CHAPTER 6

1. AutoTEMP© - The Automated TEMP Generator

1.1 Summary

This chapter gives a comprehensive description of the results of this research, which is a KBSS to aid in the generation of TEMP's, that the author has called AutoTEMP©. Version Beta 2.0 of the software release clearly demonstrates the principles of the development of a TEMP according to a conceptualised revision of the Australian CEPMAN 1 (Australian DoD, 1995), as set out in the previous chapter.

The chapter will begin with a discussion and analysis of the CSCI AutoTEMP© requirements, outlining such things as the development software and programming language used for the development of the CSCI, along with a brief description of the hardware requirement specification for this system, a brief introduction and description of Visual Basic®, and then will describe the operation and use of AutoTEMP© and its three modules.

1.2 AutoTEMP© CSCI Requirements

The only requirement for developing the CSCI as stated by Nissyrios (1995a) was a Windows® environment such as Windows® for Work Groups (WFWG) version 3.11, or Windows® 95 based on an IBM compatible computer. It is envisaged that the CSCI will be compiled into a single executable file (EXE) depending on the application development software language.

1.2.1 The Selection of the CSCI Development Software

A number of software packages had been considered by the author for this task. Taking into consideration the SRS as documented in Appendix V, possible candidates for the software development that were PC based, at the time of its conception were:

- Modsim II®
- VP-Expert®
- HyperCard®

- Microsoft® Access
- Layout®
- DataEase®
- ObjectVision®
- Visual Basic®
- Delphi® (Visual Pascal)

As discussed in the SRS, Appendix V, the CSCI was to be developed using an application programming language that did not require vast amounts of specialised programming, such that it alleviates the necessity of software coding to a base minimum, and in particular one with a fast learning curve. Due to these requirements, the most prominent application software development tools were Visual Basic® (VB) or DELPHI (visual PASCAL).

However, DELPHI was and still is a lot more complex than VB and hence would require a long learning curve, the author estimated six months, as compared to one month for VB. This is not to say that it wasn't more than adequate to suit the task ahead, however, VB was chosen as the software and programming language for the development of AutoTEMP© due to reasons summarised below as follows:

- Event driven
- Alleviated the necessity of software coding to a bare minimum
- Quickest and easiest way to create Window applications
- Fast learning curve (about one month)

1.2.2 Computer Resource Requirements

1.2.2.1 Computer Software Requirements

The following software was required for CSCI development (Nissyrios, 1995a):

- DOS for Windows® 95 Version 4.00.950
- Microsoft Windows® 95
- Microsoft Office Professional Version '95
- Microsoft Access 2.0
- Visual Basic® Professional (VBP) Version 4.0

- Microsoft Project Version 4.0

1.2.2.2 Computer Hardware Requirements

Recommendation was an IBM personal computer with the following minimum characteristics (Nissyrios, 1995a):

- Three year on-site warranty.
- 90 Mhz Pentium PCI Processor.
- 16 MegaBytes of Random Access Memory (RAM).
- 17 Inch Multi-Sync Monitor (27mm Dot Inch)
- 1.2 GigaByte Hard Disk Drive (HDD)
- 512K Cache (Pipe-Line Burst).
- 64-Bit PCI, 2MB Video Card.
- 1.44 MegaByte Floppy Disk Drive (FDD).
- Twin Speed CD-ROM Drive.
- 16-Bit Sound Blaster Card.
- Ethernet compatible in both thin-net and twisted pair formats[1].

1.3 Visual Basic®

Visual Basic® is the quickest and easiest way to create applications for the Microsoft Windows® operating system. The Visual Basic® programming system allows you to create attractive and useful applications that fully exploit the graphical user interface (GUI). Visual Basic® is more productive by providing appropriate tools for the different aspects of GUI development.

The graphical user interface can be created for the application by drawing objects in a graphical way. It's simply a matter of setting the properties on these objects to refine their appearance and behavior. The second step is to make this interface react to the user by writing code that responds to events that occur in the interface. Using Visual Basic® the user can create powerful, full-featured applications. Some of these features are (Microsoft® Corporation, (1995b)):

[1] A network capability is essential for developing a multi-user, single session support system.

- Data access features that allow you to create databases and front-end applications for most popular database formats, in particular Microsoft® Access.
- Object Linking and Embedding (OLE) allows the user to use the functionality provided by other applications, such as Microsoft® Word for Windows® word processor, Microsoft® Excel spreadsheet, and Microsoft® Project project planning system.
- The completed application is a true .EXE (executable) file that uses a run-time Dynamic Link Library (DLL) that the user can freely distribute.

1.3.1 Working with Visual Basic® 4.0

Originally the author began with Visual Basic® version 3.0, professional edition, and Windows® for Workgroups 3.11 as the software platform, this combination however, was superseded with the release of Windows® 95 and Visual Basic® version 4.0, professional edition, so as to keep up with the technology and not to mention the additional features in both of these software packages. An outline of some of the new features in Visual Basic® 4.0, professional edition are (Microsoft® Corporation, (1995b):

- OLE custom controls
- Insertable projects as controls
- Development environment extensibility
- Conditional compilation
- Settable fonts and font sizes
- Menu and toolbar negotiation
- Improved debug window
- Data access object (DAO)
- New data-bound controls
- 32-bit support
- Microsoft Jet 2.5 and Microsoft Jet 3.0 databases
- TabStrip control
- Toolbar control
- StatusBar control
- ProgressBar control
- TreeView control
- ImageList control

- Slider control

1.3.2 Steps to Creating a Visual Basic® Application

There are three main steps to creating an application for Windows® in Visual Basic® (Microsoft Corporation, 1995b):

1. Create the interface.
2. Set properties.
3. Write code.

1.3.2.1 Creating the Interface

Forms are the foundation for creating the interface of an application. You can create forms to add windows and dialog boxes to your application. You can also use them as containers for items that are not a visible part of the application's interface. For example, you can have a form in your application that serves as a container for graphics that you plan to display in other forms.

The first step in building a Visual Basic® application such as AutoTEMP©, is to create the forms that will be the basis for that application's interface. Then you draw the objects that make up the interface on the forms you create.

1.3.2.2 Setting Properties

The next step is to set properties for the objects that have been created. The Properties window provides an easy way to set the properties for all objects on a particular form.

1.3.2.3 Writing Code

The *Code window* is where the Visual Basic® code for the application is written. Code consists of language statements, constants, and declarations. Using this code window, you can view and edit any of the code in the application.

1.3.2.4 Creating Event Procedures

Code in any Visual Basic® application is divided into smaller blocks called *procedures*. An *event procedure*, contains code that is executed when an event occurs (such as when a user clicks a button). An event procedure for a control combines the control's actual name (specified in the Name property), an underscore(_), and the event name. For example, if you want a command button named Command1 to invoke an event procedure when it is clicked, then you use the event procedure Command1_Click, and so on.

1.3.3 Structure of a Visual Basic® Application

All applications can contain several different types of files, such as (Microsoft® Corporation, (1995b)):

- Form modules (.FRM) contain the visual elements of a form, including all the control on the form and Basic code associated with that form.
- Standard (.BAS) and class (.CLS) modules contain Basic code.
- Custom controls (.VBX or .OCX) include specialised controls, as well as enhanced versions of standard controls.
- A single resource file (.RES) contains strings and bitmaps used by the application.

1.3.3.1 How an Event-Driven Application Works

An *event* is an action recognised by a form or control. Event-driven applications execute Basic code in response to an event. Each form and control in Visual Basic® has a predefined set of events. If any one of these events occurs, Visual Basic® invokes the code in the associated event procedure as mentioned previously.

Although objects in Visual Basic® automatically *recognises* a predefined set of events, you determine if and how they *respond* to a particular event. When you want a control to respond to an event, you write event procedure code for that event.

Many objects recognise the same event, although different objects can execute different event procedures when the event occurs. For example, if a user clicks a form, the Form_Click event procedure executes; similarly, if a user clicks a command button named Command1, the Command1_Click event procedure executes. This is what happens in a typical event-driven application such as AutoTEMP© (Microsoft® Corporation, 1995b):

1. The application starts and the startup form is automatically loaded and displayed.
2. A form or control receives an event. The event can be caused by the user (for example, a keystroke), by the system, or indirectly by your code (for example, a Load event when your code loads a form).
3. If there is an event procedure corresponding to that event, it executes.
4. The application waits for the next event.

1.3.3.2 Event Driven vs. Traditional Programming

In a traditional or "procedural" application, the application itself rather than an event controls the portions of code that execute. Execution starts with the first line of executable code (like a line-by-line assembler) and follows a defined path through the application, calling procedures as needed.

In event-driven programs, a user action or system event executes an event procedure. Thus, the order in which your code executes depends on which events occur, which in turn depends on what the user does. This is essence of graphical user interfaces and event-driven programming: The user is in charge, and your code responds.

1.4 A Description of AutoTEMP©

The AutoTEMP© CSCI is primarily designed to aid in the automatic generation of TEMP's, however, in order to accomplish this task, it required a minimum of two modules, one for entering the data needed to fill the contents of the TEMP document, as well as a separate module for automatically generating it. An additional module was also included, this is a tutorial of the US phased acquisition process, of which is described in chapter 4, section 4.2.3. These three modules are depicted in Figure 1-1.

The header screen that the user sees once AutoTEMP has been invoked is depicted in Figure 1-2, as is evident from the three options to which module the user wishes to enter, under the File menu.

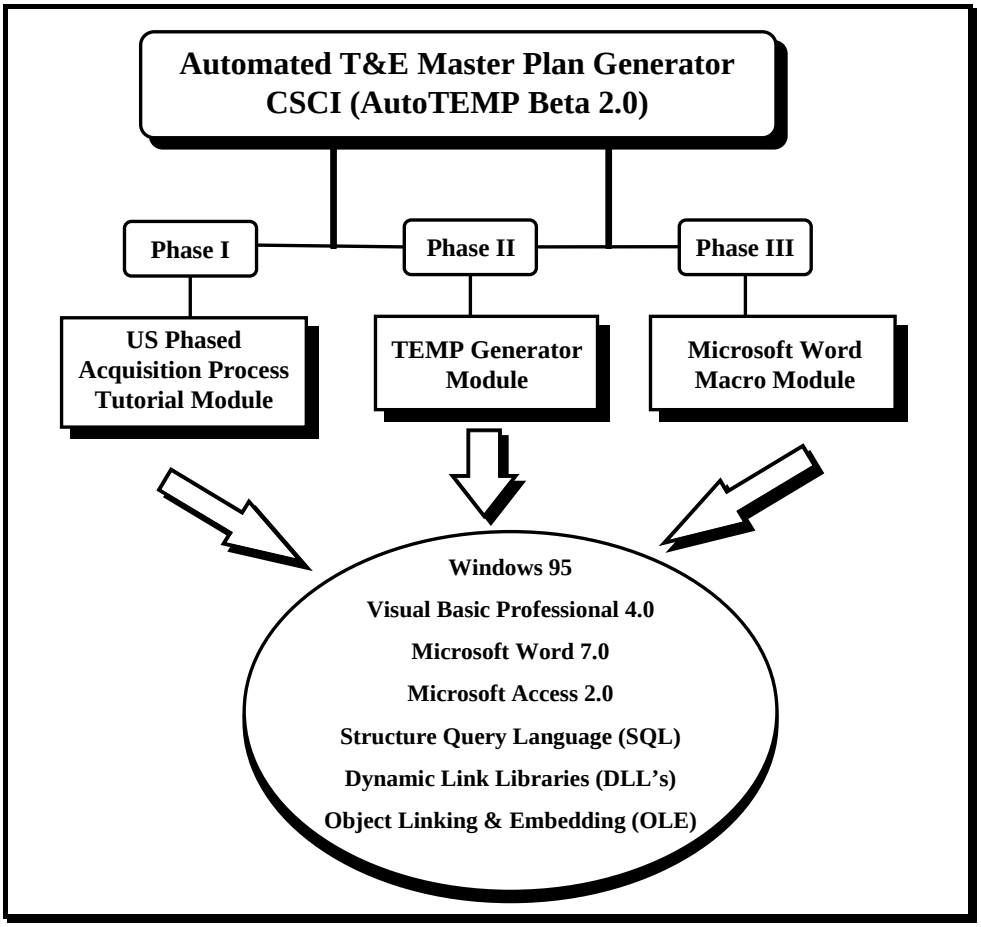

Figure 1-1 (AutoTEMP© Beta 2.0 CSCI Module Reticulation)

1.4.1 Module I - US Defence Phased Acquisition Process Tutorial

This CSCI module was developed as a means of educating ARDU personnel about the Phased Acquisition Process (PAP) which also encapsulates a concise description of the Australian CEPMAN 1 TEMP format as per chapter 5, and Appendix VI. The US version was chosen at the time as this was the best documented literature on the PAP and reasons as mentioned previously in the earlier chapters of this dissertation.

1.4.1.1 Features of Module I

The tutorial presents the user with a walk through user-friendly graphical software medium for educating oneself with testing and the PAP as well as the TEMP format. It is best to begin this tutorial by pressing the Mission Need Button, read the text and continue with Phase 0, right through to Phase IV, or similarly in an ad hoc fashion, pressing any button on the screen will provide information on the topic. You will notice that some words in the text are coloured green, and some blue.

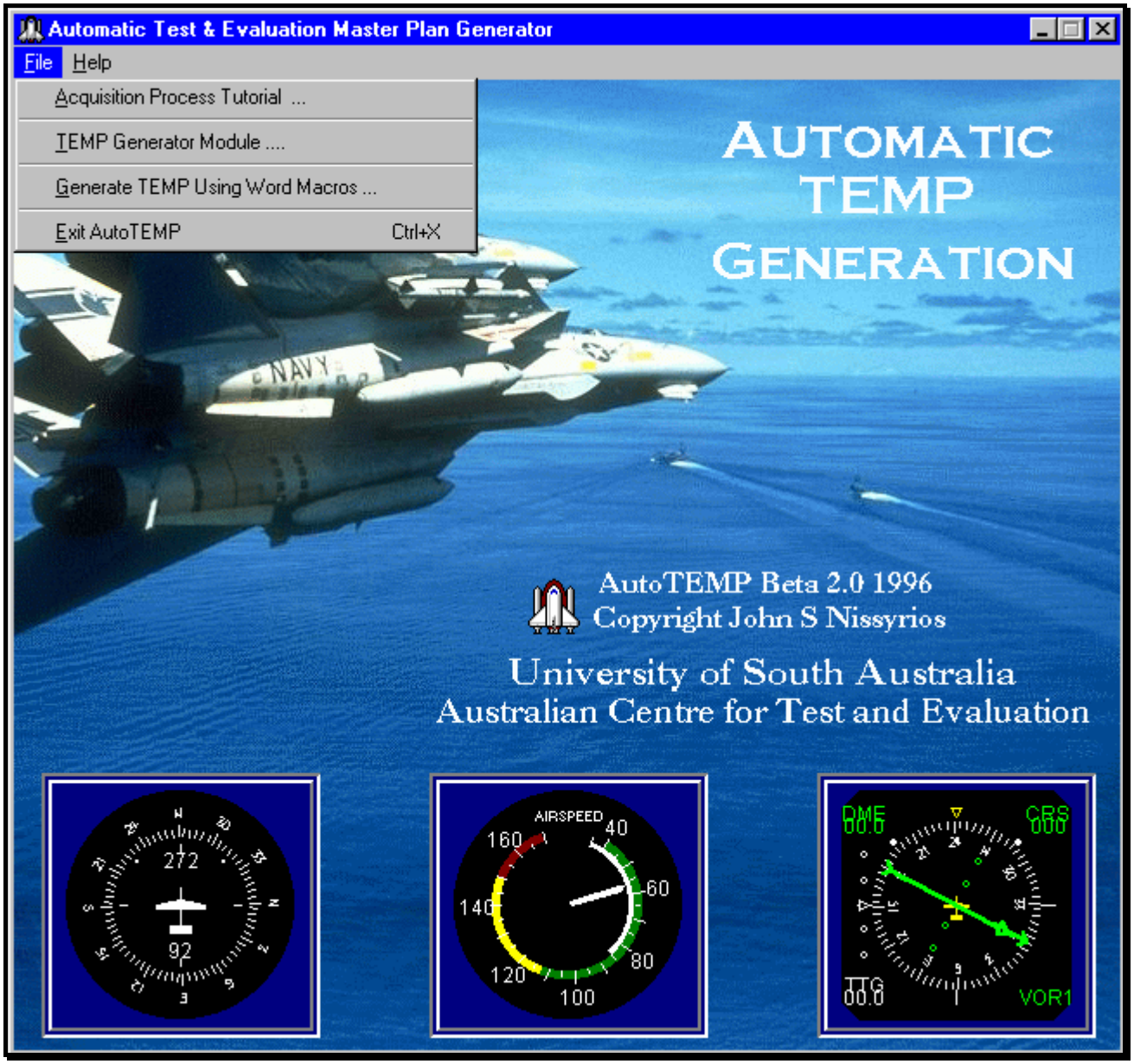

Figure 1-2 (AutoTEMP© Header Form)

By simply clicking on the green entries which represent hyperlinks[2] the software will provide other information about that entry. The hypertext entries or words are easy to locate, because first of all they are green and underlined, and second, the cursor changes from a pointer to a hand with the index finger over the text. This particular user screen (known as forms in Visual Basic®) referred to as the Module I "Home Page (HP)" is depicted in Figure 1-4. With reference to Figure 1-2, once the user has selected the first option, under the file menu, they will be presented with a help screen, and pre-tutorial information, such as instructions, background, as well as the option for printing this form. This introductory form is shown in Figure 1-3.

Simply by clicking with the left mouse button on any of the buttons shown on the menu bar, apart from the print and close buttons, which will print and close this form respectively, this will open new forms about that button pressed, similar to the one depicted in Figure 1-3.

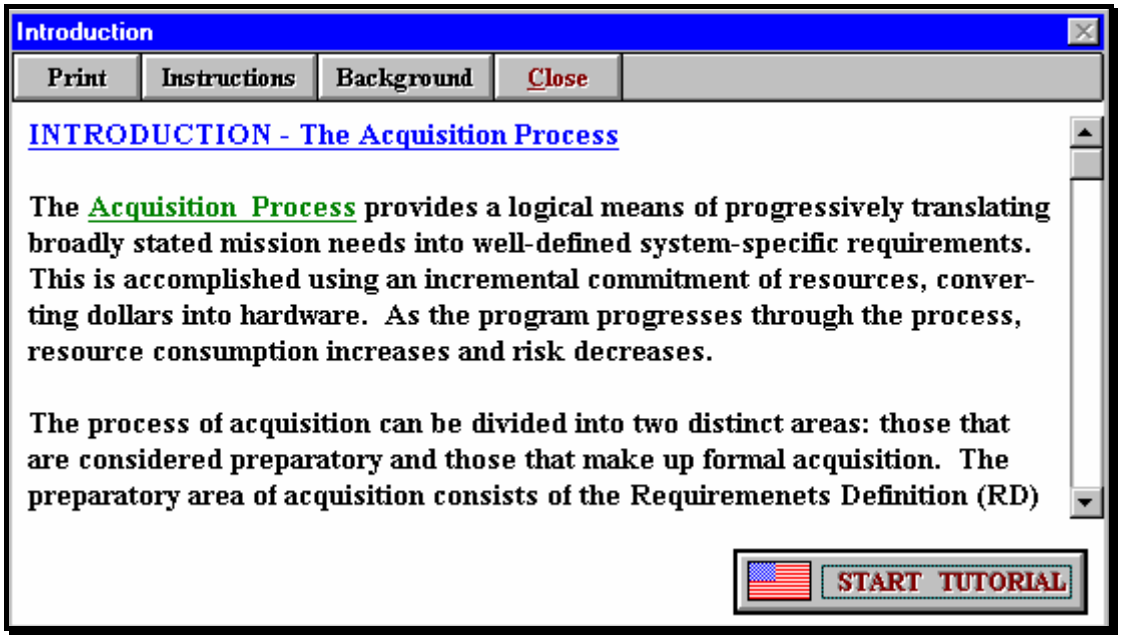

Figure 1-3 (Module I Introductory Form)

[2] Hypertext is a method of presenting information where selected words in the text can be "expanded" at any time to provide other information about the word. That is, these words are links (known as *Hyperlinks*) to other documents, which may be text, files, pictures, anything (Krol, 1992).

In order to instigate the tutorial, and load up the PAP Home Page, you simply press the Start Tutorial button with the icon of the US flag (clearly illustrating that this tutorial is Primarily US based, the exception being the TEMP information and format) as depicted in Figure 1-3.

The form depicted in Figure 1-4, presents the user with a number of options, they can either press buttons and educate themselves by reading, what the author refers to as on-line reading, or they have the option of printing the text shown on the form and reading the hard copy in their own leisure. So it really does depend on the liking of the person using the software.

The user knows whether they have "visited" a particular site (one of the button on the form in Figure 1-4) because the colour of the button changes from grey to purple, this is one of the user-friendly features of this module. The five phases, phase 0 to IV present the user with a comprehensive description of that phase with hyperlinks embedded that will guide you to other topics, and then a summary of this information. For illustrative purposes Phase II and the summary forms are depicted in Figure 1-5 and Figure 1-6 respectively.

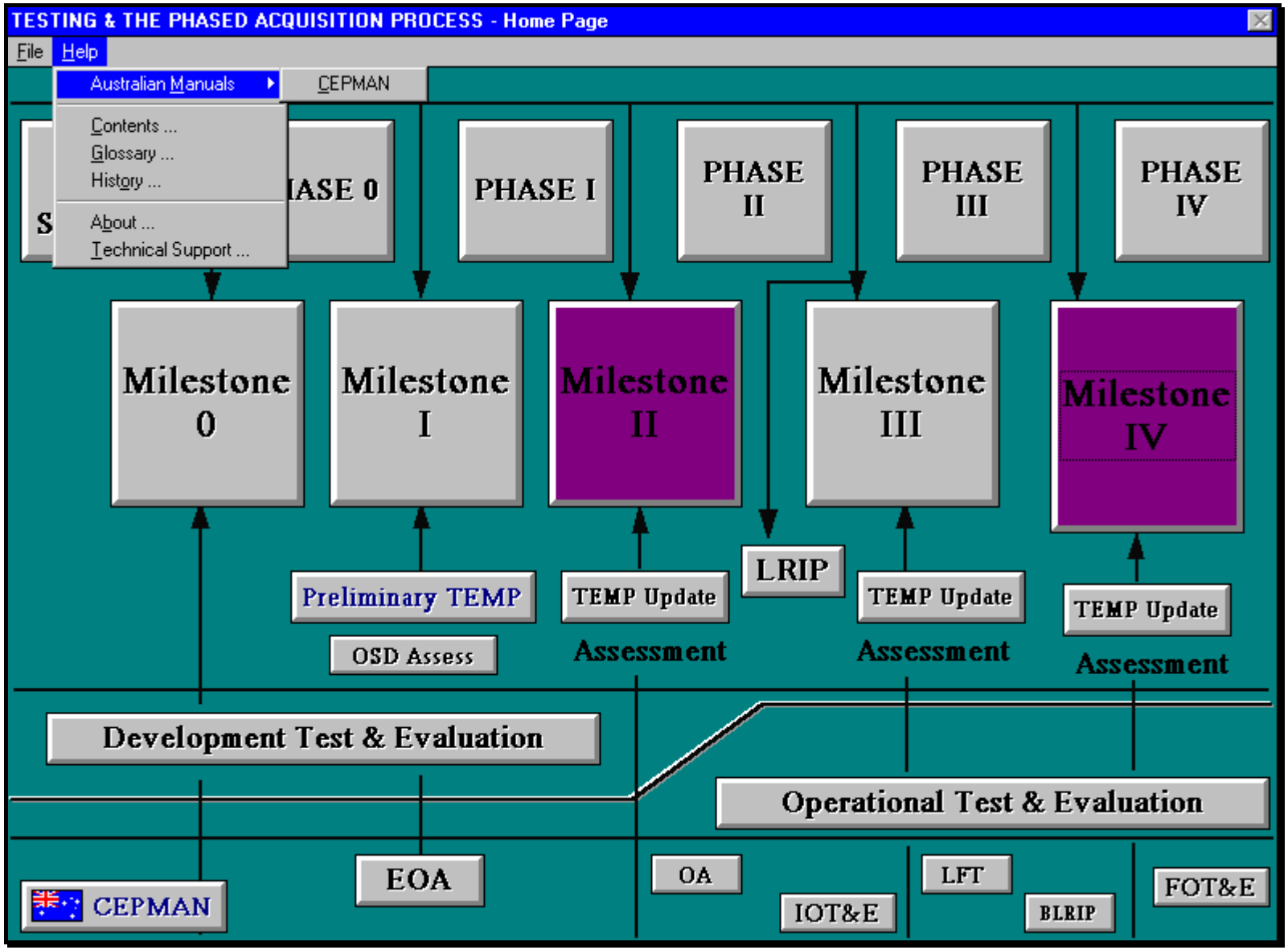

Figure 1-4 (Module I Home Page)

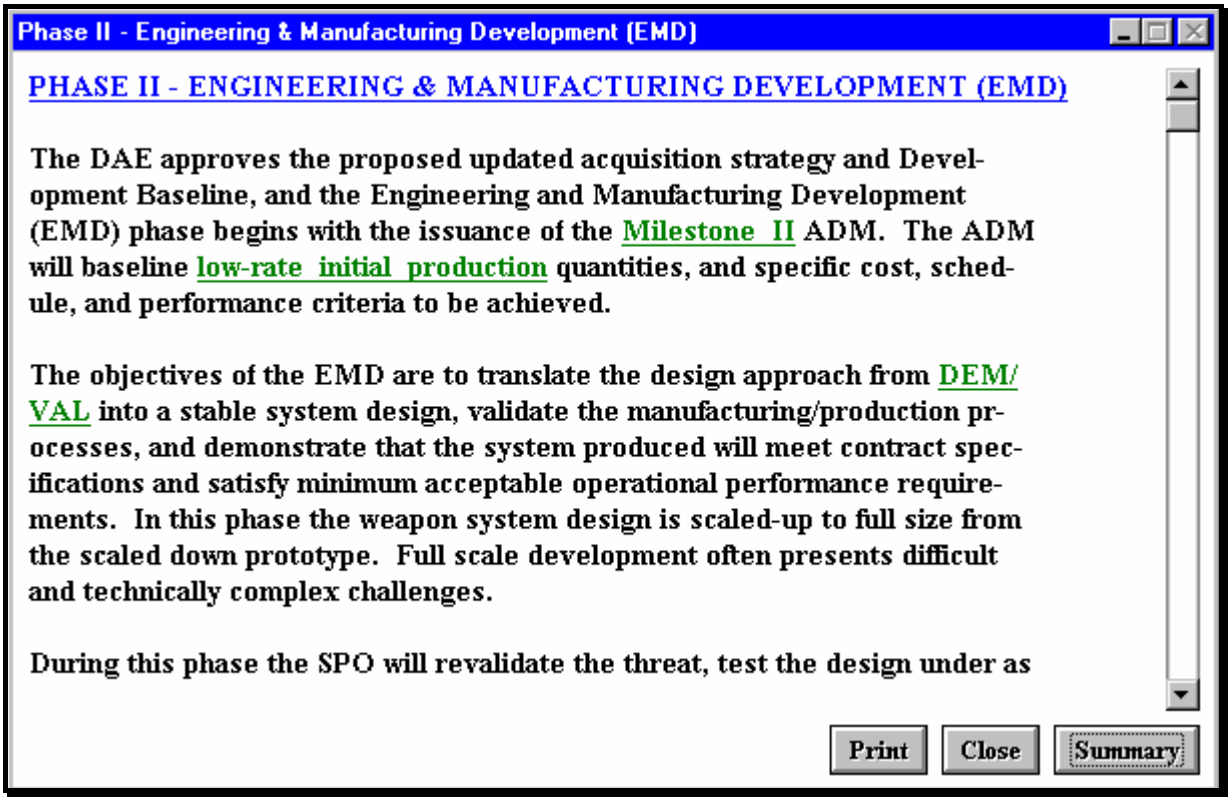

Figure 1-5 (Phase II - EMD Form)

As is evident from the above diagram, there are also scroll bars in each form, that allow the user to scroll down the form as it is being read, and also gives it versatility in its development[3]. In Figure 1-4, some of the other user-friendly features of this module are the Contents, Glossary, and History options under the Help menu.

[3] A handy 16-bit Control box (.VBX) of Visual Basic® known as Multitext.vbx, of which the author downloaded from one of the many Visual Basic® pages on the Internet, incorporates this facility.

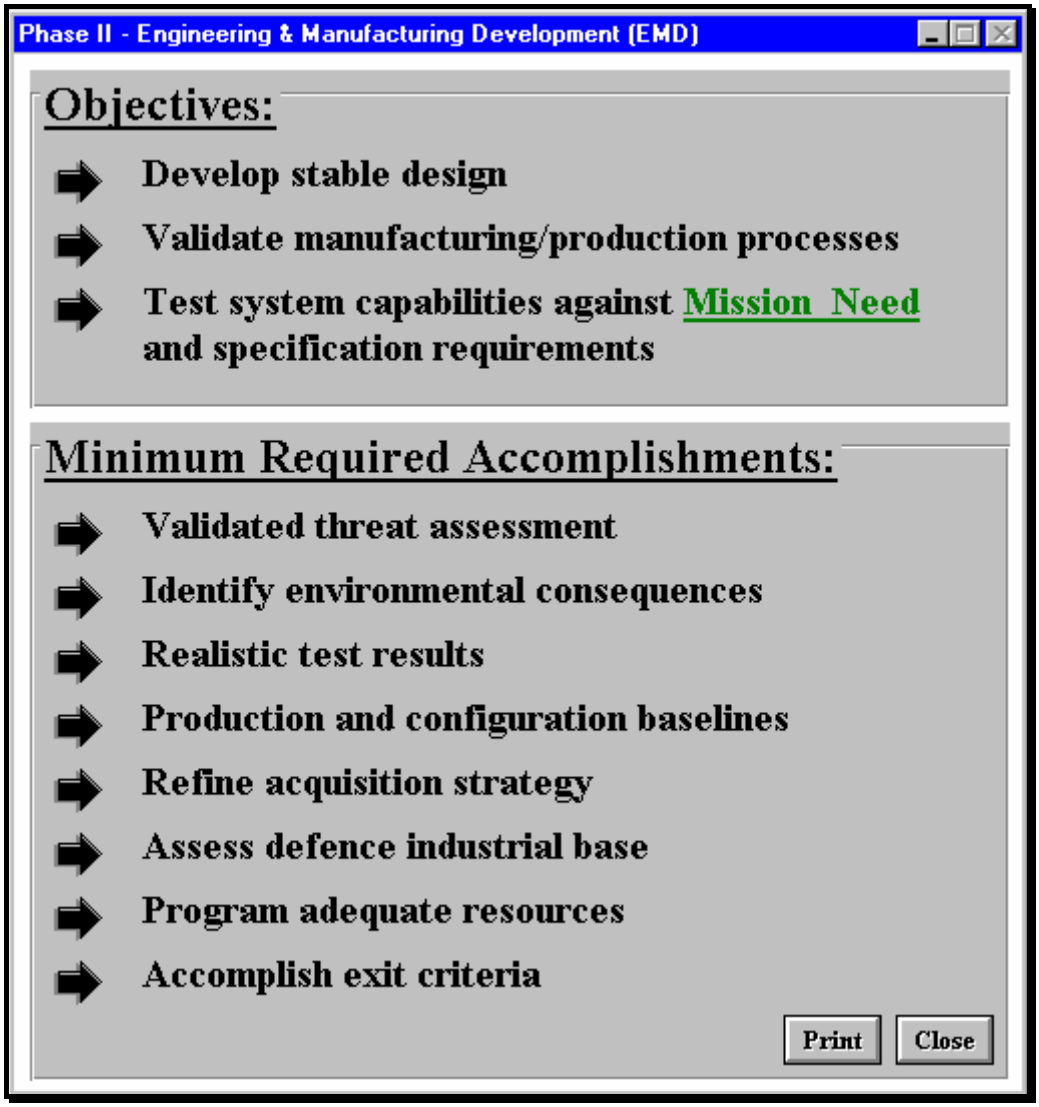

Figure 1-6 (Phase II - EMD Summary Form)

The Contents form lists all possible forms that the user could open of which there are 49 of them. Figure 1-7 shows all the possible forms that could be opened whilst Module I is active.

Acquisition Program Baseline (APB)
Air Force Operational Test & Evaluation Centre (AFOTEC)
Availability of Test Schedules
Background
Beyond Low-Rate Initial Production (BLRIP)
Capital Equipment Procurement Manual (CEPMAN)
Criticality Levels for Test & Evaluation
Developmental Operational Test & Evaluation (DT&E)
Early Operational Assessment (EOA)
Five Types of Test & Evaluation
Follow-on Operational Test & Evaluation (FOTE)
Independent Validation & Verification (IV&V)

Initial Operational Test & Evaluation (IOTE)
Instructions
Integrated Program Summary (IPS)
Introduction
Lethality
Live Fire Test & Evaluation (LFT&E)
Logistic Support Analysis (LSA)
Low-Rate Initial Production (LRIP)
Milestone 0 - Concept Studies Approval
Milestone 1 - Concept Demonstration Approval
Milestone 2 - Development Approval
Milestone 3 - Production Approval
Milestone 4 - Major Modifications Approval (As required)
Mission Need Statement (MNS)
Operational Assessment (OA)
Operational Test & Evaluation (OT&E)
Operational Test Agency (OTA)
Phase 0 - Concept Exploration and Definition (CE)
Phase 0 - Summary
Phase 1 - Demonstration and Validation (DEM/VAL)
Phase 1 - Summary
Phase 2 - Engineering and Manufacturing Development (EMD)
Phase 2 - Summary
Phase 3 - Production and Deployment (PD)
Phase 3 - Summary
Phase 4 - Operations & Support (OS)
Phase 4 - Summary
Production Acceptance Test & Evaluation (PATE)
Test & Evaluation Master Plan (TEMP)
Test & Evaluation Master Plan (TEMP) Format
Test & Evaluation Master Plan (TEMP) Update
Testing and the Phased Acquisition Process
Triangle of Measurement & Instrumentation and Test & Evaluation
Types and Applications of Test & Evaluation
Under the Office of the Secretary of Defence (OSD)
Verification & Validation (V&V)
Vulnerability

Figure 1-7 (Module I Contents)

The Glossary form, depicted in Figure 1-8, is designed so that the user can easily choose the letter of the alphabet (in the Index box) that the particular acronym or abbreviation starts with, press the letter which is a button and the software will then give you a list of all the acronyms or abbreviations that start with that letter. It is then simply a matter of clicking on the hypertext acronym or abbreviation, and the software will show you the meaning of the word in the right text box as is shown in Figure 1-8. As well as having the capability to scroll

down as the user sees fit, the user can obtain a hard copy of the list of acronyms and abbreviations, of which there are approximately 100 (all that the tutorial uses), by hitting the Print List button.

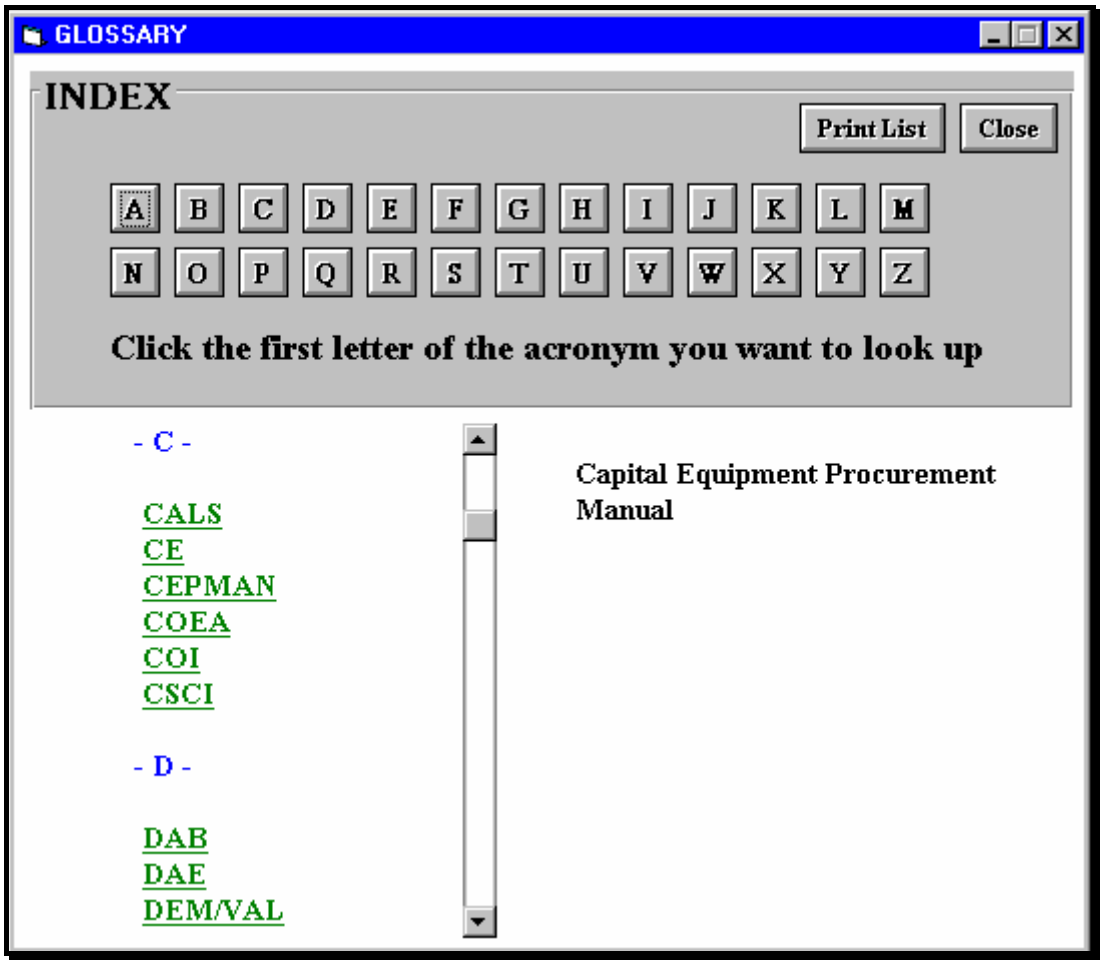

Figure 1-8 (Module I Glossary Form)

The history form depicted in Figure 1-9, has some intelligent software code encrypted in it that keeps a track of what forms are open, and allows you to quickly invoke a form that was opened during the tutorial. This feature was incorporated so as help the user navigate through the tutorial with some ease and direction, considering the 50 or so forms that can be opened at any one time.

Figure 1-9 (Module I Help History Form)

Finally, as mentioned previously, the major modification to US PAP in Figure 1-4, is the TEMP format, it outlines the Australian CEPMAN 1 format as per Appendix VI, and also incorporates Annex A and B to Chapter 14, part2 of CEPMAN 1 (Australian DoD, 1995), which are the Types and Applications of T&E as well as a description of the TEMP format, respectively. The TEMP Format form is illustrated in Figure 1-10.

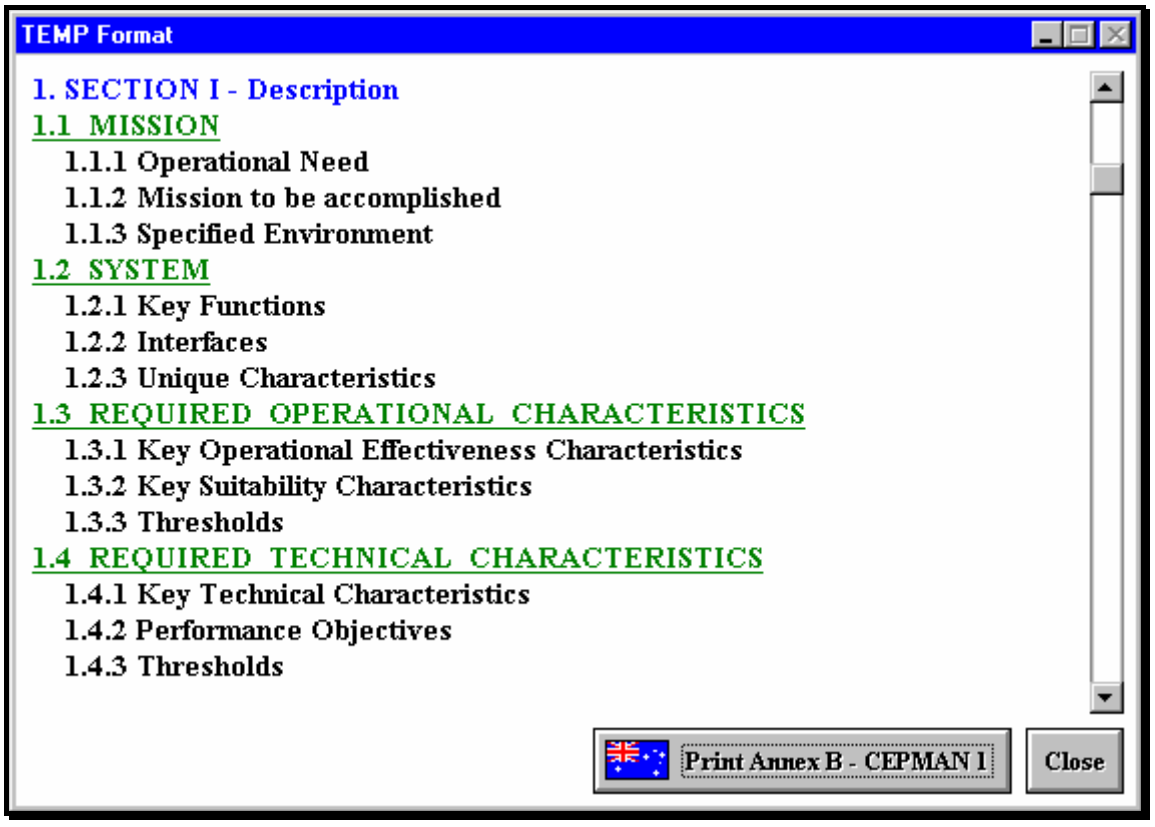

Figure 1-10 (Module I TEMP Format Form)

Annex B in Figure 1-10 is the description of the TEMP format according to the CEPMAN 1 (Australian DoD, 1995), and is depicted in Figure 1-11.

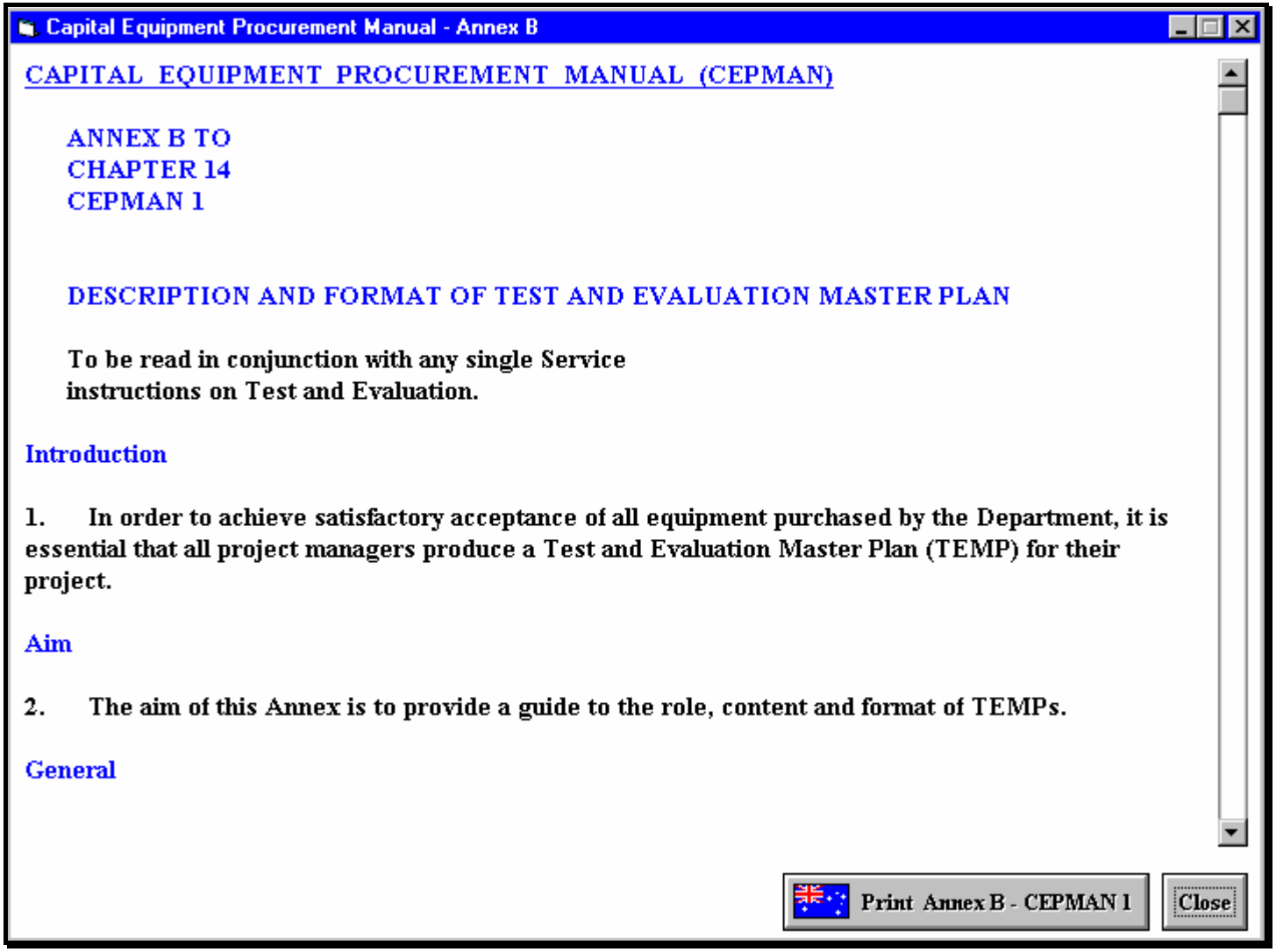

Figure 1-11 (CEPMAN 1 - Annex B Form)

Figure 1-12 illustrates Annex A of CEPMAN 1, using hyperlinks and a contents page, which once clicked on invoke information on that topic. It also allows you to print the form, the hard copy of which would look exactly like the figure.

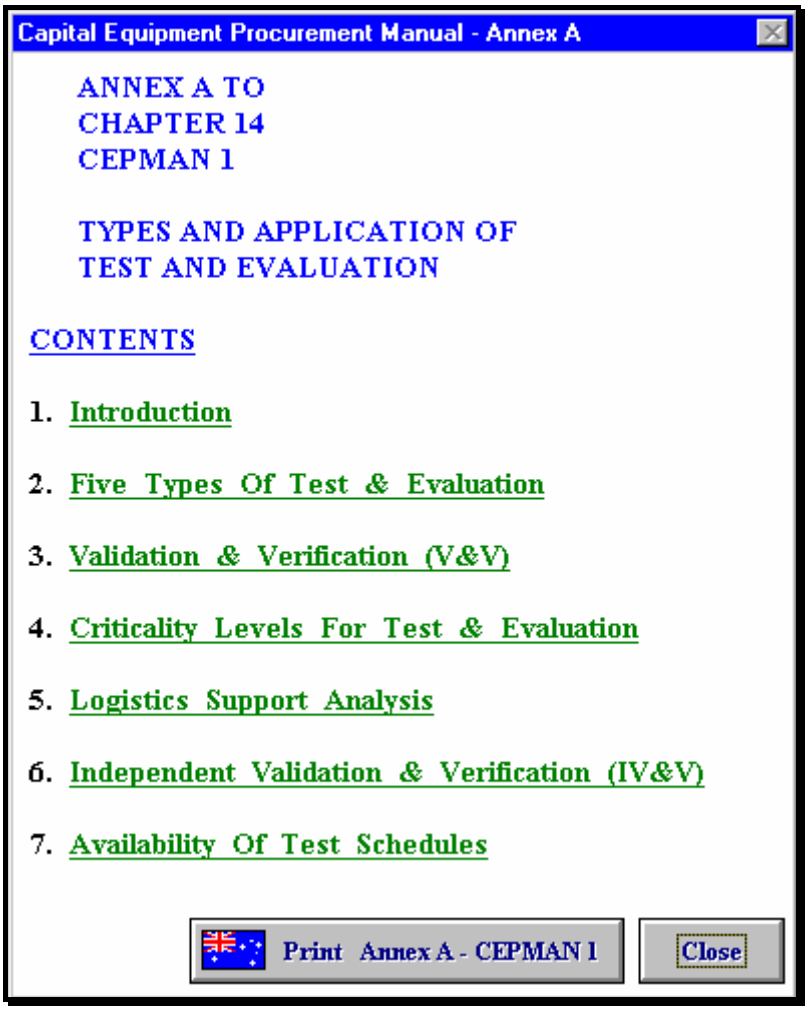

Figure 1-12 (CEPMAN 1 - Annex A Form)

1.4.2 Module II - TEMP Generator

This module is designed to allow the user to enter the necessary data required to fill the TEMP document, and hence in doing so populate the database used to store the data.

1.4.2.1 Features of Module II

This module follows a similar format with that of the of the previous module, so as to stay in "synch" and not confuse the user. That is to say, it is also a hyperlinked operated CSCI. In order to activate this module it is simply a matter of selecting the "TEMP Generator Module" under the File menu of Figure 1-2. This will invoke a similar introductory screen as that shown in Figure 1-3. This form is illustrated in Figure 1-13.

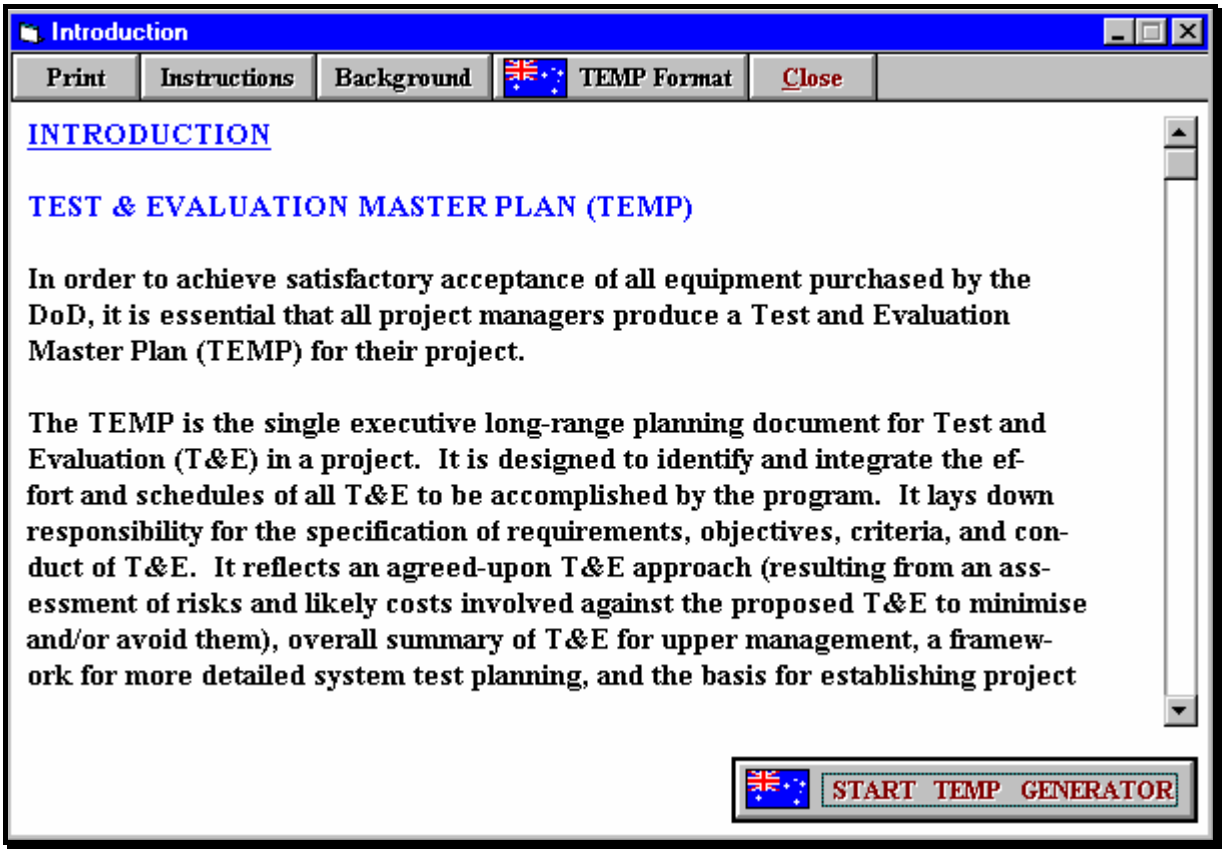

Figure 1-13 (Module II Introductory Form)

For those user's or in particular ARDU personnel who are quite literate on the PAP, they don't have to start with module I, and rather skip to module II, "dive into the deep end" and begin entering data. In the advent of this occurring the author has included the TEMP Format button as is shown in the menu bar, which invokes that information from module I, as is illustrated in Figure 1-11.

By pressing the "Start TEMP Generator" button shown in Figure 1-13, the software launches the form shown in Figure 1-14. This form allows the user to enter all their personal particulars as is shown[4]. Each TEMP created is assigned a default TEMP ID integer. In order to create a new TEMP you simply click on the "Create New TEMP" button, this action increments the TEMP ID Number by one and goes to the next record in the AutoTEMP© database.

[4] This form is also explained in section 11 of Annex I in Appendix VI (Description and Format of the T&E Master Plan)

The author has created some "dummy" TEMP's and populated the database for demonstration purposes. As is evident there are three dummy TEMP's shown, namely HTS, Submarine Mark III, and Falcon Air Fighter - 56, all of which are dummy names. By pressing the "Load Previous TEMPs" button, this will acknowledge all TEMPs previously written in the text box to the right of the Comments text box. By selecting the TEMP that needs to be updated or what have you, simply by clicking on the appropriate one, this will display this form with that record of information, much like a pointer does in a stack.

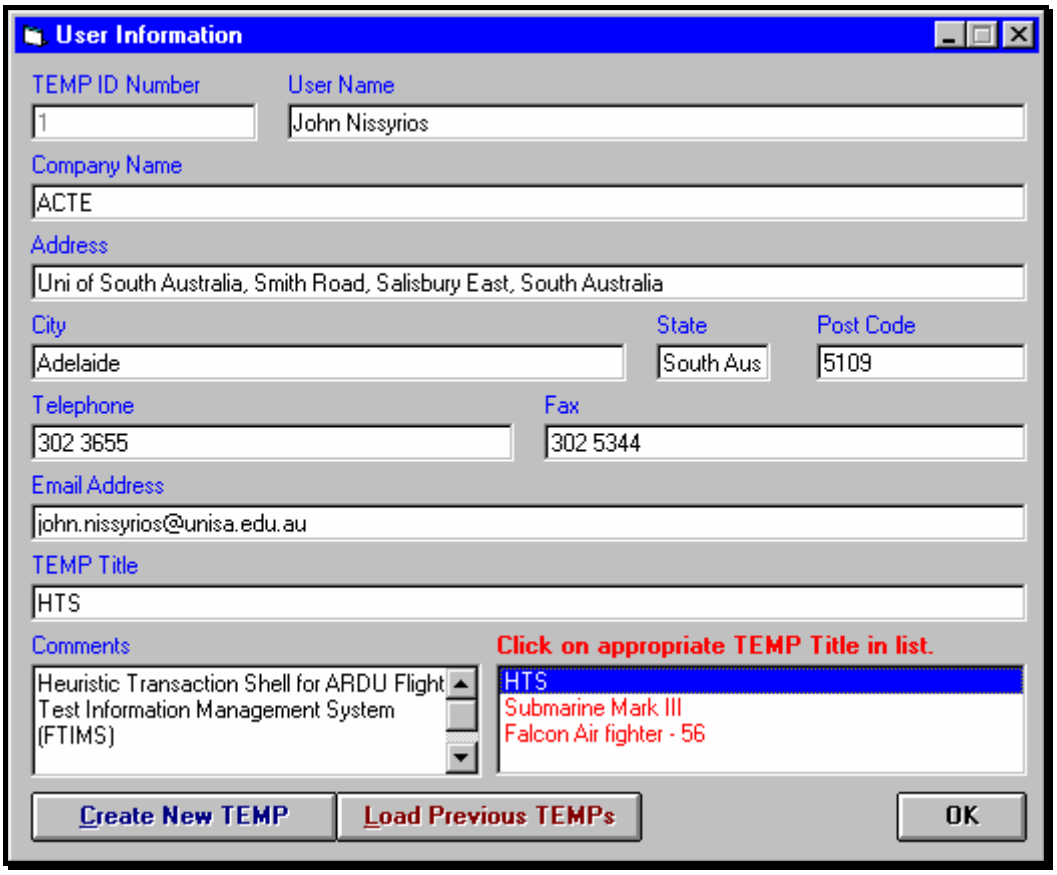

Figure 1-14 (Module II User Information Form)

Once the user has finished filling out this form, pressing the "OK" button will invoke this modules Home Page shown in Figure 1-15. This form is a hypertext Contents page of the TEMP format conceptualised in chapter 5, Figure 5-7 and is summarised in Table 5-1. This form also has the History and Glossary help facilities of module I, as well as the capability of invoking the user information form at any time to see which TEMP is being developed.

It is now a matter of invoking each section one at a time, and filling out each sub-section aspiring to that section of the TEMP.

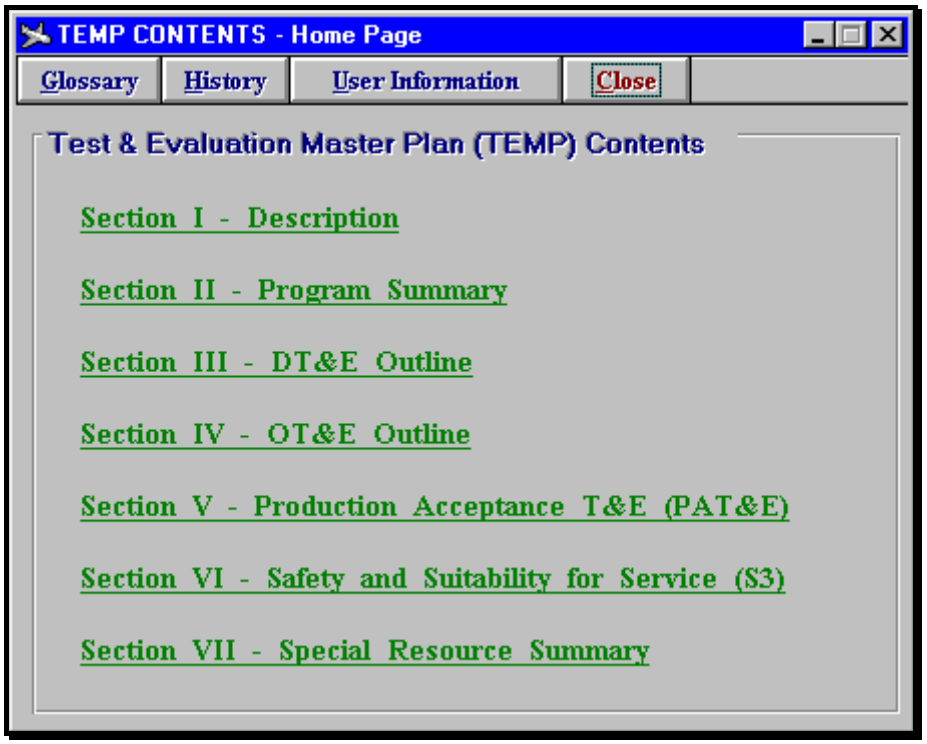

Figure 1-15 (Module II Home Page)

There is no other way to learn about the software and in particular this module without trying it out for oneself, however, for demonstration purposes, certain forms will be briefly analysed pertaining to the progression of Figure 1-15. Section 1.4 is illustrated in Figure 1-16. This form prompts the user to enter the thresholds for the operational effectiveness and suitability characteristics with the help of a calculator and definitions of each characteristic, imposed on the user. You'll notice that, the form also a "TEMP Format" button, this button invokes the Australian CEPMAN 1 TEMP format description at the position of the particular section of the form, in this case section 1.4.

Another form distinguishing sections 1.5 and 1.6 respectively is shown in Figure 1-17. This form allows the user to choose the hardware type of the system, and thus in doing so automatically assigns suitable Required Technical Characteristics for the user. For demonstration purposes the author has conveniently chosen Computers. These characteristics are clearly defined in Table 1-1 of Annex I in Appendix VI.

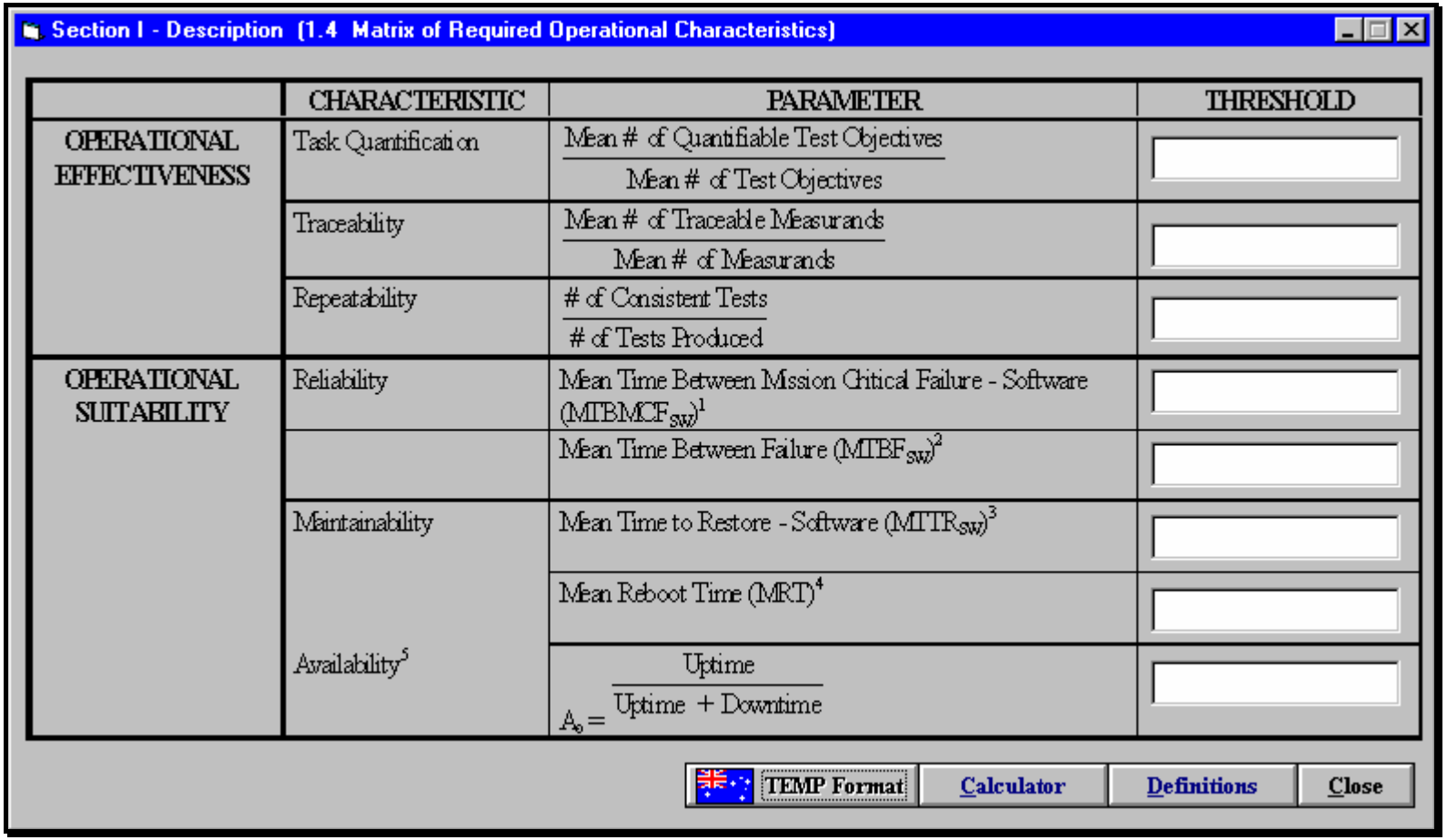

Figure 1-16 (Section 1.4 - Matrix of Required Operational Characteristics Form)

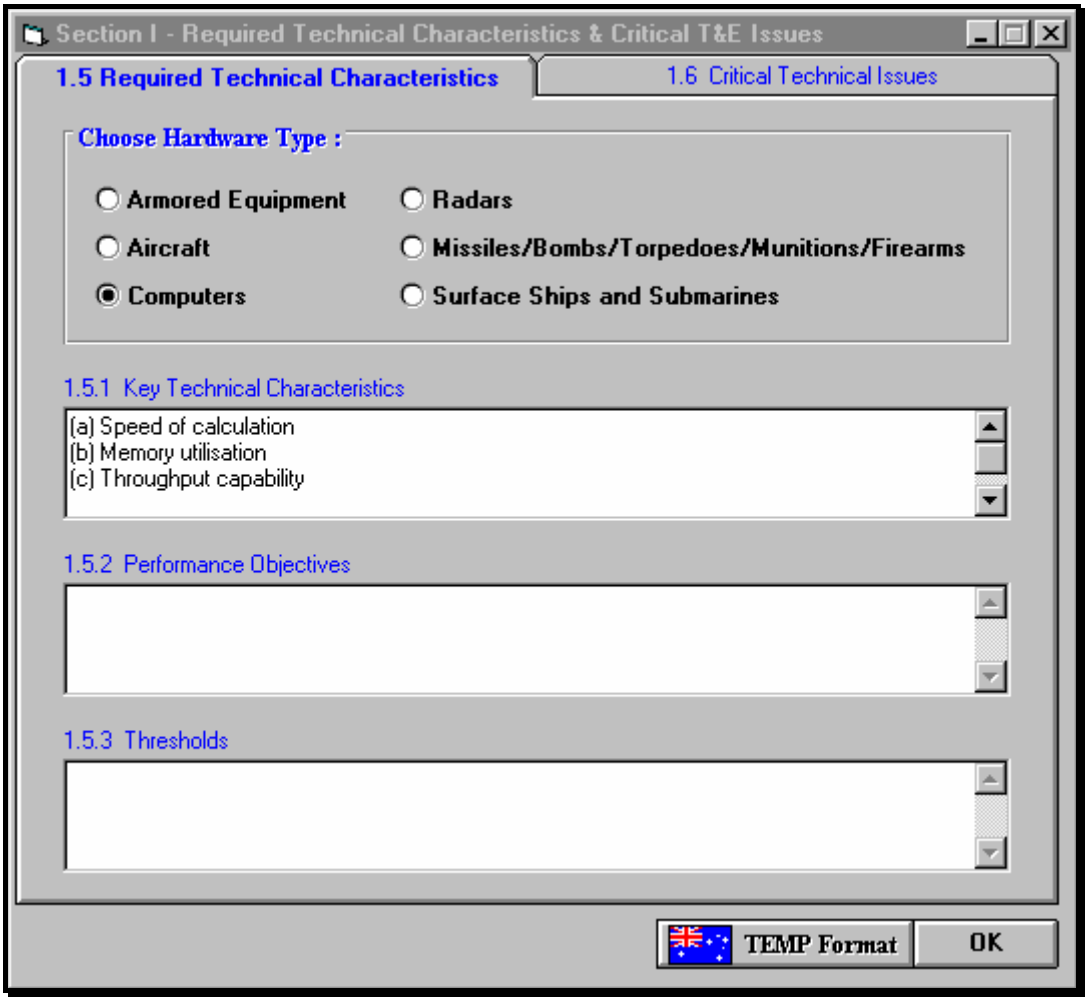

Figure 1-17 (Section 1.5 & 1.6 - Required Technical Characteristics & Critical T&E Issues Form)

As is evident the TabStrip Control is made use of in this form so as to incorporate more than one section on the one form. This 16-bit control box (.OCX) has decreased the amount of design time considerably, especially considering all the sub-sections of a document like a TEMP, it would of have implied a separate form for every sub-section, and meant a very tedious module development stage, and even more so a very difficult task for the user entering the data, having to change, open and close a form each time.

Another typical form for entering data is that of section 3.0, illustrated in Figure 1-18. This form would of had to be broken up into 6 individual forms, had the TabStrip Control not been used. This form shows section 3.2.2 selected for data entry.

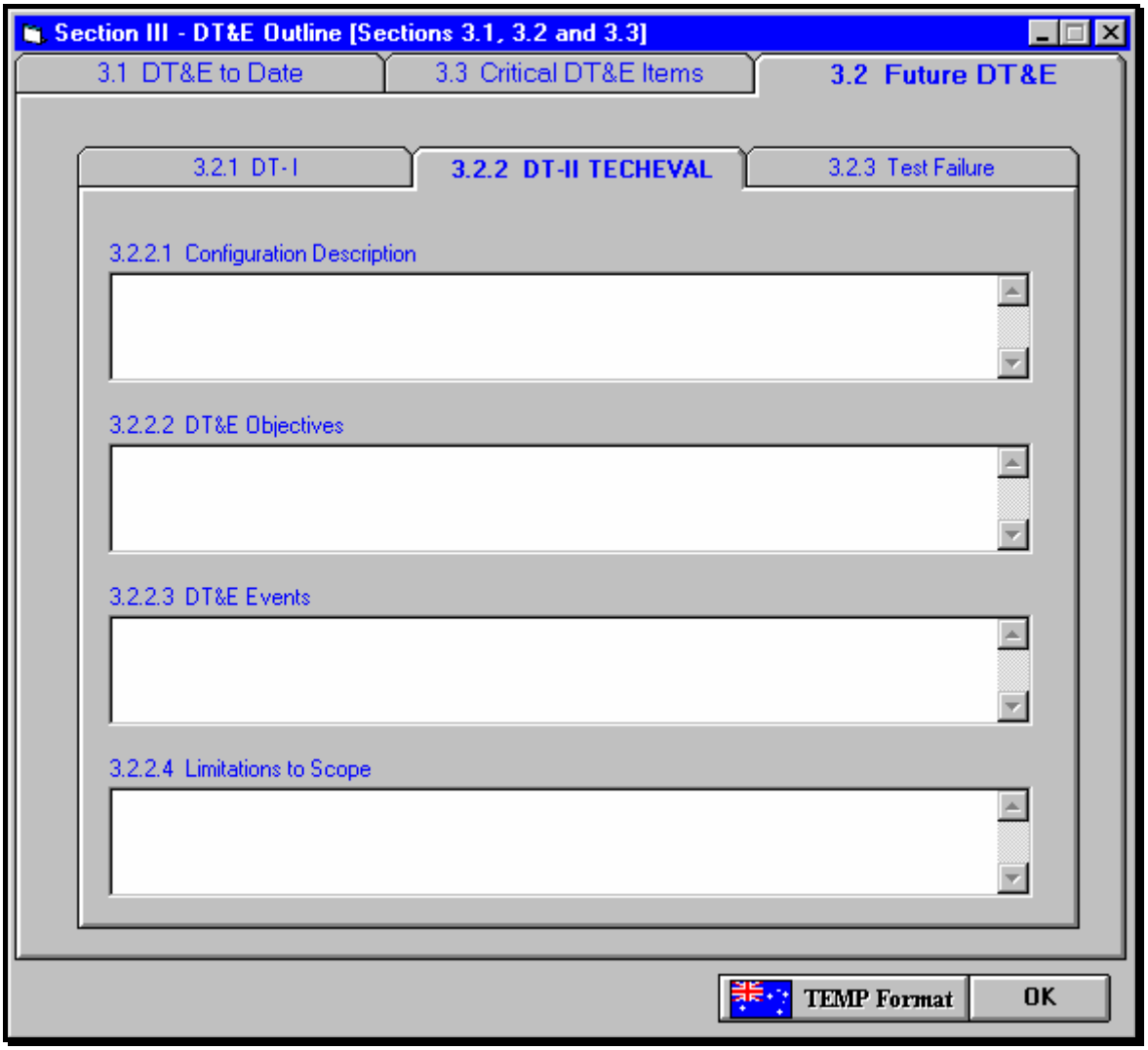

Figure 1-18 (Section 3.0 - DT&E Outline Form)

The final form that is of interest is section 7.2.4, illustrated in Figure 1-19. It details all the test phases and allows the user to enter the date, test site and test system for that phase. The test system column, uses a control known as a DBCombo Box Control. Simply put, the programmer can fill it with predefined text options, and hence the user can choose any one of these options, in this case, a Unix Workstation, IBM PC, or a Macintosh, to fill in that box. However, this type of control cannot learn, i.e., can be updated during run-time, only during design-time. There are other controls that incorporate this facility.

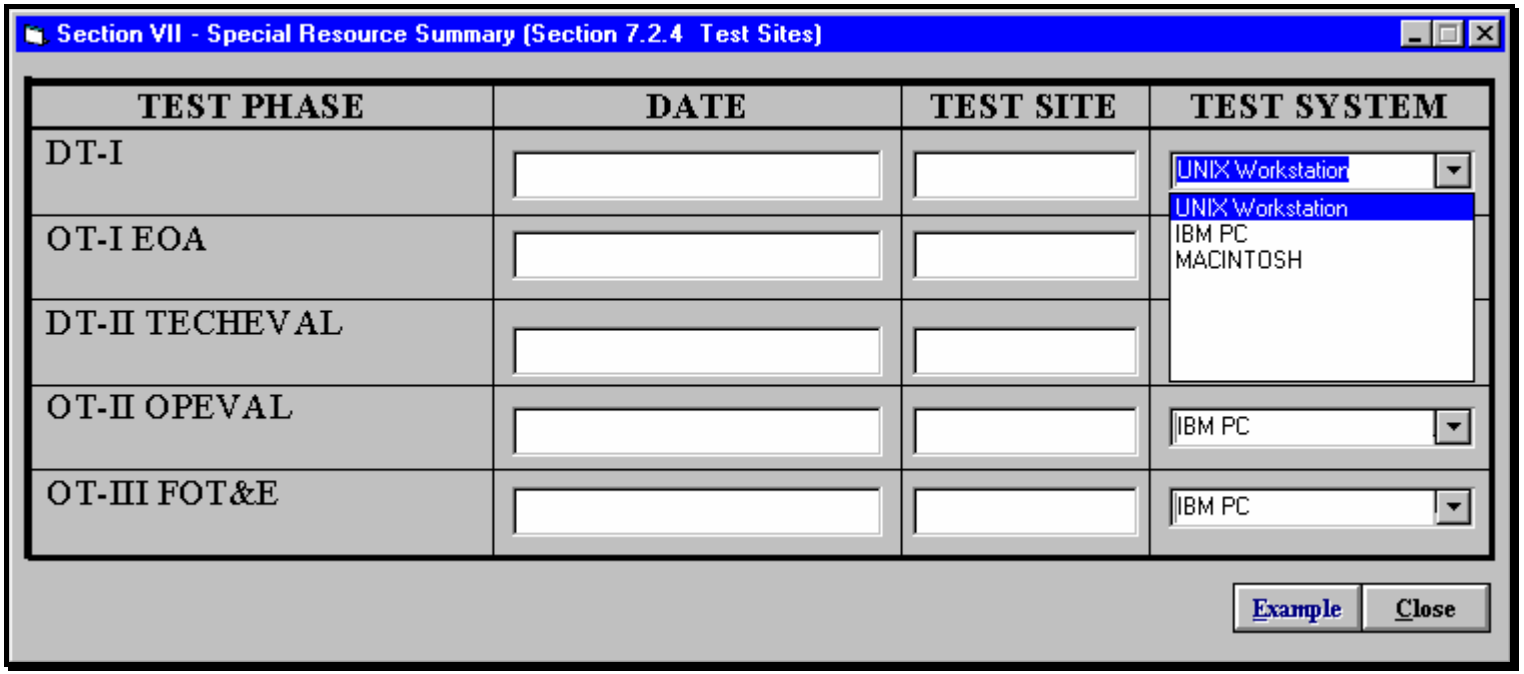

Figure 1-19 (Section 7.2.4 - Test Sites Form)

1.4.2.2 Communication Mechanisms

As is insinuated in Figure 1-1, all the three modules communicate with other applications such as Microsoft® Word 7.0 for the development of the TEMP document via the use of Word macros, and Microsoft Access® 2.0 for the storage and access of data entered by the user needed to fill the TEMP document. The mechanisms used for this communication is Dynamic Data Exchange (DDE), Object Linking and Embedding (OLE) Automation, and Structure Query Language (SQL). These three mechanisms are defined and discussed in the following sections.

1.4.2.2.1 Dynamic Data Exchange

As described by Microsoft Press (1994), DDE is a mechanism supported by Microsoft® applications in Windows that enables two applications to "talk" to each other. DDE automates the manual cutting and pasting of information between applications, providing a faster vehicle for updating information. More specifically, DDE essentially provides three capabilities (based on Microsoft Press (1994):

- You can request information from an application. For example, in a DDE conversation with Microsoft Access®, Word or Visual Basic® macro can request the contents of a record or range of records in a Microsoft Access® database.
- You can send information to an application. In a DDE conversation with Microsoft Access®, a Word or Visual Basic® macro can send text to a record or a range of records in that database.
- You can send commands to an application. For example, in a DDE conversation with Microsoft Access®, a Word or Visual Basic® macro can send a command to open a database from which it wants to request information. Commands sent an application must be in a form the application can recognise.

The Microsoft Press (1994) also states that two applications exchange information by engaging in a DDE *conversation*. In a DDE conversation, the application that initiates and controls the conversation is the *client* and the application that responds is the *server*. The role of the client and server application is best described by Figure 1-20. Each conversation is identified by a separate *channel* number.

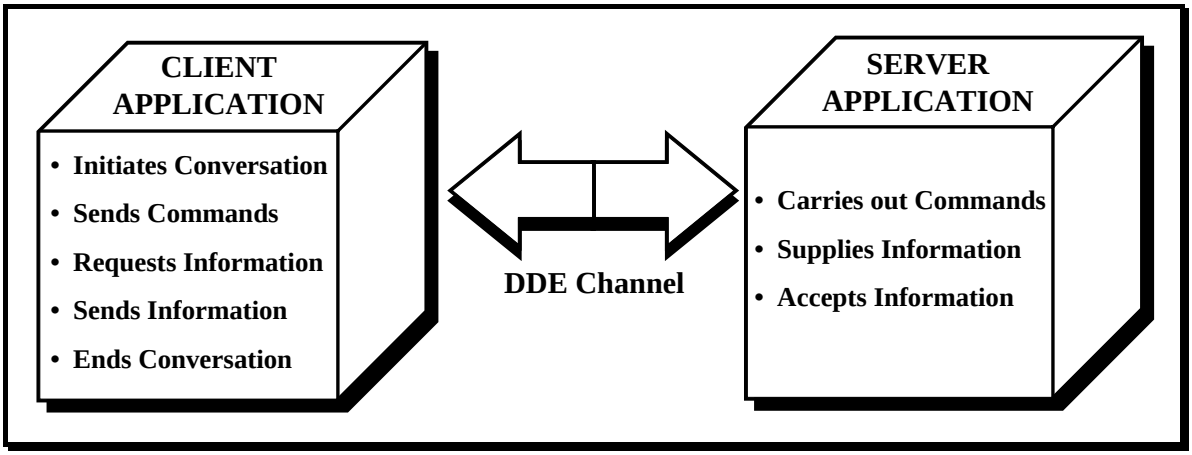

Figure 1-20 (The Roles of the Client and Server Applications in DDE (based on Microsoft Press (1994))

A key requirement for a DDE conversation is that both applications be running. If an application is not running, a client can not initiate a DDE conversation with it. For that reason, a macro that initiates a DDE conversation usually includes instruction's that carry out the following three steps (Microsoft Press, 1994):

1. Determine whether the application you want to talk to is running.
2. Start the application if it is not already running.
3. Initiate the DDE conversation.

1.4.2.2.2 Object Linking and Embedding

OLE Automation is a protocol (Microsoft Press, 1994) to replace DDE. As with DDE, an application can use OLE automation to share data or control another application.

Microsoft Press (1994) also states that in OLE automation, Word provides another application (called the "container" application) with an *object* - a unit of information similar to a topic in DDE. Word supports a single object called "Basic" for OLE automation. You use the "Basic" object to send WordBasic instructions to Word. The technique is similar to sending commands to Word through DDE, the difference being with OLE automation, WordBasic instructions can return numbers or strings directly to the container application.

This makes it possible to use the WordBasic instructions as an extension of the container application's macro or programming language[5].

1.4.2.2.3 Structured Query Language

The Structured Query Language (SQL) as stated by the Microsoft Press (1994) is an industry-standard database language used by the Microsoft Jet database engine. SQL is a database programming language with origins closely connected to the invention of the relational database by E.F. Codd in the early 1970's. Modern SQL has evolved into a widely used standard for relational databases, and is defined by the American National Interchange Standard (ANSI).

The SQL language is composed of commands, clauses, operators, and aggregate functions. These elements are combined into statements used to create, update, and manipulate databases. SQL provides both Data Definition Language (DLL) and Data Manipulation Language (DML) commands. Although there are some areas of overlap, the DDL commands allow you to create and define new databases, fields, and indexes, while the DML commands allow you to build queries to sort, filter, and extract data from the database.

The Microsoft Jet database engine provides two separate methods for accomplishing most database tasks (Microsoft Press, 1994):

- A *navigational* model that is based on moving around directly in the database records.
- A *relational* model that is based on the Structured Query Language.

Thus, the beauty of SQL is that you can implement software routines for the manipulation of the data entered by the user, such as sorting, collecting, filtering, in two or three lines, as opposed to pages of code to carry out the same task. Of the many areas that SQL was made use of, it was particularly used to implement the routines to search and locate the personal particulars record of data that belongs to the TEMP title chosen by the user to modify or create, in the user information form of Figure 1-14.

[5] It is important to note that Word can provide an object to another application for OLE automation, but it cannot use OLE automation to access objects in other applications. In other words, applications that support OLE automation, such as Visual Basic®, which can use OLE automation to access Word, but Word cannot use OLE automation to access them. In DDE terms, Word can act as a server for another application, but it cannot use another application as a server.

1.4.3 Module III - Automating the TEMP Generator (Autotemp.doc)

The task of preparing the final TEMP document named Autotemp.doc in Microsoft® Word 7.0 (herein referred to as Word), involves acquiring the requirements (user input) from AutoTEMP© fields which are inserted into a Microsoft Access® 2.0 (herein referred to as Access) database file called Autotemp.mdb. Hence, automatically generating the Autotemp.doc, the TEMP complying to the CALS conceptualised template of Figure 5-7 (detailed in Appendix VI).

The template however is not in the required Word format so as to allow for the correct DDE to take place between Access and Word. This is due to the fact that Word requires special Field Codes to establish the links between itself and other Windows applications, prior to the "transaction" taking place. This Windows application data requirement acquisition process is illustrated in Figure 1-21.

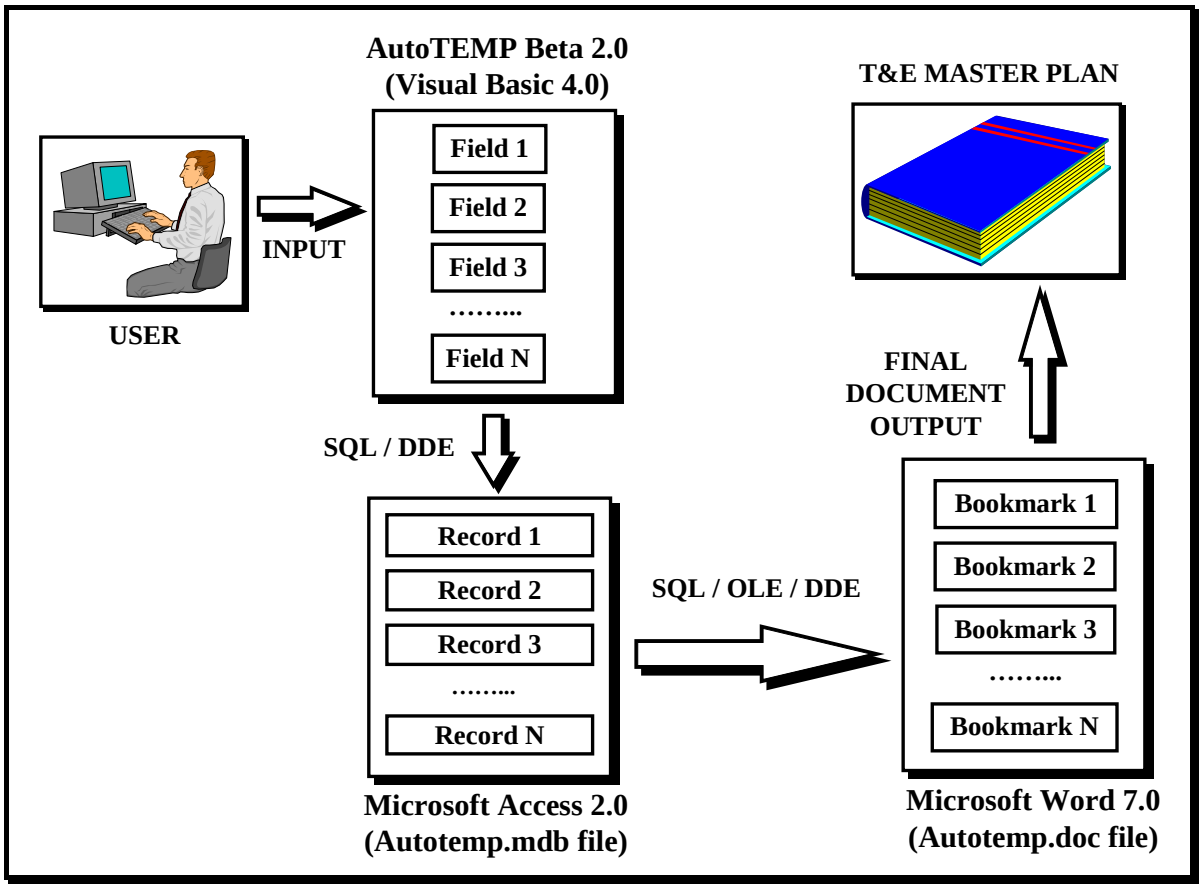

Figure 1-21 (Data Requirement Procurement Process using Windows Applications)

1.4.3.1 Data Requirement Procurement

Figure 1-21 illustrates that the Visual Basic® 4.0 module II communicates with Access via the use of SQL code. It utilises DDE to store data entered by the user into fields 1 through to N as is shown in the diagram, to the accompanying records 1 through to N in the Autotemp.mdb database file. At this stage both VB and Access are active applications. Once the user has successfully completed this task, then by closing all modules and going back to the header form and pressing the "Generate TEMP using Word Macros" option under the file menu in Figure 1-2, after a short question to double check whether or not this option was not inadvertently chosen, AutoTEMP© opens Word and automatically loads the Autotemp.doc document template that has the attached Word macros written in the WordBasic language mentioned previously. At this stage Access becomes minimised along with VB and Word is now the active application. The user is then prompted by the software telling them that the Word menu bar will be modified to accommodate a menu option for the instigation of the Word macros as is illustrated in Figure 1-22. The modification to the menu bar is illustrated in Figure 1-23.

Figure 1-22 (AutoOpen Macro Menu Bar Customizer Dialog Box)

1.4.3.2 Microsoft® WordBasic

WordBasic is a structured programming language as stated by the Microsoft Press (1994) originally modeled on the Microsoft QuickBasic® language. It combines a subset of the instructions available in standard Basic languages with statements and functions based on the Word user interface. You can use WordBasic to modify any Word command or to write your own, which are known as macros. These macros can be assigned to menus, toolbars, and shortcut keys so that they look and function like regular Word commands. Word is actually written by the use of macros, for example the menu bar facilities such as File Open, File Close, File Save, and so forth, are all sub-macros that are executed automatically each time the user invokes them.

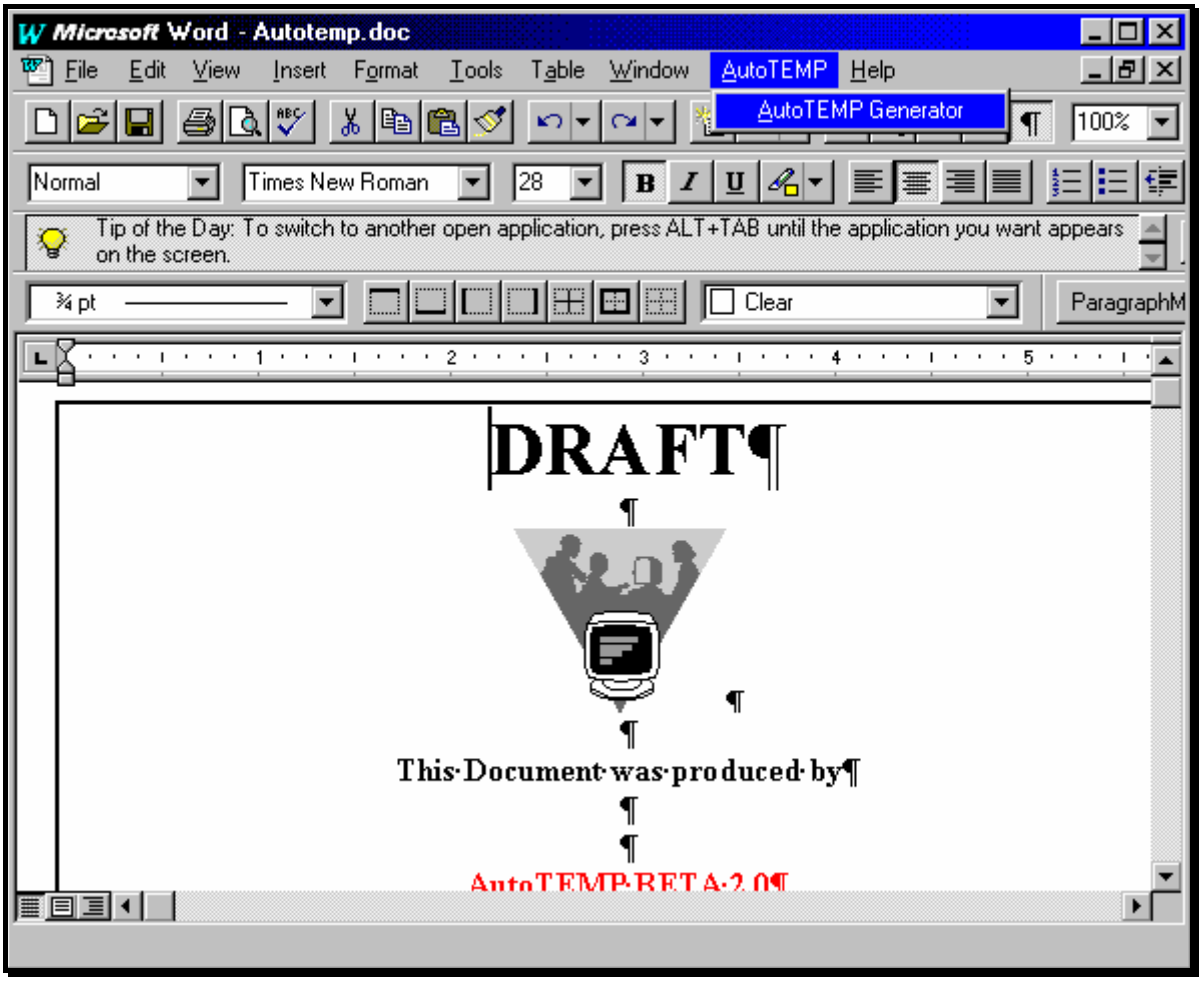

Figure 1-23 (Word Menu Bar Modification)

The only documentation apart from on-line help in Word (which is quite comprehensive), for writing Word macros is the Microsoft® Word Developer's Kit by Microsoft Press (1994), and is essentially considered as an accessory to Word, as opposed for using Word simply as a Word processor. The advantage of WordBasic over other languages and applications that could of have been used to construct the Autotemp.doc TEMP document is that VB also uses a Basic language almost identical in structure and syntax, hence the ease and compatibility whilst programming.

WordBasic allows the user to write and/or record complex macros, and the ability to insert many file types using DDE and OLE automation as described previously, including Access (.MDB) database files.

1.4.3.3 Word Macro Facility

A number of macros, approximately 150, have been written by the author, that are attached to this document whenever it is opened. Figure 1-23 shows some of the Autotemp.doc document, by choosing the Macro option under the Tools menu bar, you can access these macros. This action prompts the "Macro dialog box" as is shown in Figure 1-24.

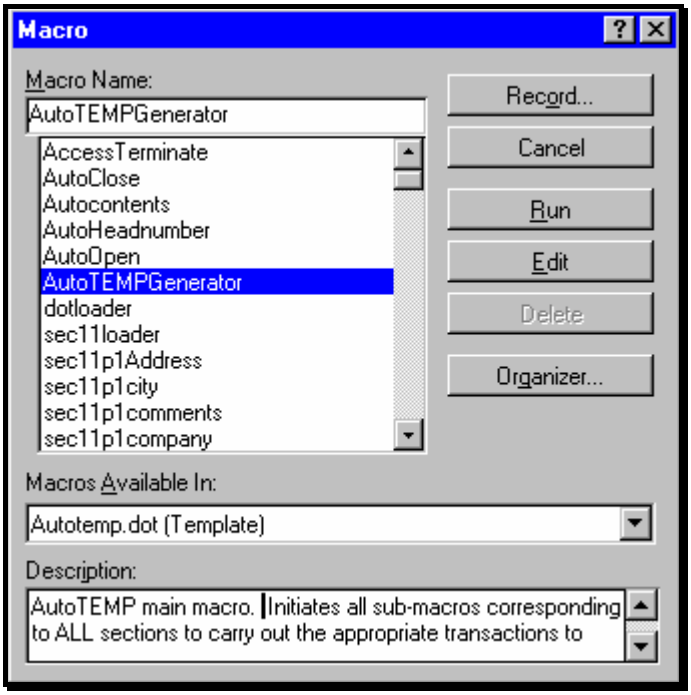

Figure 1-24 (AutoTEMP Macro Dialog Box)

Figure 1-24 illustrates that all of these 150 macros are available in the Autotemp.dot template file, which is a template containing the macros with extension .DOT as opposed to .DOC for normal Word files. The diagram illustrates the selection of the "AutoTEMPGenerator" macro. This macro is the "main program" that initiates all other sub-macros corresponding to all sections of the Autotemp.doc document, of which there are seven, to carry out the appropriate DDE and OLE automation actions between the Access database file, Autotemp.mdb and the Word file Autotemp.doc. A table of all the macros filenames and their descriptions in Autotemp.dot in chronological order, i.e., the order that they are executed, are listed in Figure 1 of Annex I in Appendix IV. Word incorporates macros that run automatically, these are listed along with their description in Table 1-1. AutoTEMP© utilises the "AutoOpen" and "AutoClose" macros. These are the macros that automatically install and un-install the AutoTEMP© menu bar modification, and open and close the Autotemp.doc

document, respectively. Hence, by simply attaching them to the Autotemp.doc document template Autotemp.dot, this causes automatic execution each time the Autotemp.doc document is opened and closed.

MACRO NAME	WHEN IT RUNS
AutoExec	When you start Word
AutoNew	Each time you create a new document
AutoOpen	Each time you open an existing document
AutoClose	Each time you close a document
AutoExit	When you quit Word

Table 1-1 (Automatic Executable Macros)

A similar dialog box prompting the user that the menu bar modification will be un-installed is executed automatically when the user closes the Autotemp.doc document, illustrated in Figure 1-25. A complete listing of all the macro routines used to generate the TEMP document Autotemp.doc, is given in Annex II of Appendix IV.

Figure 1-25 (AutoClose Macro Menu Bar Customizer Dialog Box)

1.4.3.4 Acquisition of Requirements via the use of Bookmarks and DDE

The most useful tool Word provides for identifying discrete parts of documents is the use of bookmarks. A simple use for bookmarks is just to mark a selection or location in a document. You can also use bookmarks to select text between two arbitrary locations in a document. Bookmarks are particularly useful for jumping to a specific location in a document, marking an item so that it can be referred to in a cross-reference, or generating a range of pages for an index entry. AutoTEMP© uses the bookmark feature in this module extensively to position the insertion points in the Autotemp.doc document. This allows the macros to locate the

particular bookmark, and insert the data from that particular record in the Access database file Autotemp.mdb, as is illustrated in Figure 1-21, at that insertion point. This process is continued until all the requirements, i.e., all the fields in the Autotemp.doc document are filled. A comprehensive description of each bookmark, section number, and field names, along with corresponding macro names is given in Figure 2 of Annex I in Appendix IV. The macro routines are sufficiently commented so as to guide the user through their operation. Appendix II of this dissertation contains the actual Autotemp.doc document that shows the field codes, bookmark placements and more or less "raw" document, ready for extracting and inserting requirements required to fill all of its fields at the specified bookmarks.

Once the user selects the "AutoTEMP Generator" option under the AutoTEMP menu item as is illustrated in Figure 1-23, this action instigates the commencement of the "main program" "AutoTEMPGenerator" macro, to run and begin the DDE between Access and Word, for each section until the document Autotemp.doc is filled. At each section the user will be prompted with a similar dialog box as that in Figure 1-25, instructing them that the next section will commence DDE exchange and OLE automation to fill its sub-sections fields. If there is no data to be entered into particular sections the software is instructed to fill those sections with TBD's, or To Be Determined. This action is demonstrated in Appendix III, which depicts a completed TEMP filled with TBD's, what the author would call a "dry run". This action demonstrates that all the macros operate accordingly, i.e., they do what their programmed to do. Once this action has completed, that is all sections 1 through 7 and all the Appendices have been filled, the software then updates each section with the appropriate heading numbering, and jumps to the table of contents bookmark, and inserts a table of contents. Finally, for the document to be complete as a draft at least, there are two very important sections that need planning charts inserted into, these are firstly section 1.3, which requires a diagram of the system to be inserted into, and secondly section 2.2, which requires an integrated schedule. The integrated schedule lists the following T&E aspects, normally in the form of a graphical planning chart:

- Milestones,
- Test article availability,
- Phases of DT&E, OT&E, PAT&E,
- Initial Operational Capability (IOC),

- Full Operational Capability (FOC),
- Funding, and
- Key reports.

The user is prompted at each of these sections and informed of the above, in the case of section 2.2, the user is also prompted with an example of what the schedule should look like as a guide.

1.4.4 Lessons Learnt from Sample TEMP's (Testing)

This section will briefly discuss certain problems encountered whilst developing AutoTEMP© (DT&E) as well as those discovered during operation (OT&E), focusing on the detection of bugs and the quality of the resulting TEMP document.

1.4.4.1 Developmental and Operational Related Software Bugs

A number of software bugs were detected within all three modules whilst developing the CSCI AutoTEMP©. The majority of these bugs were obvious and simple to detect and hence fixed at the time of detection. This was possible because Visual Basic® 4.0 allows two modes of operation, namely, design mode and run mode. In run mode you have access to what is known as a debug window. This feature was widely used as a means of debugging the AutoTEMP© CSCI, and is described in the following section.

The bugs that were more difficult to detect were those in module III, i.e., infested within the macros written in the Word document, Autotemp.doc. These bugs were primarily due to such things as incorrect naming of bookmarks, and field name incompatibilities within the Access database file, Autotemp.mdb. The only mechanisms for detecting those bugs were by trial and error techniques, redesign, fix, test, and so forth, i.e., via thorough T&E.

Module I had some bugs and traps with the printing routines designed to print the forms or their textual contents, especially with line feed and carriage returns, when upgrading from Windows® 3.11 to Windows® 95. Some alterations had to be made to the font types and sizes, as well as to the page alignment, in order to get it right, however these were "ironed-out" in the end, and the CSCI now operates correctly.

Module II was a very tedious module to finally complete because of the vast number of sections in the TEMP document. During its development there were certain bugs occurring

due to the volume of this module with respect to, for example, the user information form. The SQL code used in this form would not access the appropriate record aspiring to the current TEMP that the user had supposedly chosen, as well as assigning the incorrect TEMP ID number. Being new to the SQL programming language, this took longer than the author originally intended to debug, but through the use of the debug window, described in the proceeding section, this was eventually solved, by tracking the operation of the code, line by line.

Module III is now quite "bug-free" and Word no longer detects any operational errors, syntax errors, and so forth, and as previously mentioned all the information captured in the Access database, Autotemp.mdb, is correctly placed into the appropriate insertion points in the Word document, Autotemp.doc, as required.

1.4.4.1.1 Debug Window in Visual Basic®

The Debug window automatically opens at *run time* (the time when code is running). In *break mode*[6] you can use the Debug window to execute individual lines of code, view or change values of *variables* (a named storage location that can contain data that can be modified during program execution) and properties, and view *watch expressions* (a user defined expression that allows you to observe the behavior of a variable or expression). At run time, you can use it to display data or messages as the program runs. At *design time*, the time at which you build an application in the building environment by adding controls, setting control or form properties, and so on, you can view previous output to the Debug window, but you can not execute code.

1.4.4.2 User Related Problems

As stated previously, AutoTEMP© provides the user with a direct link to the TEMP format specified by the CAL compliant CEPMAN 1 instruction (Australian DoD, 1995), with the click of a button, on each form at the appropriate section, as well as a walk through tutorial on the United States Phased Acquisition Process, module I, not to mention a history and glossary form, again easily accessed by the press of a button. This enabled the user's to be directed to their particular form, backtrack, or acronym/abbreviation that they desired, as well as a reference to the most asked question, "well what should I write in this section ?". Most user's

[6] Temporary suspension of program execution while in the development environment. In break mode you can examine, debug, reset, step, or continue program execution.

simply wanted to explore for themselves, by simply clicking this button or that button, printing a form, looking up an acronym, using each module in turn and in reverse.

This is all very well when there is only one TEMP document to worry about. However, the biggest user related problem was when the user had created more than one TEMP, using the user information form, Figure 1-14, say three or four, and attempted to navigate through each TEMP, and attempting to build one or more TEMP documents. This was due to the simple fact that they would literally get lost, and simply would not know which TEMP their currently working on and which TEMP record would be used to build the TEMP document, the first ?, the second ?, and so forth. This problem was anticipated by the author, and still requires some thinking. From such lessons learnt through all these sample tests, and OT&E, it was clear that the software needed some work in this area, and would have to be looked at within the next version of AutoTEMP©.

It should be noted at this stage, that the software written, namely, AutoTEMP© Beta 2.0, is intended solely for the demonstration of concepts, that is, the ability to conceptualise and automatically generate a TEMP from a functional requirement specification, and not as a commercial piece of software. Perhaps a later version, with some appropriate funding or sponsorship from a defence related agency, would incorporate protuberant commercially viable modifications and additions.

1.4.4.3 Quality of the Final TEMP Document

The quality of the final TEMP document produced by AutoTEMP©, ideally can only be as good as the conceptualisation of the TEMP model or template allows it to be. Merz (1995) states as per The Prince, Machiavelli:

> *"It must be remembered that there is nothing more difficult to plan, more doubtful of success, nor more dangerous to manage than the creation of a new system. For the initiator has the enmity of all who would profit by the preservation of the old institution and merely lukewarm defenders in those who would gain by the new one."*

One can only say that the effectiveness of this software in increasing the efficiency, and decreasing the time and cost in generating a TEMP has by far been accomplished, however, the fruits of the author's labor has still yet to be seen, in the eyes of the lukewarm defenders

who will gain even greater quality by the next version. It must be remembered that, this work can only be appropriately compared to that of the work of Roth (1994), with his Automated Test Planning System (ATPS) software, that is reviewed and analysed in chapter 5. In a nut shell Roth's ATPS software does not produce a formatted draft TEMP, but simply a skeleton in the form of a text document with answers entered by the user to a number of questions, that would aid in the development of a TEMP. However, on the same token it incorporates a Test and Evaluation Program Risk Assessment module that looks at the possibility of risk involved, in the T&E process, that AutoTEMP© does not. One can argue that this was not part of the original objectives of the software.