INDUSTRIAL TRAINING REPORT

Submitted By:
Puru Govind (02bce070)
BTech (CSE)

TRAINING DETAILS

Company Name: Ebix

Start Date: 13th May'2004

End Date: 14th June'2004

Address: B59A, Sector-60
Noida-201301
U.P

Aim: To produce various test cases and find bugs in software.

COMPANY PROFILE

Ebix stands for E-business insurance exchange. Ebix provides a comprehensive range of software solutions for insurance carrier, brokers and underwriting agencies. Our solutions are designed to meet the business requirements of a broad cross section of the market. Importantly these solutions can be tailored to meet your exact requirements.

Ebix business solutions are backed by an organization that understands the complex requirements of the market and a team of experienced business consultants and project managers with a successful track record of system implementations

Ebix leverages its **"Global Outsourcing Model"** to enable clients to take ownership of their requirement and helps them realize nearly 50% cost savings. Our delivery model encompasses multiple processes and methodologies to take the surprise out of projects and deliver measurable results for all of our global customers.

The Company has the Software Engineering Institute's **(SEI) Capability Maturity Model (CMM)** Level 5 certification for its development unit in India and has already secured an **ISO 9001:2000 certifications** for both its development and contact centre facilities in India.

History:-

A long history of designing innovative solutions has earmarked an era of excellence in the history of Ebix. It is committed in its endeavor to answer the ever-increasing needs of Insurance organizations all over the world.

The name change from "Delphi Information Systems" to Ebix Inc. aligned the identity of the company with its strategic focus of using the Internet to enhance the way insurance business is transacted, through solutions that encompass both e-commerce and web-enabled agency management systems.

An independent provider, Ebix Inc. employs hundreds of financial sector and technology professionals who provide products, support and consultancy across 6 continents.

Founded in 1976, Ebix Inc. (NASDAQ: EBIX), formerly Delphi Information Systems, is a profitable leading international supplier of software and solutions to the BFSI industry.

Ebix Inc.’s products feature fully customizable and scaleable software designed to improve the way insurance professionals manage all aspects of BFSI distribution, including: marketing, sales, service, accounting and management.

Both the analyst and the media community have recognized Ebix for its innovation and technology expertise.

PLATFORM

AIX

AIX (**Advanced Interactive eXecutive**) is a proprietary operating system developed by IBM based on UNIX System V. Before the product was ever marketed, the acronym *AIX* originally stood for **Advanced IBM Unix**.

AIX has pioneered numerous network operating system enhancements, introducing new innovations later adopted by Unix-like operating systems; it is often one of the first operating systems to implement a new innovation in software architecture as a sophisticated software technology.

The scalable AIX 5L 5.3 supports up to 64 central processing units and two terabytes (TB) of random access memory. The JFS2 file system—first introduced by IBM as part of AIX—supports computer files and partitions up to 16 TB in size.

Development

AIX V1, introduced in 1986, was based on System V Release 3. IBM later ported AIX to the RS/6000 platform as **AIX/6000**; since 1989, AIX has served as the RS/6000's primary operating system. In developing AIX, IBM and INTERACTIVE Systems Corporation (whom IBM contracted) also incorporated source code from Berkeley Software Distribution 4.2 and 4.3.

In the SCO v. IBM lawsuit filed in 2003, the SCO Group alleged that (among other infractions) IBM misappropriated licensed source code from UNIX System V Release 4 for incorporation into AIX. IBM maintains that their license was irrevocable.

Supported architectures

AIX v1 supported IBM PS/2 Micro Channel architecture PCs and the IBM RT.

AIX v2 supported 6150-series IBM RT systems.

AIX v3 introduced support for the IBM POWER architecture.

AIX v4 introduced support for the PowerPC architecture and the PCI bus.

AIX v5 introduced support for the IA64 architecture.

AIX v5.1 was the last version to support the Micro Channel architecture.

AIX on IBM Mainframes

In 1988, IBM announced AIX/370. AIX/370 was IBM's first attempt to offer Unix-like functionality for their mainframe line, specifically the System/370. AIX/370 was released in 1990 with functional equivalence to System V Release 2 and 4.3BSD as well as IBM enhancements. With the introduction of the ESA/390 architecture, AIX/370 was rebranded as AIX/ESA in 1991 and ran on the System/390 platform. Unlike AIX on its other platforms, AIX on the mainframe never ran as the host operating system, but rather as a guest under VM. AIX on the mainframe had little success and UNIX functionality was instead added as an option with the existing mainframe operating system, MVS, which became MVS/ESA OpenEdition in 1993.

Versions

AIX 5L 5.3, August 2004

- NFS Version 4 support
- Advanced Accounting
- Virtual SCSI
- Virtual Ethernet
- Simultaneous multithreading (SMT) support
- Micro-Partitioning support
- JFS2 quota support
- JFS2 filesystem shrink support

AIX 5L 5.2, October 2002

- Minimum level required for POWER5 hardware
- Support for MPIO Fibre Channel disks
- iSCSI Initiator software
- Dynamic LPAR support

AIX 5L 5.1, May 2001

- Minimum level required for POWER4 hardware and the last release that supported Micro Channel architecture
- Introduction of 64-bit kernel, installed but not activated by default
- JFS2
- Static LPAR support
- The L stands for Linux affinity
- Trusted Computing Base (TCB)

AIX 4.3.3, September 1999

- Added online backup function
- Workload Management (WLM)

AIX 4.3.2, October 1998

AIX 4.3.1, April 1998

AIX 4.3, October 1997

- Support for 64-bit architecture

AIX 4.2.1, April 1997

- NFS Version 3 support

AIX 4.2, May 1996

AIX 4.1.5, August 1996

AIX 4.1.4, October 1995

AIX 4.1.3, July 1995

- CDE 1.0 became the default GUI environment, replacing Motif X Window Manager.

AIX 4.1.1, October 1994

AIX 4.1, August 1994

AIX v4, 1994

AIX v3.2 1992

AIX v3.1

- Introduction of Journaled File System (JFS)

AIX v3, February 1990

- Developer release licensed only to OSF; the LVM was incorporated into OSF/1.
- SMIT was introduced.

AIX v2

- last version was 2.2.1.
- AIX v1, 1986
- last version was 1.3.

Interfaces

Graphical

The Common Desktop Environment (CDE) is AIX's default graphical user interface. As part of Linux Affinity and the free AIX Toolbox for Linux Applications (ATLA), open-source KDE and GNOME desktop are also available.

Text-based console

SMIT, also known as smitty, is a text based System Management Interface Tool for AIX. It allows a user to navigate a menu hierarchy of commands, rather than using the command line. Experienced system administrators make use of the F6 command which generates the command line needed for complex tasks.

SMIT and smitty are the same program, however smitty is the text-based version, and SMIT is a graphical version which runs under X windows. If you are on a text based terminal, running the smit program will invoke the text-based version.

LANGUAGE USED

BASIC

BASIC (***B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode*[1]) is a family of high-level programming languages. Originally invented in 1964 by John George Kemeny and Thomas Eugene Kurtz at Dartmouth College, it was designed to allow students not in science fields to use computers. At the time all computer use required writing custom software, which was something only scientists and mathematicians tended to do. It became widespread on home microcomputers in the 1980s, and remains popular to this day in a handful of heavily evolved dialects.

History

Background

Prior to the mid-1960s, computers were highly expensive tools used only for special-purpose tasks. In a simple form of batch processing, these machines ran only a single "job" at a time. During the 1960s, however, computer prices started to drop to where even small companies could afford them, and their speed increased to the point where they often sat idle without jobs to run.

Programming languages of the era tended to be designed, like the machines on which they ran, for specific purposes (such as scientific formula processing or text editing). Since single-job machines were expensive, the tendency was to consider execution speed the most important feature of a language. In general, these specialized languages were difficult to use and used widely disparate syntax.

It was at this time that the time-sharing system concept started to become popular. In such a system the processing time of the main computer is "sliced up" and each user is given a small amount in alternation. The machines were fast enough for most users to feel they had a single machine all to themselves. In theory, timesharing reduced the cost of computing tremendously, as a single machine could be shared among hundreds of users.

Early years — the mini computer era

The original BASIC language was invented in 1963 by John Kemeny and Thomas Kurtz and implemented by a team of Dartmouth students under their direction. BASIC was designed to allow students to write programs for the Dartmouth Time-Sharing System. BASIC was intended to address the complexity issues of older languages with a new language designed specifically for the new class of users the time-sharing systems

allowed — that is, a "simpler" user who was not as interested in speed as in simply being able to use the machine. In the following years, as other dialects of BASIC appeared, Kemeny and Kurtz' original BASIC dialect became known as *Dartmouth BASIC*.

The eight design principles of BASIC were:

- Be easy for beginners to use.
- Be a general-purpose programming language.
- Allow advanced features to be added for experts (while keeping the language simple for beginners).
- Be interactive.
- Provide clear and friendly error messages.
- Respond fast for small programs.
- Not require an understanding of computer hardware.
- Shield the user from the operating system.

The language was based partly on FORTRAN II and partly on ALGOL 60, with additions to make it suitable for timesharing and, later, text processing and matrix arithmetic. BASIC was first implemented on the GE-265 mainframe which supported multiple terminals. Contrary to popular belief, it was a compiled language at the time of its introduction.

The designers of the language decided that it should remain in the public domain so that the language would become widespread. They also made it available to high schools in the Dartmouth area and spent a considerable amount of effort in promoting the language. As a result, knowledge of BASIC became relatively widespread for a computer language and BASIC was implemented by a number of manufacturers, and became fairly popular on newer minicomputers like the DEC PDP series and the Data General Nova. In these instances the language tended to be implemented as an interpreter, instead of (or in addition to) a compiler.

Several years after its release, highly-respected computer professionals, notably Edsger W. Dijkstra, expressed their opinions that the use of GOTO statements, which existed in many languages including BASIC, promoted poor programming practices.[2] Some also derided BASIC as too slow or too simple.

Explosive growth — the home computer era

Commodore BASIC V2.

Notwithstanding the language's use on several minicomputers, it was the introduction of the Altair 8800 microcomputer in 1975 that truly spread BASIC. Most programming languages were too large to fit in the small memory most users could afford on these machines, and with the slow storage on paper tape (or later audio cassette tape: disks of any kind were not available at any price for some years) and the lack of suitable text editors, a small language like BASIC was a good fit. BASIC also had the advantage that it was fairly well known to the young designers who took an interest in microcomputers at the time as a result of Kemeny and Kurtz's earlier proselytizing. One of the first to appear for this machine was Tiny BASIC, a simple BASIC implementation originally written by Dr. Li-Chen Wang, and then ported onto the Altair by Dennis Allison at the request of Bob Albrecht (who later founded Dr. Dobb's Journal). The Tiny BASIC design and the full source code were published in 1976 in DDJ.

MSX BASIC version 3.0

Newer companies attempted to follow the successes of MITS, IMSAI, North Star and Apple, thus creating the home computer revolution; meanwhile, BASIC became a standard feature of all but a very few home computers. Most came with a BASIC interpreter in ROM. Soon there were many millions of machines running BASIC around the world, likely a far greater number than all the users of all other languages put together.

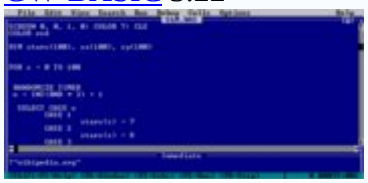
In 1975, Micro-Soft (then only two people — Bill Gates and Paul Allen) released Altair BASIC. The version written for the Altair was co-authored by Gates, Allen and Monte Davidoff. Versions of Microsoft BASIC then started appearing on other platforms under license, and millions of copies and variants were soon in use; it became one of the standard languages on the Apple II. By 1979, Microsoft was talking with several microcomputer vendors, including IBM, about licensing a BASIC interpreter for their computers. A version was included in the IBM PC ROM chips and PCs without floppy disks automatically booted into BASIC.

There are more dialects of **BASIC** than there are of any other programming language. All but a very few home computers of the 1980s had a ROM-resident BASIC interpreter.

The BBC published BBC BASIC, developed for them by Acorn Computers Ltd, incorporating many extra structuring keywords, as well as comprehensive and versatile direct access to the operating system. It also featured a fully integrated assembler.

Maturity — the personal computer era

GW-BASIC 3.22



Microsoft QuickBASIC 4.5

Many newer BASIC versions were created during this period. Microsoft sold several versions of BASIC for MS-DOS/PC-DOS including BASICA, GW-BASIC (a BASICA-compatible version that did not need IBM's ROM) and Quick BASIC. Turbo Pascal-publisher Borland published Turbo BASIC 1.0 in 1985 (successor versions are still being marketed by the original author under the name PowerBASIC).

These languages introduced many extensions over home computer BASIC, such as improved string manipulation and graphics support, access to the file system and additional data types. More important were the facilities for structured programming, including additional control structures and proper subroutines supporting local variables.

However, by the latter half of the 1980s newer computers were far more complex. At the same time, computers had progressed from a hobbyist interest to tools used primarily for applications written by others, and programming as a whole became less important for the growing majority of users. BASIC started to fade, though numerous versions remained available.

BASIC's fortune reversed once again with the introduction of Visual Basic from Microsoft. Though it is somewhat difficult to consider this language to be BASIC, because of the significant shift in paradigm towards an object-oriented and event-driven language. While this could be considered an evolution of the language, few of the features of Dartmouth BASIC, such as line numbers and the `INPUT` keyword, remain.

Many BASIC dialects have also sprung up in the last few years, including Bywater BASIC, True BASIC and REALbasic. Many other BASIC variants and adaptations are

authored by hobbyists, equipment developers, and others, as it is relatively easy to develop interpreters and compilers for BASIC

PROJECT DETAILS

I worked on project “Infinity” from 13 May 2004 to 14 June 2005. This was insurance based project. The Company also continues to support but no longer sells it.

Other details are as follows:-

Platform: - AIX (a UNIX variant)

Language: - BASIC

This project was 25 years old. The clients were facing some problems in one of its modules. So I was given the job of generating the test cases and testing them against the software. Or to find other bugs.

The data flows in the following manner through modules:-

Each client is provided with a software package. It connects to the main Infinity server which was situated in Atlanta, US. Any error message was sent to the spooler for print outs.

MY ROLE

BUGS FOUND

Following are few bugs I found in software:-

- When the host server wasn't found, the error message wasn't sent to the spooler. Even corrected this problem.
- When the policy PAB was chosen, a flag was set by mistake which later creates problem during logging. Just reported this bug. Could not rectify it.
- Few files were appended with some junk characters while software modifies it. Reported this bug. Could not rectify it.

OTHER WORK

- Created a shell script which copies the test file automatically to the main server for testing. Other people working in project told me later that they used this script till the end.
- Created a shell script which changes the permission of it as soon it is created.