THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 59, No. 9, November 1980
Printed in U.S.A.

Designer's Workbench:

Philosophy

By L. A. O'NEILL

(Manuscript received May 1, 1980)

Designer's Workbench (DWB) increases the productivity of the design-aids user by buffering the user from the idiosyncrasies of the application programs and the computers on which they run. These aids are used during the electrical design, physical design, and test development of both custom electronics and printed wiring boards. The design-aids programs were left unchanged on their original computers and are accessed via an interlocation computing network using the capabilities of the PWB/UNIX operating system. All job control, file management, and data translation required for the programs are provided automatically. In addition to describing the design constraints applied during the development of DWB, this paper also introduces the three other papers in this issue that describe in detail the user, programmer, and production environments created by DWB.*

I. INTRODUCTION

The most significant challenge facing the computer industry over the next decade is to increase the productivity of those who use the computers. Until now, the end users have relied primarily upon programmers to develop their applications. The continual reduction in hardware costs is producing such a proliferation of computers that it will not be possible to hire and train enough programmers to develop all the software needed. One approach to resolve this difficulty is to provide an environment in which the programmer can be more productive. That is what has been done in the Programmer's Workbench

* UNIX is a trademark of Bell Laboratories.

version of the UNIX time-sharing operating system.[1] Another approach is to develop a higher level language that makes it easier for the end user to instruct the computer to perform the operations required. This is the approach that most data-base management systems have taken by providing a simple query language for the user to specify what information is desired.[2] Designer's Workbench (DWB) was designed so that the end users could perform most operations for themselves while providing an improved development environment for the application programmer.

DWB provides a high-level language for the user of circuit design-aids programs, thereby removing most tedious error-prone tasks that, in the past, have reduced the circuit designer's productivity. These design aids are a set of programs used in the design, fabrication, and testing of both printed wiring boards and integrated circuits. The various programs perform functions such as simulation, design verification, prototype construction, and test generation. DWB is the product of two separate organizations that, while having similar support responsibilities, used different programs and processes specifically tailored to their product. While some of the programs supported were locally developed by these organizations, others were provided and supported by other organizations. In addition, some of these programs ran on several different computers at other locations. Because of the number of programs involved and limited programmer resources, it was not feasible to consider the usual approach of rewriting the individual programs on a single computer to simplify and unify their use.

Instead, another approach was taken. The application programs were left unchanged on their present computers, and DWB acts as an intelligent terminal to prepare and submit jobs to the appropriate computer via an interlocation computer network. Such an approach would not have been economically feasible without the excellent communication capabilities and variety of the programming utilities provided on the PWB version of the UNIX operating system. By using PWB/UNIX software, the DWB developers were freed from many routine, yet complicated, programming tasks, and the constraint of using specific computers was removed. The developers of DWB could concentrate on designing an interactive front-end to these remote application programs. DWB has gained immediate production acceptance because it automates the routine tasks that people dislike doing and computers do well. Thus, people enjoy the assistance DWB provides because it frees them for more creative tasks.

DWB serves as a high-level language that is oriented toward the user rather than the programmer or computer. By inserting an interactive computer between the designer and the application programs, DWB

only requires users to learn one language related to their background; the computer does the remainder of the job. The remote application programs are not moved or changed. The minicomputer uses a dial-up capability to reach the other computers and automatically reformats the data to the specific application program language. In this environment, training is reduced by economically providing on-line tutorial text. In addition, DWB helps the users manage their work by keeping track of files and the status of the jobs submitted.

This paper describes the scope of the design aids application and the specific constraints applied that resulted in the current implementation of Designer's Workbench. The contents of the other three DWB related papers in this issue are then discussed. Finally, the initial user reaction as well as future implications of the Designer's Workbench approach are mentioned.

II. DESIGN AIDS APPLICATION

Designers Workbench (DWB) was conceived to integrate two sets of design aids that had evolved over a period of five years. These sets were in daily production use in the transmission area of Bell Laboratories at the North Andover, Massachusetts and Holmdel, New Jersey locations.[3] They had some common components, they used many of the same machine-aided graphical layout and documentation systems, and they used similar aids to generate manufacturing tests for digital circuits. Their differences were also significant. One was used to develop manufacturing information for circuit packs, both wired prototypes and printed circuit boards (PCB), while the other was used for the design and development of custom electronics, from the silicon chip through the PCB interconnection levels. The capabilities that had to be integrated consisted of:

(*i*) Logic simulation.
(*ii*) Automatic wiring of prototypes.
(*iii*) Acceptance and diagnostic tests.
(*iv*) Testability analysis.
(*v*) Machine-aided layout.
(*vi*) Documentation.

The goal of the initial implementation of DWB was to simplify the generation of the digital test information. Thus, only a subset of the above capabilities was programmed. Five major capabilities were incorporated. The first, HIWIRE[4] is used for automatic prototype construction and to provide the physical design data required for guided probe diagnosis of bad circuits. Two logic simulators, LAMP[5] and LASAR[6]

are used for test generation and evaluation of different classes of circuits. The fourth, TMEAS[7] is used to evaluate the testability of a digital circuit and to enable the designer to determine the effect of various techniques to enhance its testability. The fifth capability is an interface that reformats the data from the previous programs into machine-readable form that will execute on specific test machines.

III. IMPLEMENTATION CONSTRAINTS

The complexity of the integration effort becomes clear when we realize that the test generation process in New Jersey consisted of 17 programs running on 7 different computers. The process in Massachusetts consisted of 24 programs running on 11 computers. To accomplish this task economically, the following constraints were applied.

(*i*) Leave application programs on present computers.
(*ii*) Do not change the input languages of these programs.
(*iii*) Keep the amount of data to be transmitted small.
(*iv*) Make the new system easy to learn.

The first constraint would allow all existing programs to be used as well as make it possible to access any future programs that would be developed on other machines. The availability of the Bell Laboratories interlocation network meant that the DWB developers did not have to be concerned with the physical and software problems of establishing such a facility. PWB/UNIX software provided the commands to transfer data between specified computers. Therefore, DWB had only to provide the next protocol levels to select the appropriate computer for execution, send the necessary data, and then return the results to the user.

Only by retaining the input languages was it possible to avoid major reprogramming effort, especially on programs supported by other organizations. To satisfy this constraint, it was necessary to provide a translation mechanism so that the data could be reformatted to the appropriate application language. The development of these translators was simplified by the availability of the PWB/UNIX compiler writing utilities, LEX and YACC.[8] To avoid a proliferation of translators, a simple internal data base structure was selected and then each specific language was converted into and out of this data base. In addition to the data translators on DWB used for the circuit descriptions, translators had to be provided for the input signals applied to the circuits because the different programs used different formats. To provide a common command language for various application programs, DWB also must translate commands into the specific form required by the individual programs. This has been done to a limited extent for some of our current applications. Significant effort will have

to be applied to command translation in the future if DWB is to reach its full potential for buffering the user from program differences.

The third constraint, on the amount of data transmitted, led to a distributed data base. The internal data base of DWB is never sent to another machine. Only enough data are transmitted in source form so that the appropriate data structure can be populated on the remote computer. Thus each application program can retain its own data base. Whenever data from remote computers are needed for other DWB applications, it is reacquired by DWB. Checks are made to determine if editing has occurred on the remote computer, and the user is informed of differences which may have to be reconciled. Some data, such as graphical layouts, do not need to be returned to DWB. These data are appropriately marked to indicate what version of the DWB data base was used in their production, and may then be distributed to the end user. The existence of separate data bases required that DWB provide checks to insure that only consistent information is retained.

The fourth constraint was satisfied by providing an inexpensive, interactive environment where the user could be led through the use of the system. The UNIX operating system on a minicomputer provided just such an environment. Not only does DWB prompt the user to supply all required information, but extensive on-line tutorials are also provided to answer any questions that may arise.

One other major constraint was applied in the development of DWB. It was obvious that DWB would have to be an evolving system to track changes in product line and technology. Thus, it was designed from the start to be easily modified and maintained. Throughout the design and implementation phases, modularity and testability were emphasized. Whenever possible, the program code and the data were kept separate so that changes could be introduced without recompiling. Finally, a production control system was instituted to insure that the user would not be inconvenienced by future development work.

The application of these constraints and the use of the PWB version of the UNIX operating system led to the economical implementation desired. The specification phase required 13 programmer months and an elapsed time of 6.5 months. The implementation of the initial thread required 25 programmer months and an elapsed time of 3.5 months. The provision of the entire production set of applications required 29 programmer months and an elapsed time of 5.5 months. Finally, the provision of the production control system required 21 programmer months and an elapsed time of 3.5 months. These times and efforts are remarkably small when compared to the effort required to implement other design-aids systems on a new computer. Many application programs are dependent upon the operating system under

which they were written, and the original development of these programs required many programmer years of effort. Furthermore, the flexibility of the approach has made it easy to add other applications and programs to the Designer's Workbench.

The testing of a design-aids system with its evolving application environment differs from the testing of a typical software product. The formal integration testing by the developers consumed only 8 programmer months for the first thread (successful run of an application) and 10 programmer months for the full capability. However, the end users are continually testing and evaluating the system and recommending changes to the specifications. Relatively little code was changed to correct bugs, but much has been modified to satisfy evolving user requirements. Maintenance, including the modifications required to accommodate changes in operating systems, has required less than one programmer half-time since the system went into production.

IV. THE USER ENVIRONMENT

The view of Designer's Workbench seen by the end user is described in "Designer's Workbench—The User Environment."[9] This paper presents the requirements that were placed upon the implementation to gain the acceptance of the user community. Particular emphasis was placed upon the human-machine interface to insure that the new user would not be frustrated by the many details needed to run different programs on different machines. The methods by which the user is shielded from the file-naming conventions, data translations, job control, and error handling are illustrated. The processes by which the user creates and maintains data and submits jobs are described. These jobs may be submitted either interactively or to a batch queue. For batch jobs, DWB keeps track of the status so that the user can easily determine if the remote program has run successfully or not. For interactive jobs, DWB performs the login process for the target machine and provides an encryption mechanism to keep the required passwords secure. Then either a clear channel is provided for normal interactive operation on the remote computer or DWB can translate the commands into the form accepted by the target machine and interpret the replies for the user.

V. THE PROGRAMMER ENVIRONMENT

The view of Designer's Workbench seen by a programmer who is to add a new capability to DWB is presented in "Designer's Workbench—The Programmer Environment."[10] This paper emphasizes that DWB was designed to provide consistency, simplicity, modularity, reliability, and ease of maintenance. The reasons that these attributes were stressed are presented, as well as the software engineering techniques

utilized. We describe the file structure used to simplify the access of data. This is needed both to describe the circuit and to submit the job with complete accounting information. Then the mechanisms used to customize each job for submission and to translate the data to the form required by the application programs are presented. The value of PWB/UNIX utility programs is discussed, as well as the effect that the UNIX languages had on the implementation. Finally, the methods used to retain status information and maintain records are presented. The overall purpose of this paper is to illustrate the simplicity with which DWB could be implemented and new applications could be added in the PWB/UNIX environment.

VI. PRODUCTION ENVIRONMENT

The methods used to keep a system such as DWB current in a production situation are described in "Designer's Workbench—Providing a Production Environment."[11] If a system such as DWB is to be successful, careful provision must be made to insure that it can be modified, either to correct bugs or add new capabilities, without interfering with the end user. It is imperative that the software be reliable to maintain the user's confidence. To accomplish this, a simple yet comprehensive production software control facility has been provided for DWB. The source code is controlled, multiple versions are concurrently available, and an extensive testing procedure is followed. The Source Code Control System (SCCS) provided by the PWB/UNIX system is used to keep track of all sources, both program and data, needed to reconstruct any previous version of DWB. In addition, several versions of DWB are concurrently available so that debugging in a complete system environment, testing, and production can occur simultaneously, without interference. Three versions are available to the user: the current one, a test one containing new features, and the last production version. The last version is used if a new release appears to cause trouble with a capability that worked previously, then the user can revert to a good version until the problems are corrected. To simplify the control of this production operation, a set of utility librarian programs are provided. The utilities automate the process to insure that all functions are performed correctly and consistently. Other utilities are provided to automate the record-keeping and accounting functions; these provide usage data that aid the developers in isolating potential problem areas and guide future development efforts.

VII. USER REACTION

The initial user response to the capabilities of DWB has been quite favorable. In general, the basic philosophy has been accepted, and the

users have made many suggestions on additional capabilities that they would like to have provided. The training of new users is indeed simpler, and the number of requests to solve JCL-related problems has greatly diminished. In contrast, the demands placed upon the translators have greatly increased. When these programs were used only by experienced, trained users, the users would compensate for deficiencies. Novice users, however, placed additional strains on the translators. Therefore, additional effort has been placed in this area to make the translators more robust. The users have requested that the remainder of the design aids process be implemented so that the circuit description developed for the testing function can be used in the physical design process. Work is in progress to add links to the other interactive layout and design aids systems used throughout Bell Laboratories. To accomplish this, a second input language is being added to DWB. This is the language used to populate the data base of those other systems. With the implementation of these capabilities, DWB will be able to satisfy its original goal of integrating the complete design process.

The most significant request for a new capability was to augment the current batch capability with an interactive method for executing programs on remote computers. The user community wanted the same type of tutorial guidance and assistance that they presently have on DWB made available for remote interactive programs. Interactive operation is a much more complicated task than batch submission, because DWB would have to analyze each command submitted by the user and each response received from the remote computer. With DWB serving as a buffer, the user would see similar commands for all applications while each computer would see its specific command language. An initial version of this capability is available, but more work is needed to generalize the handling of messages, such as error return codes, received from the remote systems.

Another class of interactive applications that must be integrated into DWB are those programs that rely heavily on graphical editing. As a minimum, DWB provides for simplified file transfer to and from the graphics system so that the user can concentrate on learning and using the graphical capability itself. Within Bell Laboratories, many new graphical systems are being rewritten to take advantage of the UNIX operating system. When these graphical programs exist on the same computer as DWB, it will be possible to switch between DWB and the graphical system as necessary in the design process.

VIII. CONCLUSIONS

It has been found that the Designer's Workbench approach greatly increases user acceptance of design-aids programs. The vast majority of the previous complaints about design aids have been removed by

shielding the user from the idiosyncrasies of the computers, both the job control language and the input languages of the programs. In addition, most of the tedious, error-prone steps of data encoding, file management, consistency verification, and job customization have been removed or greatly simplified for the novice user without restricting the flexibility provided for the experienced user. Furthermore, DWB has created an environment in which new applications can be readily written and integrated with the existing aids. Although DWB was designed to handle a specific set of application programs, the capability is perfectly general and can easily be applied to other sets. Adding new applications is straightforward because of the software engineering techniques used. Thus a general-purpose capability has been developed that can be used to simplify the task of the end users and thus increase their productivity.

IX. ACKNOWLEDGMENTS

The authors of this set of papers would like to acknowledge the significant contributions of C. G. Savolaine in defining, specifying, and implementing the initial version of DWB. In addition, they would like to thank A. Devito, D. S. Evans, J. M. Franke, T. V. Gaudet, J. E. Gorman, J. W. Kasinskas, B. T. McNamara, R. P. Snicer, C. A. Verbinski, and E. D. Walsh for their contributions to DWB, and a significant number of users who have driven this work. We would especially like to thank R. B. Hawkins, E. B. Kozemchak, and J. Logan for their advice and support in realizing DWB.

REFERENCES

1. T. A. Dolotta and J. R. Mashey, "An Introduction to the Programmer's Workbench," Proc. 2nd Int. Conf. on Software Engineering (October 13–15, 1976), pp. 164–168.
2. B. Schneiderman, *Databases: Improving Usability and Responsiveness*, New York: Academic Press, 1978.
3. L. A. O'Neill et al., "Designers Workbench—Efficient and Economical Design Aids," Proc. 16th Design Automation Conference (June 25–27, 1979), pp. 185–199.
4. D. S. Evans and L. A. O'Neill, "An Integrated System for the Design of Printed Wiring Boards," Electro '76 Professional Program, Session 26 (1976), Boston, Mass.
5. S. G. Chappell et al., "LAMP: Logic Analyzer for Maintenance Planning, Logic-Circuit Simulators," B.S.T.J., *53*, No. 8 (October 1974), pp. 1451–1476.
6. *The Users Manual for the Teradyne P400 Automatic Test Plan Generation System*, Boston: Teradyne Inc., 183 Essex St., Mass.
7. J. Grason, "TMEAS, a Testability Measurement Program," Proc. 16th Design Automation Conference (June 25–27, 1979), pp. 156–161.
8. S. C. Johnson and M. E. Lesk, "Language Development Tools," B.S.T.J., *57*, No. 6, Part 2 (July–August 1978), pp. 2155–2175.
9. J. R. Breiland and R. A. Friedenson, "Designer's Workbench: The User Environment," B.S.T.J., this issue, pp. 1765–1790.
10. P. H. McDonald and T. J. Thompson, "Designer's Workbench: The Programmer Environment," B.S.T.J., this issue, pp. 1791–1807.
11. T. J. Thompson, "Designer's Workbench: Providing a Production Environment," B.S.T.J., this issue, pp. 1809–1823.