Development Techniques for Generic Software

Richard L. Hamilton

Bell Laboratories
Holmdel, New Jersey 07733

1. INTRODUCTION

In developing the first version of a generic implementation of X.25, Levels 2 and 3, we examined three development techniques: table-driven finite state machine implementation, an integrated testing environment, and top-down design. While not designed as an experiment, we monitored the project closely and compared the product with other implementations of X.25 at Bell Laboratories to evaluate potential benefits and penalties.

2. TECHNIQUES

2.1 Finite State Machine

A finite state machine (FSM) is a powerful tool for both specifying and implementing protocols. This technique was used in the X.25 specification and has been discussed in the literature[1,2,3,4]. A table-driven implementation of the FSM was chosen to facilitate changes and simplify coding. We were interested in what effect this technique would have on program size, speed of execution, coding time, and debugging time.

2.2 Testing Environment

Contrary to common practice, we made a testing environment before coding. The complexities of a communications protocol, especially X.25, require careful attention to the problems of verifying that an implementation of that protocol does in fact perform correctly. In addition, we felt that the process of verification should start as early as possible in the development process. The testing environment, which runs under the UNIX* operating system, let us test the FSM and its tables very early in the coding process. We were able to integrate new modules easily and test them thoroughly using this tool.

2.3 Top Down Design

In designing and implementing a solution, we followed a top-down approach. This made it possible to have a "running" version at all times, with unwritten modules replaced by dummy routines. This was not rigorously followed in coding because it was often more sensible to code all of the routines that performed one function even if that meant coding some low-level functions early. Doing this still let us always have a running version, but simplified testing.

3. MEASUREMENTS

Our main method for evaluating these techniques was comparison with existing implementations of X.25 at Bell Laboratories. We measured the size and execution speed of both our implementation and the existing ones and ran some simple complexity metrics.

---

* UNIX is a Trademark of Bell Laboratories

We used the testing environment to help modify and transport existing implementations of both Level 2 and Level 3 to a new environment, which gave us the opportunity to compare our versions with the existing ones in terms of the ease of making modifications. We kept a log of program bugs found and the effort it took to fix them, for all of the implementations, and tried to characterize the types of problems found.

4. CONCLUSION

A combination of a table-driven finite state machine realization, a comprehensive testing environment, and a top-down approach was used to produce an implementation of X.25, Levels 2 and 3. In comparison with other, ad hoc, X.25 implementations, we found that our solution ran as much as 20% faster, but was about 35 to 40 percent bigger. We were able to explain all but 11% of that difference in terms of added function or added flexibility. A McCabe complexity metric showed little difference between the implementations.

Comparison of time spent debugging showed that our approach was superior to the ad hoc methods, both in terms of number of errors detected and time taken to correct those errors. Even so, the testing environment was shown to be a significant aid in debugging the other implementations when compared to other testing techniques. Although not intended as a controlled experiment, the data collected during development support using these techniques in similar circumstances.

REFERENCES

[1] Bochmann, Gregor V., "A General Transition Model for Protocols and Communication Services," IEEE Transactions on Communications, vol. COM-28, no. 4, April 1980.

[2] Bochmann, Gregor V. and Tankoano Joachim, "Development and Structure of an X.25 Implementation," IEEE Transactions on Software Engineering, vol. SE-5, no. 5, September 1979.

[3] Bochmann, Gregor V. and Carl A. Sunshine, "Formal Methods in Communication Protocol Design," IEEE Transactions on Communications, vol. COM-28, no. 4, April 1980.

[4] Danthine, Andre A. S., "Protocol Representation with Finite-State Models," IEEE Transactions on Communications, vol. COM-28, no. 4, April 1980.

THE VIEWGRAPH MATERIALS
for the
R. HAMILTON PRESENTATION FOLLOW

DEVELOPMENT TECHNIQUES

FOR GENERIC SOFTWARE

X.25 DEVELOPMENT

OBJECTIVES	TOOLS
Portable	C language, minimal set of primitive functions
Maintainable	Testing/ development environment
Flexible	Table-driven finite state
Modifiable	Layered approach

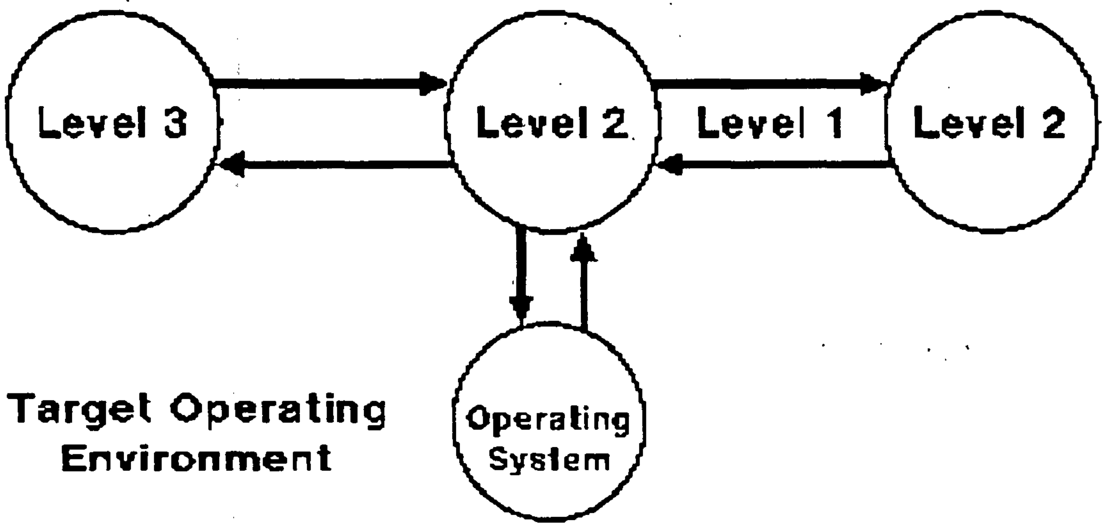
R. Hamilton
Bell Labs

DEVELOPMENT ENVIRONMENT

UNIX™ Operating System

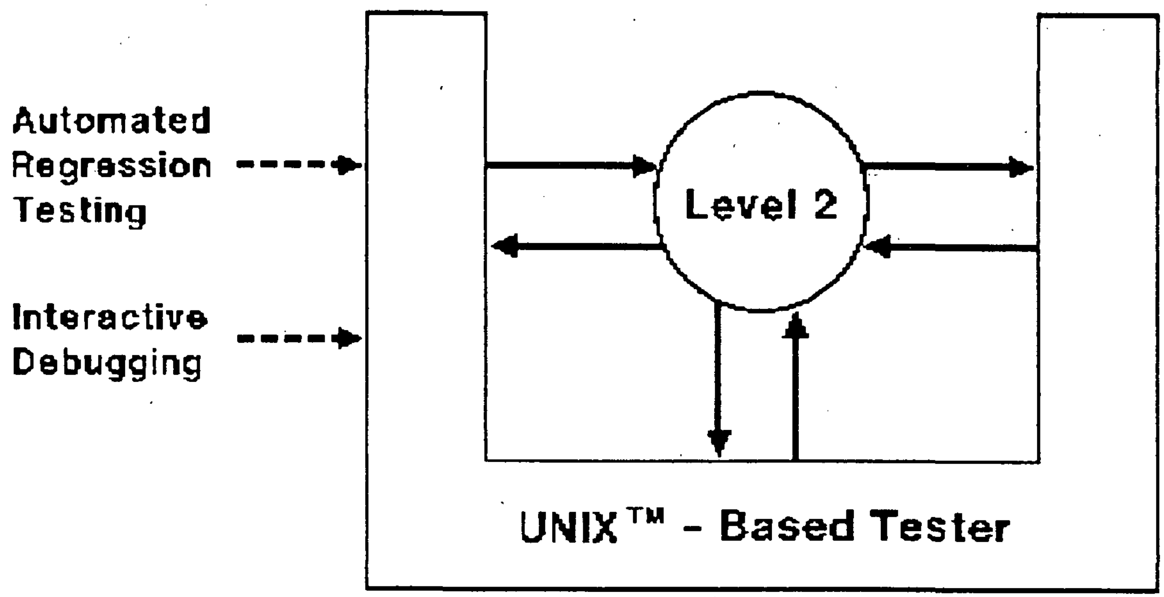
- Make
- AWK
- SCCS
- Shell

LEVEL 2 -- NORMAL ENVIRONMENT



Target Operating Environment

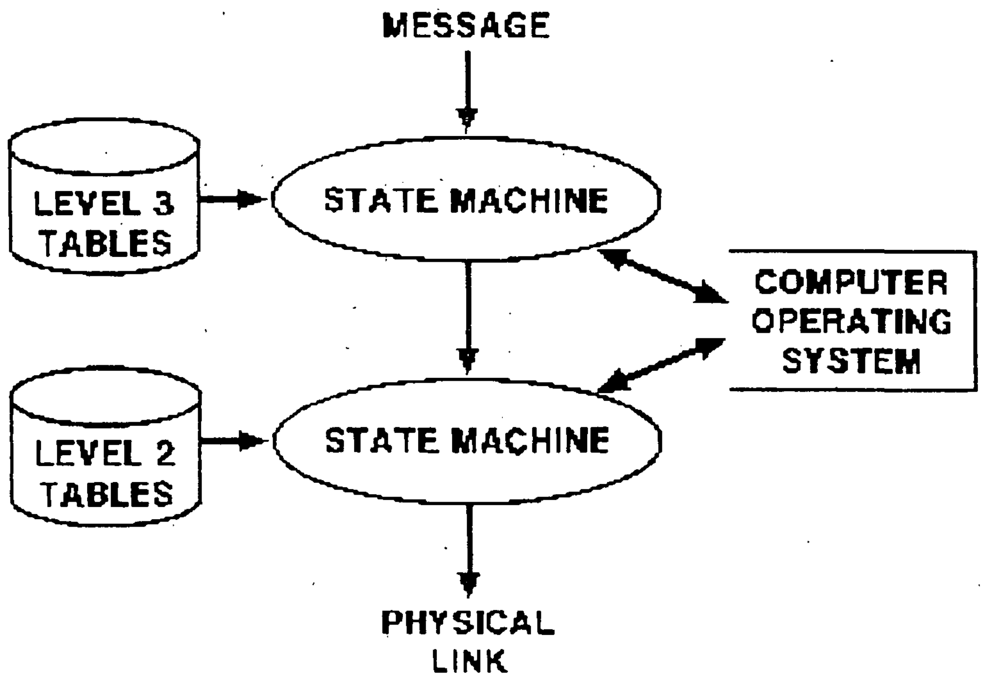
LEVEL 2 -- TESTING ENVIRONMENT



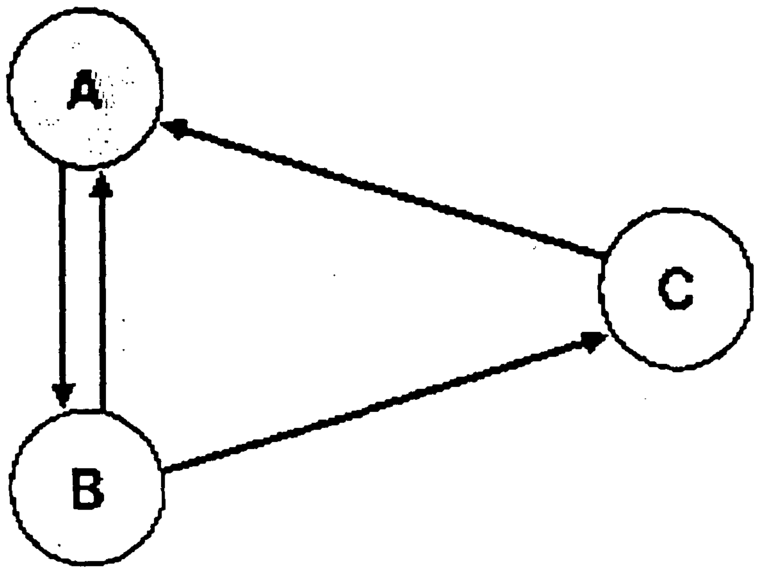
FINITE STATE MACHINE

- **Table-driven**
- **Hierarchical**
- **Parallel**

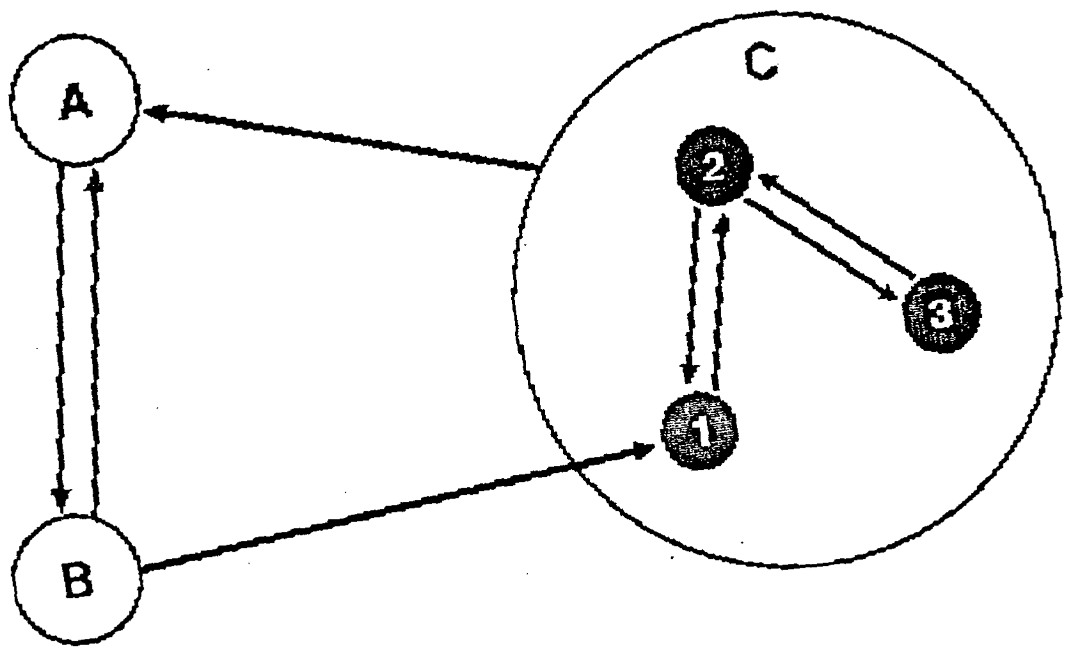
X.25 IMPLEMENTATION



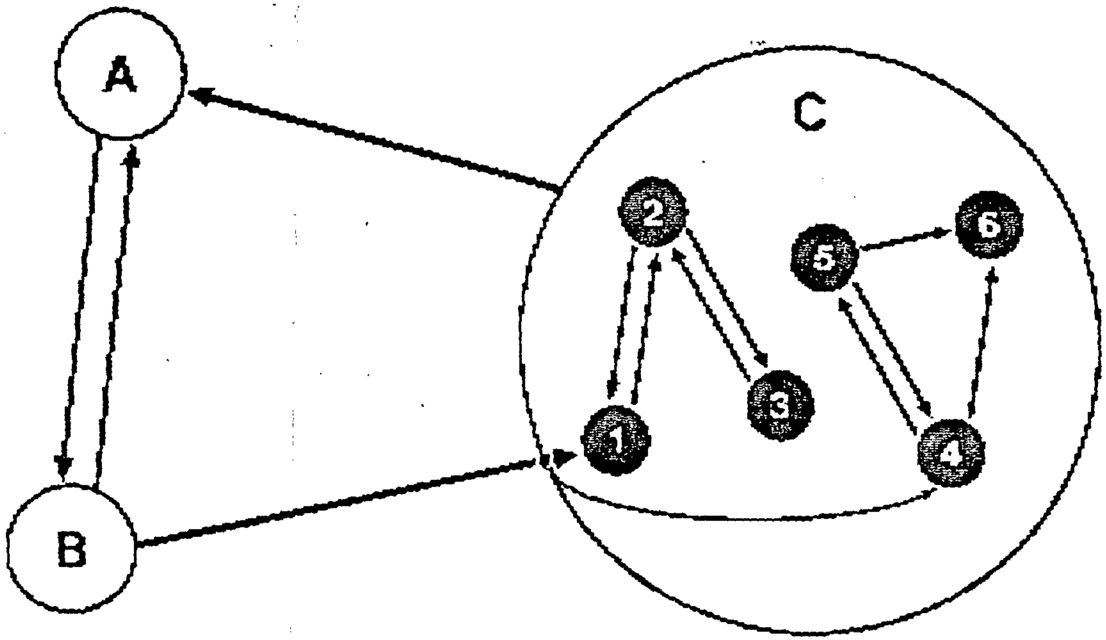
FINITE STATE MACHINE



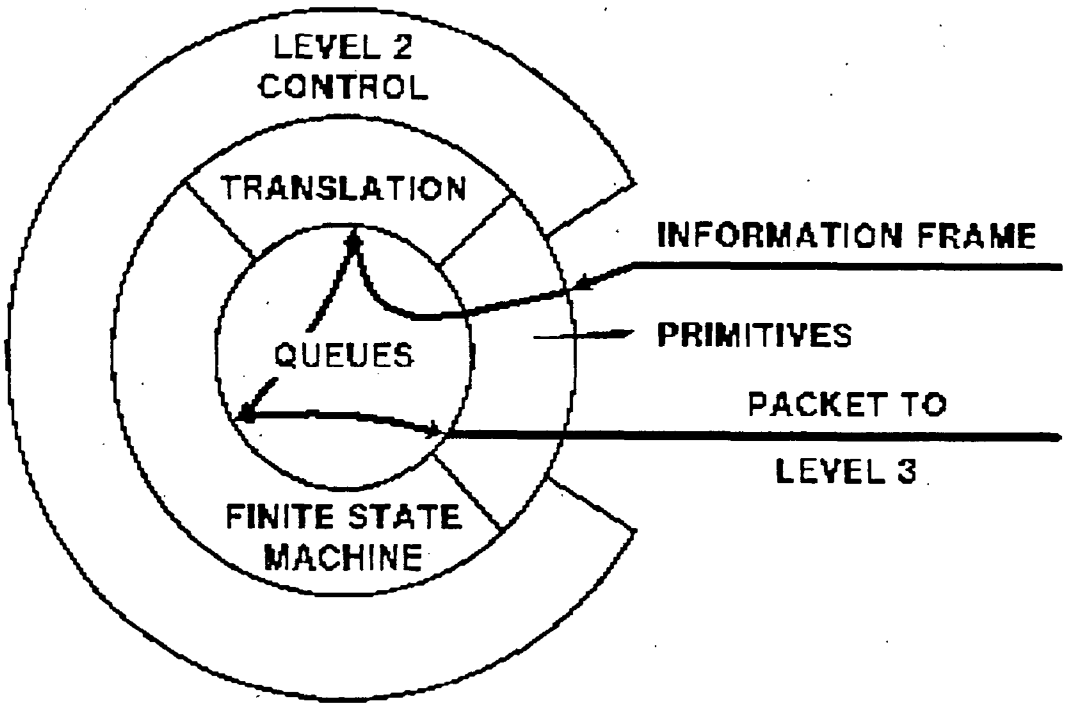
FINITE STATE MACHINE
HIERARCHICAL STATES



FINITE STATE MACHINE
HIERARCHICAL PARALLEL STATES



LAYERED STRUCTURE



LINES OF CODE

LEVEL 2	LINES OF CODE	% DIFFERENCE
• Existing	1039	
• Generic	1846	+78%

LEVEL 2	LINES OF CODE	% DIFFERENCE
• Existing	1590	
• Generic	2252	+42%

SIZE MEASUREMENTS
IN BYTES

LEVEL 2	TEXT	DATA	=	TOTAL	% DIFFERENCE
• Existing	5688	56	=	5744	
• Generic	6766	1236	=	8002	+39%

LEVEL 3	TEXT	DATA	=	TOTAL	
• Existing	6818	268	=	7086	
• Generic	8558	926	=	9484	+34%

Note: All programs compiled under the 8086 cross-compiler with the optimize option, without primitives, and without any debugging aids included

LEVEL 2 SIZE DIFFERENCES

Added function	
• Channel No.	200
• Timer routines	272
• Disconnect	186
Added flexibility	
• Action overhead	248
• Channel select	52
• Multi-table FSM	200
• Table clarity	192
• Optional prims	100
TOTAL	1450
Actual difference	2258
Bytes unaccounted for	808

MEASUREMENTS

Size - 35 to 40% larger

Speed - 0 to 20% faster

Complexity - Equivalent