evolucionó rápidamente y fue revisado de 1961 a 1965.

Fue en 1968 cuando salió la primera versión ANSI del lenguaje, siendo revisada posteriormente en 1974 (COBOL ANS-74), 1985 (COBOL ANS-85), y en 2002 (COBOL ANS-2002); se prepara otra revisión para el año 2008.

COBOL fue un lenguaje preponderante durante décadas, pero actualmente se usa básicamente en entidades bancarias y otros sistemas informáticos añejos, en los que se utiliza principalmente para mantener el código existente.

2.3.1.4 ALGOL

Se denomina Algol a un lenguaje de programación. La voz es un acrónimo de las palabras inglesas Algorithmic Language (lenguaje algorítmico).

Fue muy popular en las universidades durante los años 60s, pero no llegó a cuajar como lenguaje de utilización comercial.

Sin embargo, Algol influyó profundamente en varios lenguajes posteriores que sí alcanzaron gran difusión, como Pascal, C y Ada.

Hacia 1965 dos corrientes se distinguieron sobre el tema de un sucesor para Algol. Como resultado se definieron los lenguajes Algol W que es un lenguaje minimalista, rápidamente implementado y distribuido y, por otra parte, Algol 68 que para la época está en la frontera entre un lenguaje para programar en él y un lenguaje para investigar sobre él.

2.3.1.5 APL

A Programing Language. Es un intérprete, desarrollado por IBM a finales de los años 60. Fue definido por Kenneth Iverson. Es un lenguaje muy conciso, con una sintaxis muy sencilla. Está orientado a trabajos con matrices, con la que se pueden hacer todo tipo de operaciones lógicas, aritméticas, incluso se pueden inventar las operaciones que se quieren hacer con las matrices.

Es de una potencia tremenda. Una sola sentencia puede traducirse en miles de ellas en otros lenguajes, como por ejemplo Fortran.

Un ejemplo, un lenguaje de simulación de circuitos, SIAL desarrollado por D. Manuel Alfonseca ocupaba cerca de 25 000 sentencias en Fortran-Assembler y en APL todo el programa cabía en dos folios.

A pesar de ser un lenguaje de tan alto nivel también es capaz de manipular a escala de bits.

Tiene la propiedad de que desde una rutina se puede reescribir otra, lo que lo hace muy apropiado para la fabricación de compiladores.

Sus problemas radican en que:

- Necesita teclado especial para poner los operadores lógicos y simbólicos.
- Es tan potente que es difícil de documentar.

2.3.1.6 SIMULA

Es el primer lenguaje de programación orientada a objetos (OOP) que luego de varios años de su desarrollo, casi todos los modernos lenguajes comenzaron a utilizar sus principios de orientación a objetos. Así fue como se popularizaron términos como clases, objetos, instancias, herencia, polimorfismo, etc. Por su parte Simula 67 fue oficialmente lanzado por sus autores Ole Johan Dahl y Kristen Nygaard en la Conferencia de Trabajo en Lenguajes de Simulación IFIO TC 2, en Lysebu cerca de Oslo en mayo de 1967

2.3.1.7 PL/1

Acrónimo de Programming Language 1 (Lenguage de Programación 1), fue propuesto por IBM hacia 1970 para responder simultáneamente a las necesidades de las aplicaciones científicas y comerciales, disponible en las novedosas plataformas de utilidad general IBM 360 y más adelante IBM 370.

Este lenguaje tenía muchas de las características que más adelante adoptaría el lenguaje C y algunas de C++. Desafortunadamente, IBM registra el nombre del lenguaje como forma de mantener control sobre su desarrollo, lo que disuadió a otras empresas de dar ese nombre a sus implementaciones. No siendo posible encontrar un único lenguaje para diversas plataformas, los potenciales usuarios del lenguaje prefirieron no adoptarlo a pesar de sus múltiples innovaciones, que incluían multiprocesamiento, recursión, estructuras de control modernas, facilidades para la puesta a punto, asignación dinámica de espacio para estructuras de datos, procedimientos genéricos, etc.

Sin embargo, dentro de los usuarios de IBM, el lenguaje se utilizó con bastante intensidad, y el proyecto Multics utilizó PL/1 como lenguaje de desarrollo para su sistema de operación.

PL/1 fue probablemente el primer lenguaje comercial cuyo compilador estaba escrito en el lenguaje que compilaba.

2.3.1.8 BASIC

Originalmente fue inventado en 1964 por John George Kemeny (1926-1993) y Thomas Eugene Kurtz (1928-) en el Dartmouth College. En los años subsiguientes, mientras que otros dialectos de BASIC aparecían, el BASIC original de Kemeny y Kurtz era conocido como BASIC Dartmouth.

BASIC fue diseñado para permitir a los estudiantes escribir programas usando terminales de computador de tiempo compartido. Las siglas significan ***B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode*. BASIC estaba intencionado para facilitar los problemas de complejidad de los lenguajes anteriores, con un nuevo lenguaje diseñado específicamente para la clase de usuarios que los sistemas de tiempo compartido permitían: un usuario más sencillo, a quien no le interesaba tanto la velocidad, sino el hecho de ser capaz de usar la máquina. Los diseñadores del lenguaje también querían que permaneciera en el dominio público, lo que contribuyó a que se diseminara.

Los ocho principios de diseño de BASIC fueron:

- Ser fácil de usar para los principiantes.
- Ser un lenguaje de propósito general.
- Permitir que los expertos añadieran características avanzadas, mientras que el lenguaje permanecía simple para los principiantes.
- Ser interactivo.
- Proveer mensajes de error claros y amigables.
- Responder rápido a los programas pequeños.
- No requerir un conocimiento del hardware de la computadora.
- Proteger al usuario del sistema operativo.

2.3.1.9 ISWIM

Es una notación algorítmica en el estilo de un lenguaje de programación diseñada por Peter J. Landin y descrita por primera vez en su artículo, Los próximos 700 lenguajes de programación, publicado en la revista Communications of the ACM, en 1966. El nombre del lenguaje es el acrónimo de la frase en inglés "If you See What I Mean".

Si bien nunca fue implementado, su influencia fue decisiva en el desarrollo de la programación funcional y se pueden contar los lenguajes SASL, Miranda y ML como sus sucesores más directos.

2.3.1.10 CPL

Es un acrónimo inglés de Basic Combined Programming Language (Lenguaje de Programación Básico Combinado). Fue diseñado por Martin Richards de la Universidad de Cambridge en 1966 debido a las dificultades experimentadas con el lenguaje de programación CPL durante los años 60. El primer compilador implementado fue escrito en 1967 mientras Richards visitaba el MIT. El lenguaje fue descrito por primera vez en un proyecto presentado en una conferencia informática en 1969. Años después, Dennis Ritchie lo utilizó como base para desarrollar C.

Es un lenguaje de programación ordenado, potente y muy fácil de adaptar a diferentes arquitecturas. Se popularizó en los programas de arranque de las computadoras (bootstraps en inglés) debido a sus compiladores simples y compactos, algunos con capacidad para correr en sólo 16 kilobytes. Inclusive algunos sistemas operativos fueron escritos total o parcialemente en BCPL (TRIPOS y Amiga Kickstart entre otros).

La principal razón de la capacidad de adaptación a las diferentes arquitecturas es la estructura de su compilador, el que fue dividido en dos partes. La cara visible del mismo interpretaba el código fuente y generaba código máquina para una máquina virtual; la otra cara del compilador tomaba dicho código máquina y lo traducía al código necesario para la arquitectura deseada.

No mucho después, este diseño de compiladores se hizo popular; pero el compilador de Richards fue el primero en definir una máquina virtual para este propósito. Algunos de los lenguajes que utilizan el mismo formato son Java y Pascal.

El lenguaje tiene la peculiaridad de tener sólo un tipo de dato: la palabra (word en inglés) compuesta de una cantidad fija de bits elegidos generalmente para coincidir con el tamaño de una palabra de la arquitectura correspondiente. La interpretación de cualquier valor es determinado por los operadores utilizados para procesarlos los utilizaba como enteros, como punteros, etc. Para poder lograr esto, la implementación carecía de un chequeo de tipos. Para evitar estos errores, entonces, se desarrolló la notación húngara.

2.3.1.11 FORTH

Es un lenguaje de programación interactivo atípico, inventado (aunque su autor y seguidores prefieren decir descubierto) por Charles H. Moore en los años 1960. Una de sus importantes características es la utilización de una pila de datos para pasar los argumentos entre las palabras, que son los constituyentes de un programa en Forth.

2.3.1.12 LOGO

Es uno de los pocos lenguajes de programación con instrucciones en español. Logo es un lenguaje de alto nivel en parte funcional en parte estructurado, de muy fácil aprendizaje (por lo cual, generalmente es el lenguaje de programación preferido para trabajar con niños y jóvenes).

Existen varios intérpretes de Logo en español (y más de 100 en total), entre ellos: LogoWriter, WinLogo, Logo Gráfico, XLogo, MSWLogo y LogoEs. XLogo, MSWLogo y LogoES tienen la particularidad de ser además Software Libre.

2.3.1.13 PASCAL

Es un lenguaje de programación de alto nivel y propósito general, desarrollado por Niklaus Wirth, profesor del Instituto tecnológico de Zurich, Suiza. Lo creó pensando en un lenguaje didáctico para el aprendizaje de técnicas de programación, "una disciplina sistemática basada en determinados conceptos fundamentales". Estos conceptos más tarde se tornarían motivo de controversia entre los que creen que este lenguaje tiene utilidad limitada a los medios académicos, como Brian W. Kernighan. Con el tiempo se ha convertido además en un estándar de los lenguajes de programación más usados.

La primera versión preliminar apareció en 1968 y el primer compilador aparece a finales de 1970.

A partir de los años setenta se convirtió en el sucesor de ALGOL en el entorno universitario. Pascal permite construir programas muy legibles. Wirth es también autor del lenguaje Modula-2 y de Oberon.

El lenguaje de programación Delphi es una versión orientada a objetos moderna del lenguaje Pascal y es ampliamente utilizada en la industria de software.

En la ciencia de la computación, el lenguaje de programación estructurado Pascal es uno de los hitos en los lenguajes de programación y esta todavía vigente hoy en día.

2.3.1.14 SMALLTALK

Es un lenguaje orientado a objetos diseñado por Alan Kay durante los años setenta en el Palo Alto Research Institute de Xerox (conocido como Xerox Parc). Smalltalk era un elemento de la primera computadora personal con interfaz gráfica, el "Alto" (que nunca fue llevado al mercado pero de cuyas ideas nació el Macintosh).

Smalltalk es considerado el primero de los lenguajes orientados a objetos (OOP). En Smalltalk TODO es un objeto, incluso los números enteros. Smalltalk se basó en ideas de Simula (un lenguaje de simulaciones).

Pero Smalltalk no es meramente un lenguaje, sino un entorno completo, prácticamente un sistema operativo que se ejecuta encima de una "máquina virtual". Esto asegura su máxima portabilidad entre plataformas.

A pesar de ser un lenguaje muy simple, muy poderoso, y que promueve buenas prácticas de programación, Smalltalk (hasta el 2002) no había llegado a ser un lenguaje muy popular. Esto se debe a la poca aceptación de lenguajes interpretados en los años 1980 y 1990. A pesar de esto algunas empresas como Digitalk, Visual Works e IBM llegaron a tener relativo éxito con la plataforma como una herramienta de desarrollo rápido que competía contra herramientas como Power Builder y en menor grado contra Visual Basic que eran muy populares en los años 80 y 90.

En 1995, Sun Microsystems lanzó Java que es un lenguaje fuertemente influenciado por Smalltalk. y que popularizó el concepto de lenguajes interpretados con recolectores de basura.

2.3.1.15 C

Es un lenguaje de programación creado en 1969 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell basándose en los lenguajes BCPL y B. Al igual que sus dos predecesores, es un lenguaje orientado a la implementación de Sistemas Operativos (los sistemas operativos Linux y UNIX están escritos mayormente en C), pero se ha convertido en un lenguaje de propósito general de los más usados.

Se trata de un lenguaje no fuertemente tipado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Un ejemplo es la posibilidad de mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos. Destaca su gran riqueza de operadores y expresiones.

Existe un estándar ISO de 1986 denominado ANSI C. En teoría, un lenguaje 100% ANSI C sería portable entre plataformas y/o arquitecturas pero en la práctica esto no es siempre cierto.

2.3.1.16 PROLOG

Proveniente del francés Programation et Logique, es un lenguaje de programación bastante popular en el medio de investigación en Inteligencia Artificial.

Se trata de un lenguaje de programación ideado a principios de los años 70 en la universidad de Aix-Marseille por los profesores Alain Colmerauer y Phillipe Roussel. Inicialmente se trataba de un lenguaje totalmente interpretado hasta que, a mediados de los 70, David H.D. Warren desarrolló un compilador capaz de traducir Prolog en un conjunto de instrucciones de una máquina abstracta denominada Warren Abstract Machine, o abreviadamente, WAM. Desde entonces Prolog es un lenguaje semi-interpretado.

Prolog se enmarca en el paradigma de los lenguajes declarativos, lo que lo diferencia enormemente de otros lenguajes más populares tales como Fortran, Pascal, C, etc.

En todos los mencionados, las instrucciones se ejecutan normalmente en orden secuencial, es decir, una a continuación de otra, en el mismo orden en que están escritas, que sólo varía cuando se alcanza una instrucción de control (un bucle, una instrucción condicional o una transferencia).

Los programas en Prolog se componen de cláusulas que constituyen reglas del tipo "modus ponens", es decir, "Si es verdad el antecendente, entonces es verdad el consecuente". No obstante, la forma de escribir las cláusulas es al contrario de lo habitual. Primero se escribe el consecuente y luego el antecedente. El antecedente puede ser una conjunción de condiciones que se denomina secuencia de objetivos. Cada objetivo se separa con una coma y puede considerarse similar a una instrucción o llamada a procedimiento de los lenguajes imperativos. En Prolog no existen instrucciones de control. Su ejecución se basa en dos conceptos: la unificación y el backtracking. Gracias a la unificación, cada objetivo determina un subconjunto de cláusulas susceptibles de ser ejecutadas. Cada una de ellas se denomina punto de elección. Prolog selecciona el primer punto de elección y sigue ejecutando el programa hasta determinar si el objetivo es verdadero o falso.

En caso de ser falso entra en juego el 'backtracking', que consiste en deshacer todo lo ejecutado situando el programa en el mismo estado en el que estaba justo antes de llegar al punto de elección. Entonces se toma el siguiente punto de elección que estaba pendiente y se repite de nuevo el proceso. Todos los objetivos terminan su ejecución bien en "verdadero", bien en "falso".

2.3.1.17 INTERCAL

Es un lenguaje de programación esotérico diseñado para ser extremadamente difícil de entender. Es una parodia de los lenguajes de programación FORTRAN y COBOL.

Fue creado por Don Woods y James Lyons, estudiantes de Princeton, en 1972. La versión actual, C-INTERCAL, es mantenido por Eric S. Raymond. Los autores originales dicen que INTERCAL significa "Compiled Language With No Pronounceable Acronym" ("Lenguaje compilado sin un acrónimo pronunciable").

Una de las características más peculiares de INTERCAL es que en lugar de tener la sentencia GOTO (desaconsejada por Edsger Dijkstra) tiene la instrucción COME FROM #, que indica que cuando se ha ejecutado la sentencia # se salta a la siguiente sentencia después de COME FROM #.

2.3.1.18 SCHEME

Es un lenguaje funcional (si bien impuro, ya que, por ejemplo, sus estructuras de datos no son inmutables) y un dialecto de Lisp. Fue desarrollado por Guy L. Steele y Gerald Jay Sussman el la década de los setenta e introducido en el mundo académico a través de una serie de artículos conocidos como los Lambda Papers de Sussman y Steele.

La filosofía de Scheme es decididamente minimalista. Su objetivo no es acumular un gran número de funcionalidades, sino evitar las debilidades y restricciones que hacen necesaria su adición.

Así, Scheme proporciona el mínimo número posible de nociones primitivas, construyendo todo lo demás en base a este reducido número de abstracciones. Por ejemplo, el mecanismo principal para el control de flujo son las llamadas recursivas finales.

Scheme fue el primer dialecto de Lisp que usó ámbito estático o léxico (en lugar de dinámico) de forma exclusiva. También fue uno de los primeros lenguajes de programación con continuaciones explícitas. Scheme ofrece también gestión automática de memoria (recolección de basura).

Las listas son la estructura de datos básica del lenguaje, que también ofrece arreglos entre sus tipos predefinidos. Debido a su especificación minimalista, no hay sintaxis explícita para crear registros o estructuras, o para programación orientada a objetos, pero muchas implementaciones ofrecen dichas funcionalidades.

2.3.1.19 BOURNE SHELL

Era la shell por defecto de la versión 7 de Unix, y sustituyó la Tompson shell, cuyo ejecutable tenía el mismo nombre: sh. Fue desarrollado por Stephen Bourne, de los Laboratorios Bell de AT&T, y vio la luz en la versión 7 de Unix distribuida a colegios y universidades. Todavía es una shell muy popular para cuentas Unix. En la mayoría de los sistemas Unix el programa binario de la Bourne shell o un programa compatible se encuentra en /bin/sh.

Los principales objetivos de la Bourne shell eran aprovechar dos características claves del kernel de la versión 7:

la lista de parámetros (argumentos) mucho más larga, limitada a 8192 bytes (anteriormente 127).

las variables de entorno. Éstas eran una nueva característica de la versión 7 y permitía pasar mucha información a los programas a través del arranque.

La Bourne shell también fue la primera en destacar la convención de usar el descriptor de archivo 2 para mensajes de error, permitiendo un control del programa mucho mayor durante la creación del script manteniendo los mensajes de error separados de la información.

2.3.1.20 MODULA-2

Lenguaje de programación cuyo autor es Niklaus Wirth, autor también del lenguaje Pascal.

Como novedad respecto a este último lenguaje, introduce el concepto de módulo, y de encapsulamiento. Del código contenido en un módulo, sólo se facilita una interfaz pública, permaneciendo el resto oculto para un desarrollador ajeno (encapsulado), lo que facilita el mantenimiento de dichas estructuras de programación.

Este concepto de módulo constituye el antecedente de las clases u objetos que se observan en el concepto moderno de Programación Orientada a Objetos (POO), sin embargo, la incapacidad de declarar múltiples instancias de los módulos, así como la ausencia de todo tipo de herencia, impiden decir que Modula-2 sea un lenguaje orientado a Objeto.

Modula-2 se utiliza principalmente en las universidades por su excelente adaptación a la enseñanza de un lenguaje estructurado.

2.3.1.21 DBASE

Fue el primer Sistema Gestor de Bases de Datos SGBDR, usado ampliamente para microcomputadoras , publicado por Ashton-Tate para CP/M, y más tarde para Apple II, Apple Macintosh e IBM PC bajo DOS donde se convirtió en uno de los títulos de software más vendidos por un buen número de años. dBASE nunca pudo superar exitosamente la transición a Microsoft Windows y terminó siendo desplazado por productos más nuevos como Paradox, Clipper, y FoxPro.

Incorporaba un lenguaje propio interpretado y requería un LAN PACK para funcionar sobre red local. La versión IV llegó finalmente en 1988.

dBASE fue vendido a Borland en 1991. Al poco tiempo promovió una casi intrascendente versión 5, de la que llegó a haber versión para Windows. Luego vendió los derechos de la línea de productos en 1999 a dataBased Intelligence, Inc. (dBI) que sigue comercializando nuevas versiones, llamadas dBASE Plus, desde 1999.

Durante la primera mitad de los '80s muchas otras compañías produjeron sus propios dialectos o variaciones del producto y lenguaje. Estos incluyeron FoxPro (ahora Visual FoxPro), Quick-Silver, Clipper, Xbase++, FlagShip, y Harbour. Todos ellos son llamados informalmente como xBase o XBase.

El formato subyacente de dBASE, el archivo dbf, es ampliamente utilizado en muchas otras aplicaciones que necesitan un formato simple para almacenar datos estructurados.

2.3.1.22 VISUAL BASIC

Es un lenguaje simple pensado para programadores inexpertos, guiado por eventos, y centrado en un motor de formularios poderoso que facilita el rápido desarrollo de aplicaciones gráficas. Su sintaxis, derivada del antiguo BASIC, ha sido ampliada con el tiempo al agregarse las características típicas de los lenguajes estructurados modernos. No requiere de manejo de punteros. Posee varias bibliotecas para manejo de bases de datos, pudiendo conectar con cualquier base de datos a través de ODBC (Informix, DBase, Acces, Mysql, SQL Server, etc) a través de ADO.

Es utilizado principalmente para aplicaciones de gestión de empresas, debido a la rapidez con la que puede hacerse un programa que utilice una base de datos sencilla, además de la abundancia de programadores en este lenguaje.

El compilador de Microsoft genera ejecutables que requieren una DLL para que sus ejecutables funcionen, en algunos casos llamada MSVBVMxy.DLL (acrónimo de "MicroSoft Visual Basic Virtual Machine x.y", siendo x.y la versión) y en otros VBRUNXXX.DLL ("Visual Basic Runtime X.XX"), que provee todas las funciones implementadas en el lenguaje. Además existen un gran número de bibliotecas (DLL) que facilitan el acceso a muchas funciones del sistema operativo y la integración con otras aplicaciones.

2.3.1.23 JAVA

El lenguaje mismo se inspira en la sintaxis de C++, pero su funcionamiento es más similar al de Smalltalk que a éste. Incorpora sincronización y manejo de tareas en el lenguaje mismo (similar a Ada) e incorpora interfaces como un mecanismo alternativo a la herencia múltiple de C++.

2.4 Generaciones

Los lenguajes de programación se dividen en 2 categorías fundamentales:
Bajo nivel: Son dependientes de la máquina, están diseñados para ejecutarse en una determinada computadora. A esta categoría pertenecen las 2 primeras generaciones.
Alto Nivel: Son independientes de la máquina y se pueden utilizar en una variedad de computadoras. Pertenecen a esta categoría la tercera y la cuarta generación. Los lenguajes de más alto nivel no ofrecen necesariamente mayores capacidades de programación, pero si ofrecen una interacción programador/computadora más avanzada. Cuanto más alto es el nivel del lenguaje, más sencillo es comprenderlo y utilizarlo.
Cada generación de lenguajes es más fácil de usar y más parecida a un lenguaje natural que su predecesores.

2.4.1 Primera Generación

Empieza en los años 1940-1950. Consistía en sucesiones de dígitos binarios. Todas las instrucciones y mandatos se escribían valiéndose de cadenas de estos dígitos.
Aún en la actualidad, es el único lenguaje interno que entiende la computadora; los programas se escriben en lenguajes de mayor nivel y se traducen a lenguaje de máquina.

2.4.2 Segunda Generación

Fines de los ´50. Se diferencian de los lenguajes de máquina en que en lugar de usar códigos binarios, las instrucciones se representan con símbolos fáciles de reconocer, conocidos como mnemónicos. Aún se utilizan estos lenguajes cuando interesa un nivel máximo de eficiencia en la ejecución o cuando se requieren manipulaciones intrincadas. Al igual que los lenguajes de la máquina, los lenguajes ensambladores son únicos para una computadora particular. Esta dependencia de la computadora los hace ser lenguajes de bajo nivel.

2.4.3 Tercera Generación

Se desarrollan en los años ´60. Los lenguajes de esta generación se dividen en tres categorías, según se orienten a dos ramas:

Procedimientos: requieren que la codificación de las instrucciones se haga en la secuencia en que se deben ejecutar para solucionar el problema. A su vez se clasifican en científicos (ej. FORTRAN), empresariales (ej. COBOL), y de uso general o múltiple (ej. BASIC). Todos estos lenguajes permiten señalar cómo se debe efectuar una tarea a un nivel mayor que en los lenguajes ensambladores. Hacen énfasis los procedimientos o las matemáticas implícitas, es decir en lo que se hace (la acción).

Problemas: Están diseñados para resolver un conjunto particular de problemas y no requieren el detalle de la programación que los lenguajes orientados a procedimientos. Hacen hincapié en la entrada y la salida deseadas.

Objetos: El énfasis se hace en el objeto de la acción. Los beneficios que aportan estos lenguajes incluyen una mayor productividad del programador y claridad de la lógica, además de ofrecer la flexibilidad necesaria para manejar problemas abstractos de programación.

2.4.4 Cuarta Generación

Su característica distintiva es el énfasis en especificar qué es lo que se debe hacer, en vez de cómo ejecutar una tarea. Las especificaciones de los programas se desarrollan a un más alto nivel que en los lenguajes de la generación anterior.

La característica distintiva es ajena a los procedimientos, el programador no tiene que especificar cada paso para terminar una tarea o procesamiento. Las características generales de los lenguajes de cuarta generación son:

- Uso de frases y oraciones parecidas al inglés para emitir instrucciones
- No operan por procedimientos, por lo que permiten a los usuarios centrarse en lo que hay que hacer no en cómo hacerlo
- Al hacerse cargo de muchos de los detalles de cómo hacer las cosas, incrementan la productividad.

2.4.5 Quinta Generación

Alrededor de la mitad 1998 surgieron grupos de herramientas de lenguajes de quinta generación, los cuales combinan la creación de códigos basadas en reglas, la administración de reutilización y otros avances.

Programación basada en conocimiento. Método para el desarrollo de programas de computación en el que se le ordena ala computadora realizar un propósito en vez de instruirla para hacerlo.

3. ARQUITECTURAS DE PROGRAMACIÓN

Existen diversas formas de hacer un programa, una de las más comunes es hacer un sistema y tener que instalarlo en cada una de las máquinas que lo va a utilizar. Otra forma es tener una computadora dedicada a almacenar datos y contar con una aplicación que almacene los mismos, llamada cliente-servidor, la cual es la más utilizada actualmente.

Con el desarrollo de Internet ha crecido una nueva tendencia, trabajar desde cualquier lugar en cualquier momento. Las aplicaciones cliente-servidor, tienen la desventaja de tener que estar en la oficina para utilizarlos, en cambio una arquitectura orientada a Web permite que podamos conectarnos desde cualquier lado para realizar nuestro trabajo.

¿Qué es una arquitectura de software?

Una Arquitectura Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La arquitectura software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema de información.

Una arquitectura software se selecciona y diseña con base en unos objetivos y restricciones.

Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura software de tres capas para implementar sistemas en tiempo real.

La arquitectura software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos. Toda arquitectura software debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea de computación.

Cliente-servidor

La arquitectura cliente-servidor llamado modelo cliente-servidor o servidor-cliente es una forma de dividir y especializar programas y equipos de cómputo a fin de que la tarea que cada uno de ellos realizada se efectúe con la mayor eficiencia, y permita simplificarlas.

En esta arquitectura la capacidad de proceso está repartida entre el servidor y los clientes.

En la funcionalidad de un programa distribuido se pueden distinguir 3 capas o niveles:

- Manejador de Base de Datos (Nivel de almacenamiento),
- Procesador de aplicaciones o reglas del negocio (Nivel lógico) y
- Interface del usuario (Nivel de presentación)

En una arquitectura monolítica no hay distribución; los tres niveles tienen lugar en el mismo equipo.

En un comienzo, los mainframes concentraban la funcionalidad de almacenamiento (#1) y lógica (#2) y a ellos se conectaban terminales tontas, posiblemente en sitios remotos.

En el modelo cliente-servidor, en cambio, el trabajo se reparte entre dos ordenadores. De acuerdo con la distribución de la lógica de la aplicación hay dos posibilidades:

Cliente liviano (o cliente fino): si el cliente solo se hace cargo de la presentación.

Cliente pesado (o cliente grueso): si el cliente asume también la lógica del negocio.

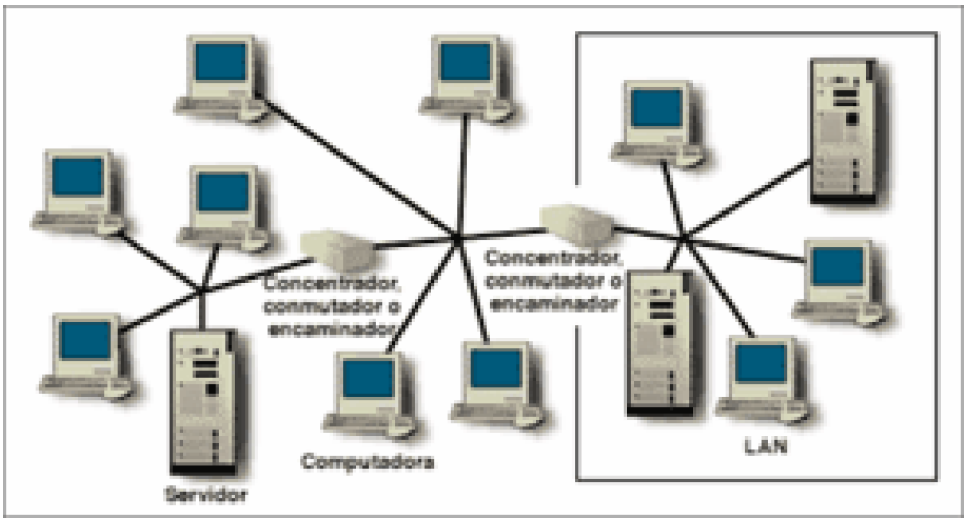
En la actualidad se suele hablar de arquitectura de tres niveles, donde la capa de almacenamiento y la de aplicación se ubican en (al menos) dos servidores diferentes, conocidos como servidores de datos y servidores de aplicaciones.

Ventajas de la arquitectura cliente-servidor

El servidor no necesita tanta potencia de procesamiento, parte del proceso se reparte con los clientes.

Se reduce el tráfico de red considerablemente. Idealmente, el cliente se conecta al servidor cuando es estrictamente necesario, obtiene los datos que necesita y cierra la conexión dejando la red libre para otra conexión.

Figura 1. Gráfico de Arquitectura para aplicaciones cliente-servidor



Arquitectura Orientada a Servicios

La Arquitectura Orientada a Servicios (en inglés Service-oriented architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios.

Al contrario de las arquitecturas orientadas a objetos, las SOAs están formadas por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación (p.ej., WSDL). La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Java o .NET). Con esta arquitectura, se pretende que los componentes software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar; así, un servicio C Sharp podría ser usado por una aplicación Java.

Diseño y desarrollo de SOA

La metodología de modelado y diseño para aplicaciones SOA se conoce como análisis y diseño orientado a servicios. La arquitectura orientada a servicios es tanto un marco de trabajo para el desarrollo de software como un marco de trabajo de implantación. Para que un proyecto SOA tenga éxito los desarrolladores de software deben orientarse ellos mismos a esta mentalidad de crear servicios comunes que son orquestados por clientes o middleware para implementar los procesos de negocio. El desarrollo de sistemas usando SOA requiere un compromiso con este modelo en términos de planificación, herramientas e infraestructura.

Cuando la mayoría de la gente habla de una arquitectura orientada a servicios están hablando de un juego de servicios residentes en Internet o en una intranet, usando servicios web. Hay un juego de estándares de los que se habla ligados a los servicios web. Incluyen los siguientes:

- XML
- HTTP
- SOAP
- WSDL
- UDDI

4. EL IMPACTO ECONÓMICO DEL DESARROLLO DE SOFTWARE EN BENEFICIO DE UN PAÍS Y RAZONES PARA ACTUALIZAR EL PENSUM DE ESTUDIOS DE LOS BACHILLERES EN COMPUTACIÓN EN GUATEMALA

En la actualidad con la rápida evolución con la que se produce hardware en el mundo no es de extrañar que el área de software este en el mismo camino.

Esta área ha sido desarrollada por varios países desde mediados de los ochenta teniendo su mayor producción en los últimos años. La India uno de los países más grandes de Asia con un crecimiento acelerado en su economía.

Este explosivo crecimiento ha sido posible en gran parte por el rápido y persistente incremento de sus exportaciones de software, lo cual ha generado un renovado optimismo acerca de la posibilidad que, por esta vía, se contribuya que los países en desarrollo puedan reducir la brecha tecnológica que les separa de los países industrializados.

Guatemala es un país con mucho potencial en el área de desarrollo de software ya que cuenta con ciertas ventajas sobre otros países:

- Su posición geográfica, ya que está colocada cerca de una de las economías más grandes del mundo.
- Similitud de horarios en la región, que ayudaría a proveer una solución de soporte con menos tiempo de espera.
- El crecimiento que empieza a tener en el sector debido a la creación de empresas dedicadas al desarrollo de software.

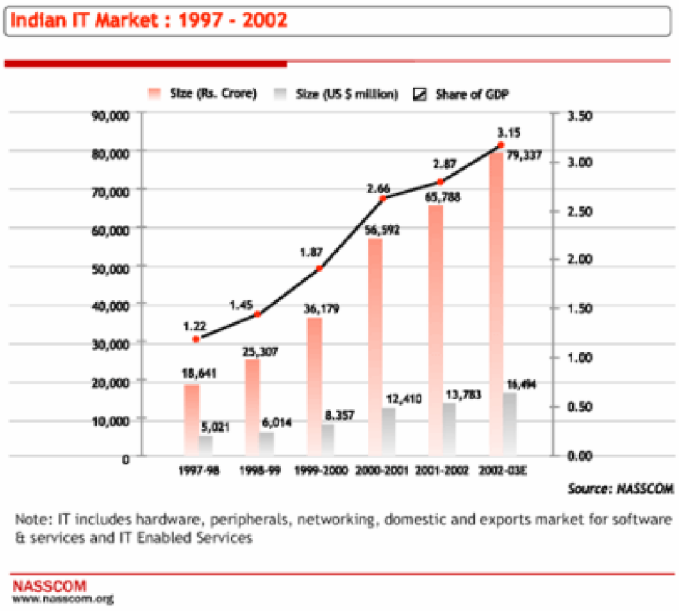
- Reconocimiento a nivel Americano de productos de software de buena calidad hechos por empresas guatemaltecas.

En resumen Guatemala tiene un escenario bastante alentador en el área de desarrollo de software fácilmente explotable que podría ayudar a la economía nacional como lo ha hecho con varios países alrededor del mundo.

CASO DE ESTUDIO: "LA INDIA"

La India ha tenido un incremento acelerado y muy grande en el mercado de TI en los últimos años.

Figura 2. Gráfica de crecimiento de mercado de tecnologías de la información de La India: 1997 - 2002



Nota: Tecnologías de la Información incluye hardware Periféricos, redes, mercado para software y servicios local y de exportación y Servicios para Tecnologías de la información.

Claves del Éxito de las TI en la India

Entre las ventajas competitivas que han impulsado el crecimiento explosivo del Sector de las Tecnologías de Información en India se encuentran las siguientes:

Tecnología de Microprocesadores

Al significativo auge de la poderosa industria informática a escala mundial ha contribuido la emergente y rápida transformación de la tecnología de microprocesadores ha tenido efectos revolucionarios en esta industria.

En primer lugar, este desarrollo ha potenciado una industria que produce software y hardware necesario para permitir a individuos, organizaciones, pequeñas empresas y corporaciones explotar directamente los beneficios de tal potencia informática. Además, ha transformado sustancialmente otras industrias, las cuales pueden ahora usar la capacidad de almacenar información y ejecutar instrucciones para automatizar y modificar la forma en la cual ellos dirigen y administran sus procesos y operaciones. Lidera una gran expansión del tamaño y envergadura del sector servicios, incluyendo finanzas, banca, comercio, entretenimiento y educación.

India, Segunda Fuerza Laboral Angloparlante

Alrededor de 280.000 personas calificadas laboran en la Industria del software y de servicios en la TI de India, reflejando la segunda fuerza laboral científica angloparlante del mundo. Con una población de más de 1.000 millones de habitantes y con 200 millones de jóvenes entre 15 a 30 años, como un reflejo de la inversión que el gobierno ha realizado en educación tanto en Universidades como en Institutos de elite orientados a fomentar el sector IT, cuyos ramos son impartidos en Inglés, se estima que 150.000 ingenieros se gradúan anualmente en India, de los cuales 68.000 son Ingenieros en Software.

Conocimiento, Mano de Obra Barata y Calificada

Las crecientes perspectivas que ofrece el sector (trabajo permanente, conexión con el exterior, salarios superiores a la media, acciones en las empresas) lo convierte en una gran atracción para jóvenes que buscan mejores perspectivas. Como consecuencia se están incrementando las postulaciones a los Institutos y Universidades especializadas, a las cuales sólo ingresan los que exhiben excelentes calificaciones en matemáticas, previo a un estricto proceso de selección.

India posee una alta calidad del recurso humano, asociado al beneficio de ser de un costo altamente efectivo. La mano de obra en India es eficientemente calificada y barata respecto a la competencia internacional, reflejada en el hecho que el costo promedio de un programador de software en India es 1/6 de su similar en Estados Unidos u otro país europeo desarrollado.[1]

Política Gubernamental de Fomento de la TI

El Gobierno ha jugado un rol determinante en el desarrollo del Sector de las Tecnologías de Información en India. Apoyando el auge exportador del sector TI y su descentralización, el gobierno ha creado centros de desarrollo tecnológico en varias ciudades indias.

Se han creado los Electronics Hardware Park (EHTP) unidades diseñadas para concentrar todos los requerimientos específicos de una sector electrónico globalmente.

Los Software Technology Park of India (STPI) se han creado como ventana única de solución proporcionada por organizaciones ligadas a la exportación de software y servicios. Ella provee facilidades de infraestructura en servicios de transmisión de datos de alta velocidad (HSDC).

[1] http://www.gobernabilidad.cl/modules.php?name=News&file=article&sid=521

Incentivos a las TI

Considerando las perspectivas de crecimiento de la TI, el gobierno de la India ha implementado una serie de incentivos para potenciar el auge de este sector, entre ellos destacan:

- 100% liberación impuestos en exportaciones en TI
- Liberación pago impuestos a abastecedores de software. Esto estimulará a pequeñas empresas a captar pedidos.
- Excepción impuestos donaciones computadores de más dos años uso a instituciones educacionales, hospitales e instituciones gubernamentales.
- Depreciación de TI productos a tasa de un 90% en tres años.
- Excepción arancel para software usado en TI.[2]

La India se basa en un plan muy ambicioso para tener estas características. Este plan contempla un lapso de 10 años (1998-2008) para realizar sus principales expectativas:

- Tener infraestructura suficiente para tener a la mayor parte del país conectado por fibra óptica, medio satelital y wireless y tener acceso a Internet, Extranets e Intranets a través del país.
- Lograr una exportación de 50 billones de dólares anuales de la Industria de Tecnologías de Información, Software y Servicios.
- Lograr que el porcentaje de personas que puedan acceder a una computadora cambie de 1 de cada 500 a 1 de cada 50 con total acceso al Internet. [3]

[2] http://www.gobernabilidad.cl/modules.php?name=News&file=article&sid=521
[3] http://www.nasscom.in/upload/38370/action_plan_1.pdf

CASO DE ESTUDIO: “COSTA RICA”

Un ejemplo más cercano de este auge de la exportación de Software de alta calidad es Costa Rica. Siendo un país Centro Americano tiene un acelerado crecimiento en el desarrollo de software de calidad.

El éxito de la industria de desarrollo de software depende principalmente de la existencia de recurso humano calificado, es por esta razón que en Costa Rica se desarrollo un Programa de Apoyo a la Competitividad del Sector Software y este ha dado especial énfasis a la formación de recurso humano a través del Estudio de Fortalecimiento de los Centros de Enseñanza y la Actualización Curricular, el cual se llevo a cabo de febrero del 2002 a octubre del 2003.

El estudio tenía como objetivo proponer recomendaciones curriculares y acciones para que la industria de desarrollo de software en Costa Rica pueda contar con personal adecuado en término de tres pilares:

a) Idoneidad: Es la capacidad y habilidad de los graduados para desempeñarse de manera óptima de acuerdo a las necesidades y demandas del entorno.
b) Calidad: Resultado de la formación de los estudiantes de acuerdo con los planes de estudio propuestos por cada centro de enseñanza y la preparación recibida por parte del cuerpo docente, los servicios, equipo e infraestructura disponible a los mismos.
c) Cantidad: Numero de profesionales, la cual debe estar relacionada con la demanda.

Estos tres pilares deben de tomarse en cuenta para el desarrollo de una estrategia tendiente a fortalecer el capital humano en apoyo a la competitividad de la industria de desarrollo de software. [4]

[4] Chacón Aguilar, Laura Julia. Desarrollo de la Industria del Software en Guatemala. Tesis Ing. En Sistemas, Universidad de San Carlos de Guatemala, Facultad de Ingeniería. 2007

Actualización constante del pensum de estudios de los bachilleres en computación en Guatemala

En el resumen de las estrategias utilizadas por La India y Costa Rica, se presentan dos puntos claves para que la exportación de software sea un pilar importante en la economía del país.

- Personal altamente capacitado
- Apoyo del Gobierno para desarrollar personal altamente capacitado.

Debido a que Guatemala es un país con población mayormente pobre, los estudiantes que salen de diversificado buscan trabajos inmediatos que les pueda ayudar a pagar su universidad o mantener a su familia, siendo en la mayoría de casos la segunda opción.

Las empresas de desarrollo de software necesitan personal altamente capacitado para producir, sin embargo los bachilleres no llegan a los estándares establecidos por estas empresas ya que tienen poco o ningún conocimiento de las tendencias actuales del mercado de desarrollo de software, lo que los obliga a tomar sub-empleos en los que la mayoría desperdician el tiempo utilizado en el bachillerato.

Por lo anterior es necesario una actualización en el pensum de los bachilleres en computación, con la colaboración de las empresas de desarrollo de software y el Ministerio de Educación, para definir un punto en el que los graduados de bachilleres de computación provean una fuerza utilizable de trabajo calificada para el desarrollo de software, siendo de beneficio para estos y las empresas de desarrollo de software.

5. COMPARACIÓN DE PENSUM DE ESTUDIO DE PROGRAMACIÓN DE BACHILLERES EN COMPUTACIÓN VRS. EMPRESAS EXPORTADORAS DE SOFTWARE EN GUATEMALA (SOFEX)

Desde que las ciencias en la computación comenzaron a invadir el mercado y a convertirse en una necesidad, se adoptó esta disciplina en el pensum de los estudiantes, principalmente en los bachilleres en computación. Sin embargo la tecnología ha avanzado con pasos agigantados y la educación se fue quedando atrás, lo que ha traído como resultado la existencia de un currículo educativo obsoleto. La pregunta ahora es ¿cuentan los bachilleres en computación con las herramientas para enfrentarse a un mercado competitivo en la elaboración de software?

En Guatemala existe la gremial de exportadores de software llamada SOFEX que desde su fundación se ha caracterizado por estar a la vanguardia en el uso de herramientas por lo que se ha considerado como un parámetro de comparación entre las herramientas utilizadas y las que los alumnos de bachillerato en computación aprenden en sus establecimientos educativos.

Las arquitecturas Cliente – Servidor y Orientada a Servicios, han sido utilizados por las empresas que componen a SOFEX (Gremial de Exportadores de Software de Guatemala), teniendo la tendencia a utilizar la Arquitectura Orientada a Servicios porque cada vez más personas se familiarizan con el uso del Internet en sus trabajos y hogares.

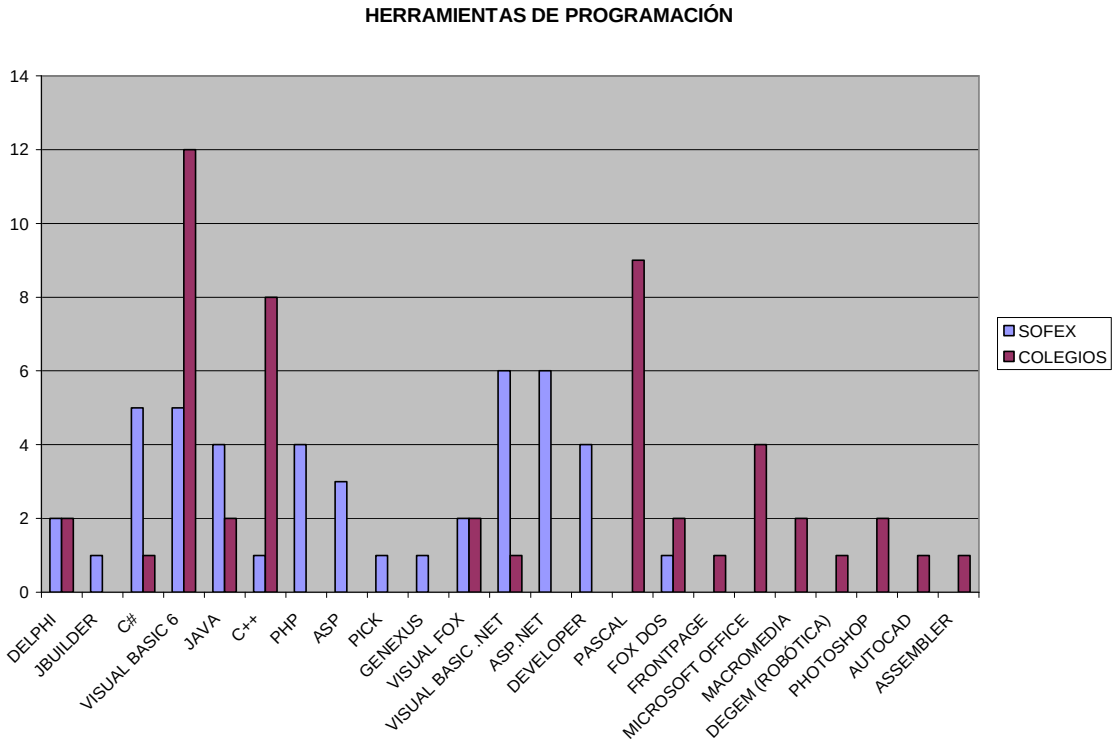
Para hacer el estudio comparativo se verificaron las herramientas que se enseñan en los centros educativos a los alumnos de bachillerato en computación y las herramientas utilizadas por las empresas agremiadas a SOFEX, los resultados fueron los siguientes:

Tabla I: Herramientas utilizadas y su frecuencia de utilización en empresas de SOFEX y la muestra de colegios que imparten bachillerato en computación

HERRAMIENTAS	SOFEX	COLEGIOS
DELPHI	2	2
JBUILDER	1	0
C#	5	1
VISUAL BASIC 6	5	12
JAVA	4	2
C++	1	8
PHP	4	0
ASP	3	0
PICK	1	0
GENEXUS	1	0
VISUAL FOX	2	2
VISUAL BASIC .NET	6	1
ASP.NET	6	0
DEVELOPER	4	0
PASCAL	0	9
FOX DOS	1	2
FRONTPAGE	0	1
MICROSOFT OFFICE	0	4
MACROMEDIA	0	2
DEGEM (ROBÓTICA)	0	1
PHOTOSHOP	0	2
AUTOCAD	0	1
ASSEMBLER	0	1

En la tabla I se resumen los resultados obtenidos de la investigación de herramientas utilizadas por las empresas afiliadas a SOFEX y la muestra de colegios que imparten el bachillerato en computación.

Figura 3. Gráfica de la frecuencia de utilización de las herramientas de programación en las empresas afiliadas a SOFEX y la muestra de colegios que imparten el bachillerato en computación

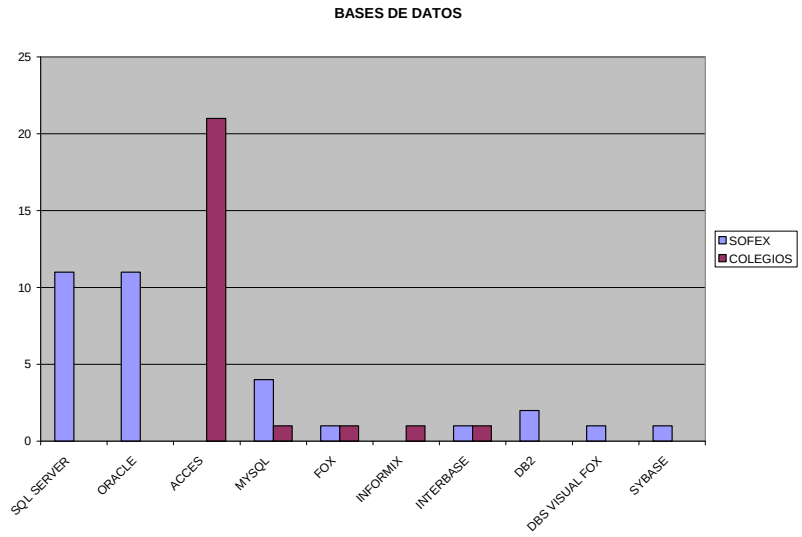


Se puede verificar en la gráfica que los lenguajes más utilizados por los colegios es Pascal y Visual Basic 6. en SOFEX dominan las herramientas como Visual Basic .NET, Developer y Visual Basic 6.

Tabla II: Bases de datos utilizadas y su frecuencia de utilización en empresas de SOFEX y la muestra colegios que imparten bachillerato en computación

BASE DE DATOS	SOFEX	COLEGIOS
SQL SERVER	11	0
ORACLE	11	0
ACCES	0	21
MYSQL	4	1
FOX	1	1
INFORMIX	0	1
INTERBASE	1	1
DB2	2	0
DBS VISUAL FOX	1	0
SYBASE	1	0

Figura 4. Gráfica de bases de datos utilizadas y su frecuencia de utilización en empresas de SOFEX y la muestra colegios que imparten bachillerato en computación

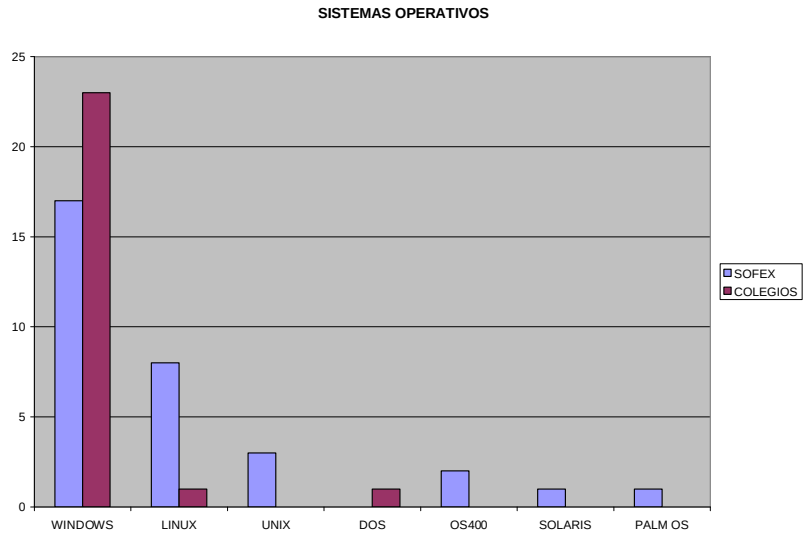


Se lográ visualizar que las empresas centran su actividad de bases de datos en programas como SQL SERVER y ORACLE, y los colegios utilizan ACCESS, la cual como se logra ver en la gráfica no es utilizada por ninguna empresa en SOFEX.

Tabla III: Sistemas operativos utilizados y su frecuencia de utilización en empresas de SOFEX y la muestra colegios que imparten bachillerato en computación

SISTEMAS OPERATIVOS	SOFEX	COLEGIOS
WINDOWS	17	23
LINUX	8	1
UNIX	3	0
DOS	0	1
OS400	2	0
SOLARIS	1	0
PALM OS	1	0

Figura 5. Gráfica de sistemas operativos utilizados y su frecuencia de utilización en empresas de SOFEX y la muestra colegios que imparten bachillerato en computación



En la gráfica se logra visualizar que el sistema operativo más utilizado, tanto en SOFEX como en los colegios, es Windows, seguido por el sistema operativo Linux utilizado únicamente por un colegio.

6. DISCUSIÓN

El uso de herramientas de programación se ha expandido y con ello la necesidad de programadores capacitados. La carrera de Bachillerato en computación rápidamente adoptó muchos adeptos que soñaban con programar. Sin embargo, la gremial SOFEX, de respetado prestigio por estar a la vanguardia en el uso de herramientas para programar, utiliza herramientas que los bachilleres en computación desconocen. ¿Será necesaria una reestructuración curricular en el pensum de estudios de los Bachilleres?, ¿será la carrera de bachilleres sólo una introducción a las ciencias de la computación, la cual necesita de una ingeniería en sistemas para crear programadores aptos a las necesidades de las empresas? Como se verá a continuación, algunas de las herramientas que utilizan las empresas son del conocimiento de los bachilleres, pero otros, están lejos de ser conocidas por ellos.

En el principio de la programación o el desarrollo de programas o sistemas, se tenían pocas personas que podían realizar la tarea, es decir, que todos los sistemas eran desarrollados por el mismo grupo de personas. Esto debido a que los sistemas se desarrollaban de forma específica, como la programación utilizando el código binario. Posteriormente se avanzó dando como resultado los lenguajes ensambladores, que además de facilitar el trabajo de programación de una computadora, incrementaron el número de personas para programarlas.

La programación de computadoras no ha variado mucho desde entonces, se siguen utilizando personas para esta tarea, pero la cantidad de personas o empresas que requieren programas o sistemas se ha incrementado de manera gigantesca, provocando la multiplicación del trabajo y la necesidad de tener más personas que se encarguen de hacer estos programas.

La educación en Guatemala, particularmente la de los bachilleres en computación, tuvo sus inicios en el gran desarrollo que tenia la sistematización de las empresas. Lastimosamente la educación de estos bachilleres se quedó muy rezagada a comparación del avance que ha tenido la tecnología.

Actualmente, en Guatemala se ha implementado la gremial de exportadores de software que busca competir con otros países en el desarrollo de soluciones para las necesidades de las empresas. Pero SOFEX se enfrenta al problema de que el número de personas que desarrollan el software es limitado, casi escaso, o no tienen el conocimiento que se necesita para poder laborar en estas empresas.

Después de hacer el estudio, se hace evidente que las instituciones que enseñan programación a los bachilleres en computación, se han quedado atrasadas en el desarrollo de las habilidades de los programadores, y que el bachillerato en computación se ha vuelto únicamente un pase para entrar a la universidad.

En el análisis de resultados se observa que Pascal es una de las herramientas más ampliamente conocidas por los estudiantes, como se muestra en la figura número 3. La herramienta pascal fue creada con el único propósito de educar a los jóvenes que estaban interesados en la programación, razón por la cual fue incorporada en el currículo educativo y es la más ampliamente conocida por los estudiantes, por lo que se interpreta que esta herramienta es obsoleta para la empresas agremiadas a SOFEX.

C fue desarrollado con el propósito de implementar sistemas operativos, tales como Linux y Unix, pero tiene tanta versatilidad que se ha utilizado para el desarrollo de otras aplicaciones, es incluido en el currículo de algunos colegios por la misma versatilidad.

SOFEX casi no utiliza C en sus sistemas, como se muestra en la figura número 3 aunque hay una variación de esta herramienta que se denomina C# que esta tomando su propio camino, pero en general esta herramienta no es de utilidad para SOFEX.

Visual Basic, como su nombre lo indica es una herramienta visual derivado del antiguo lenguaje Basic, con características de lenguajes estructurados. Como se muestra en la figura número 3, los alumnos están identificados con el programa, es decir les parece atractivo en la práctica, y, esto promueve un entusiasmo por aprenderlo. Afortunadamente es una de las herramientas más utilizadas por SOFEX, como se muestra en la figura número 3, lo que nos indica que esta herramienta es útil que se aprenda para que haya una relación educativa – SOFEX que sirva en ambos sentidos.

Por otro lado, .NET es una de las más recientes herramientas hechas para programar software, desarrollada para facilitar la unión de programas, bases de datos y plataformas en los cuales pueda correr. Considerando lo anterior se puede ver claramente porqué en la figura número 3 se demuestra el poco conocimiento del mismo entre los alumnos. Por su parte SOFEX utiliza esta herramienta como una de las más utilizadas para el desarrollo de los programadores, en esta herramienta si se evidencia una clara separación en la relación educativa - SOFEX.

DELPHI, viene a innovar el antiguo pascal. Mientras Pascal fue elaborado para el área educativa, DELPHI, que aparte de ser visual, viene con muchas herramientas para conexiones a bases de datos, manejos de archivos y una interesante variedad de herramientas para el desarrollo de programas. Como se muestra en la figura número 3, ésta es una herramienta con la que los alumnos pueden tener un avance rápido debido a su familiaridad con Pascal pero lastimosamente en SOFEX, solamente la utilizan dos empresas.

JAVA este lenguaje ha sido desarrollado para lograr utilización multiplataforma, siendo descendiente directo de C, posee muchas características del mismo, como el polimorfismo y la herencia. Este lenguaje sin embargo tiene muy poco porcentaje de alumnos que lo conocen y manejan, como se muestra en la figura número 3. Sin embargo en SOFEX resulta ser una de las herramientas de más utilidad para el desarrollo de programas o sistemas.

Aunque los Bachilleres en computación tengan conocimiento de Pascal, Visual Basic, C, esto no los hace competitivos en el mercado laboral, ya que el único lenguaje en el que van de la mano está empezando a ser reemplazado por su evolución Visual Basic .NET y las empresas de punta utilizan principalmente herramientas como Java y las herramientas propias de .Net. La comparación de las herramientas conocidas por los Bachilleres y las herramientas utilizadas por las empresas muestra que los dos grupos sólo coinciden en Visual Basic. Este resultado es alentador para las autoridades educativas que luchan por actualizar sus currículos educativos y graduar bachilleres que con su conocimiento y aptitudes puedan llegar a hacer que Guatemala crezca y se encuentre entre los mejores exportadores de software a nivel latinoamericano.

Haciendo una comparación entre los manejadores de bases de datos que se utilizan en SOFEX y la muestra de colegios que imparten bachillerato en computación, se puede ver una clara brecha entre los dos. Esto se muestra en la figura número 4 en la que se puede ver claramente que los manejadores de bases de datos más utilizados en el mercado son: “SQL Server, Oracle y MySql”, sin embargo en la muestra de colegios que imparten bachillerato en computación enseñan, en su gran mayoría, el manejo de Access del paquete de Microsoft Office.

El manejo de los diferentes sistemas operativos también se ve marcado en la comparación. Las empresas afiliadas a SOFEX manejan varios sistemas operativos que son totalmente desconocidos para los bachilleres en computación, tal es el caso de Unix, AS400, Solaris y Palm OS.

El sistema operativo más conocido y manejado tanto en las empresas como en la muestra de colegios que imparten bachillerato en computación es Windows, figura número 5, siendo el más popular actualmente. Hay un caso en el que aparte de manejar Windows, también manejan Linux, dándoles a los alumnos una ventaja en el mercado.

Sería recomendable que los colegios adoptaran el uso del sistema operativo Linux con el fin de ampliar los conocimientos de los alumnos, aprovechando que el licenciamiento del mismo es muy accesible.

CONCLUSIONES

1. El pensum de estudios de la carrera de bachillerato en computación no prepara a los alumnos como programadores listos para ser contratados por las empresas desarrolladoras de software agremiadas a SOFEX.

2. Las herramientas, que actualmente son impartidas a los bachilleres en computación como parte de su pensum son: Pascal, C, Visual Basic.

3. Las herramientas más utilizadas actualmente por las empresas desarrolladoras de software que conforman SOFEX son: Herramientas de .NET, Visual Basic y Java.

4. Las únicas herramientas que actualmente se utilizan tanto en la educación como en las empresas desarrolladoras de software agremiadas a SOFEX es Visual Basic y Delphi.

5. El pensum de estudios de la carrera de bachillerato en computación no prepara a los alumnos como programadores listos para ser contratados por las empresas agremiadas a SOFEX.

6. Tomando en cuenta los resultados de la estadística se observa una separación entre las instituciones educativas que preparan programadores y las empresas desarrolladoras de software que están agremiadas a SOFEX.

RECOMENDACIONES

1. El sector académico, colegios y Ministerio de Educación y las empresas desarrolladoras de software deben realizar una revisión constante del pensum de estudios de los bachilleres en computación, para facilitar la incorporación de los alumnos al mercado laboral.

2. Para que un bachiller en computación sea apto para trabajar en empresas que desarrollan software –construyendo software- es necesario una reestructuración de la clase de programación en su pensum.

3. Utilizar el lenguaje Java o alguna de las herramientas de .NET, por su diversidad de funciones en el último año de bachillerato, para que los programadores salientes sean más competitivos en el mercado laboral. Ver apéndice A.

4. La enseñanza de la programación se ha reducido a "la enseñanza de una herramienta", se recomienda modificar la metodología en una que enseñe a programar; por ejemplo, enseñar el uso de algoritmos, pero que se utilice una herramienta actualizada para que los graduandos sean más competitivos en el mercado.

5. Se debe ampliar el pensum de estudios en cuanto a técnicas de programación y control de calidad del software, independientemente de la herramienta de desarrollo.

6. Si un bachiller en computación esta interesado en trabajar como programador, se le recomienda ampliar sus conocimientos mediante el estudio de una carrera en sistemas y ciencias de la computación.

7. Realizar prácticas de graduación en el área de desarrollo de las empresas para que los graduados puedan aprovechar sus conocimientos.

8. Realizar un proyecto de graduación para la evaluación de los conocimientos adquiridos por los alumnos.

9. Utilizar otros sistemas operativos que permitan al alumno conocer más formas de administrar los recursos de la computadora. Linux sería una buena herramienta debido a su licenciamiento accesible.

10. El Ministerio de Educación debe buscar apoyo en empresas internacionales para la adquisición de licencias necesarias de los programas de desarrollo más utilizados en el mercado.

REFERENCIAS ELECTRÓNICAS

1. **José Guillermo Valle, James Gildardo Gutierrez.** Definición arquitectura cliente servidor. España. 2005. http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml#quees

2. **Wikipedia**. **Cliente-servidor.** 2006 http://es.wikipedia.org/wiki/Cliente-servidor

3. **Wikipedia. Arquitectura software.** 2006 http://es.wikipedia.org/wiki/Arquitectura_de_software

4. **Wikipedia. Arquitectura Orientada a Servicios.** 2006. http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

5. **Wikipedia. Programación.** 2007. http://es.wikipedia.org/wiki/Programaci%C3%B3n

6. **Desarrollo de la industria tecnológica de información en India.** 2004. http://www.gobernabilidad.cl/modules.php?name=News&file=article&sid=521

7. **Information Technology Action Plan**. Gobierno de la India. http://www.nasscom.in/upload/38370/action_plan_1.pdf

BIBLIOGRAFÍA

1. **Chacón Aguilar, Laura Julia**. Desarrollo de la Industria del Software en Guatemala. Tesis Ing. En Sistemas, Universidad de San Carlos de Guatemala, Facultad de Ingeniería. 2007.

APÉNDICE A

Se recomiendan las siguientes páginas para actualizarse en las herramientas más utilizadas en el mercado.

- Desarrollador 5 Estrellas, sitio fundado por Microsoft para capacitar a los programadores de Latinoamérica:

  http://www.mslatam.com/latam/msdn/comunidad/dce2005/

- Sun Capacitación, sitio fundado por Sun Microsystems para la capacitación de personal en java:

  https://www.suntrainingcatalogue.com/eduserv/client/cmsearch.do;jsessionid=0F43141F81BC398FF413225E1C6A43F1.tomcat1?catId=3485&l=es_MX

APÉNDICE B

Las empresas encuestadas agremiadas a SOFEX fueron:

- OPEN-CONSULT
- BYTE
- ALDEAS SYSTEM
- STRATEGIC ANALISIS DE CENTRO AMÉRICA
- MFSI
- SOFTWARE Y SERVICIOS DE AUTOMATIZACIÓN, S.A.
- COINSA
- CONSULTE
- VIA ASESORES, S.A.
- GYSSA
- MEGASOLUCIONES
- SEGA
- ASEINFO
- BDGSA
- ICON
- SAVSA
- SITECPRO

COMISIÓN DE SOFTWARE DE GUATEMALA
SofeX
Aplicaciones desarrolladas para el mundo.

AGEXPORT
ASOCIACIÓN GUATEMALTECA DE EXPORTADORES

Guatemala, 25 de junio de 2007

Ingeniero
Marlon Pérez Turk
Director de Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala
Presente

Estimado Ing. Pérez,

Por este medio deseamos hacer constar que los datos respecto a empresas de desarrollo de software empleados por el estudiante Erik Arnulfo Santizo Bardales para realizar su trabajo de tesis titulado "ANÁLISIS COMPARATIVO ENTRE EL PENSUM DE PROGRAMACIÓN DE COMPUTADORAS DE LA CARRERA DE BACHILLERES EN COMPUTACIÓN Y LAS NECESIDADES DE LAS EMPRESAS DESARROLLADORAS DE SOFTWARE EN EL ÁREA METROPOLITANA DE GUATEMALA" fueron obtenidos de empresas que son miembros de la Comisión de Software de Exportación de Guatemala (SOFEX), representando la muestra aproximadamente el 66% de las empresas afiliadas.

Esperamos que esta información sea de su utilidad, y agradecemos el interés de la Universidad de San Carlos de Guatemala en promover este tipo de investigaciones que serán sin duda de gran utilidad para promover el desarrollo del país.

Atentamente,

Jorge Arturo Rivera Perezgil, MSE
Coordinador Académico
Comisión de Software de Guatemala - Sofex

15 Avenida 14-72 Zona 13, Guatemala, C.A. - PBX: (502) 2422 3400 - Fax: (502) 2422 3434
e-mail: sofex@agexport.org.gt

APÉNDICE C

Los colegios encuestados fueron:

- Colegio Salesiano Don Bosco
- Liceo Fraternidad Cristiana
- Colegio Bilingüe El Prado
- Suger Montano
- Bilingüe en Computación
- Colegio San Jorge
- CEIS
- LA VID
- Colegio San Pablo
- SHADDAI
- ELIM
- Colegio Justo Rufino Barrios
- Colegio Euroamericano
- MAYALAND
- Colegio San José de la Encarnación
- Colegio Suizo Américano
- Tecnológico André Ampere
- Liceo Victoria y Libertad
- SM Cortijo
- Centro Integrado de computación y turismo
- Colegio de computación electrónica y electricidad
- IMB-PC

- Liceo Americano Tecnológico de computación
- Liceo Ixchel
- Colegio Científico Integrado