ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S15CSE02

Automation From Pictures: Producing Real Time Code from a State Transition Diagram*

Andrew J. Kozubal
Mail Stop H820
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Abstract

The state transition diagram (STD) model has been helpful in the design of real time software, especially with the emergence of graphical computer aided software engineering (CASE) tools. Nevertheless, the translation of the STD to real time code has in the past been primarily a manual task. At Los Alamos we have automated this process. The designer constructs the STD using a CASE tool (Cadre Teamwork) using a special notation for events and actions. A translator converts the STD into an intermediate state notation language (SNL), and this SNL is compiled directly into C code (a *state program*). Execution of the state program is driven by external events, allowing multiple state programs to effectively share the resources of the host processor. Since the design and the code are tightly integrated through the CASE tool, the design and code never diverge, and we avoid design obsolescence. Furthermore, the CASE tool automates the production of formal technical documents from the graphic description encapsulated by the CASE tool.

I. INTRODUCTION

Structured analysis and design methods often make use of the state transition diagram (STD) to model real time systems.[1] A CASE tool, such as Cadre Teamwork/RT[2], can partially automate the STD methodology, but the programmer is left with the task of converting the STD into run time code. The programmer takes into account numerous factors, such as task priority, task synchronization, and pending for multiple events, to produce efficient code, and often the resulting code bears little resemblance to the STD. Using a two-step procedure, we have achieved significant automation of this process.

The translation of the STD into code is based on work done previously to develop a language that is based on the STD paradigm. The state notation language (SNL) [3] was developed to simplify programming of time-constrained sequential operations that are driven by events. During extensive experience with the SNL on the Ground Test Accelerator and the Advanced FEL at Los Alamos,[4,5] the SNL evolved into a powerful tool for implementing real time, automatic control. Subsequently, we developed a tool to capture relevant coding information about the STD within the CASE environment and translate it into SNL syntax. Below, we describe the salient features of the SNL, and explain how the translator is used to produce a complete SNL module from the STD.

* Work supported and funded under the Department of Defense, US Army Strategic Defense Command, under the auspices of the Department of Energy.

II. STATE NOTATION LANGUAGE

We designed the SNL to be consistent with the STD methodology and applicable to the existing run time environment that we use at the Los Alamos National Laboratory.[6–8] Following the Mealy convention for STDs, we specify both the events and the actions on the transition between states, and allow only the state name to appear in the state as in Figure 1.

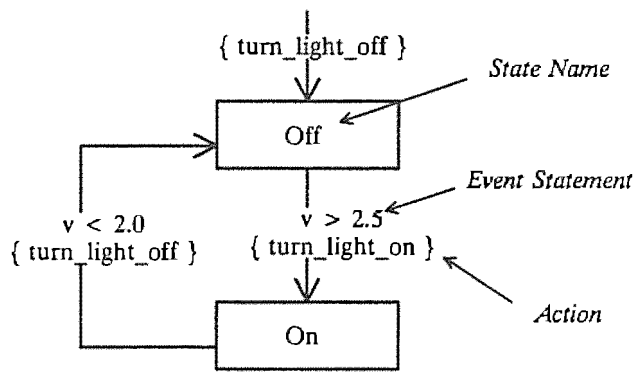

Figure 1. Example of a State Transition Diagram.

In the above example there is only a simple relational expression, which involves one event, the change in the value of variable "v". The SNL is designed to handle more complex event expressions, as well as multiple events. Events may be associated with database channels and time delays. Actions may include calculations, outputs to database channels, and calls to procedures.

Rather than invent yet another new language, we based the SNL on a comprehensive subset of C, along with some relatively minor additions to handle events, actions, and states. We simplified the coding by allowing the programmer to associate run time database channels with a C variable. Figure 2 shows the complete program that implements the STD in Figure 1 in SNL syntax.

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S15CSE02


```
program detect_HV_level;

float           v;
assign    v to "HV_PS_01:output_volts";
monitor   v;

short           light;
assign    light to "HV_PS_01:hv_light";

ss testHV {
     state Init {
          when () {
               light = 0;
               pvPut(light);
          } state Off
     }
     state Off {
          when (v > 2500.) {
               light = 1;
               pvPut(light);
          } state On
     }
     state On {
          when (v < 2000.) {
               light = 0;
               pvPut(light);
          } state Off
     }
}
```

Figure 2. A SNL Control Program.

A complete program contains a *program* statement, a declaration section, and one or more *state sets* (designated by "ss" in the SNL). Within a program, multiple state sets correspond to multiple STDs. Some of the SNL features include:

Statement	Description
program	Provides a name for run time execution.
assign	Assigns or associates a variable with a database channel.
monitor	Causes the channel value to be returned asynchronously whenever it changes by a significant amount.
ss	Specifies the start of a state set.
state	Specifies a state by name.
when	Specifies a transition, with the corresponding events. *When* is followed by the event and action statements and the next state.
pvPut	Function to put a value to a database channel.
pvGet	Function to get a value from a database channel.

The SNL is block structured, as in C. A state may have multiple *when* statements, corresponding to multiple transitions from that state. Other features of the language include: (1) macro definitions within database names, (2) network connection status of database channels, (3) access to channel alarm status, and (4) synchronization through event flags. A state notation compiler generates efficient reentrant C code from the SNL.

On the target processor a sequencer program initiates and controls the execution of a task for each state set. The sequencer establishes connections to database channels and handles asynchronous events, such as might occur on a monitored database channel or loss of a network connection.

III. INTEGRATING THE SNL INTO THE CASE TOOLSET

The user first builds a model within the Teamwork environment. By following existing conventions for real time analysis and design[9], the Teamwork will provide various checks on the design. The specification for a program begins at a process bubble within a data flow diagram (DFD). An example is show in Figure 3.

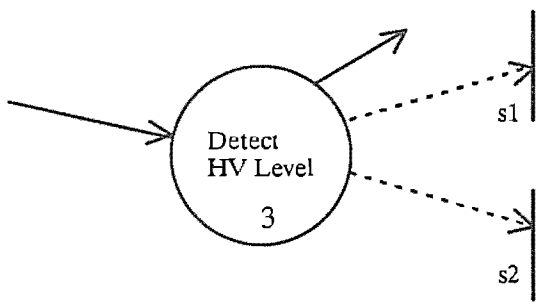

Figure 3. Part of a DFD Showing Control Connections to C-Specs.

Two "control flows" (dashed lines) from bubble 3 in this DFD connect to the control specifications (C-Specs), s1 and s2. Each C-Spec contains a STD, which corresponds to a state set in the program. Declarations, and other header information are placed in the process specification (P-Spec) that is contained within the DFD bubble. The events and actions are placed in the STD on the transitions. Because actions could be very complex — too many characters to fit conveniently on the STD — each action must be specified as the name of a P-Spec.

A translator builds the SNL program from the Teamwork model. This translator accesses the CASE model database using the Cadre Teamwork/Access interface routines[10]. To use the translator the user specifies the model name, the bubble number (default bubble is 0), and the output file for the SNL program. The topography of the DFD and STDs determine the program structure, and the contents of the STDs and P-Specs determine the details.

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S15CSE02

IV. EXPERIENCE AND FUTURE PLANS

We have used the translator on only a few simple test cases. The CASE tool methodology is a little awkward to use, especially when the programmer must go back and forth between the CASE environment and the run time environment during program debugging. On the other hand, programmers have been highly pleased with the SNL. We are investigating the idea of designing a graphic editor that would be more appropriate than the CASE environment.

Although we have made no measurements, we estimate that the use of the SNL rather than C has saved significant programming time, and that performance approaches that of programs written in C.

V. CONCLUSION

The STD paradigm is useful for implementing real time control. Automating the translation from STD to a run time program is expected to introduce fewer coding errors and provide better design documentation. Acceptance of this methodology may depend on providing a more user friendly graphic interface.

VI. REFERENCES

[1] Ward, Paul T. and Stephen J. Mellor, *Structured Development for Real Time Systems*, 3 vols., Prentice-Hall Inc., Yourdon Press, 1985.

[2] Teamwork/RT User's Guide, Release 4.0, Cadre Technologies Inc., Providence, RI.

[3] Kozubal, A. J., L. R. Dalesio, J. O. Hill, and D. M. Kerstiens, "A State Notation Language for Automatic Control," Los Alamos National Laboratory report LA-UR-89-3564, November, 1989.

[4] Atkins, W., "Using the EPICS Sequencer Tool to Automate the GTA Vacuum System, " submitted to Particle Accelerator Conference, San Francisco, California, May 6—9, 1991.

[5] Wilson, William L., Marcus W. May, and Andrew J. Kozubal, "Rapid Development of a Measurement and Control System for the Advanced Free-Electron Laser," submitted to Thirteenth International Free-Electron Laser Conference, Santa Fe, New Mexico, August 26—30, 1991.

[6] Kozubal, A. J., L. R. Dalesio, J. O. Hill, and D. M. Kerstiens, "Run Time Environment and Applications Tools for the Ground Test Accelerator Control System," in *Accelerator and Large Experimental Physics Control Systems*, D. P. Gurd and M. Crowley-Milling, Eds. (ICALEPCS, Vancouver, British Columbia, Canada, 1989), pp. 288—291.

[7] Dalesio, L. R., "The Ground Test Accelerator Database: A Generic Instrumentation Interface," in *Accelerator and Large Experimental Physics Control Systems*, D. P. Gurd and M. Crowley-Milling, Eds. (ICALEPCS, Vancouver, British Columbia, Canada, 1989), pp. 288—291.

[8] Hill, J. O., "Channel Access: A Software Bus for the LAACS," in *Accelerator and Large Experimental Physics Control Systems*, D. P. Gurd and M. Crowley-Milling, Eds. (ICALEPCS, Vancouver, British Columbia, Canada, 1989), pp. 288—291.

[9] Hatley, Derek J. and Imiaz A. Pirbhai,, *Strategies for Real Time System Specification*, Dorset House Publishing Co., 1987.

[10] Teamwork/Access, Teamwork/RT User's Guide, Release 4.0, Cadre Technologies Inc., Providence, RI.