BBC: A Blackboard Generating System (Extended Abstract)

Joshua Levy
levy@linc.cis.upenn.edu
Dr. Tim Finin
tim@linc.cis.upenn.edu

University of Pennsylvania
Computer and Information Science
March 1987

1. Introduction

This paper describes a black board system generator, Black Board Builder in C (BBC). This generator is unique because it is a compiler producing the black board as a separate program, and because this black board is in C. A discussion of the advantages of this general architecture is followed by a discussion of the features of a specific implementation.

BBC's architecture is similar to a compiler's.† It accepts an input program which has been created previously, and generates a separate output program; it is not interactive. The input language used is similar to C, and the output language generated is C.

Lately there has been an evolution of AI techniques and systems from LISP and PROLOG based implementations running in special environments (e.g. LISP machines) toward implementation in conventional languages (C or Pascal) running in general environments (UNIX or VMS). Examples of this movement include Teknowledge's expert system shells S.1 and M.1, and the availability of AI tools as additions to conventional languages such as Objective-C (similar to SmallTalk) and C++ (an object oriented

†It is actually patterned after a compiler generator (yacc) available on UNIX systems.

language).‡ The technical advantages of an implementation in a conventional environment are discussed below, but there is another advantage: UNIX environments are far more common than LISP machine environments, and the same is true of UNIX programmers compared to LISP machine programmers. The ultimate goal of the BBC project is to bring black board ideas into the C and UNIX environment in a way which will feel comfortable to conventional programmers.

2. Architectural Advantages

The general architecture described above leads to several advantages over conventional (interpreted, LISP based) black board systems. Some advantages are the result of inherit abilities of C (portability, number of users), others result from having a generated program separate from the generator, and finally, some are produced by generating a program in the local ''native language'' (C in the case of UNIX). Five advantages of BBC generated black boards will be discussed:

- Integration
- Speed and Size
- Portability
- Preprocessing
- Utilities

Perhaps the biggest advantage of creating C programs as output is the ease of integrating other programs written in C. Since C is becoming the high level language for low level work, it is increasingly true that routines to control robots, speech devices, etc. are written in C. Since many black board systems interface with these devices, being written in C is a big advantage. Furthermore, many programs with which a black board might want to communicate, data base systems, rule based expert systems, and so forth,

‡TRC, a program to translate rules into C (or Pascal) code is a microcosm of this. It was developed because close interaction was required with a Pascal program which simulated the problem an expert system was needed to solve, and no expert system shell had a good enough Pascal interface.

already have C interfaces. Lastly, there are large C subroutine packages available which are useful to almost any programmer, including black board programmers.

Two other advantages of BBC are speed and size. An advantage is gained by generating C code instead of LISP, since C compilers on most machines create faster code than LISP compilers. Speed improvements, moreover, should not be considered a footnote in a sales brochure. Many black board applications (and AI applications in general) must operate in real time. Speech analysis, robot control and submarine tracking are all black board applications in which real time response is necessary. LISP systems also tend to be big, especially the portable dialects like Common LISP which might require 2-4 Megabytes of memory to do real work. C systems tend to be much smaller.

A third advantage is portability. Since BBC creates high level C code, its output is quite portable. Blackboards could be developed on a SUN, for example, and used on a VAX, an IBM PC, or a specialized robot controller. The compactness and speed gained from using C can make otherwise marginal host computers acceptable.

Because there is an input language, as opposed to interactive input, this language can be changed by simple preprocessors or generated by other programs. For example, BBC's input language could easily be changed from C-like to LISP-like using a preprocessor. Also, the input to BBC can easily be generated from other programs, a graphical black board design tool, or a black board building expert system.

UNIX environments are now approaching LISP machine environments in terms of utilities available to help programmers. UNIX includes debuggers, cross referencers, profilers, and flow tracers, all of can be used on the black boards generated by BBC. The implementation of BBC uses naming conventions designed to make these utilities easy to use, especially debuggers.

3. Implementation

These ideas were tested by building a prototype black board generator with the described architecture. This generator, also called BBC, runs on a VAX 785 under Ultrix 1.2 (a derivative of 4 BSD). It is designed to be portable to any UNIX system, as is the code it produces.† This implementation is being used to build a black board to monitor UNIX systems, a prototype version of which has been completed.

BBC's input language is fairly simple. Types of objects placed on the black board are specified by C structures. A routine to control the black board is named. (It can either be written by the user, or one of the defaults can be specified.) Lastly, knowledge sources are described in ways similar to the description of C structures. Each knowledge source has four possible slots: one to specify a C routine to be called to determine if the knowledge source should fire; one to specify a routine to call if the knowledge source is fired; one to specify a routine to be called to initialize the knowledge source before it is ever fired; and one to specify a routine to cleanup after the last firing of the knowledge source.

BBC automatically produces three types of routines for the programmer to use in developing black board systems. These are designed to be the interface between the black board, and the knowledge sources which use it. The most basic routines are the add, delete, change, and read routines, which provide black board data access. Next are the searching routines. These return objects from the black board if they meet certain criteria. Several of these routines are created, some are very general, and can be used to specify arbitrary searches, while others are specific, and designed to implement common searches. Finally, routines are created to integrate the black board's knowledge sources

† Two exceptions: the debugger assumes that sizeof(int) == sizeof(int*) and the search by regular expression routines use code specific to BSD systems, and will not run on Bell UNIXs.

and its control routine.

BBC provides other services as well. A history list is kept on the black board. This list can be used for debugging, to see what has been done recently; to help the control routine determine which knowledge sources are overdue for use; or to allow the knowledge sources to reason about themselves. There is also a built in debugger which allows black board developers to print the contents of the black board, check the history list, "single step" through the black board's operation, etc. Finally, sample control routines are distributed with BBC. These can either be used 'as is' or be the starting point for developing customized controllers.

4. Conclusions and Future Work

The architecture of a black board building tool shapes the black boards it creates and their possible uses. This work has shown the viability of a compiler and C based architecture for black board generators. The black boards created by a prototype system have most of the advantages of being C programs, but do not require the amount of work required to write a black board from scratch in C. These black board systems are especially appropriate where speed, portability, or ease of integration with C tools are required.

Several interesting lines of research are open. One involves RPC, Sun Microsystem's remote procedure call system. RPC provides a general framework for C programs on one machine to call C routines on other machines. It could provide an easy way to distribute the black boards created by BBC. Another lies in making the black board a more powerful data structure. It is currently a collection of typed objects, but more powerful representations could be implemented. A frame based black board might be especially useful and interesting. Finally, other black board generators have means of grouping the black board objects and the knowledge sources which use them. This could

5. References

This paper was influenced by two very different schools of thought, previous work on black board generators, and previous work on UNIX tools. The following black board works are listed by influence, most influential first.

[Nii]
H. Penny Nii, William C. White, Nelleke Aiello; *Joy of AGE-ing: An Introduction to the AGE-1 System*, Heuristic Programming Project, Stanford, 1981.

[Hayes-Roth]
Barbara Hayes-Roth, *BB1: An architecture for blackboard systems that control, explain, and learn about their own behavior*, Stanford, HPP-84-16 or STAN-CS-84-1034, 1984.

[Balzer]
Robert Balzer, Lee Erman, Philip London, Chuck Williams; *Hearsay-III: A Domain-Independent Framework for Expert Systems,* **Proceedings of AAAI**, 1980.

The following work refers to UNIX tools. [K&P] is a practical introduction both to creating programs in UNIX, and to the philosophy they embody. [YACC] and [TRC] are specific tools which generate C code as output, and finally, [RPC] is an example of a library of useful routines which could be especially useful to a black board builder.

[TRC]
Daniel D. Kary, *The TRC Reference Manual*, Computer Science Department, North Dakota State University: 1986. This article and source code for TRC was posted to mod.sources, an ARPAnet bulletin board.

[K&P]
Brian W. Kernighan and Rob Pike, **The UNIX Programming Environment**, Prentice-Hall, Englewood Cliffs: 1984.

[RPC]
Remote Procedure Call Programming Guide, Sun Microsystems: 1986. Source code and documentation for RPC was posted to mod.sources.

[YACC]
S. C. Johnson, *YACC: Yet Another Compiler Compiler*, Computing Science Technical Report #32, Bell Laboratories, Murray Hill, NJ, 1978.

be experimented with as a method of reducing programming mistakes.