Verifiable Embedded Real-Time Application Framework

Pao-Ann Hsiung, Feng-Shi Su, Chuen-Hau Gao, Shu-Yu Cheng, and Yu-Ming Chang
Department of Computer Science and Information Engineering,
National Chung Cheng University, Chiayi 621, Taiwan, R.O.C. E-mail: hpa@computer.org

Abstract

A new application framework called Verifiable Embedded Real-Time Application Framework (VERTAF) is proposed for embedded real-time application development, with the aim of reducing design errors and increasing design productivity. VERTAF is an integration of three technologies: object-oriented, software component, and formal verification. It consists of five software components: Implanter, Modeler, Scheduler, Verifier, and Generator. Experiences of using VERTAF show a significant increase in design productivity through design reuse, and a significant decrease in design time and effort through design verification.

1. Introduction

Current technology in designing embedded real-time software is still immature. Software engineers tend to use a very rudimentary trial-and-error design technique, developing everything from scratch, applying only unit-testing, and producing sub-optimal software. In this work, an integration of three technologies is proposed: *object-oriented* technology, *software component* technology, and *formal verification* technology. This technology integration results in the *efficient* design of *verifiable* embedded real-time software such that design productivity is increased, design time and error are reduced, and heuristically optimal designs produced.

Besides object-oriented (OO) and software component technology, our proposed application framework called *Verifiable Embedded Real-Time Application Framework* (VERTAF) also incorporates *formal verification* technology into the application design process. We use *model checking*, which is defined as follows. Given a real-time system description S and a temporal property specification ϕ, model checking answers if S satisfies ϕ. A real-time system is modeled by a set of *Timed Automata* (TA) [1]. A temporal property is specified using *Timed Computation Tree Logic* [2].

Using VERTAF, a developer can devote more time and effort to the actual application tasks, instead of real-time system peculiarities. Even program verification can be accomplished automatically by VERTAF since it has integrated formal verification into its design process. VERTAF is modularized into five software components that can be used at different stages of application development, namely *Implanter*, *Modeler*, *Scheduler*, *Verifier*, and *Generator*.

Although object-oriented technology has been applied to the design of real-time systems in several proposed work, there have been very few works on the development of OOAF for real-time application design. Two recently proposed frameworks are *Object-Oriented Real-Time System Framework* (OORTSF) [3] and SESAG [4] [5]. OORTSF and SESAG are simple frameworks that have been applied to avionics software development. Some design patterns were proposed related to real-time application design. Code can be automatically generated. But, there are still some scheduling and real-time synchronization issues left not handled such as asynchronous event handling and protocol hooking.

2. VERTAF Components

Figure 1 illustrates a *component-view* of VERTAF in the industry standard *Unified Modeling Language*.

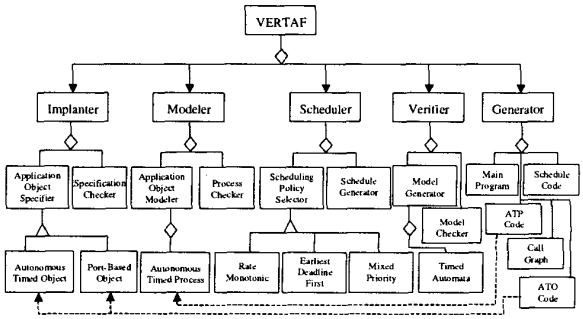

Figure 1. Component View of VERTAF

VERTAF consists of five basic software components: *Implanter*, *Modeler*, *Scheduler*, *Verifier*, and *Generator*. The given sequential order is the sequence in which they are used. In general, a user may use the five components of VERTAF as follows. Given a software application to be designed, an engineer identifies and specifies the objects that are specific to the application using the *Implanter* component, in which a uniform object model called *Autonomous Timed Object* (ATO) is provided. The application objects are then transformed by *Modeler* component into uniform process models, each of which represent a real-time task. *Scheduler* checks the

0-7695-1134-1/01 $10.00 © 2001 IEEE

schedulability of the tasks and schedules them using a scheduling algorithm. *Verifier* proves the feasibility of the scheduled set of tasks by showing if they satisfy all given real-time and embedding constraints. *Generator* generates code based on decisions made by other components.

3. Experimental Results

An industrial application example developed using VERTAF is presented in this section: a *cruiser* example consisting of 12 tasks used to control the vehicle speed under different circumstances. As shown in Figure 2, AICC (*Autonomous Intelligent Cruise Controller*) system application [10] was developed and installed in a Saab automobile by Hansson et al. The AICC system can receive information from road signs and adapt the speed of the vehicle to automatically follow speed limits.

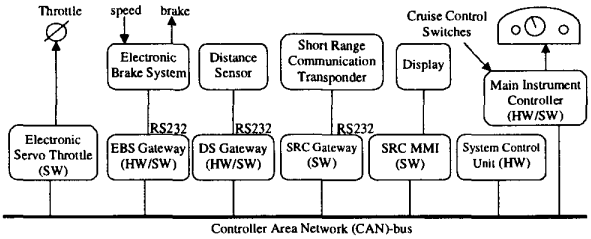

Figure 2. AICC Example: System Architecture

As shown in Figure 3, there are five domain objects specified by the designer of AICC for implementing a *Basement* system. Basement is a vehicle's internal real-time architecture developed in the *Vehicle Internal Architecture* (VIA) project [10], within the *Swedish Road Transport Informatics Programme*. Each object may correspond to one or more tasks. *Process Table* and *Call-Graph* generated by the *Modeler* component are as shown in Table 1 and Figure 3, respectively. There are totally 12 tasks performed in 5 objects. Two different resources were identified in VERTAF, namely, SRC and Display. This application took 5 days for a real-time system designer using VERTAF. The same application took the same designer 20 days to complete development. This significant decrease in design time was because VERTAF automatically extracted tasks and constraints from object specifications.

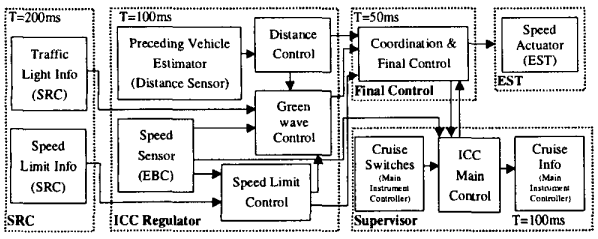

Figure 3. AICC Example: Call-Graph

Table 1. AICC Example: Task Table

Index	Task Description	Object	Period (ms)	Execution Time (ms)	Deadline
1	Traffic Light Info	SRC	200	10	400
2	Speed Limit Info	SRC	200	10	400
3	Vehicle Estimator	ICCReg	100	8	100
4	Speed Sensor	ICCReg	100	5	100
5	Distance Control	ICCReg	100	15	100
6	Green Wave Control	ICCReg	100	15	100
7	Speed Limit Control	ICCReg	100	15	100
8	Coordination	Control	50	20	50
9	Cruise Switches	Supervisor	100	15	100
10	ICC Main Control	Supervisor	100	20	100
11	Cruise Info	Supervisor	100	20	100
12	Speed Actuator	EST	50	5	50

SRC: *Short Range Communication*,
ICCReg: *ICC Regulator*, EST: *Electronic Servo Throttle*

4. Conclusion

An object-oriented application framework, called VERTAF, was proposed for embedded real-time systems application development. It was a result of the integration of three different technologies: *object-oriented* technology, *software component* technology, and *formal verification* technology. The integration resulted in verifiable objects and components, and thus eliminated design errors at an early stage. Different levels of re-use, including object-level and component-level, increased design productivity and decreased overall design effort and time.

References

[1] R. Alur and D. Dill, "Automata for modeling real-time systems," *Theoretical Computer Science*, Vol. 126, No. 2, pp. 183–236, April 1994.

[2] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," in *Proceedings IEEE Logics in Computer Science*, 1992.

[3] W.-B. See and S.-J. Chen, "Object-oriented real-time system framework," in *Domain-Specific Application Frameworks* (M. E. Fayad and R. E. Johnson, eds.), pp. 327–338, John Wiley, 2000.

[4] P.-A. Hsiung, "RTFrame: An object-oriented application framework for real-time applications," in *Proceedings of 27th International Conference on Technology of Object-Oriented Languages and Systems*, pp. 138–147, IEEE CS, Sept 1998.

[5] P.-A. Hsiung, "Object-oriented application framework design for real-time systems," in *Proceedings of the 4th International Symposium on Real-Time and Media Systems* (RAMS'98), pp. 221–227, Taipei, Taiwan, September 1998.

[6] D. B. Stewart, R. A. Volpe, and P. K. Khosla, "Design of dynamically reconfigurable real-time software using port-based objects," *IEEE Transactions on Software Engineering*, Vol. 23, No. 12, December 1997.

[7] K.-H. Kim, "APIs for real-time distributed object programming," *IEEE Computer*, Vol. 33, No. 6, pp. 72–80, June 2000.

[8] P.-A. Hsiung and F. Wang, "User-friendly verification," in *Proceedings of* FORTE/PSTV '99, October 1999.

[9] P.-A. Hsiung, "Embedded software verification in hardware-software codesign," *Journal of Systems Architecture — the Euromicro Journal*, Vol. 46, No. 15, pp. 1435–1450, Elsevier Science, the Netherlands, December 2000.

[10] H. A. Hansson, H. W. Lawson, M. Stromberg, and S. Larsson, "BASEMENT: A distributed real-time architecture for vehicle applications," *Real-Time Systems*, Vol. 11, No. 3, pp. 223–244, 1996.