Among Harvard's Libraries: A new software interface for cataloging in Widener Library

A New Software Interface for Cataloging in Widener Library

William Hays

Two years ago in these pages, Kenneth Carpenter[1] and Michael Kaplan[2] described the ten-year history of the automation of cataloging in Widener Library. Over that time, work with the on-line catalog shifted from mainframe terminals to personal computers, the process of cataloging books became much more efficient, the accuracy of data entry increased, and productivity increased dramatically. What had been repetitive, mundane tasks at a computer terminal became, on the PC, part of an automated batch process or were transformed by the use of macros, editable or recordable sequences of commands attached to compound keystrokes. With the introduction of additional desktop software for cataloging and connecting to databases other than HOLLIS, PCs became "technical services workstations," realizing the promise of having a computer on every desk in the Department of Cataloging Services. Michael Kaplan, Team Leader of the Database Management Team and Coordinator of OCLC and RLIN Operations for Harvard College Library, is principally responsible for having led the department in this area, and he sustains an international reputation for his work.[3]

Cataloging productivity reached its highest level in 1995. Moreover, since the data entry component of cataloging had been reduced, repetitive strain injuries were rarely reported. Yet at the same time, the department was in a precarious situation with regard to its desktop software. At the center was Cornell Telnet, an outdated DOS terminal emulation program used to connect to HULPR, the technical services module of HOLLIS. Most of the critical, time-saving pieces were implemented with share-ware programs such as the macro development program NewKey or local developments also in the DOS environment. Many secondary pieces of cataloging software, such as the Cataloger's Desktop from the Library of Congress, were just starting to be issued for Windows, the new standard for PC operating systems. Though in general there was backward compatibility for DOS programs within the Windows environment, a mismatch in communications protocols prevented us from integrating the new Windows software with the older DOS applications. Additionally, some of the motivating features of Windows such as multitasking and the interoperability of true Windows applications did not apply to DOS programs. The very advancements in technology that allowed us to make gains prevented us from moving forward into a total Windows environment.

Finding the Right Software

A search was undertaken to find a suitable Windows terminal emulation package that would support a connection to HULPR and permit us to use macros in the Windows environment. Technical service departments in libraries are a very narrow market, and few applications have been written specifically for working with library systems. In fact, only one Windows terminal emulation product, TCP3270 from McGill University, was being used for NOTIS-type library systems on IBM mainframes such as HOLLIS, and two years ago, its then-current version contained only a very limited macro facility, one that could not duplicate our capabilities under DOS. A new version, originally due out in the fall of 1994, did not become available until January of 1996. Our hopes were pinned to this software, but initial tests showed that it was not stable. After six months of testing new sub-releases and trying to retrofit the older but stable version with back-door client applications to provide macro capability, we were left with nothing. A brief attempt to adapt OCLC's new Passport for Windows for connecting to HULPR also proved untenable.

An alternative course was to search for a generic IBM terminal emulation package that could be adapted for working with HULPR, including the specialized ALA character set. Rodney Goins, head of Automation Services for Harvard College Library, suggested WRQ, a Seattle-based company that specialized in terminal

1 Kenneth Carpenter, "Cataloging Books in Widener," *Harvard Library Bulletin*, n.s., 6 (Spring 1995) : 3-8.

2 Michael Kaplan, "From Worksheet to Workstation: Technical Services Workstations in Harvard College Library's Cataloging Services Department," *Harvard Library Bulletin*, n.s., 6 (Spring 1995) : 8-12.

3 Michael Kaplan, ed., *Planning and Implementing Technical Services Workstations* (Chicago : American Library Association, 1997).

emulation software used primarily in the corporate setting. On 1 July 1996, I consulted their website and found the product that was to change markedly the way in which the department staff catalogs books. WRQ sent me an evaluation copy of Reflection for IBM, and within two weeks the basic problem of connecting to HOLLIS and HULPR on the mainframe and displaying the data properly was solved. I knew we had a winner.

On first evaluating Reflection, it was clear that it had many features that would enable the development of new cataloging capabilities. Foremost, it contains its own scripting language, Reflection Basic, that is very similar to Visual Basic, the Windows development platform and language for writing graphical user interface applications. With Reflection Basic it would be possible to move beyond the simple macros of Newkey into a much more powerful programming environment. One compelling feature is the ability to send commands in a script program just when the mainframe is able to process them. As all users of HOLLIS know, the average response time is almost instantaneous, but occasionally it is noticeably lengthy. Other applications with macro capabilities such as NewKey or TCP3270 required specifying an amount of time for the macro to pause, which often failed when the response time of HOLLIS was longer than expected. Reflection is always successful and always as fast as HOLLIS.

Another feature in Reflection script programs derives from having a full-fledged programming environment with text handling capabilities. Previously, selecting a field for copying from a bibliographic record required the user to meticulously identify the beginning and ending of the field. Although HOLLIS stores and transmits data fields as separate units, the display program did not usually separate them except by placing them on separate lines. With Reflection it is possible to select or identify an entire field by clicking just once anywhere on the field.

Aside from the benefits from local programming, Reflection's own application interface lets staff customize their own HULPR sessions and allows them, not the developer, to configure the program to run a script program from a keystroke, a mouse click on a button, or a pop-up menu item. As a policy, it is important for the department to work with the individual staff member to find a comfort zone where potential ergonomic injuries are avoided. Reflection allows a much wider range of variation for adapting the workstation to the user than we have had before.

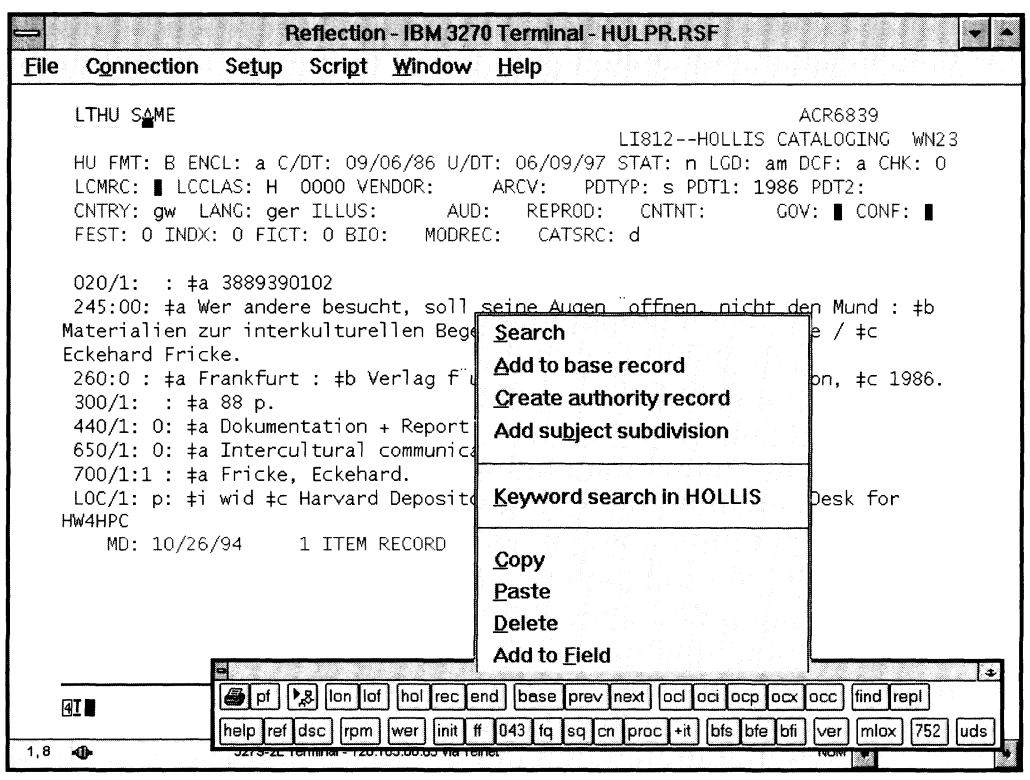

Figure 1. The pop-up menu on a bibliographic record.

PROJECT PRELIMINARIES

Having identified Reflection as a good potential solution to the department's software problems was only a beginning. To implement a software change for the seventy-member department had implications for all technical services throughout Harvard College Library. Administrative and systems units would also have to be involved. HCL Automation would have to support the implementation of the software on a local server, with access through the local area network. The Office for Information Systems (OIS), which traditionally prescribed and developed the software to be used for connecting to HOLLIS and HULPR, would have to verify the accuracy of the data transmission. And though funds to purchase licenses for Reflection were available through the bequest of Bartol Brinkler, a distinguished former member of the department, administrative approval would be required.

Jane Ouderkirk, head of the Cataloging Services Department in Widener Library, provided encouragement and the administrative support to get the project on track. The initial plan was 1) to develop the essential script programs covering existing functionality and yet showing the new kinds of labor-saving features and 2) to prepare a proof-of-concept demonstration, both to be done over the summer of 1996. For input from an accomplished cataloger, I planned to work closely with Bruce Trumble, team leader of the German/Scandinavian Team, who was willing to test out new script programs as they were developed. In addition, HCL Automation and OIS would be consulted on issues relating to them.

INITIAL DEVELOPMENT-SUMMER, 1996

With Reflection, duplicating the macro functionality we had under DOS would be easy, but there was no need to limit the solution by the artificial constraints of the old system. At the outset, I envisioned that the way in which catalogers worked could be better taken into account, especially those catalogers who created original records. For example, very few headings, i.e. data fields on a bibliographic record, are unique to that record. Most headings are given in "authorized form" and are taken from some other source. Therefore, much of a cataloger's work involves consulting other records in HOLLIS or other databases while centered around a "record in progress." There needed to be a way to move around the database between related records without keying possibly lengthy search strings. Optimally, there should be a way to search every heading by just pointing to it. Also, the type of search, such as "author" or "title," should be determined by the type of heading, yet allow the cataloger to easily change any parameters as needed. Similarly, when a heading is identified for copying to the record being created, the script program should allow only valid fields to be added. Programming with Reflection made this kind of sophistication possible since the rules of cataloging could be incorporated into a script program to structure the interface. The particular solution was a pop-up menu that is displayed when a field is selected by mouse click or keystroke, as is shown in figure 1. The user then selects the menu item corresponding to the desired action for the selected field.

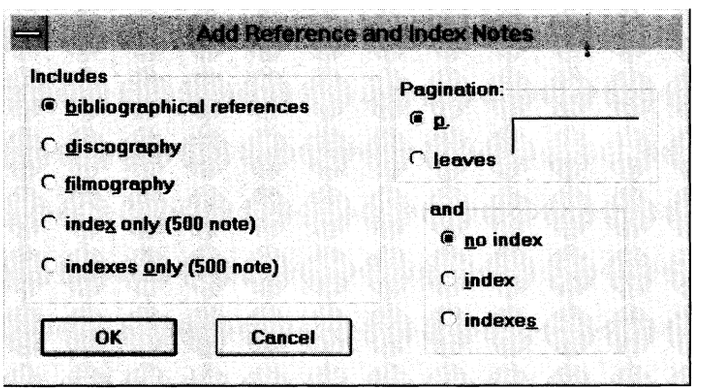

Figure 2. Dialog box for adding a bibliographic references or index note.

Graphical user interfaces (GUIs) are intended to facilitate data entry. For instance, one benefit of GUIs comes from providing valid choices for a particular data field, rather than relying on the operator to key any given item correctly. The danger with GUIs is that identifying the correct choice can be very time consuming between scrolling listboxes, clicking checkboxes, and tabbing through buttons. Since catalogers are expert users, they do not need a list of choices to locate a correct data value and so are more likely to find standard GUI controls an obstacle rather than a help.

Though the Reflection programming language does not provide the full gamut of Windows capabilities, it does provide enough features that choices in a Windows dialog box can be optimized for the most likely case depending on the context. Figure 2 shows the dialog box for adding bibliographic reference or index notes where the default action of selecting "OK" will add the standard note "Includes bibliographical references." to a record. The next most likely scenario would be to add pagination. Typing the appropriate page numbers will add them to the pagination box without having to locate the data entry position. For example, typing "87-92" followed by the Enter key will add the note "Includes bibliographical references (p. 87-92)." Since the new interface consisted of add-on windows rather than a complete reformulation of the main screen, editing directly into the terminal screen emulator is still possible, even fundamental. Nevertheless, most catalogers are using the program scripts with dialog boxes and find that their work is expedited by them.

Many features in the new Reflection interface are invisible to the user. Especially during the local processing, the record is checked for errors and inconsistencies, and the cataloger is notified only when there is a problem. Scripts ask the cataloger only for relevant input because it is able to scan the record to determine what is needed.

Implementation-Fall/Winter 1996-1997

On 9 September 1996, a meeting was held with representatives from the Harvard College Library Administration, HCL Automation, the Widener Library Cataloging Services Department, OIS, and WRQ that eventually led to a licensing agreement to purchase 107 copies of the software to be distributed over the local area network. In this initial round of implementation, Reflection was to be made available only to Cataloging Services and the Judaica, Middle Eastern, and Slavic Divisions. Development of the new cataloging interface continued from September through January with extensive collaboration with some of the original monograph catalogers in the department, especially Daryl Boone (English Team) and Judy Amory (German/Scandinavian Team). Implementation for original catalogers on the German/Scandinavian, English, and French/Italian teams was spread throughout December and January. Later on, I worked with other teams that required specialized Reflection scripts.

Working with Michael Kaplan of the Database Management Team in February and March provided an opportunity to demonstrate the power of Reflection programming to perform database cleanup operations. One of these projects was to examine 90,000 records that had potential duplicates in HOLLIS that had not been resolved in the recently completed RECON project. Over the course of a few weeks, the program merged 65,000 records that met criteria not specified in RECON. This saved an estimated eight months time for a full-time staff member to manually make the same changes.

In March and April, Reflection was to be set up in the Area Studies Department in Widener. In working with Michael Hopper of the Middle Eastern Division and Elizabeth Vernon of the Judaica Division, it became clear that their specialized needs were greater than could be provided for them. In particular, the Area Studies Department had responsibility for acquisitions processing in addition to cataloging. Macro writing under the old system was something that many people in the department had learned. For the most part it consisted of writing or recording a sequence of keystrokes including literal commands and required no special training. If a macro needed to be customized for a particular individual or task, it was usually something the user could solve. In contrast, the new Reflection script programs are uniform, imposed from above, and not customizable by users. They have the benefit of ensuring uniformity of practice, but they cannot adapt to unforeseen needs. To write a script program with Reflection Basic demands programming skills, and any developed software, even if it fell into the modest category of "macro," would require dedicated support for local software development, something Widener had never done before. Rather than add an insupportable burden to the centralized development process, I wrote a program called Reflection Personal Macros (RPM) that allowed staff who were not programmers to write their own macros in much the same way that they had done with NewKey. The program included a script-

ing language, an editing environment, and a means of converting a macro to a Reflection Basic Script for easy distribution among the staff. The Judaica Division has used RPM extensively to create macros for their local operations.

For the Serials Cataloging Team, the introduction of Reflection means a critical change in workflow. Cataloging for CONSER, the national program for cooperative serials cataloging, had been primarily performed using OCLC, the national bibliographic database with its own software. Often the same record had to be manually entered into both HOLLIS and OCLC. Reflection scripts now provide the ability to convert and move records from HULPR to OCLC directly on the PC, which lets the CONSER cataloger work almost exclusively in HULPR, the preferred environment.

Evaluation

All of the original monograph catalogers on the English, French/Italian, and German/Scandinavian teams have been using Reflection since January 1997. For those team members, original cataloging productivity has increased 10.7 percent, comparing the period January to May, 1996 to the same period in 1997. Moreover, the increase occurred in the first few months after the introduction of the new interface, and this increase is on top of the highest levels of productivity to date. No measurements are available to assess improvements in accuracy but, given the automation of data entry from authoritative sources and the simplification of the copying procedure, many of the sources of possible errors in data entry have been eliminated. Individuals claim a variety of benefits, particularly that less time is spent on the mechanics of data entry and that more time is devoted to the intellectual aspects of cataloging. More verification and authority work takes place because it is easier to do than before or it takes place automatically, with the cataloger alerted only in the event of a discrepancy. As Kathleen Hunter Rutter, team leader of the French/Italian Team noted, "the quality of original cataloging has increased even more than the quantity."