ToolBook as a component: the place of ToolBook in the software toolkit.

Patricia Paterson, Roger Beresford, John Rosbottom
Department of Information Science
University of Portsmouth
Milton Campus
Southsea
Hampshire, PO4 8JF

E-mail: Patersonp@uk.ac.port.cv
Beresfordra@uk.ac.port.cv
Rosbottomj@uk.ac.port.cv

Abstract

When treated as a component in an integrated environment, ToolBook becomes an important resource in the software toolkit. In this paper, we examine the following uses of ToolBook:-

- as a component of a communications software interface.
- as a component of an open model for hypermedia with dynamic linking.
- as a component of a larger software design tool.
- as a component within an expert system learning environment.

We consider the ways in which it is possible to communicate between different packages by using the facilities of Dynamic Data Exchange (DDE), Object Linking and Embedding (OLE) and sharing resources using Dynamic Link Libraries (DLLs)

We examine how ToolBook makes use of these facilities in certain specific instances, and with what degree of success.

To illustrate the use of the underlying communications paradigm, we refer in detail to the following applications designed and implemented using ToolBook at the University of Portsmouth-

- A guide to the Joint Academic Network (JANET) which offers learning and reference material and automatic connection to JANET services.
- A partner to Microcosm, another hypermedia package developed at the University of Southampton DECS taking advantage of the strengths of each package.
- Tutorial and advice modules within a Designer Support Package developed as part of the EC RACE GUIDANCE project.
- A tutorial component and good interface in association with an expert system implemented in XiPlus.

ToolBook as a component: the place of ToolBook in the software toolkit

Introduction

Asymetrix describe ToolBook as "a Software Construction Set for Windows". There are basically two things that one can do with any construction set. Firstly, one may treat it as a closed environment and use it to build applications which are self contained. For example, one may build models made entirely of Lego bricks or Meccano strips, and play with them in isolation. Alternatively, one may use the elements of the construction set to build units which themselves then become part of wider, open applications. The Lego or Meccano-based model may be a ride or side-show in a fairground, or the model engine may be used to drive rides on a fairground made of an entirely different material.

It is in this second way of using a construction set that we are interested; namely, the use of ToolBook as a component of, or interface to, another package. When treated as a component in an integrated environment, ToolBook becomes an important resource in the software toolkit. In this paper, we examine the following uses of ToolBook:-

- As a component of a communications software interface.
- As a component of an open model for hypermedia with dynamic linking.
- As a component of a larger software design tool.
- As a component within an expert system learning environment.

Communicating with Windows

Microsoft Windows is a graphically oriented multi-tasking operating system for the IBM PC. It allows the user to run a number of programs, or applications, concurrently, each within its own area of the screen, or "window". The user manipulates the software by operating through the relevant window or windows. Windows also provides menus by which commands may be issued to the relevant applications, and dialogue boxes allow the application to communicate directly with the user when prompting for information. The user can move easily between applications, and data can be transferred from one application to another by making use of the traditional clipboard cut-and-paste methods.

There is also a message-passing protocol, Dynamic Data Exchange (DDE), which allows inter-application communication and transfer of information. Using DDE, a modification in one application results in automatic updating of any associated applications which use the same information. Recent developments of the windows environment, specifically Windows for Workgroups, mean that the exchange of data and command messages is feasible between instances of compatible software on different computers connected to a Local Area Network. This extension to the basic Clipboard and DDE facility is significant and if fully exploited will lead to a wide range of composite applications, (Lent, 1993).

A DDE conversation occurs when two or more applications or instances of applications pass messages back and forth. The messages are initiated by a client instance, which may request data exchange or request command execution. A server application or instance responds to requests, and either server or client may terminate the conversation.

DDE supports both permanent and temporary exchanges of information. Temporary exchanges are simple transactions and are known as cold links. The two types of permanent exchange are known as warm links and hot links. With warm links, the sender tells the receiver that new information exists but does not transmit it until told to do so. With hot links, information flows whenever the sender has something new to transmit. (Hardaker, 1993)

OpenScript, ToolBook's programming language, provides DDE-support commands and messages to allow ToolBook to act as the client or server in a DDE conversation, or even both, as when multiple instances of ToolBook are running, but it does not support DDE Advise protocol, which establishes permanent hot links between applications, and enables the same data to be held by automatically by both applications.

A DDE-event command or message is sent when information is exchanged between applications or instances, and the word Remote is an essential component of the relevant ToolBook OpenScript statement. For example, when ToolBook is the DDE client, getRemote, setRemote and executeRemote commands communicate with the server application. If a DDE channel is not currently open, ToolBook automatically sends the message WM_DDE_INITIATE to open a channel to the server. The channel will normally remain open until ToolBook has executed all pending handlers (groups of statements with a common purpose), as, for example, when a user turns a page. The default behaviour is then for ToolBook to close the channel by automatically sending the WM_DDE_TERMINATE message, but this can be altered by using the keepRemote command. DDE is useful but can be complicated to set up. DDE links are fragile, and in order to maintain a link, one application must store the complete path of another's document. Reorganising a hard disk or e-mailing to someone with different directory structures causes problems.

A more advanced form of information exchange is implemented via Object Linking and Embedding (OLE). The main objective of this development is data or document integration. For example, suppose one is running both a word processor, such as Word, and a spreadsheet, such as Excel, at the same time, and that a table from Excel forms part of a Word document. In the current implementation of OLE, a single click on the Excel table selects it for repositioning or resizing; a double click on the table activates the Excel application for use. In the next version, (OLE 2.0) still under development by Microsoft, double-clicking on the table will activate the Excel application within the word processor; the menubar changes to display Excel rather than Word menu items and updating takes place within the environment of the word processor. If, in turn, the spreadsheet table should contain a graphic from a drawing application, it would be possible to tunnel through the document rather than activating external windows (Schatzman,1993).

If this functionality is supported within ToolBook, it will open the way to an extremely interesting range of uses of ToolBook as a component of other applications.

Windows also supports Dynamic Link Libraries (DLLs) which are separate files written in C, Pascal or assembly language, that contain functions which Windows applications can use to share code and resources. Windows itself consists largely of Dynamic Link Libraries (DLLs) and these can be used to extend the capabilities of ToolBook. For example, it is possible to link to a DLL and call its functions to perform tasks not available in OpenScript, such as driving sound or keyboard devices. DLLs can also intercept and translate Windows messages to execute OpenScript statements, and they can send Windows messages from other applications to execute or evaluate OpenScript statements. ToolBook uses Windows DLLs to perform actions such as activating windows and displaying dialogue boxes, whose appearance and behaviour are controlled by functions in the appropriate DLL rather than through scripts and properties.

ToolBook as a component of a communications software interface

JANET stands for Joint Academic NETwork, which is a private Wide Area Network (WAN) that interconnects local computer networks in Research Councils, Universities, Colleges of Higher Education and other parts of the academic community in the UK. JANET currently runs the Coloured Book protocols over an X.25 packet switch network. It is connected to the public packet switched networks in the UK and hence, via the Internet, to academic networks throughout the world. By its nature, it is a large and complicated network, and new or casual users often find it very confusing. A guide to using JANET that provides both a reference source and a tutorial, as well as the potential for automated access to the relevant communications network is highly desirable. This section describes such a guide, under development by the University of Portsmouth. The first prototype was produced as a final year BSc project. (Fosker, 1993)

The JANET guidebook, displayed in ToolBook, consists of four main sections. These contain information about how to use the Guide, an introduction to JANET, details of Services available with automatic access and a Glossary.

At present, access to JANET within the University of Portsmouth is via a dumb terminal or via a PC running a communications package which makes the PC appear to be a dumb terminal. The University of Portsmouth uses a VAX (which is called CSOVAX under the JANET name registration scheme) as a mailserver. ToolBook was used to provide the reference source and the tutorial, and to act as an easy to use front-end to the JANET services via the communications package. Crosstalk for Windows was chosen as the communications software because it is a full Windows application, supports Windows DDE and has a high level script language that allows Crosstalk to perform functions such as automatic logging on to a remote computer.

The Crosstalk for Windows scripting language is based on the Crosstalk Application Script Language (CASL). Crosstalk scripts are usually created using the Windows Notepad editor, but any editor capable of outputting plain ACSII text may be used.

Crosstalk scripts are given the .XWS file extension. A script compiler checks the script and outputs a new compiled version with the file extension .XWC. Only this version of the file is needed to run the script.

The JANET Guide ToolBook interacts with Crosstalk using Windows DDE. Crosstalk provides a powerful example of Windows DDE but its implementation and use differ slightly from that of ToolBook. For example, when initiating a DDE conversation from Crosstalk, one must first declare the integer variable dde_channel and then use the DDEINITIATE command, as in the example below, so that a DDE channel may be assigned to the conversation.

```
INTEGER dde_channel
DDEINITIATE dde_channel,"TOOLBOOK","SYSTEM"
```

This process, which is automatic in ToolBook, is further illustrated in example scripts later in this paper.

When the user loads the JANET Guide ToolBook, the book automatically loads Crosstalk using the OpenScript run command. This script also requests that Crosstalk run the INITIAL.XWS Crosstalk script that causes Crosstalk to drop any connections that it may have and minimise its window. However, the majority of the interaction between Crosstalk and ToolBook occurs when the user requests the JANET guide to connect to a remote computer service via the service selection buttons on the relevant ToolBook page. The following scripts show the interaction between ToolBook and Crosstalk when using the National Information on Services and Software (NISS)
Gateway service.

a) Script to connect user to the NISS Gateway service: ToolBook is the client application and uses Remote commands as described earlier in this paper. The system responds to a click on the NISS Gateway button by loading the NISS module, transferring across information on user id and password obtained earlier in the ToolBook session, and then running the module.

```
to handle buttonUp
hide group "NISS"
show field "hold on"

--Communicates with Crosstalk via DDE
executeRemote "[bye]" application "XTALK" topic "SYSTEM"
executeRemote "[Load(NISS)]" application "XTALK" topic "SYSTEM"

setRemote "USERID" to text of field "username" of page "password"
application "XTALK" topic "SYSTEM"
setRemote "PASSWORD" to text of field "password" of page "password"
application "XTALK" topic "SYSTEM"
executeRemote "[go]" application "XTALK" topic "SYSTEM"
```

```
executeRemote"[execute(NISS)]" application "XTALK" topic "SYSTEM"

hide field "hold on"
show group "NISS"
end buttonUp
```

b) Script that Crosstalk loads and runs as requested by the given OpenScript example: It maximises the Crosstalk window and then communicates with the CSOVAX by mimicking a user typing commands at the terminal. The script automatically logs the user into the CSOVAX using the variables passed to it by the DDE conversation with ToolBook. When it has logged the user into the NISS gateway service, it waits, checking to see if the user has finished with the NISS system. Then it automatically loads and runs the logoff script ready for implementation when the user has finished.

```
LABEL CALLPAD
MAXIMISE

LABEL WAKEUP
REPLY "|"
REPLY "|"

LABEL LOGON
WAIT FOR 'ETHERPAD'
WAIT 2 TICKS
REPLY "call csovax"
WAIT FOR 'Username'
REPLY USERID
WAIT FOR 'Pass'
REPLY PASSWORD

LABEL PAD
WAIT FOR 'CSO $'
WAIT 2 TICKS
REPLY "pad call uk.ac.niss"
WAIT FOR 'Terminal type?'
REPLY "vt200"


LABEL LOGOFF
CHAIN "LOGOFF"
END
```

c) Script that Crosstalk uses to logoff from the network and return user to ToolBook: Crosstalk is the client application and ToolBook is the server application. The logoff script waits for the user to finish using the remote service and return to the CSOVAX prompt CSO $. It then issues the VAX VMS command logoff, drops the connection to the CSOVAX and minimises the Crosstalk window. As noted earlier, the DDE channel has to be explicitly activated in Crosstalk.

```
LABEL LOGOFF
WAIT FOR 'CSO $'
REPLY "LOGOFF"
WAIT FOR '*** Cleared'
BYE
MINIMISE
INTEGER dde_channel
DDEINITIATE dde_channel,"TOOLBOOK","SYSTEM"
DDEEXECUTE dde_channel,"get bringWindowToTop(sysWindowHandle)"

END
```

The application as implemented was satisfactory as a prototype, but the main problem inherent in the present design is that all details are encoded in developed software. It does not support easy customisation for different environments, in that the links are program- defined, rather than user defined, and require access to both ToolBook and Crosstalk code. Thus, problems with fragile links are encountered. Further work currently being undertaken seeks to build on the prototype and to make the specific links generic under the control of the user.

ToolBook as a component of an open model for hypermedia with dynamic linking

In order to circumvent some of the problems engendered when linking applications with code embedded within the actual programs, it is possible to utilise the clipboard to set up and maintain links. One approach to managing the problem of a multiplicity of applications is illustrated by the use Microcosm makes of DDE, (Fountain et al, 1990).

Microcosm is an open model for hypermedia, under development at the University of Southampton, which provides a platform which should enhance the capabilities and usability of a hypermedia system. It was designed to overcome some of the problems common to many current hypermedia systems, including the authoring effort involved in inserting links into documents, problems with read-only media such as CD-ROMs, and difficulties withproprietary document formats that often make it impossible to take documents from one system and use them in another. Also, most hypermedia systems are closed, and additional drivers which may be added to extend the system are not bi-directional in the sense of being able to make links back into the hypermedia program.

In essence, Microcosm makes it possible to browse through large bodies of multimedia information or to add links and further information to the system. Documents may be of many different types, such as text, video, graphics, sound, and one of the most important new concepts is that no information about links is held in the document data files themselves. Instead, there is a separate links database which holds all the information referring to links. There may be several linkbases associated with a set of hyperdocuments. This allows for the provision of both public linkbases for general access, as may be provided by the original author, and private linkbases as generated by the individual user. The private linkbase is analogous to pencil notes in the margin of a book.

There are three levels of generality of link sources: generic links, where the user may follow the link after selecting the given anchor at any point in any document; local links, where the user may follow the link after selecting the given anchor at any point in the current document; and specific links, where the user may follow the link only after selecting the anchor at a specific location in the current document. Specific links may be made into buttons, as with ordinary hypermedia.

Generic links are of particular interest here. When a new document is created, the author will have access to all the generic links that have already been defined for the system. Links may be to processes as well as to documents, and any Windows application where it is at least possible to select objects and copy them to the clipboard is a candidate for Microcosm.

The DDE (Dynamic Data Exchange) message passing protocol used by Windows was found to meet the requirements of Microcosm. Using the DDE capabilities, the generic link facility can be extended to any application running in the same environmment. Applications integrate into Microcosm in this way include ToolBook, Guide, Superbase and Word, and in theory there is no limit to the number and type of application that can be integrated in this way.

ToolBook as a component of a larger software design tool

Software engineering is the term usually applied to the design and building of large software systems that are built by teams of people rather than by individuals, uses engineering principles and which may contain both technical and non-technical aspects (Sommerville, 1989). Various tools exist to help designers, including graphical design editors, text processors, communications software, testing and debugging tools and project management tools. These often complicated tools themselves usually contain further help for the software designer and builder. It is the use of ToolBook as a component of a new software design tool that we now consider.

GUIDANCE stands for "Generic Usability Information for the Design of Advanced Network Communications in Europe" and is part of the RACE programme for Research and Development of Advanced Communication in Europe. The GUIDANCE project (RACE 1067) involves a new user-centred methodology, the Enabling States approach (Whitefield et al, 1992) and close attention to usability principles is an essential component.

A new software tool, the Designer Support Package (DSP), has been developed as part of the project (May et al, in press). Current implementation has resulted in the development of three strands of the DSP: support for the GUIDANCE method itself (the Design

Environment), support for accessing the usability principles (the Principles hyperdocument) and support for learning how to use the GUIDANCE method (the Tutorial).

The Design Environment allows the user to construct a task model of the application, automatically generates the necessary tables which are part of the methodology and provides full navigational support between the tables, task model and reference materials. The Principle hyperdocument and the Tutorial were prototyped as separate ToolBook books able to be called up at will from the Design Environment.

The Principles hyperdocument contains full definitions and descriptions of the twelve generic usability principles as identified by the GUIDANCE project. These are: Compatibility, Coherence (Consistency), Simplicity, Redundancy, Salience, Reversibility, Transparency, Completeness, Controllability, Flexibility, Feedback and Support Orientation. Research suggests Compatibility should have a higher priority than Coherence (Consistency), (Wang, 1992) and that, together with Simplicity, application of these three usability principles should be considered first. Application of the other principles depends on the context of the design, but an understanding of the issues involved is clearly necessary and the current Principles hyperdocument is designed to offer advice in the following way.

For each generic principle, a description and rationale is given. The principle is illustrated with examples of general use in the areas of Hypermedia, Computer Supported Co-operative Work (CSCW) and Communications Services Integration. Further examples are given of the application of specific instances of the principles as relevant to the design of a multimedia multiauthor document production (MMMADP) interface. These are based on the case study in Multimedia Conferencing which served as a vehicle for the development of much of the GUIDANCE work and which is illustrated in the Tutorial.

The Tutorial contains information on how to use the GUIDANCE method, the Designer Support Package and the Principles hyperdocument. It provides a Guided Tour to introduce the user to navigation within and between the various elements of the design tool, and to the contents of these elements. There are examples of the use of the GUIDANCE method, the DSP and the application of the relevant Principles in the design of the MMMADP interface referred to earlier. In its present form, the Tutorial functions as a modified multimedia interactive on-line help facility, with navigation supported by overview diagrams which continually update.

For operational reasons, the first prototype DSP was eventually implemented in Lisp on a Macintosh IIcx with an A3 colour monitor. The early work on Principles and the Tutorial, implemented in ToolBook, was modified to run in SuperCard 1.5. However, theoretical work on the GUIDANCE project has tried to formalise the expression of usability guidelines in a prescriptive manner, (Whitefield et al,1992). This implies the need for a rule-base and the application of knowledge engineering within the design environment. These in turn will place further demands upon the support offered to the designer via the

Principles hyperdocument and the Tutorial, and automatic transfer of data between the instances of the different elements of the DSP will become highly desirable. Future implementations of the DSP are expected to be PC-based and in a Windows environment. This implies the use of ToolBook and Windows DDE or OLE links between the strands of the DSP.

ToolBook as a component within an expert system learning environment.

There is no agreed definition of an expert system. Simplistically, it could be described as a program that emulates a human expert, who may in turn be considered as someone who is proficient in certain skills or knowledge. A more restrictive definition could be a program which has a large knowledge base in a restricted, specialised field. This knowledge base may be derived from one or more human experts, and the expert system uses inferential reasoning to perform tasks for which a human expert would otherwise be required.

XiPlus is a shell based expert system construction kit produced by Inference Corporation and release 3.50 R0 is a recent version produced for Windows. Earlier versions of XiPlus were not known for the friendliness of the user interface, and it was hoped that the Windows version would be more intuitively usable. However, these expectations were not realised. Since ToolBook provides an excellent means of designing interfaces, and both ToolBook and XiPlus are designed to interface with other packages, one might assume that combining the two would be productive.

Systems or products built under XiPlus are known as Applications, and all Application filenames end in .APC. Code can be written to call XiPlus from within ToolBook, or ToolBook from within XiPlus.

a) Script to load XiPlus from ToolBook, using a button.:
If the full path to the XiPlus executable code is not defined by the DOS path command, it is necessary to include the full path name to the executable file for XiPlus and the full file name for the XiPlus Application. This example assumes that XiPlus for Windows is held in the C directory and that the Application is called myexample

```
to handle buttonUp
run C:\XIPW\XI3.EXE myexample.APC
end buttonUp
```

b) Script to load ToolBook from XiPlus:
To call up ToolBook whilst running an XiPlus instance, it is necessary to write code into the text file of the knowledgebase. Because XiPlus is an expert system designed for backward chaining, it is necessary to build procedures into the

program's code which will trigger the desired commands. This can be achieved by installing a condition in the rule base that checks to see if a given form has been "done". "Doing a form" means that the system sends a preconstructed form to the screen, the user completes it via the keyboard and, on exiting the form, the necessary condition fires. In this example, the form is known as "details". There are three rules shown below. The first fires if the details are unknown, and "does the form"; the second fires when the details from the form are known and the ToolBook application is not required; the third fires when it is required to get data from "Mybook" within the ToolBook application.

```
if details are unknown

then do form details
and details are entered

if  detail is somethingtrue
and evidence is some
then {do whetever is relevant}
and condition is known

if  detail is somethingtrue
   and evidence is none
then report more investigation is needed
   and command save state result
   and command dos tbook.exe mybook.tbk
   and condition is known
```

As can be seen from the example, the interfacing between ToolBook and XiPlus requires considerable understanding of the structure of the application language and is embroiled in the semantics of the backward chaining rulebase. To date, only the transfer of control has been solved; the transfer of information for use within the rulebase has, so far, proved intransigent.

Conclusions

We have considered how ToolBook is currently being employed as a component or element of the software toolkit used in an integrated environment. At its current stage of development, Window's software allows real-time exchange of information between applications using Dynamic Data Exchange and real time integration of information between applications using Dynamic Link Libraries to store common code and to execute common tasks. An exciting future development will be when one can call and run one application actually within another, for example, by embedding the relevant icons and then activating them. In this way, tunnelling through a number of applications should be possible. This tunnelling through layers of applications seems to be permitted by the draft specifications released for OLE 2.0, but the Microsoft guidelines do not advise this practice, (Andrews, 1993). However, there are still problems with OLE and one can have automatically updated linking, or unbreakable embedding, but not automatically updated, unbreakable links. Further development of reliable dynamic linking really awaits the coming of a new, object-oriented approach to the underlying operating system, the promise of which is dangled in CAIRO the code name of the development project being undertaken by Microsoft, (Anon, 1992).

References

Andrews, D. (1993) Microsoft Demonstrates OLE 2.0 BYTE's Essential

Guide to Windows, Spring 1993, p 22

Anon. (1992) CAIRO is Microsoft's Future. BYTE's Essential Guide to Windows, Autumn 1992, p 13

Fosker, M. (1993) A Hypermedia Guide to JANET. BSc Project Dissertation, University of Portsmouth.

Fountain, A., Hall, W., Heath, I. and Davis, H.(1990) MICROCOSM: An Open Model for Hypermedia with Dynamic Linking. CSTR 90-12, University of Southampton.

Hardaker, M. (1993) The Exchange Mechanism. Windows User. May, 1993, pp 143-146.

Lent, A. (1993) Workgroups By The Numbers. BYTE's Essential Guide to Windows, Spring 1993, pp 32-35.

May, J., Byerley, P., Denley, I., Hill, B., Adamson, S., Paterson, P. and Hedman, L. (in press) The Enabling States Method. In: Byerley, P., Barnard, P. and May, J. (Eds) Usability and Integrated Services: Design Issues and Methodology. Elsevier.

Schatzman, B.D. (1993) Next-Generation OLE. Byte. March 1993, pp 209-210.

Sommerville, I. (1989) Software Engineering, 3rd ed. Addison-Wesley. p3.

Wang, E. (1992) Usability Evaluation for Human-Computer Interaction. Thesis (PhD). Lulea University of Technology, Sweden

Whitefield, A., Byerley, P., Denley, I., Esgate, A. and May, J.(1992) Integration of services for human end-users (1): Design principles, enabling states analysis and a design method. In: Byerley, P. and Connel, S. (Eds). Integrated Broadband Communications: Views from RACE - Usage Aspects. North-Holland Studies in Telecommunication Vol 18. Elsevier.