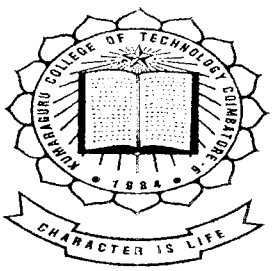


SURROUND CODE MIGRATION SYSTEM

By

Mithin.S

Reg. No : 71202621022

of

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION

ENGINEERING

In partial fulfillment of the requirements
for the award of the degree

of

MASTER OF COMPUTER APPLICATION

June, 2005

BONAFIDE CERTIFICATE

Certified that this project report titled

" SURROUND CODE MIGRATION SYSTEM "

is the bonafide work of Ms. ***MITHIN .S*** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Project Guide.

HOD/CSE.

Coimbatore:

Date : 24/04/2005

Certified that the candidate **Ms.MITHIN.S** with Register Number.**71202621022** was examined by us in the Project Work viva voce Examination held on:

Internal Examiner

External Examiner

ABSTRACT

ABSTRACT

"Amisys Advance" is a health care insurance product developed by AMISYS, LLC for HP3000 platform. This product is sold to health care insurance providers like CENTENE, UCARE, HEALTH ADVANTAGE etc. Those providers modified the code of Amisys Advance according to their policy specifications and conditions. These modified codes are known as Surround Codes.

Hewlett-Packard announces the withdrawal of support and obsolescence plan for the HP 3000 series of machines. So there is a need in migrating the Amisys Surround Codes. AMISYS decided to move to HP9000 platform. All the Surround Code for its customers is being migrated from HP3000 platform to HP9000 platform. Unified Process conversion methodology is followed for all phases based on ETVX model. Our project aims to convert code which is in HP3000 to HP9000 (HP-UX). The tasks include

- Conversion of the code using the TOOL developed by TCS.
- Execution of the shell scripts/jobs in the UNIX server.

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I sincerely thank **Dr.K.K.Padmanaban**, **Principal, Kumaraguru College of Technology**, Coimbatore for giving me the opportunity to undertake this full-time industrial project.

I am grateful to **Dr. S.Thangaswamy, H.O.D, Department of Computer Science and Engineering**, for providing me with valuable guidance in this project.

My sincere thanks to **Mr.A.Muthukumar,.M.C.A.,M.Phil., Course coordinator, M.C.A** and faculty guide in the college, **Mr.M.Manikantan, M.C.A., M.Phil., Lecturer Department of Computer Science and Engineering** for their constant motivation and suggestions made it possible for me to learn a lot and for their continued encouragement and support during the course of this project.

I would like to sincerely thank **Mr. K.Subramanian, Group Leader** and **Ms.A.Chandrakala, Project Leader**, of "AMISYS SURROUND CODE MIGRATION SYSTEM", TCS, for giving me the valuable opportunity to undertake this project in their esteemed organization.

I am very grateful to **Mr.R.Vinodh, Team Leader - JCL**, and **Mr.GuruPrasad Team member**,"AMISYS SURROUND CODE MIGRATION SYSTEM", TCS, Chennai who guided me in the organization and was instrumental in the successful completion of my project.

Finally I would like to thank the **entire team of "AMISYS SURROUND CODE SYSTEM"** Project, and the **HR Department TCS**, Ambattur and **my family** and **friends** who were instrumental in the successful completion of this project.

CORPORATE PROFILE

TATA CONSULTANCY SERVICES

CORPORATE PROFILE

Tata Consultancy Services is a division of Tata Sons, the holding company of the **$10.4 billion Tata Group**, India's best-known business conglomerate that was established in 1968.

TCS has been India's largest IT enterprise, as well as Asia's largest independent software and services organization. It is the single largest software services exporter from India, and services clients in over 55 countries around the world. With over 100 branches globally, TCS employs over 27,000 consultants and serve hundreds of clients, providing IT and business consulting services to organizations in government, business and industry in India and abroad. TCS is an organization with multiple **SEI CMM Level 5 centers**. It is the first company to be assessed at PCMM Level 4 in the world. It is India's first global billion-dollar software organization.

The main focus of TCS is on IT consulting/services, outsourcing and business process management. The industry offerings are structured into 10 industry practices including Banking, Financial services, Insurance, Telecom, Manufacturing, Media and Entertainment, Retail and Consumer goods, Transportation, Healthcare and Life sciences, Energy & Utilities and s-Governance.

TCS has developed IT solutions for over 800 clients all over the world. Some of the major clients of TCS are American Express, AT&T, Bombay Stock Exchange, British Airways, British Telecom, Citibank, Compaq, IBM Microsoft, Nokia, Reserve Bank of India, Ford Motor Company, General Motors, US Government - Department of Defense and the Singapore Airlines

TCS believes that IT is a key factor for social change and is committed to several community development ventures. The vision of TCS is to be among the global top 10 by 2010 and to help customers

achieve their business objectives by providing innovative, best-in-class consulting, IT solutions and services.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

LIST OF FIGURES

LIST OF ABBREVIATIONS

LIST OF ABBREVIATIONS

1.	**HP**	HEWLETT PACKARD
2	. **FTP**	FILE TRANSFER PROTOCOL
3.	**MPE**	MULTIPROGRAMING EXECUTION
4.	**JCL**	JOB CONTROL LANGUAGE
5.	**QTP**	QUERY TRANSACTION PROCESS
6.	**KSAM**	KEYED SEQUENTIAL ACCESS METHOD
7.	**CSAM**	C-INDEXED SEQUENTIAL ACCESS METHOD
8.	**IQA**	INTERNAL QUALITY AUDIT
9.	**EQA**	EXTERNAL QUALITY AUDIT
10.	**MF COBOL**	MICRO FOCUS COBOL

CHAPTER 1

INTRODUCTION

CHAPTER I

INTRODUCTION

1.1 PROBLEM DESCRIPTION

The AMISYS product is implemented on HP3000 platform. The AMISYS product is a HEALTH CARE INSURANCE project. Hewlett-Packard made an announcement on the withdrawal of support and obsolescence plan for the HP 3000 series of machines. So AMISYS decided to move it's product to HP9000. All the Surround Codes for it's customers like Health Advantage, Centene is being migrated from HP3000 platform to HP9000 platform.

The objective is to migrate the large and complex system to current technology platforms to provide better functionality. It is to redevelop the Cognos Powerhouse based 'AMISYS' application systems user interface to a UNIX ENVIRONMENT.

1.2 SCOPE OF THE SYSTEM

- Analysis for the existing application to determine its functionality.
- Extensive testing of the Shell script's to ensure that it works in exactly the same way as the existing application (Functionality not changed).
- Technical development for the application involves migrating

  - HP3000 JCL to HP-UX Posix Shell
  - HP3000 Command Files to HP-UX Posix Shell
  - HP3000 Cobol to HP-UX MF Cobol
  - HP3000 Copybooks to HP-UX Copybooks
  - HP3000 Questions Files to HP-UX Posix Shell
  - HP3000 Quick to HP-UX Quick
  - HP3000 Quiz to HP-UX Quiz

- HP3000 QTP to HP-UX QTP
- Fcopy to UNIX copy commands
- FTP to FTP on Unix
- KSAM to CISAM
- Suprtool to HP-UX Suprtool
- Maestro job scheduler to Maestro for Unix
- NBSpool and Vista to TBD replacement products
- Any ASKplus to SQL
- Security 3000 commands to UNIX equivalent commands.

1.3 SCOPE OF THE WORK

- Migrate the large and complex system to current technology platforms.
- Analysis for the existing application to determine its functionality.
- Extensive testing of the Shell script's to ensure that it works in exactly the same way as the existing application (Functionality not changed).
- Convertion of JCL and Suprtool
- Extensive conformity testing to ensure 100% migration
- Involvement in tool development/enhancement

1.4 SYSTEM ENVIRONMENT

This section provides the overview of the hardware and software requirements for the system migration

1.4.1 Hardware Specification:

- HP 9000 PC
- Pentium IV Processor
- 40 GB HDD
- 256 MB RAM

1.4.2 Software Specification:

- WINDOWS XP Professional, HP-UX
- ORACLE 9i version 9.0.3
- COGNOS Power House 8.43.B
- MF COBOL version 9.2
- Screen Jet(used as emulator for HP-UX)
- Edit Plus version 2
- Compare It version 3.0

CHAPTER II

ANALYSIS

An analysis model of the module is defined, which helps to specify the various requirements as well as facilitate easy design of the subsystem.

2.1 SYSTEM ANALYSIS

The AMISYS product is a HEALTH CARE INSURANCE product. Health Insurance is the protection against medical costs associated with illness and injuries. It usually provides direct payment or reimbursement for treatment of illnesses and injuries. The premium can be payable in a lump sum or in installments. The cost and range of protection depends on provider and the policy conditions

2.1.1 Existing System

The existing system "AMISYS" runs on HP 3000 MPE platform. The interface screens are developed in Cognos Powerhouse which is a specification-based development language. HP3000 JCL programs are written to use the Business Logics (written in HP3000 COBOL) in the HP 3000 server. TurboIMAGE acts as a database. Some of the files that are used to run the applications are

- HP3000 JCL,
- HP3000 COBOL,
- HP3000Command Files,
- HP3000 Quick,
- HP3000 Quiz,
- HP3000 QTP,
- HP3000 Copybooks,
- HP3000 Questions Files,

- FTP,
- KSAM Files,
- Suprtool,
- Maestro Job Scheduler,
- Adager Commands

2.1.1.1 Demerits of Existing System

The following are the demerits of the existing system:

- Inflexible to modifications/up gradations
- Lower productivity due to more development efforts
- Doesn't leverage other technology investments
- Expensive maintenance costs
- Obsolete

2.1.2 Proposed System

The proposed system should run on HP 9000 Platform. HP-UX refers to HP 9000 server. **ORACLE 9i** is the database used. In the Proposed system various files are used to run the application. Some of them are given below

- HP-UXPosixShell,
- HP-UXMFCobol,
- HP-UXCommandFiles ,
- HP-UXQuick,
- HP-UXQuiz,
- HP-UXQTP,
- HP-UXCopybooks,
- FTP on UNIX CISAM,
- HP-UXSuprtool,
- Maestro for UNIX,

- Oracle Commands.

2.1.2.1 Special Features of the proposed System

- Accessibility
- Business Flexibility
- Productivity
- Rapid Time-To-Market
- Provides Excellent User Interface

2.1.3 Migration Methodology

Unified Process conversion methodology is followed for all phases based on ETVX model ETVX is no proprietary methodology/process but is a general term to explain the activities and deliverables in the various phases of the project.

The following ***diagram*** represents the ***ETVX*** process.

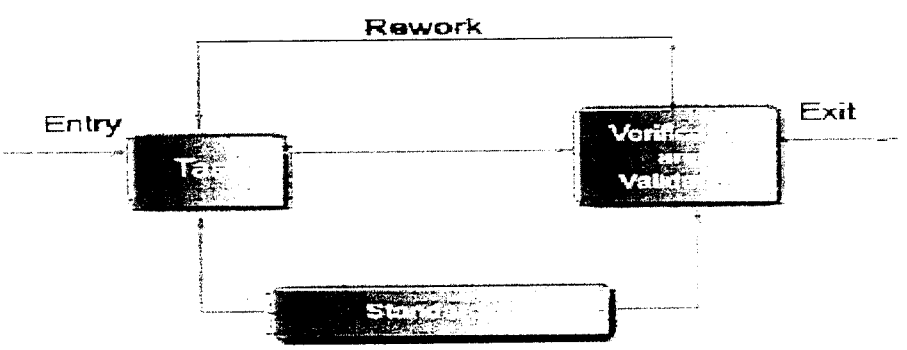


Figure 2.1: ETVX –Process Model

A high level description of the process has been described in table1.

Table 2.1: High Level Description of the ETVX Process

Entry Criteria	A checklist of conditions that must be satisfied before beginning the activity.
Tasks	A set of tasks that need to be carried out.
Validation	A list of validation tasks to verify the quality of the work items produced by the activity.
Exit criteria	A checklist of conditions that must be satisfied before the activity is completed.

2.1.4 Phases in Migration

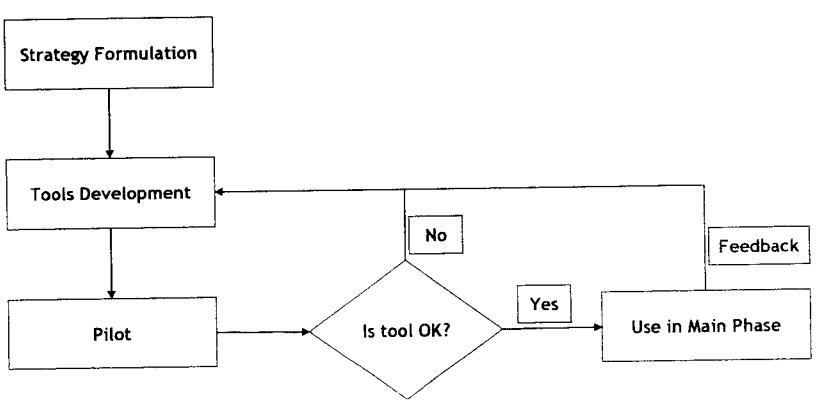


Figure 2.2: Phases in Migration

2.1.4.1 Analysis and Planning phase

The application that is to be migrated is analyzed and a detailed design is done. A conversion strategy document is prepared. The deployment strategy is finalized.

2.1.4.2. Application Migration phase

The application is migrated to the target platform and tested. The test cases and data are supplied by the customer. During this phase the codes from HP3000 will be migrated to a complete HP9000 platform. This phase would be carried out manually.

2.1.4.3. Acceptance testing phase

The final migrated application is tested for acceptance by the customer and deployed as per the deployment strategy decided in the analysis and planning phase.

2.1.5 Migration Process

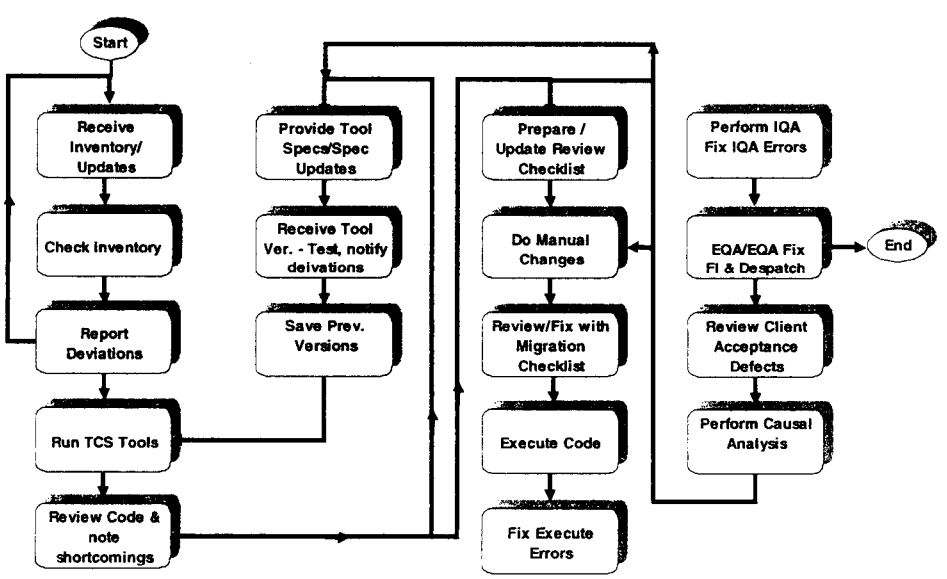


Figure 2.3: Migration Process

2.2 SOFTWARE REQUIREMENTS SPECIFICATION

2.2.1 Subsystems of Amisys

This System is made up of more than 40 sub-systems, each addressing a distinct functional requirement. For the purpose of acquiring an over all view of the system and its various sub-systems and their relative functionality, the entire sub-systems are categorized into the following three groupings.

2.2.1.1 SystemAdministrationSub- Systems

Sub-systems that control the configuration, access, and management of other sub-systems in AMISYS.

2.2.1.2 Functional sub-systems

Sub-systems that are directly related to the core functionality of AMISYS.

2.2.1.3 General Management Sub-systems/Utilities

Sub-systems that enhance the effectiveness of the application through interfaces to external systems or features that add utility to the users.

2.2.2 Non-Functional Requirements

2.2.2.1 Performance Requirements

The system will get the details from the database as quick as possible as the system uses the optimized queries.When using the high power server on a distributed system the system will work more efficiently.

2.2.2.2 Maintainability Requirements

The system will be easy to translate to any language needed for the client of AMISYS.The system can run on any platform only when the user has emulator software.

2.2.2.3 Security Requirements

Authorized access for the details stored in the database will be given for only persons who are authorized by the Provider.The access to the database will be full for all the authorized users.

2.2.2.4 Look And Feel Requirements

The screens will be created with a single template and it will not have drastic color changes.

2.2.2.5 Usability Requirements

The system will be easy to work for those who have already worked on the system in HP3000 platform as the functionalities are same and it will be easy for the new users if they had undergone a good training. The System being an interactive system it will be easy to use and work with.

2.2.2.6 Legal Requirements

The System is being developed to comply with the Health Insurance Laws in United States.Warning messages will be shown to the users if the action of the user does not comply with the laws.

2.3 USER INTERFACE SPECIFICATION

The System can separately run on HP-UX platform and it can also run on special software which acts as an emulator for HP-UX Operating System from any other Operating System. In developing environment ScreenJet, a

special emulator software for emulating HP-UX from Windows XP Professional. The user can use the product either from HP-UX platform itself or he can use emulator software.

The user interface screen in existing system is in COGNOS PowerHouse and in the proposed system it is being developed using HP-UX PowerHouse. The screens will be a menu based interactive interface screen. The screens will be having help facilities which can be used at any time.

2.4 ASSUMPTIONS

The System after developed is assumed to work as perfectly as when it was running on the HP3000 platform. The Strategies that are going to be used for migrating the system are assumed to be perfect.

2.5 CONSTRAINTS

The System is being developed for only HP-UX platform not for other operating systems. The Production from every team member should be consistent because every thing is sent to the onsite that is AMISYS frequently.

The new users should be trained vigorously to know the functionality of all sub systems and their modules. The Server that is going to be used should be very fast in responding as the client can use this system as a distributed system.

2.6 RISKS

When the user is not having the HP-UX Operating System he should have to use emulator software like Screen Jet. If the team members of the project lack speed in working other persons may have to suffer because all the modules are interconnected.

If the user does not have a good training in handling the system and if he makes mistakes then the system will not an efficient system.

CHAPTER 3

DESIGN

CHAPTER III

DESIGN

The design process maps the 'what to do' of the requirements specifications to the 'how to do' of the design specification. It identifies the software components, specifies relationships among components, defines program structure and provides a blue-print for implementation. The design phase starts with a process through which requirements are translated into a representation of the system.

3.1 SYSTEM ARCHITECTURE

The Software Architecture describes how the Existing and Proposed system are modeled. It specifies various components, technologies and terminologies that are used in the existing and in the proposed system. Being a migration project, the design of the proposed system architecture will be preferred more. Every technology used in the existing system should have to be mapped and migrated into the proposed system.

3.1.1. Existing system architecture

Various components of the Existing System (HP 3000 machine) in Multi Programming Executive (MPE) Operating System are Job Control Programs (JCLs), MPE COBOL Programs, Powerhouse Programs. TurboIMAGE database. Keyed Sequential Access Method (KSAM) files, Flat files, MPE intrinsics, C and FORTRAN functions and Third party products.

The following Figure depicts the system architecture of the HP 3000 machine in Multi Programming Executive (MPE) Operating system.

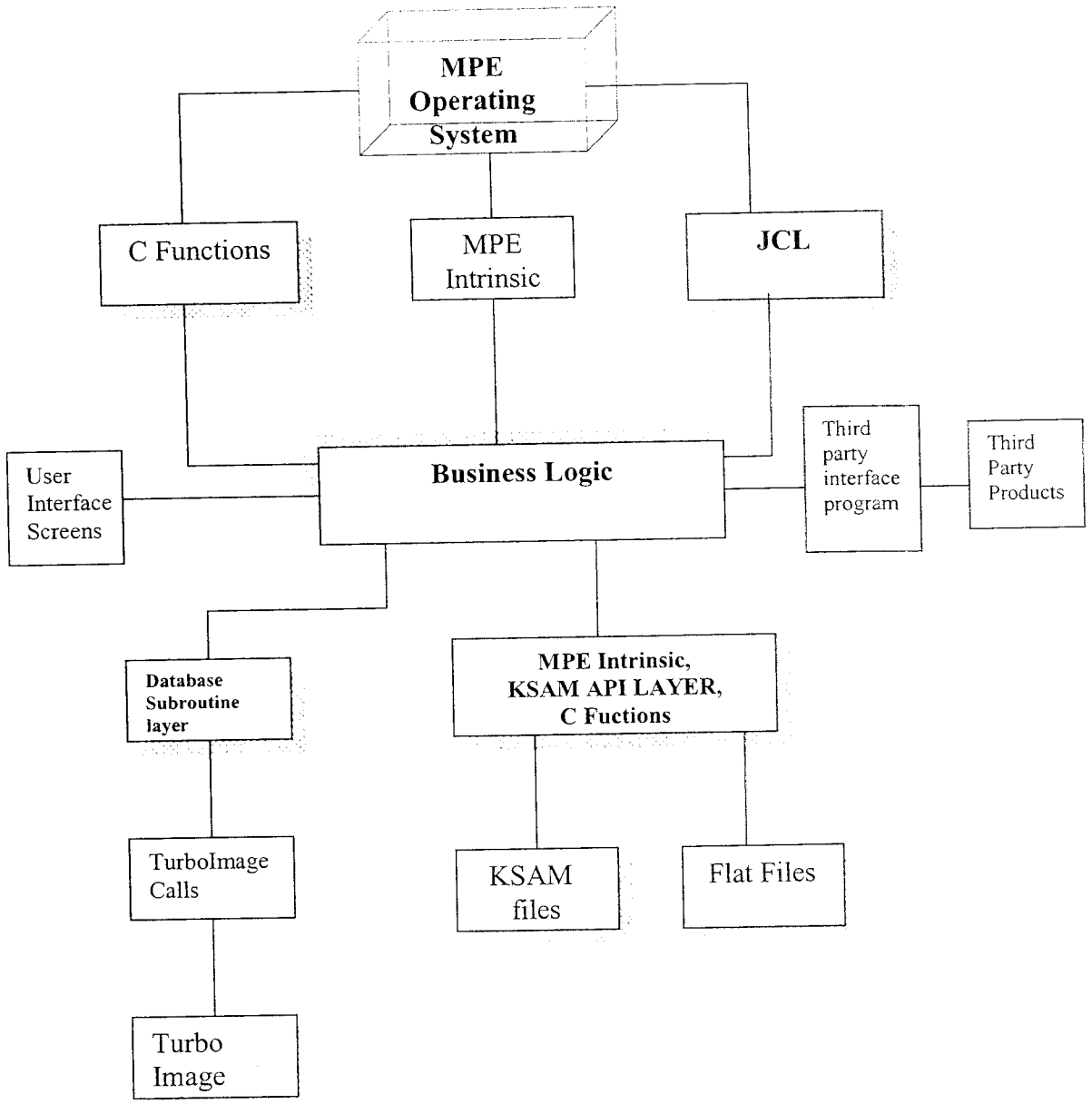


Figure 3.1: Architecture of HP 3000 MPE Operating System

3.1.1.1 Job control programs

The Job Control Programs are written in Job Control Language in MPE operating system. User defined job control programs can also be written and they can be used in MPE operating system. A job is a collection of related job steps. A job is identified by a JOB statement. Jobs are used when we want the computer to carry out several instructions on its own. There is no need for us

to log in when the job starts. The job prints out a report listing what it did and whether it was Successful.

Jobs placed in a series and entered through one input device form an input stream. The operating system reads an input stream into the computer from an input/output (I/O) device or an internal reader. The input device can be a card reader, a magnetic tape device, a terminal, or a direct access device. An internal reader is a buffer that is read from a program into the system as though it were an input stream.

3.1.1.2 MPE Cobol programs

Business Logics is the core logic area where the implementations deals with the policies provided by the health insurance company in various forms. These logics are written in MPE COBOL. This interacts with every part of the system. MPE COBOL is written exclusively for MPE Operating System.

3.1.1.3 Powerhouse programs

Powerhouse programs are written to develop the screens for the application. The version used is **POWERHOUSE 3.10**. A PowerHouse is part of an application development environment that allows creating business applications quickly and easily. PowerHouse for UNIX is divided into the following separate, yet integrated *components*:

3.1.1.4 PDL

The PowerHouse Definition Language (PDL) allows creating and maintaining a PowerHouse dictionary. As the backbone of all PowerHouse systems, the PowerHouse dictionary stores definitions of the data used by your PowerHouse applications.

3.1.1.5 QDESIGN, QUICK and Debugger

QUICK is an interactive screen processor with a powerful development tool. QUICK screens are used by data-entry operators and other end-users to process data quickly or to browse effortlessly through their files. **QDESIGN** is a screen designer. It is used to build data entry and retrieval screen systems. Debuggers analyze and control QUICK screens as they run.

3.1.1.6 QUIZ

QUIZ is the PowerHouse report writer. It takes the information which the user requests and gives it a structure. The information is automatically displayed in columns with headings. The key to the simplicity of QUIZ lies in its relationship with the data dictionary. QUIZ references the rules and standards defined in the data dictionary by the application designer when it formats your report.

3.1.1.7 Query Transaction Processor

QTP is a high-volume transaction processor. It gives the power to change the data in your files in one sweep. Because QTP is easy to use and designed for fast, high-volume file updating, it should be used by someone who is familiar with the implications of updating active files.

3.1.1.8 Turbo IMAGE database

Turbo IMAGE act as a database in HP3000 (i.e. MPE Operating System). There is a common interface layer for Turbo IMAGE calls in MPE. The database calls are layered and have access to the TurboIMAGE database.

TurboIMAGE is a database management system that operates only on HP 3000 computer systems. HP bundles TurboIMAGE with current HP 3000 systems, and has done so throughout most of the life of the 3000 platform. Consequently, the vast majority of HP 3000 applications are based on the

TurboIMAGE DBMS. TurboIMAGE offers programmatic access to the data through specialized system procedures called intrinsics. This distinguishes TurboIMAGE from relational databases such as Oracle or SQL/Server. TurboIMAGE is not a relational database. It is based on a network data access model. The TurboIMAGE intrinsic interface is very different from the embedded SQL that is used to access relational databases such as Oracle or SQL/Server.

TurboIMAGE/XL is a set of programs and procedures that can use to define, create, access, and maintain a database. The primary benefit of the TurboIMAGE/XL database management system is time savings.

3.1.1.9 KSAM Files

(KSAM Files – Keyed Sequential Access Method)

KSAM is the Keyed Sequential Access Method database system that comes with the MPE operating system. The KSAM access method makes it possible to access files sequentially, or directly, or using Keyed Access.

Keyed access requires that a field in each record be identified in advance as a key field. When data is written to a KSAM file, the KSAM access method not only writes the data, it also writes the key field to an index, which can then be used to retrieve the data record. This allows applications to access records from the middle of a file without having to search the file sequentially, or keep track of relative record numbers.

KSAM API layer is built to access the KSAM files. In other words, the Keyed Sequential Access Method (KSAM) is a method of organizing records in a file according to the content of key fields within each record. As implemented for the HP 3000 computer system, KSAM/3000 is similar to and competitive with other indexed sequential access methods. Every record in a KSAM file contains a primary key field whose contents determine the primary logical sequence of records in the file. Other key fields can also be defined so that the file can be sequenced in alternate orders. The order in which records are physically written to the file, the chronological order, can be the same as the primary key sequence or it can be unrelated to any logical sequence.

3.1.1.10 Flat files

KSAM file writes the key field to an index, which can then be used to retrieve the data record. Those data are written to FLAT files which are nothing but data files.

3.1.1.11 MPE intrinsics

Many programs use procedures or subroutines to handle recurring tasks. In the MPE operating system many of these tasks are performed by a set of system procedures called intrinsics. Intrinsics are available to any process which has the capabilities required to call the intrinsic.

Intrinsics are no different from procedures that you would write for yourself, except that the details of performing a task are invisible. The term intrinsic refers to any external system or subsystem. However, under MPE this term has a more specific meaning. To qualify as a true Hewlett-Packard documented and user-callable intrinsic, it must meet the following criteria:

- An intrinsic is a Hewlett-Packard supported external interface to an operating system or subsystem service.
- An intrinsic performs type and bounds checks on parameter values before it uses them, thus protecting the operating system and the user from one another.
- An intrinsic is documented in a Hewlett-Packard manual.
- If an intrinsic is enhanced, its interface, capabilities, and feature set remain backward compatible.
- A process may call an intrinsic from any Hewlett-Packard supported programming language.
- An intrinsic differs from other system library procedures Hewlett-Packard subsystems and applications can also provide interfaces that meet the definition of an intrinsic.

3.1.1.12 C and FORTRAN Functions

The Operating System consists of various C and Fortran functions that are used to carry out the functions of the operating system.

3.1.1.13 Third Party Products

The third party products used are:

3.1.1.13.1 Suprtool

It is a software tool for the HP 3000 and HP 9000 .It extracts data quickly. It does many data processing functions like copying, selecting, sorting, reformatting, printing using files and databases. It links data from several files into one.

It provides FAST serial processing of "flat" files, KSAM files, TurboIMAGE, Oracle, Allbase and Eloquence databases. Suprtool provides fast batch reporting on your HP 3000 and HP 9000. With Suprtool, we can select records from a flat file, keyed file, or database file to feed into your report program. Suprtool performs best when we are selecting less than 50% of the entries. Once selected, we can rearrange and sort them, or calculate a total. The report program then needs to read only the subset of qualified records.

Suprtool solves many other common problems and routine data processing tasks that normally would require programming, such as archiving and deleting old transactions, checking the total amount of a day's transactions, or verifying that the debits and credits balance out to zero Suprtool is often used as a simple report writer, though it may be described more accurately as a "data dumper" or "record printer" than "report writer".

3.1.1.13.2 Maestro job scheduler

Maestro job scheduler schedules the jobs that should have to be run on a particular date and time. Every details such as data, time, job no and

mandatory inputs should have to be given in advance so that these scheduler schedules the job at the right time.

3.1.1.13.3 NBSpool

NBSpool is the report writer. These Spools will collect the data in a report format and it will send the contents to the printer. This printed format of data will be used as the official one.

3.1.1.13.4 Qedit

Qedit aims to provide everything an MPE or HP-UX programmer could need to write COBOL, PowerHouse, or other programs, and to prepare documentation. Therefore, Qedit has Line mode for batch editing and full-screen mode for interactive editing. On HP terminals, Qedit's full-screen mode is called Visual mode.

Qedit's Visual mode is a powerful but friendly full-screen editor designed specifically for programmers. It gives full access to the editing capabilities of your terminal in block-mode, with low system overhead. You can move, copy, mark and delete blocks of text with Visual's cut-and-paste functions, and page backward and forward through your file with function keys. To use Visual mode, user must have an HP terminal or an HP terminal emulator (e.g., Reflection from WRQ).

3.1.1.13.5 FTP

FTP is an old and rugged protocol that manages to copy files across networks. It allows data exchange of systems that are almost incompatible to each other due to different operating systems, different file structures or different character sets. At the present time, there are many web interfaces implemented by FTP. Different security problems could occur while using of FTP. Firstly, FTP sends the password in clear text. If a hacker uses a sniffer, he can get a list of usernames and passwords, provided he is on the

same network segment or situated between client and server. Second, data is sent in clear text, too. This may not be a problem when downloading new software with the anonymous account, but it could be quite a headache when transferring confidential information.

3.1.2. Proposed system architecture

Various components of the Proposed System (HP 9000 machine) in Hewlett Packard – UNIX (HP-UX) Operating System are HP- UX Posix Shell Scripts, MF COBOL Programs, Powerhouse Programs., Oracle Database, Circular - Indexed Sequential Access Method(C - ISAM) files. Flat files, C functions and Third party products

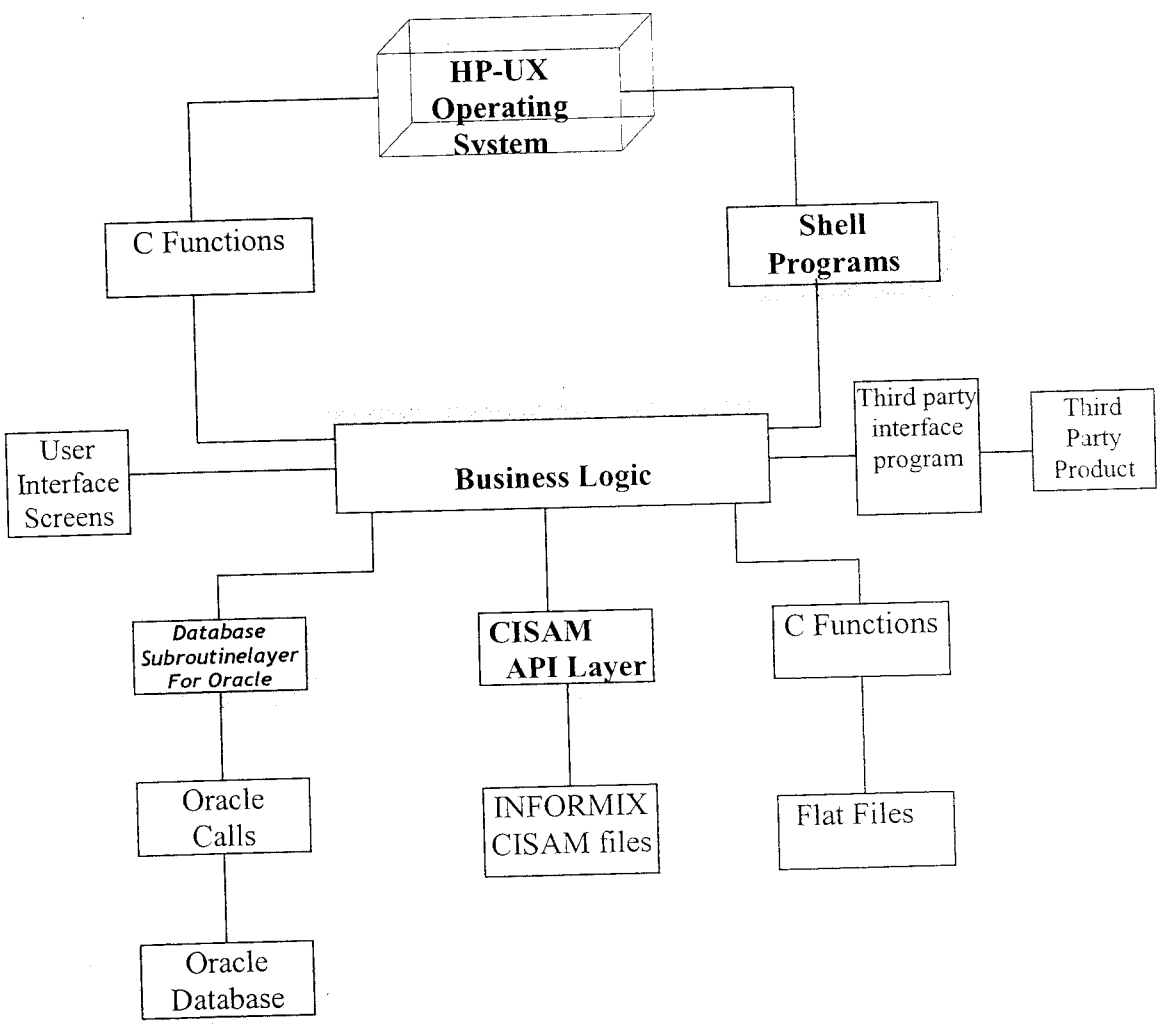


Figure3.2: **Architecture of HP (9000)- UX Operating System**

3.1.2.1 MF COBOL programs

Business Logics is the core logic area where the implementations deals with the policies provided by the health insurance company in various forms. These logics are written in MF COBOL. This interacts with every part of the system. MF COBOL is written exclusively for MF Operating System.

3.1.2.2 Powerhouse programs

Powerhouse programs are written to develop the screens for the application. The version used is POWERHOUSE 8.43.B.PowerHouse is part of an application development environment that allows creating business applications quickly and easily. PowerHouse for UNIX is divided into the following separate components:

3.1.2.2 PDL

The PowerHouse Definition Language (PDL) allows creating and maintaining a PowerHouse dictionary. As the backbone of all PowerHouse systems, the PowerHouse dictionary stores definitions of the data used by your PowerHouse applications.

3.1.2.4 QDESIGN, QUICK and Debugger

QUICK is an interactive screen processor with a powerful development tool. QUICK screens are used by data-entry operators and other end-users to process data quickly or to browse effortlessly through their files. QDESIGN is a screen designer. It is used to build data entry and retrieval screen systems. Debugger analyzes and control QUICK screens as they run.

3.1.2.5 QUIZ

QUIZ is the PowerHouse report writer. It takes the information which the user requests and gives it a structure. The information is automatically displayed in columns with headings. The key to the simplicity of QUIZ lies in its

relationship with the data dictionary. QUIZ references the rules and standards defined in the data dictionary by the application designer when it formats your report.

3.1.2.6 Query Transaction Processor

QTP is a high-volume transaction processor. It gives the power to change the data in your files in one sweep. Because QTP is easy to use and designed for fast, high-volume file updating, it should be used by someone who is familiar with the implications of updating active files.

3.1.2.7 ORACLE database

Oracle 9i improves on the features embedded within previous versions of the database to provide a scalable, high-performance platform with embedded support for data warehousing and business intelligence requirements included new extraction, transformation and loading (ETL) features, extended support for **on-line analytical processing (OLAP)** and improvements to data mining features.

An Oracle database stores data in physical data files and allows controlled user-access to these files through a set of operating system processes. These processes are started during the instance startup. Because they work silently, without direct user interaction, they're known as background processes. To enable efficient data manipulation and communication among the various processes, Oracle uses shared memory, known as Shared Global Area (SGA). These background processes and the shared memory segment together are referred as an Oracle instance.

3.1.2.8 C – ISAM Files

(C - ISAM Files – C - Indexed Sequential Access Method)

Informix C-ISAM files are used as a substitute for the KSAM files in the source environment. Informix C-ISAM is a library of C functions that

efficiently manages ISAM files. In the UNIX environment MF COBOL supports a modified version of C-ISAM. However the creation and loading of the C-ISAM files in MF COBOL on UNIX environment is done using the file handler utilities. This utility always produces a new file from the existing file unless an error condition occurs.

3.1.2.9 Flat files

The files produced by the C - ISAM files known as FLAT files which are nothing but data files.

3.1.2.10 MPE intrinsics

The MPE intrinsics are only for MPE Operating system.This was remove in the HP – UX platform itself

3.1.2.11 C Functions

The Operating System consists of various C functions that are used to carry out the functions of the operating system.

3.1.2.12 Third Party Products

The third party products used are

3.1.2.12.1 Suprtool

It is a software tool for the HP 3000 and HP 9000 It extracts data quickly. It does many data processing functions like copying, selecting, sorting, reformatting, printing using files and databases. It links data from several files into one.

It provides FAST serial processing of "flat" files, KSAM files, TurboIMAGE, Oracle, Allbase and Eloquence databases.

Suprtool provides fast batch reporting on your HP 3000 and HP 9000. With Suprtool, we can select records from an flat file, keyed file, or

database file to feed into your report program. Suprtool performs best when we are selecting less than 50% of the entries. Once selected, we can rearrange and sort them, or calculate a total. The report program then needs to read only the subset of qualified records.

Suprtool solves many other common problems and routine data processing tasks that normally would require programming, such as archiving and deleting old transactions, checking the total amount of a day's transactions, or verifying that the debits and credits balance out to zero Suprtool is often used as a simple report writer, though it may be described more accurately as a "data dumper" or "record printer" than "report writer".

3.1.2.12.2 Maestro job scheduler

Maestro job scheduler schedules the jobs that should have to be run on a particular date and time. Every details such as data, time, job no and mandatory inputs should have to be given in advance so that these scheduler schedules the job at the right time.

3.1.2.12.3 NBSpool

NBSpool is the report writer. These Spools will collect the data in a report format and it will send the contents to the printer. These printed formats of data will be used as the official one.

3.1.2.12.4 Qedit

Qedit aims to provide everything an MPE or HP-UX programmer could need to write COBOL, PowerHouse, or other programs, and to prepare documentation. Therefore, Qedit has Line mode for batch editing and full-screen mode for interactive editing. On HP terminals, Qedit's full-screen mode is called Visual mode.

Qedit's Visual mode is a powerful but friendly full-screen editor designed specifically for programmers. It gives full access to the editing capabilities of your terminal in block-mode, with low system overhead.

You can move, copy, mark and delete blocks of text with Visual's cut-and-paste functions, and page backward and forward through your file with function keys. To use Visual mode, you must have an HP terminal or an HP terminal emulator (e.g., Reflection from WRQ).

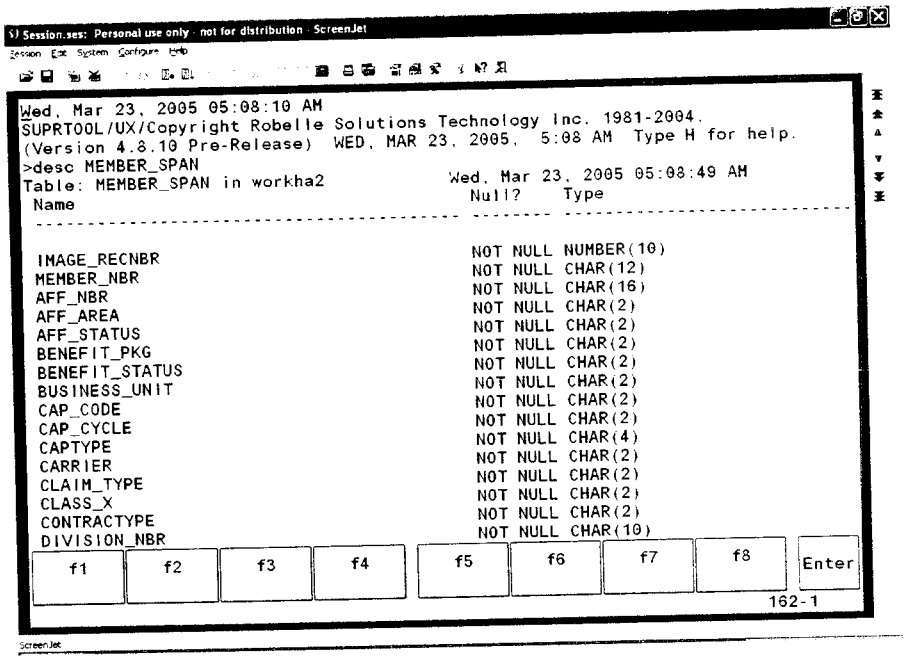
3.1.2.12.5 FTP

FTP is an old and rugged protocol that manages to copy files across networks. It allows data exchange of systems that are almost incompatible to each other due to different operating systems, different file structures or different character sets. At the present time, there are many web interfaces implemented by FTP.

Different security problems could occur while using of FTP. Firstly, FTP sends the password in clear text. If a hacker uses a sniffer, he can get a list of usernames and passwords, provided he is on the same network segment or situated between client and server. Second, data is sent in clear text, too. This may not be a problem when downloading new software with the anonymous account, but it could be quite a headache when transferring confidential information.

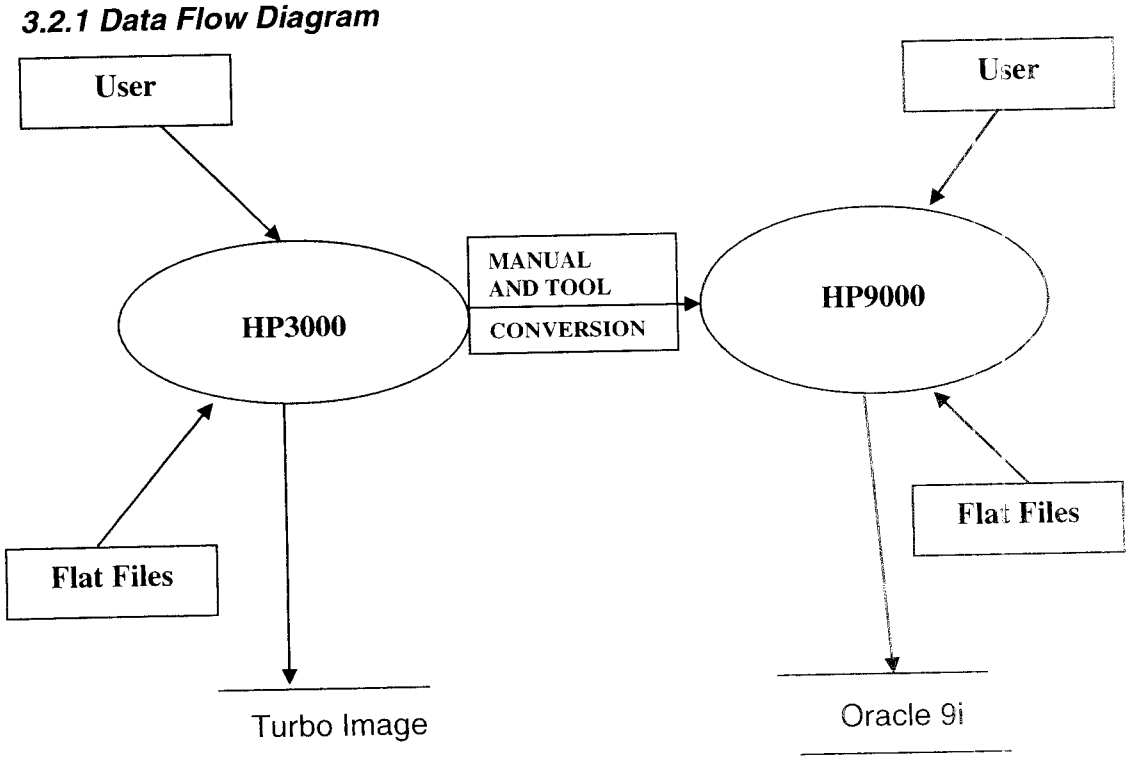
3.2 DATA DESIGN

A highly reliable and efficient Relational Database Management System is used. The client is supposed to use a distributed system as the client has more than 70 clients distributed geographically. But all the data will be finally updated in a centralized database. Many number of tables, procedures and triggers that are called for every module.

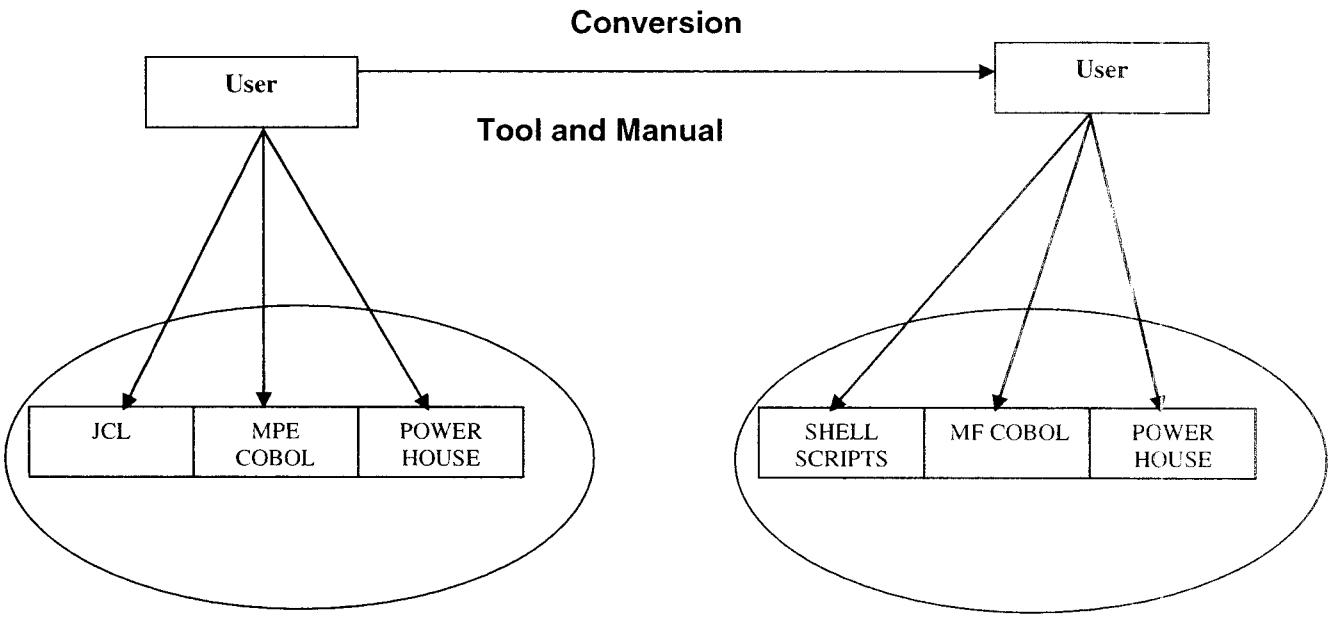


Example of a TABLE DESIGN – Data design

3.2.1 Data Flow Diagram



3.2.2 Module Diagram



3.3 TOOL DEVELOPMENT

3.3.1. Migration Tools Group

The Tools Group was established to develop and maintain tools that help in the conversion of source programs, screens and data from one environment to another. These tools have application in conversion and maintenance projects. The tools are developed using TCS developed standard conversion tools such as TABGEN, TREEGEN, Unix utilities like lex and yacc, public domain software like bison. The tools are developed for specific projects or in anticipation of projects.

Currently, the group is mainly involved in development / maintenance support of analysis/conversion tools for Euro. These tools had been previously developed by TCS and are currently being used a number of projects within TCS. The group will also develop any new tools required by user projects.

TCS believes that automation is key to good quality migration. The success of any large scale migration depends largely on the extent of automation. Automation contributes to the following:

- Consistency in migrating program code
- Reliability
- Reusability
- Efficiency in turn-around

TCS has a dedicated Tools Group that specialises in developing and customising tools according to the migration requirements. A large repository of tools has been developed by TCS. These tools are parser-based and have been proven in large migration assignments. TCS language transformation tools are based on the compiler technology.

3.3.2 Architecture of the Language Processing Tool

TCS Language Analysis and Migration tools are based on Compiler Technology. The migration tools have a front-end that takes the source program as the input. This front-end convert the source programs into an Intermediate Representation (IR) - an Abstract Syntax Tree (AST) that is suitable for migration. The front-end comprises the Lexical Analyser Module and the Parser Module for the source language. This is built using the language constructs and grammar based on the programming language manual.

The back-end works on this AST and typically converts it into target language. The back-end comprises the Transformation Module, the unparser module (to convert the AST to physical representation of the source program) and the Formatter module (to preserve the original alignment of the code and introduce correct alignment for the migrated code).

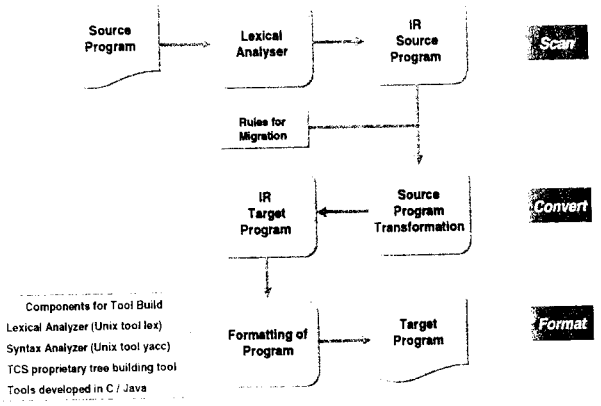


Figure 3.3: Conversion Tool Architecture

3.3.2.1 Lexical Analyser module

This Module performs the job of breaking up the input source program into tokens or lexemes. In addition this module removes white spaces and recognises comments in the input source code to avoid over-loading the parser. In addition, the parser takes care of context switching when a copybook statement is present in the input, thus making it transparent to the Parser module.

3.3.2.2 Parser module

This Module performs syntactic analysis and constructs an Intermediate Representation of the source program using the tokens / lexemes returned by the Lexical Analyser Module. This IR is usually in an Abstract Syntax Tree form. The basic assumption is that the input program is free from syntax errors. The parser performs rudimentary syntactic checks. In case of any syntax errors in the input, the parser reports all of them and terminates the processing. The IR closely resembles the source language.

3.3.2.3 Transform module

This module performs the language transformations using the AST. It traverses the AST and converts every line of the source code present in the program into equivalent code in the target language syntax. This module handles the following cases:

- Conversion of language syntax to equivalent syntax in the target
- Emulation of concepts in the source language into the target language

The transformations are performed based on the transformation rules specified in the AST. These rules are provided as specifications to the tool.

3.3.2.4 Unparser module

This module performs a traversal of the source AST and generates all the unparsed specifications into a file. This file is unformatted and unreadable and will require a formatter to make it readable. This file contains the converted target program.

3.3.2.5 Formatter Module

This module processes the output generated by the unparser module and generates the target program in the desired format. This is semantically equivalent to the source code.

3.3.3 Components and description of the Tool used

The tool consists of a lexer, a parser and a tree-walker. The lexer accepts the input stream and separates it into tokens. The valid syntaxes in which these tokens can be present in the input are implemented in the parser, which will generate a parse tree on encountering a match to one of those syntaxes.

The conversion of valid input to the target language (UNIX) is done in the tree-walker, which is used to traverse the parser-generated tree and generate the output code.

The input to our tool comes from QDESIGN, which is the PH screen development component. Screen designers use QD statements and procedures to create screens. The output would be the UNIX Shell Scripts.

3.3.3.1 Compilers

A compiler is a tool that converts a source code to a target code.

SOURCE CODE ⟶ COMPILER ⟶ TARGET CODE

In our project,

SOURCE CODE is QD.
TARGET CODE is UNIX Shell Scripts

3.3.3.2 ANTLR

ANother Tool for Language Recognition, is a language tool that provides a framework for constructing recognizers, compilers, and translators from grammatical descriptions containing Java, C++, or C# actions.

ANTLR tool accepts grammatical language descriptions and generates programs that recognize sentences in those languages. As part of a translator, the grammars may be augmented with simple operators and actions to tell ANTLR how to build ASTs (Abstract Syntax Tree) and how to generate output.

ANTLR knows how to build recognizers that apply grammatical structure to three different kinds of input: (i) character streams, (ii) token streams, and (iii) two-dimensional trees structures. Naturally these correspond to

lexers, parsers, and tree walkers. The syntax for specifying these grammars, the *meta-language*, is nearly identical in all cases.

ANTLR is just a tool that helps to build software by automating the well-understood tedious components, but does not attempt to let you specify an entire compiler, for example, in a single description. Tools that claim this sort of thing are great for writing amazing "one liners" for publishing journal papers, but fail miserably on real projects.

By using ANTLR Tool we can minimize the number of lines in the coding which in turn increases the speed and efficiency of the tool compared to other tools.

ANTLR can create lexers, parsers and ASTs. ANTLR is more than just a grammar definition language. However, the tools provided allow one to implement the ANTLR defined grammar by, automatically generating lexers and parsers (and tree parsers) in UNIX.

ANTLR implements a PREDICATED – LL (k) parsing strategy and affords arbitrary look ahead for disambiguating the ambiguous.

In the main program, the Lexer is instantiated first and the input source code file is passed as input. Then, the Parser is instantiated and the lexer is passed as its input. Then, for the Tree walker that is instantiated next the parse tree generated by the parser is given. The output from this main program is the target code in UNIX Shell Scripts.

ANother Tool for Language Recognition is a language tool that provides a framework for constructing recognizers, compilers, and translators from grammatical descriptions containing Unix Shell Scripts actions.

Grammar files analysis:

Grammar file contents are explained below.

3.3.3.3 The Lexer

A source file is streamed to a lexer on a character by character basis by some kind of input interface. The lexer's job is to quantify the meaningless stream of characters into discrete groups that, when processed by the parser, have meaning. Each character or group of characters quantified in this manner is called a token.

Tokens are components of the programming language in question such as keywords, identifiers, symbols, and operators. (Usually) The lexer removes comments and white space from the program, and any other content that is not of semantic value to the interpretation of the program.

The lexer converts the stream of characters into a stream of tokens which have individual meaning as dictated by the lexer's rules. The stream of tokens generated by the lexer is received by the parser. A lexer usually generates errors pertaining to sequences of characters it cannot match to a specific token type defined by one of its rules.

3.3.3.4 The Parser

A lexer groups sequences of characters it recognizes in the character stream into tokens with individual semantic worth, it does not consider their semantic worth within the context of the whole program. This is the job of the parser.

Languages are described by a grammar. The grammar determines exactly what defines a particular token and what sequences of tokens are decreed as valid. The parser organizes the tokens it receives into the allowed sequences defined by the grammar of the language.

If the language is being used exactly as is defined in the grammar, the parser will be able to recognize the patterns that make up certain structures and group these together.

If the parser encounters a sequence of tokens that match none of the allowed sequences of tokens, it will issue an error and perhaps try to recover the error by making a few assumptions about what the error was.

Usually the parser will convert the sequences of tokens that it has been deliberately created to match into some other form such as an Abstract Syntax Tree (AST).

An AST is easier to translate to a target language because an AST contains additional information implicitly by nature of its structure. Effectively, creating an AST is the most important part of a language translation process.

The parser generates one or more symbol table(s) which contain information regarding tokens it encounters, such as whether or not the token is the name of a procedure or if it had some specific value, the symbol tables are used in the generation of object code and in type checking, for example, so that an integer cannot be assigned to a string or whatever.

ANTLR uses symbol tables to speed up the matching of tokens, in that an integer is mapped to a particular token, then instead of matching the string that would compose a textual description of that token, the integer that represents that token is matched. This is a lot quicker.

A parser usually generates errors pertaining to sequences of tokens it cannot match to the specific syntactical arrangements allowed, as decreed by the grammar.

Both lexers and parsers are recognizers. However, traditionally, lexers are associated with processing streams of characters and parsers are associated with processing streams of Tokens.

3.3.3.5 The Tree Parser

Parsing is the application of grammatical structure to a stream of input symbols. ANTLR takes this further than most tools and considers a tree to be a stream of nodes, albeit in two dimensions. In fact, the only real difference in ANTLR's code generation for token stream parsing versus tree parsing lies in the testing of lookahead, rule-method definition headers, and the introduction of a two-dimensional tree structure code-generation template.

ANTLR tree parsers can walk any tree that implements the AST interface, which imposes a child-sibling like structure to whatever tree data-structure you may have.

While tree parsers are useful to examine trees or generate output from a tree, they must be augmented to handle tree transformations. ANTLR tree parsers support the buildAST option just like regular parsers. Without programmer intervention, the tree parser will automatically copy the input tree to a result tree.

Each rule has an implicit (automatically defined) res0ult tree; the result tree of the start symbol can be obtained from the tree parser The various alternatives and grammar elements may be annotated with "!" to indicate that they should not be automatically linked into the output tree. Portions of, or entire, subtrees may be rewritten.

3.3.4 Tool Development Aids

TCS has developed a set of proprietary tools for this purpose. These tools are specification oriented. A set of high level specifications supplied to these tools yield source code in C that can be compiled and executed to achieve the desired function. The following proprietary tools are used for developing/customising Filters:

- lex, a UNIX proprietary tool is used for Lexical Analyser Module

- yacc, a UNIX proprietary tool is used for Parser Module
- treegen is used for AST building. The set of rules specified in this AST help in performing the transformations
- unparser is used for outputting the target AST
- Formatter is used to format the intermediate output from the Unparser.

3.3.5 Languages used in tool development

- Lex and yacc specifications
- Treegen specifications
- C – for the transformation routines

The specifications to all these tools are integrated and compiled to give the resultant filter. The feature of generating program converters using generic tools facilitates efficient and accurate implementation of program conversions.

CHAPTER 4

SYSTEM IMPLEMENTATION

CHAPTER IV

SYSTEM IMPLEMENTATION

AMISYS SURROUND CODE MPE/JCL TO HP/UX SHELL SCRIPT CONVERSION TOOL

4.1 OBJECTIVE & ENVIRONMENT SET UP

To convert MPE/JCL programs to HP/UX POSIX shell script.

The tool runs on a platform that supports Java. Prerequisites are:

- Java Development kit (jdk-1.3 or above)
- Antlr package(antlr-2.7.1) (sent along with the tool)

Add following path to CLASSPATH at the end as below

Command prompt: **set CLASSPATH =% CLASSPATH %; c:\antlr-2.7.1;**

Assuming the class path batch file is run from desktop and your desktop path is C:\Documents and Settings\user id\Desktop the class path file should contain the following lines

- cd ../../.
- set PATH=%PATH%;C:\jdk1.3\bin;
- set CLASSPATH=;.;c:\jdk1.3\lib;
- set CLASSPATH=%CLASSPATH%;c:\ToAmisys\antlr-2.7.1;
- cd ToAmisys\Tool
- cmd

Save this file with <filename>.bat in desktop. You can run this file every time to use the tool

4.2 TOOL AND ITS FOLDERS

The MPE/JCL tool folder which is named as **"ToAmisys** Contains two main folders inside **(folder names are given in italics)**

1) ***antlr-2.7.1***---------◊ main parser

2) ***Tool***

The folder ***"Tool "***contains the following folders

1) ***antlr*** --- Sub parser

2) ***Controlfiles*** --- control files for "update" and "insert"MPE Commands are generated here

3) ***Output*** --- Converted shell scripts are generated here

4) ***inputpgm*** --- Programs to be converted are copied here

5) ***Reports*** --- Error, Warning report files are generated here With .report extn .

6) ***Scripts*** --- Script files for "update" and "insert" commands are generated here

7) *UCAREWarnings* --- This is folder mainly used for ucare client .It Gives general warning about Amisys product Version 10 to 11 Changes to be made in JCL.

8) ***xref*** --- This Folder is the most important of all as the input files we copy inside this folder decides the Correct conversion for each client. Every time we get a new inventory, the contents of the files inside this folder has to be changed.

"Xref" folder contains the following files and the contents of the files need to be replaced, every time we get a new inventory, without changing the names of the files.

1) ***SUBFILELIST*** --- If our inventory contains powerhouse components like quiz files and qtp files, then all the subfiles used needs to be listed inside the file.

2)*UcareDirectory Structure* ---- This file contains the Client specific directory structure mapping.

3) *UcareVariable Status* --- This file is only for UCARE Client. This will contain list of tables and fields that needs to be converted

4.3 UPDATING THE TOOL

Whenever the tool is enhanced with features, only updated files are sent to all the members.

To update the tool with more fixes and features

- Unzip the file provided into C:ToAmisys/Tool
- The xref directory needs to be updated with the current subfilelist and directory structure mapping file.

Note: The main parser antlr-2.7.1 which is present inside the "ToAmisys" remains the same, only the "Tool" folder present inside the "ToAmisys" directory is updated

4.4 RUNNING THE TOOL

1) Set the class path and Update the files inside the "xref" directory according to the client.
2) Place the JCL program in 'inputpgm' subdirectory under Tool
3) Use the following command

Command prompt: java Main < input filename > DEFAULT MODE
Command prompt: java Main –u < input filename > Client UCARE
Command prompt: java Main –c < input filename > Client CENTENE
Command prompt: java Main –t < input filename > Client TRINITY

The output (Converted Shell Scripts) will be generated in the output folder with .sh extension added to the input file name.

Note: complete name of input file should be typed with its extension. If no input program name is given, all the programs inside the inputpgm folder will be converted.

4.5 COMMON ERRORS ENCOUNTERED

There are various errors encountered after converting the code they are shown below :

- ***For Input File not found,***

  export statement with the correct path

- ***For Record Length not found,***

  REC <record length>

- ***For Open a Database / No DB OPEN statement used,***

  USEQ $AMISYS_PUB/dbopen before the SELECT statement

- ***For reducing the Compilation time of a program,***

  where rownum < 10000 in all SELECT Statements.

- ***For If there is no such input file***

  touch <path >/<file name>.dat

- ***For errors that arises due to NULL Values in the Database.***

  NULLIF

- ***For EOF on stdfile.***

  Xeq

  Exit

  !EOSUPRTOOL

CHAPTER 5

SYSTEM TESTING

CHAPTER V

SYSTEM TESTING

Testing of the application is done at various stages of development of the application to find and fix bugs at an earlier stage and get an error free, high quality product as output of the process.

The major quality *objectives of testing* are:

- The developed system is functionally correct and reliable.
- The developed system meets the objectives of the client.

5.1 TEST CASES

The test cases are common for every JCL program that has to be converted to a POSIX Shell Program in any module. The test cases that have to be carried out are shown in Appendix 2.

5.2 TEST REPORTS / CHECK LIST

Test Reports or checklist for JCL to POSIX SHELL SCRIPT migration has to be filled after every conversion which help in reviewing the code.The checklist is shown in Appendix 3.

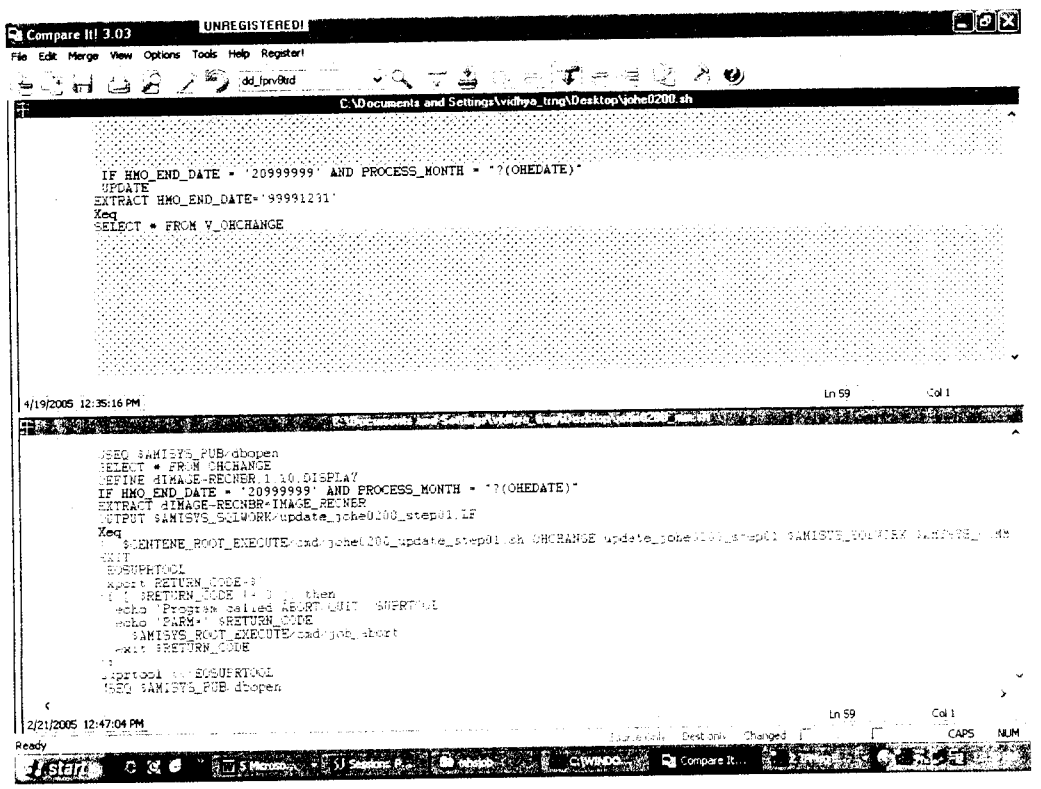
5.2 PEER REVIEWS

Peer reviews by other members of the team during the migration helps in identifying bugs and errors in the earlier stages. As a result of this, most of the bugs are curtailed in the development stage and not escalated to Internal Quality Audits (IQA) and External Quality Audits (EQA).

5.3 INTERNAL QUALITY AUDIT (IQA)

After a program is converted and assigned for delivery to the client, testing of that program is done by the IQA team. The screen is also tested simultaneously whenever the corresponding Powerhouse components are available to verify that it is functionally correct.

Testing the updated version with 'Compare it' tool:



5.5. EXTERNAL QUALITY AUDIT (EQA)

After the completion of IQA, EQA is carried by persons out of this project. This testing emphasizes on the user friendliness of the system.

5.6 TESTING PHASES

These are the broad testing phases: -

Testing Type	Explanation
Unit Testing	Testing of individual screens against the desired screen functionality.
Link Testing	Non- functional Testing of the ability to view/launch screens from other screens, based on the screen groups defined.
System Testing	An end-to-end testing focused on functionality, based on pre-defined test cases.
Regression Testing	All retests, to cross check the already tested functionality for consistency after changes in the environment, bug fixes.
Acceptance Testing	Testing the system to decide on go live of the system.

Table 5.1:Testing Phases

CHAPTER 6

FUTURE TRENDS AND CONCLUSION

CHAPTER VI

FUTURE TRENDS AND CONCLUSION

FUTURE TRENDS

The tool developed by TCS is well *intelligent* in converting 80 % of the code from HP3000 to HP9000 OS.

The architecture of the Tool is developed in such a way that it can accommodate to convert any new patterns which might evolve during the future Clients' code.

CONCLUSION

The AMISYS is a comprehensive and flexible solution to health care insurance system. The project assists the insurance companies and the customers, help provide and avail of health care services, respectively.

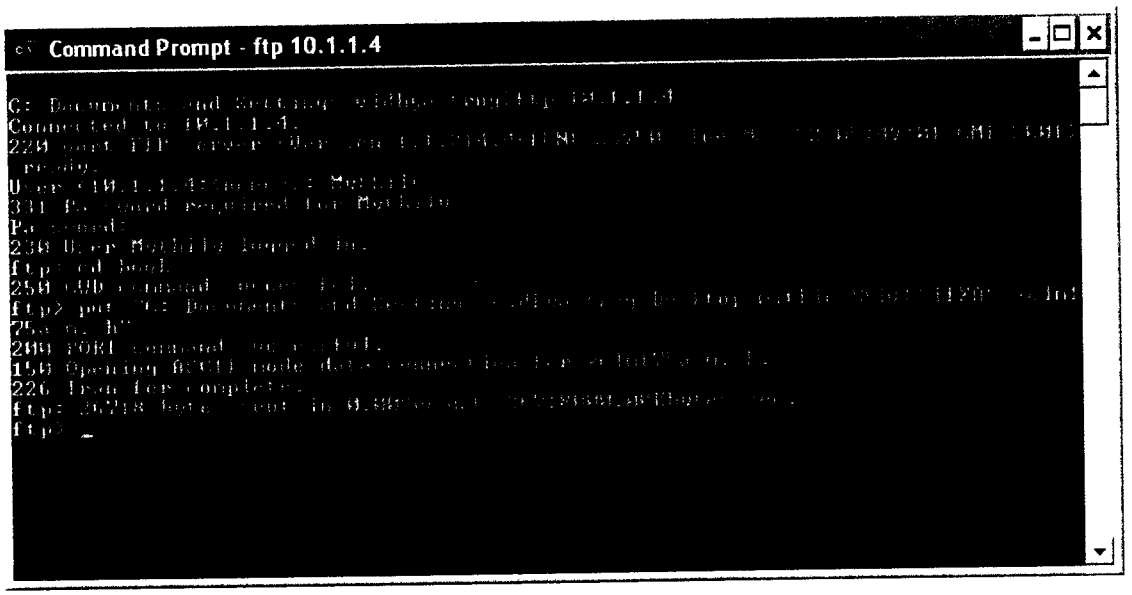
Moreover this system works in present day's technology. The system is migrated to a new platform which can be used by more than 70+ clients of AMISYS. Even though the system can be upgraded, the usage of the POSIX shell script would be the same because POSIX is the prescribed standard for UNIX environment.

The system works perfectly in the client's environment and it is ready for delivery to AMISYS' clients.

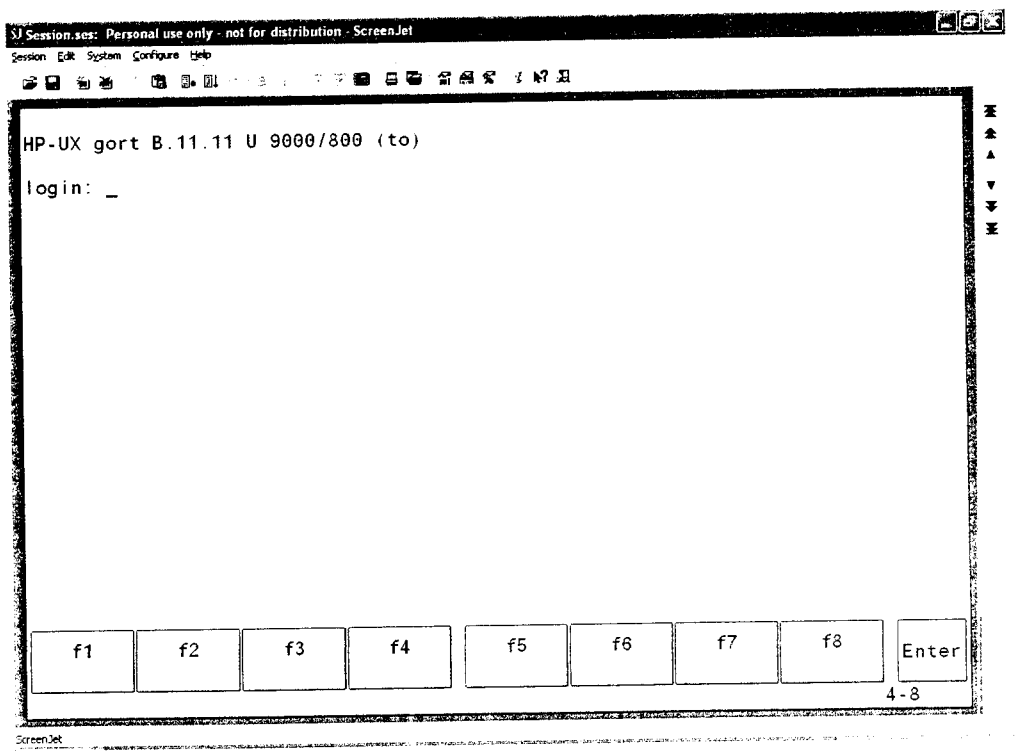
APPENDIX 1

SCREENS SHOTS

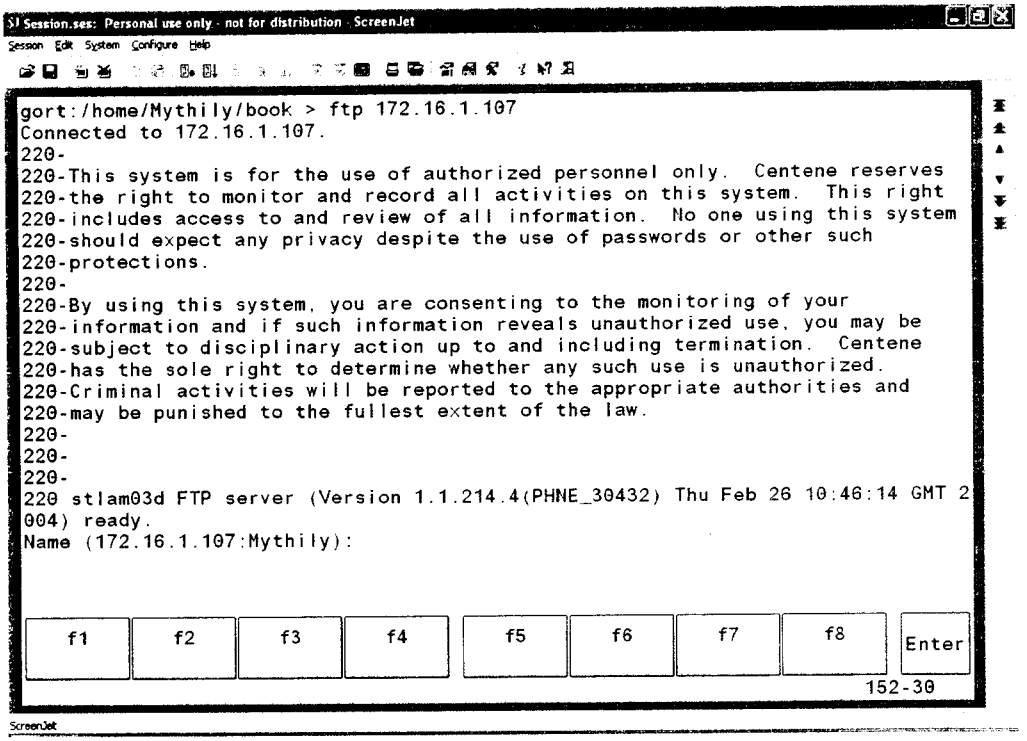
Converting from HP3000 To Windows



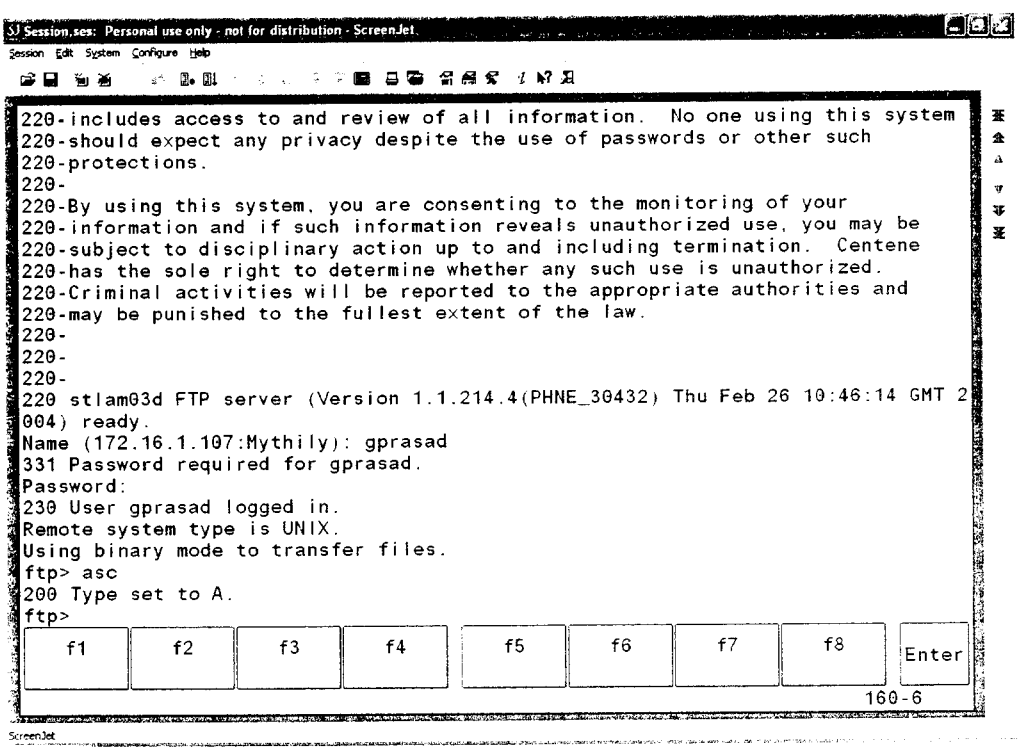
Logging into the Screenjet



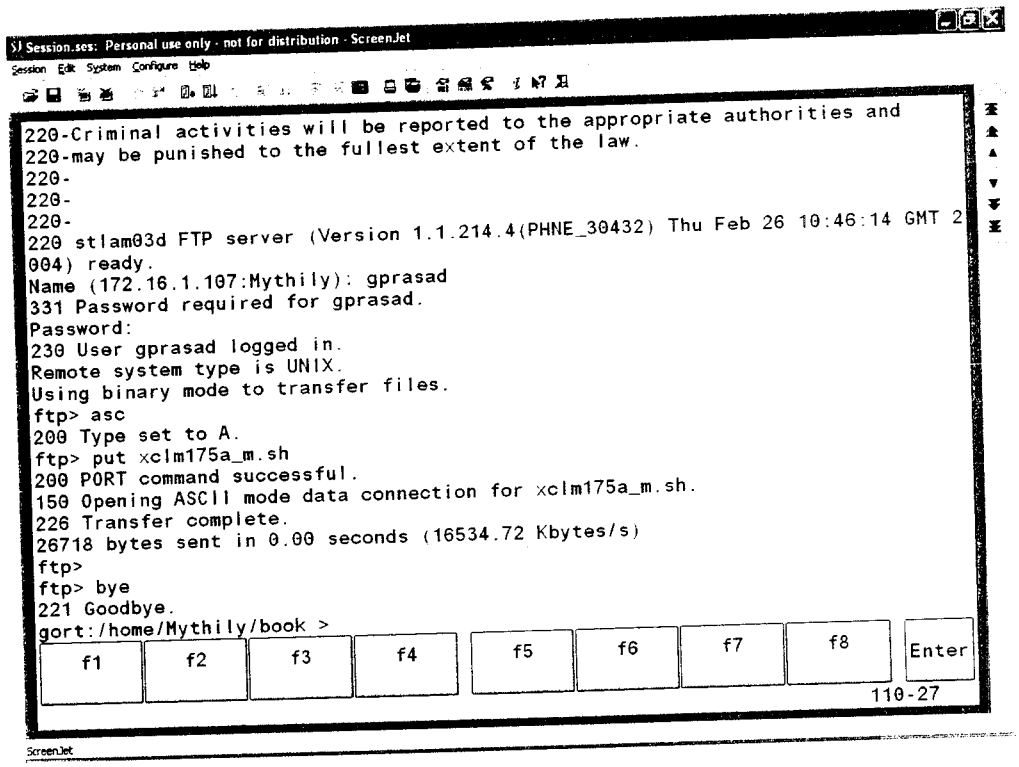
Transfer of files from Windows to Gort using FTP (Step 1)



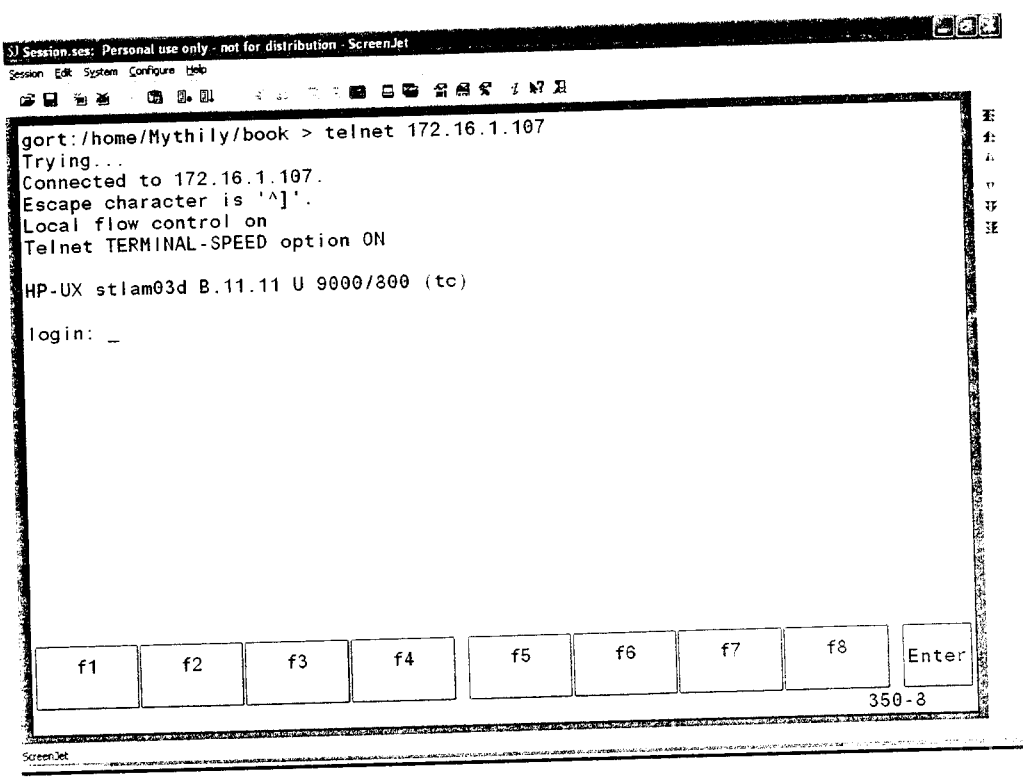
Transfer of files from Windows to Gort using FTP (Step 2)



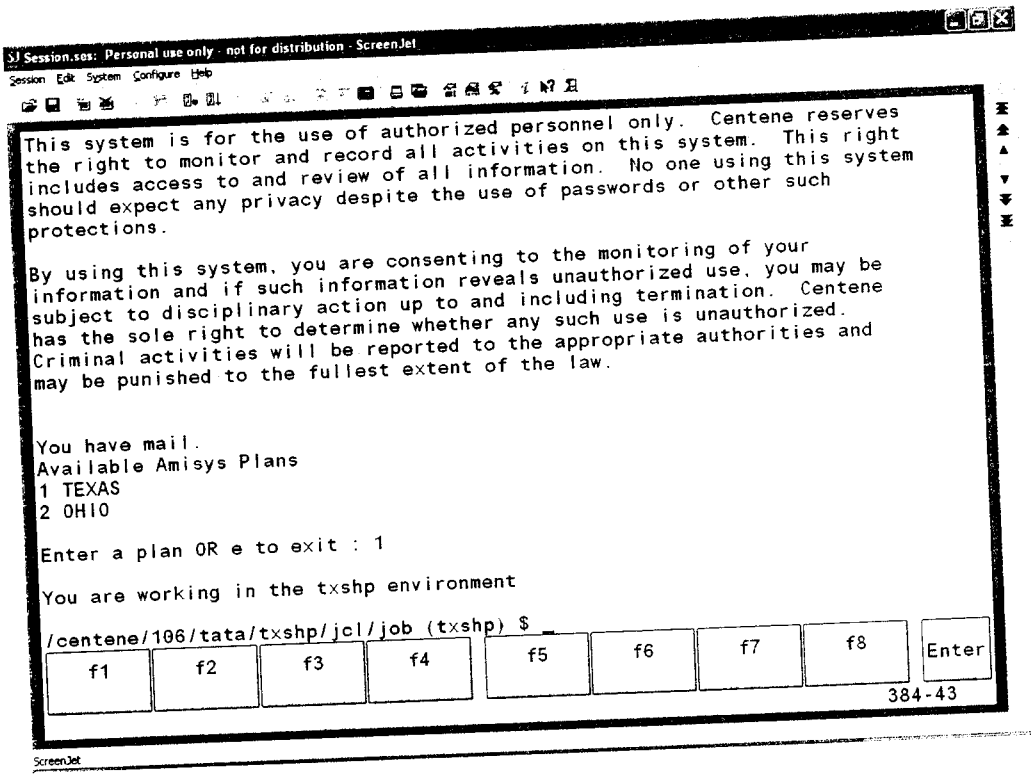
Transfer of files from Windows to Gort using FTP (Step 3)



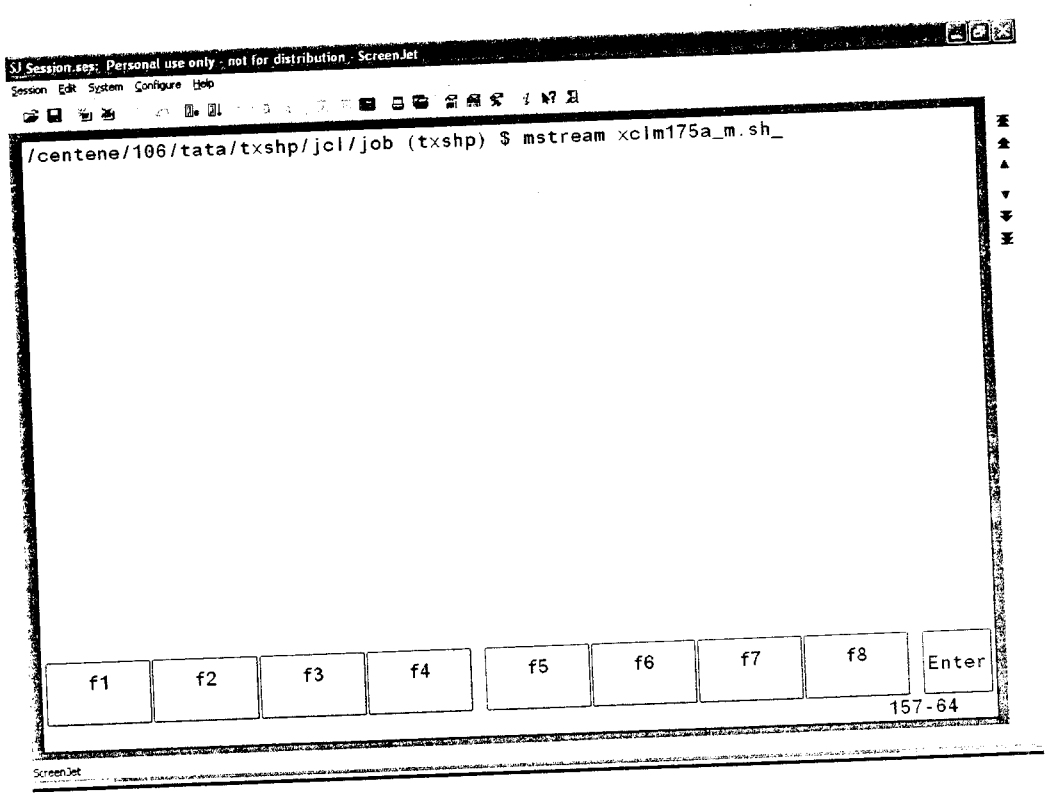
To RUN and EXECUTE the program / Job



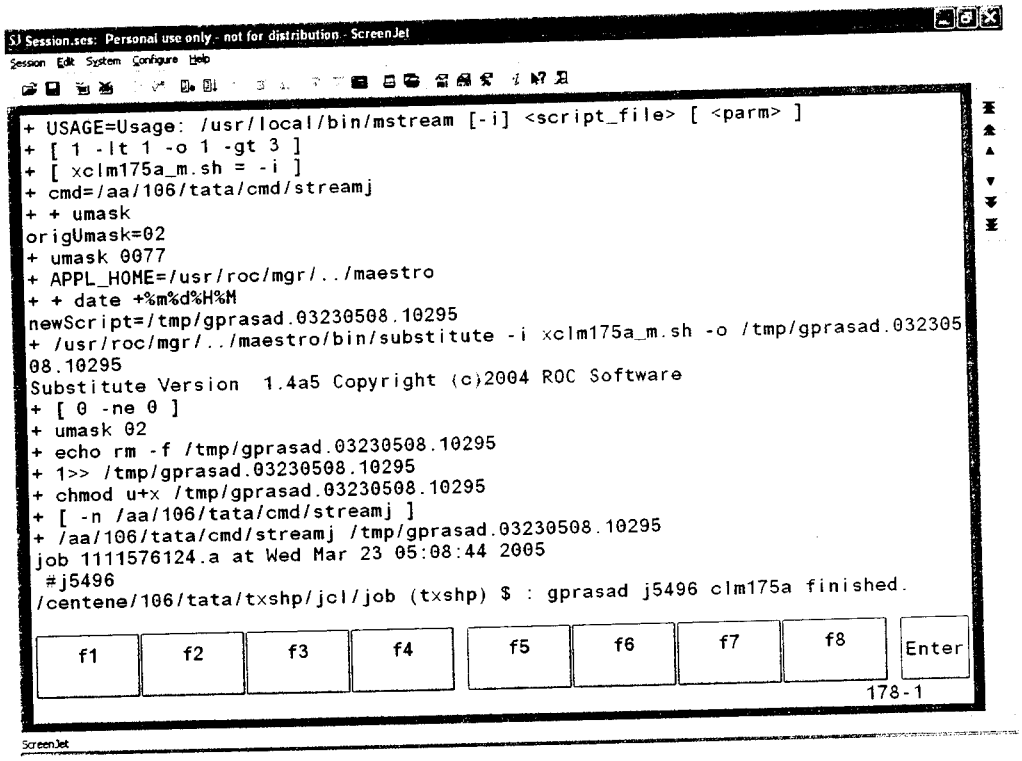
Selecting the AMISYS Job plans



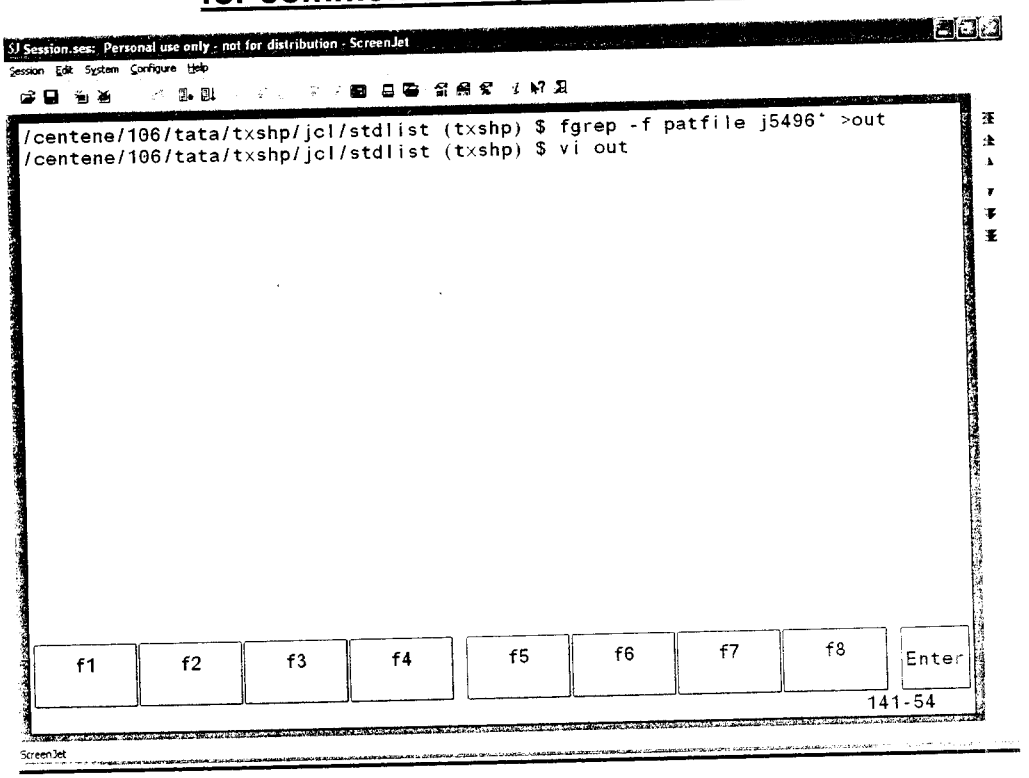
Running the Job using "mstream"



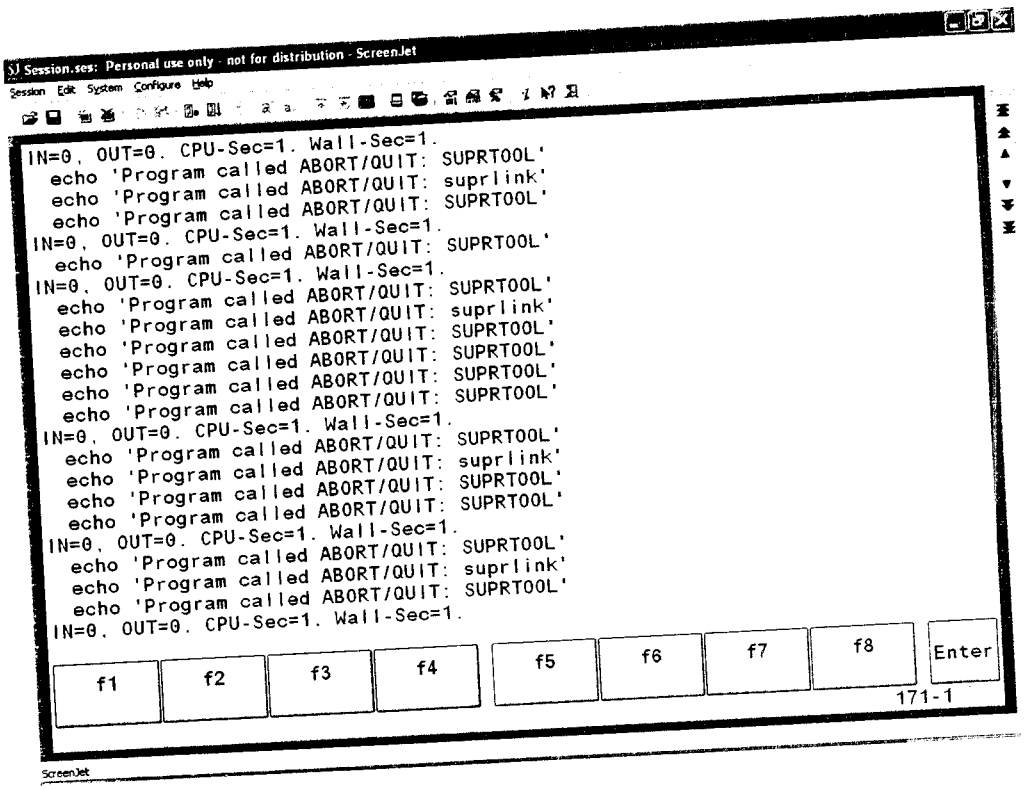
Job Finishes without any Errors

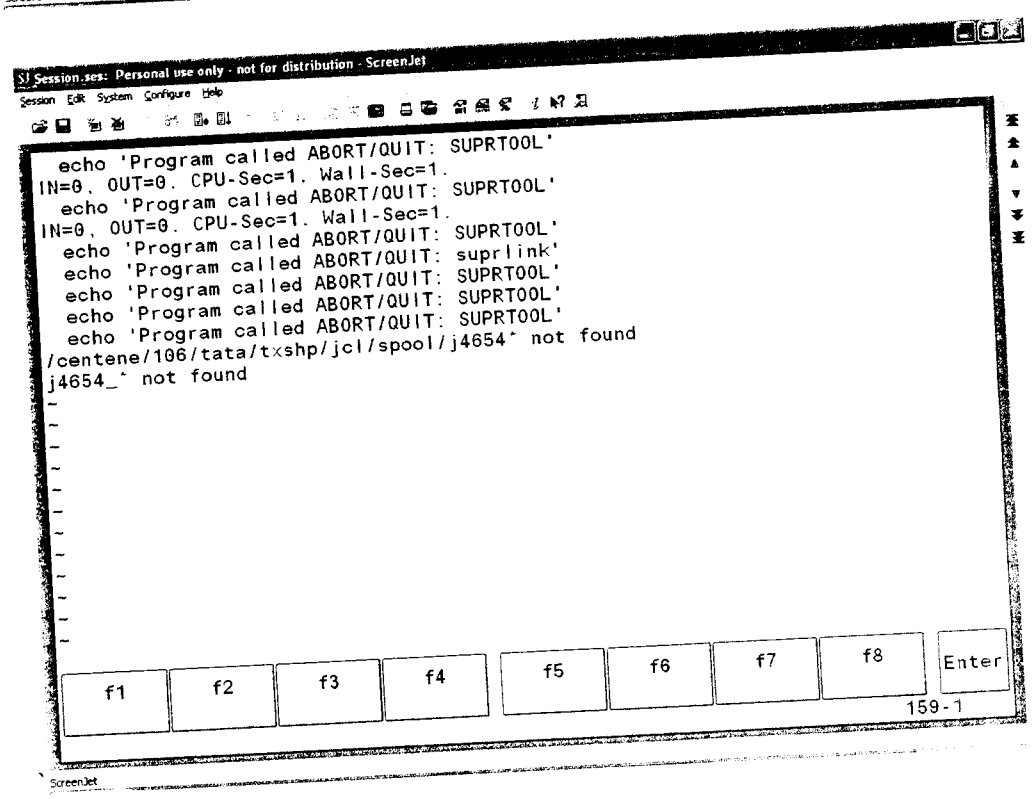


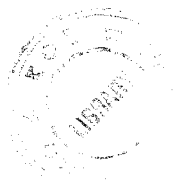
Checking the Log of the successfully executed / Finished script for common error patterns (*Step 1*)



Checking the Log of the successfully executed / Finished script for common error patterns (*Step 2*)







APPENDIX 2

TEST CASES for JCL to Shell Scripts migration

TEST CASE ID	TEST CASE DESRIPTION	EXPECTED RESULTS
SH0001	Find whether the job is created with a job number.	A file with the jobname along with the job number is created in the log directory.
SH0002	Find whether the filenames that are used in "touch" statements are created	The file with the filename specified should be created in that directory
SH0003	Find whether the filenames that are used in "rm" and "rmdat" statements are deleted	The file with the filename specified should not be in that directory
SH0004	Find whether the filenames that are used in input and output statements exists	Files used should be in their respective directories along with the specified permissions.
SH0005	Find whether the self describing files are created when the "link" keyword is used	A file with the same name but with ".sd" extension should have to be created.
SH0006	Find whether the transactions are committed into database	Check for some values whether the values are in the database.
SH0007	Find whether the subfile are accessed correctly	The last time of access should have to be matched with the time in the log file.
SH0008	Find whether the correct cobol files are accessed	The last time of access should have to be matched with the time in the log file.
SH0009	Find whether the correct powerhouse files are accessed	The last time of access should have to be matched with the time in the log file.

SH0010	Find whether the report files are generated correctly	The report files should not have any junk values in it.
SH0011	Find whether the output files specified explicitly and those specified with keywords "PRN","PERMDATA" having perfect output data.	The output files should not have any junk values.
SH0012	Find whether the inputs passed as parameters to cobol and powerhouse files are inferred correctly.	The log file should specify the inputs as passed in the original program.

APPENDIX 3

CHECKLISTS FOR JCL CHANGES

Script Name :

1. Check whether the JCL start with

   '#!/usr/bin/sh '

   and following with export statement having 'program Name' The job name should be defined using the HPJOBNAME variable.

   Example

   export HPJOBNAME='job0100'

2. Check for the following w.r.t file name:
   a. Refer the MPE source code for the following:
   b. case type should be lower
   c. Non Quiz files should have .dat extension
   d. Quiz/QTP files should have .sf extension
   e. Quiz/QTP files should not have extension when exported.
3. Check for the comment line :

   Must have only '#'
4. Check for any non-Unix code:

   E.g.: comment (or) delete lines with '{any number or text}'
5. Check if Suprtool

   begin with:

   a. "suprtool <<!EOSUPRTOOL"

   and end with:

   b. "EXIT !EOSUPRTOOL"
6. Check for Return code after the following commands: follow :

   'time' command

   'suprtool'

   'suprlink'
7. Check if JCL

   Begin with :

   . $AMISYS_ROOT_EXECUTE/cmd/job_start

   and in the end:

$AMISYS_ROOT_EXECUTE/cmd/job_end

8. Check if all 'X' statement in MPE are converted to its equiv. 'XEQ' in UNIX
9. Check if backslash are used before suprtool $ functions.
   E.g. "\$LOOKUP"
10. Check if all '#'s are converted to _NBR under suprtool.
    E.g. DIVISION# to DIVISION_NBR

11. Check for the following:
    All oracle related variables having "-"(hyphen) should be replaced with "_"(underscore).
    a. i.e., In suprtool, if a select statement is found, then all the variables present inside the suprtool will be changed as follows:

if the variable is a field-name containing "-"(hyphen) then it will be changed to "_"(underscore).

. if any other variable is present, then those variables will be maintained as they are in the original JCL.

- . In suprtool, some blocks don't have a select statement but have an Input statement. Those blocks will be changed except for variable names.
- . In suprtool, some blocks do not have either select or Input statement. Those blocks have been maintained as they are in the original JCL.

12. Check if correctly used: "select * from" instead of "get" – suprtool
13. Check if 'USEQ $AMISYS_PUB/dbopen' is used for DATABASE connection before SELECT statement in a Suprtool block.
14. Check for the following:
    1)Export statements can not have spaces on either side of the equal sign.
    2)Export statements should not have 2 equal signs.
15. Check for job_abort: To terminate a job when an error occurs, job_abort is used as follows: .
    $AMISYS_ROOT_EXECUTE/cmd/job_abort

16. Check if the JCL end with 'exit' after the job_end script.
17. Check if 'LINK' is appended to all output statement in advance suprtool except to those Output statements that have APPEND,PRN,ASCII,QUERY,ERASE option.
18. Check if all abbreviations in suprtool like 'EXT', 'I', 'O', 'DUP' are replaced with 'EXTRACT', 'INPUT', 'OUTPUT', 'DUPLICATE' respectively
19. Check for the usage of utilities like rmdat/mvdat etc., in UNIX.
20. If there is no input to time command then there should be no start and end tags.
21. HPJOBNUM need not be set inside the code. While accessing the value of the variable, it should be accessed as $HPJOBNUM
22. Every if statement should have a matching fi statement and the code should be properly indented.
23. If Deletevar <variable name> is present in MPE code then the equivalent Unix code will be export <variable name>=""
24. If the output of PURGE is redirected to $null in MPE, the $null should be replaced with /dev/null in Unix equivalent code.
25. If statement containing AND, OR statements are coded in Unix as : if [<condition> -a/-o <condition]; then
26. There should be a semicolon following the if condition, when "then" also appears in the same line
27. The job name that is assigned to HPJOBNAME should be in small case.
28. All file names should be in small case along with the path and .dat extension
29. In Suprtool, wherever LIST command appears for printing, there should not be RESET LIST, after the XEQ command.
30. Check if all 'QTP' and 'QUIZ' block of code are migrated. Verify it with migration strategy.\
31. The comment lines within 'suprtool' should have a open '{' and close '}'
32. Check whether the PURGE used with @ in MPE, should be replaced with rm -f with * in Unix and no extension should be given to the file name.

33. Check Whether exit is present after .
    $AMISYS_ROOT_EXECUTE/cmd/job_abort where the Source has ABEND statement.
34. Do not add LINK to OUTPUT if the file used is a QUIZ/QTP file.
35. LINK should not be present in SUPRLINK OUTPUT if not present in MPE code.
36. Leading and trailing spaces within quotes containing parameter values in an echo statement should be removed.
37. Check whether the rec and NOLF option is present for all files in INPUT except foe self describing files.
38. Check whether the QUIZ/QTP files are accessed with .sf in the JCL except in export statement
39. Check whether LF is appended to PRN in Suprtool-OUTPUT statement
40. Check whether QUIZ/QTP Unix equivalent code do not have /dev/null
41. Check whether the OUPUT options are the same between MPE and UNIX converted code.
42. Check whether != has replace <> in the converted code except in Suprtool block.
43. Check whether V_ is appended to the Table names used in Suprtool
44. Check whether a newline is prsent after exit statement at the end of the job
45. # should not be changed to _NBR in Suprtool, when # comes within quotes and in Extract statement.

REFERENCES

REFERENCES

Books

1 Sumitabha das "Unix Concepts & Application", Tata Mc Grew-Hill publishing company, 1998

2 Yashavant Kanetkar, "Unix Shell Programming", BPB Publication, 1996

3 David B.Horvath, "Unix for the Mainframer", Prentice Hall PTR, 1998

4 Gary De Ward Brown,"JCL Programming Bible",Wiley Dream tech publication, 2002

5 Alexis Leon."IBM Mainframe Handbook", Leon Vikas publisher, 2003

Online References

1 http://heather.cs.ucdaris.edu

2 http://www.skillsoft.com

3 http://unix.ittoolbox.com