PT Embedded Systems, **3,** (Dossier 2002), pp. 36-38, ISSN: 1389-1405; Note: the final publication is layouted differently than this submitted text.

Software for Embedded Control Systems *)

Jan F. Broenink, Gerald H. Hilderink and Dusko S. Jovanovic

Electrical Engineering, Control Laboratory, Cornelis J. Drebbel Institute for Mechatronics,
Twente Embedded Systems Initiative.
P.O.Box 217, 7500 AE, Enschede, the Netherlands
Phone: +31 53 489 2793 Fax: +31 53 489 2223
E-mail: J.F.Broenink@el.utwente.nl

Abstract

The research of our team deals with the realization of control schemes on digital computers. As such the emphasis is on embedded control software implementation. Applications are in the field of *mechatronic devices*, using a *mechatronic design approach* (the integrated and optimal design of a mechanical system and its embedded control system). The ultimate goal is to support the application developer (i.e. mechatronic design engineer) such that implementing control software according to ðo it the first time right" becomes business as usual.

1 Introduction

Modern requirements for reliable and efficiently extendable / updateable software for embedded systems, stress the availability of proper design software, assisting the complete design stretch. Especially, when *embedded control systems* are concerned, having the behavior of the complete system available as dynamic model in the design tool is crucial for effective design work.

We consider *Embedded Control Systems* (ECS) as a separate class of embedded systems: the dynamic behavior of the appliance (i.e. the 'machine'-part of the embedded system) is essential for the functionality of the embedded system (Figure 1). ECS are mostly *mission critical systems*, implying when coming across a software error, the complete system get into an undefined state, resulting in a dangerous situation. The other class of embedded systems is *Embedded Data Systems*, where the relevant behavior of the appliance can completely be described by waiting times between subsequent commands from the software. Missing deadlines decrease the quality of service, but are not fatal: it are *soft real-time* systems.

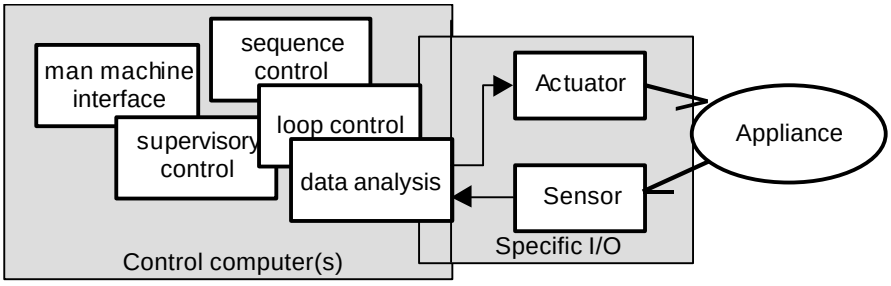

Figure 1: General architecture of an embedded *control* system

At *Embedded Control Systems*, we furthermore separate the I/O interface boards from the computer, because they are often dedicated to the ECS, although not necessary specifically developed. The software part consists of a layered structure of *controllers* and the *user interface* [1]. The *loop controllers* implement the control laws and are *hard real-time*, because missing deadlines mean system failure. *Sequence controllers* implement sequences of activities based on logical actions in time, commanding the loop controllers. Supervisory controllers contain optimization algorithms or expert systems that adapt parameters of the lower controllers. At an ECS, *computational latency* must be small compared to the time constants of the appliance. Examples are robots, production machines like wafer steppers, motor management and traction control of automobiles.

The embedded computer system is considered heterogeneous and distributed, because modern systems are often composed of existing subsystems, having their own control software and processors [2]. Furthermore, systems must be easily *scalable* and *adaptable*, to support ever changing functional specifications and evolution of computer hardware.

PT Embedded Systems, **3,** (Dossier 2002), pp. 36-38, ISSN: 1389-1405; Note: the final publication is layouted differently than this submitted text.

We advocate a *model-based* approach, using *simulation* as a key technique in the verification process (Figure 2). Starting with a model of the dynamic behavior of the appliance, verify it by simulation, the control laws can be designed and verified. These control laws need to implemented on the embedded computer. These are refined stepwise leading to a specification to the algorithm from which computer code can be generated. Also the control algorithms are integrated with user interfaces and system-specific functionality (i.e. command structures). Each refinement step is verified through simulation. The generated code is functionally identical to the control-system part of the simulated system.

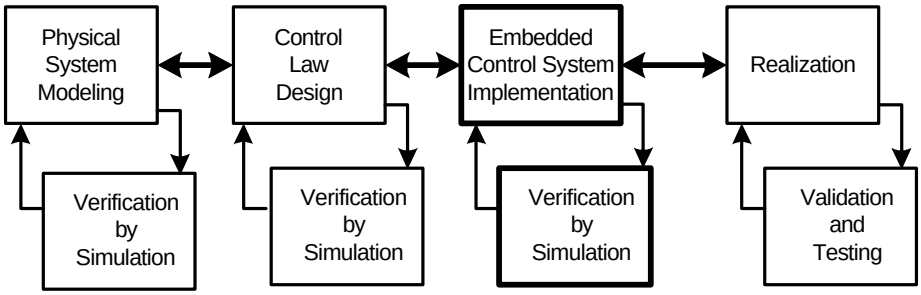

Figure 2: Design approach for Embedded Control System Software

Parallel hardware and parallel software is used, to exploit the inherent parallel nature of embedded systems and their control. Using this parallelism, control software can elegantly be described as a set of parallel processes and straightforwardly be mapped onto parallel computers. Using CSP techniques (Communicating Sequential Processes) grants a true object–oriented approach.

Using simulation as a means for verification at all stages of the design process allows for proper checking of each change. Furthermore, due to the object–oriented nature of the specification, a building–block approach can elegantly be followed. Together with the extensive use of simulation, this enables concurrent engineering: Parts of the control system (algorithm, control computer, interface hardware, appliance (i.e. plant to be controlled)), can be simulated whilst other parts are already in their final realization.

Our research is currently organized as three projects: (1) the development of a design framework elaborated upon in Section 2; (2) the enhancement of our current modeling, simulation and controller design package 20-SIM [[3]; [4]] to support for embedded control software implementation, as discussed in Section 3; and (3) the use of field busses for realization of heterogeneous networked embedded systems, presented in Section 4. We focus on applications in the field of robotics and mechatronics.

2 Software framework for real-time and parallel processing

In this subproject, a basic framework (a kind of real-time operating system) for implementing embedded–system software on heterogeneous distributed hardware is being developed. The process algebra CSP forms the theoretical basis.

CSP–type channels, together with the newly developed *link driver* concept, take care of the communication between (sequential) processes and I/O. These channels implement the data communication between processes, drawn as the connection lines in control engineering block diagrams (cf. Figure 1) or data flow diagrams (see Figure 3) when the control laws have been transformed to algorithms. Obviously, data communication serves as synchronization points. In our framework, this is implemented as a *waiting rendezvous*, according to the CSP theory, and encapsulated in the channel software. The method *calls* for the communication in traditional object-oriented software are replaced by CSP *channel communications* [[5]]. The channels offer, besides the data transport, also higher-order multithreading constructs, and are capable of dealing with single- and multi-processor environments.

*) This research is supported by PROGRESS, the embedded system research program of the Dutch organization for Scientific Research, NWO, the Dutch Ministry of Economic Affairs an the Technology Foundation STW.

PT Embedded Systems, **3,** (Dossier 2002), pp. 36-38, ISSN: 1389-1405; Note: the final publication is layouted differently than this submitted text.

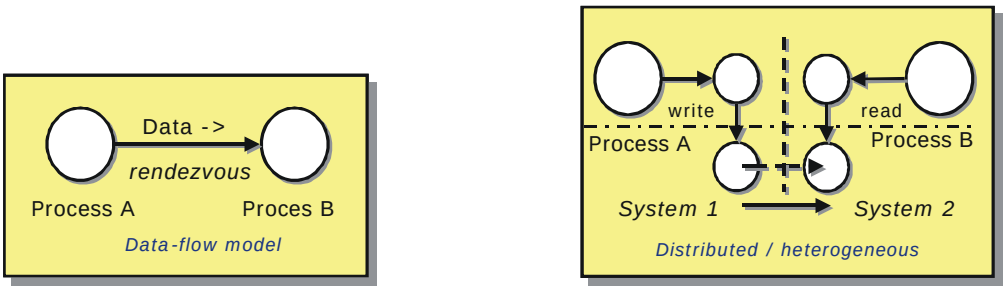

Figure 3: Dataflow (left) and our implementation (right) where the lower bubbles represent the link drivers.

Furthermore, the channels take care of the real-time priority scheduling. For this, the notion of priority and scheduling has been carefully examined and consequently it was reasoned that priority scheduling should be attached to the communicating channels rather than to the processes. Thus, the use of channels hides threads and priority indexing from the user, simplifying the use of priorities for the object-oriented paradigm. Moreover, the notion of scheduling is no longer connected to the operating system but has become part of the application instead. Thus, the parallel & distributed software-writing problem will be alleviated.

The software is written in Java, and was awarded the 100% pure Java certificate by Sun. Using the Java code as a design pattern, the library is implemented in C++ and C, because Java compilers are not yet available for contemporary control processors. Furthermore, the performance of the Java version stayed far behind our real-time requirements, due to the interpretative run-time environment of Java.

3 Design tools for embedded control system software

The purpose of this subproject is the development of a software tool and libraries, which allow for building high reliable software for heterogeneous real-time control systems in a short time at a fraction of the present day costs. The basic CSP–framework as discussed above, is used such that the real-time programming problem will be alleviated.

Emphasis is put on the transformation from a high level description (i.e. control law) to computer code for the embedded computer (Java, C++, C). Refinements, such as command structures, safety checks and data integrity algorithms, are added during the control–code development process. Each refinement step is verified through simulation. In fact, with this project we develop a design tool for part three and four of our design method (right half of Figure 2).

For the other two parts, we already have a tool, 20-SIM, prototyped in earlier research projects, and now being commercialized by Controllab Products B.V., a University spin off company. In fact, generation of C-code out of a model part is already possible, although in a rudimentary form.

For modeling the dynamic behavior of the appliance we use *Bond Graphs* [[6]; [7]; [3]]. Bond Graphs are directed graphs, showing the relevant dynamic behavior. Vertices are the submodels and the edges denote the ideal exchange of energy.

Bond Graphs are *physical-domain independent*, due to the fact that *physical* concepts are analogous for the different physical domains. Thus, mechanical, electrical, hydraulic, etc system parts are all modeled with the same graphs. Since the amount of basic physical concepts is limited, the number of basic elementary bond graph models is limited too.

Another starting point is that it is possible to write models as *directed graphs*: parts are interconnected by bonds, along which exchange of energy occurs. A bond represents the energy flow between the two connected submodels. This energy flow can be described as the product of two variables (effort and flow), letting a bond be conceived as a *bilateral signal* connection. During modeling, the first interpretation is used, while during analysis and equations generation the second interpretation is used.

Encapsulation is granted because:

- The interfaces of bond-graph submodels consist of so-called *ports*, consisting of two variables, whose product is the power exchanged through the port. For each physical domain, such a pair can be specified, for example voltage and current, force and velocity.
- The submodel equations are specified as real equalities, and not as assignments.

PT Embedded Systems, **3,** (Dossier 2002), pp. 36-38, ISSN: 1389-1405; Note: the final publication is layouted differently than this submitted text.

Differential equations are generated after model processing, where the port variables obtain a computational direction (one as input, the other as output) and the equations are rewritten to assignment statements. Simulation of bond-graph models to study the dynamic behavior is in fact repeatedly executing the model statements.

4 Real-time control on field-bus interconnected systems

The purpose of this soon to–be started subproject is to use the basic CSP-framework in the context of real–time embedded systems composed of several co–operating processors in networked environments. Application areas are for instance automotive, production machines and Personal Area Networks (PAN's).

In future, industrial and consumer applications using embedded computer do ask more and more flexibility, whereas size, production costs and time-to-market need to go down to keep, or increase, the competitiveness of the products. Especially, when a family of end products is composed of a clever combination of semi-manufactured parts, the compositional properties are essential for an efficient business management.

Typically, the constituting parts of such an modular end product have their own intelligence, i.e. processing element, and mostly are interconnected via some strictly defined interface, like a computer bus or a fieldbus. Such a bus mostly diminishes the amount of cabling, thus causing a more simpler and probably a more competitive product. Clearly, this bus system should be transparent to let the composition of parts be real plug and play. A kind of *Connection Abstraction Layer* is needed, which should be such that its latency should be negligible comparing to the processing times of the application. Note that the hard real-time control loops run over the fieldbus, which implies that per cycle the bus latency is seen twice (Figure 4).

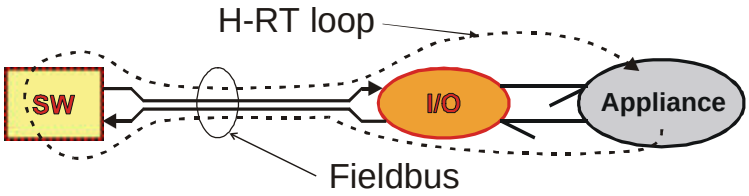

Figure 4: Hard real-time loop over an fieldbus for a distributed embedded system

Distributing the processing elements over give possibilities to tune the processing elements to the computational performance needed by its module. Furthermore, this distributedness gives possibilites to design *intelligent appliance parts*, which can be treated as *components*, making reconfiguring machines easy. These mechatronic components can than be used in the same way as software components are used in component based software development.

In this project, we will provide for an elegant way of letting work together embedded controllers using our CSP-based channel framework. Especially on scalability and on the communiation architecture of the software, we can provide a novel way of working. So, our channel framework will be used as the Communication Abstraction Layer. As physical connections, industrial field busses are used, either wired or wireless. The software in the different processing elements is described as a set of *Communicating Sequential Processes*. Processes reside on different computing elements, and *Channels*, enabling a clear interface, specify the interprocess communication. The implementation of these interprocess connections, including the scheduling is completely encapsulated in the channel itself, see Figure 3 (right side).

5 Conclusion

By using an integrated approach in designing embedded (control) systems, it is possible to elegantly exploit the advantages of concurrent engineering, and to use a mechatronic or systems approach to design. The latter truly supports *flexible hardware-software co-design*, which becomes crucial in modern embedded system development.

Due to the co-operation with industry via the project structure of STW-PROGRESS (funding the two subprojects presented in Section 3 and Section 4), real-life industrial embedded systems will be designed using tools and ideas being developed, providing for feedback to the research.

PT Embedded Systems, **3,** (Dossier 2002), pp. 36-38, ISSN: 1389-1405; Note: the final publication is layouted differently than this submitted text.

References

[1] S. Bennett, *Real-Time Computer Control: An Introduction*. London, UK: Prentice-Hall, 1988.

[2] H. Kopetz, *Real-Time Systems: design principles for distributed embedded systems*: Kluwer Academic Publishers, 1997.

[3] J. F. Broenink, "Computer-aided physical-systems modeling and simulation: a bondgraph approach," in *Faculty of Electrical Engineering*. Enschede, Netherlands: University of Twente, 1990.

[4] J. v. Amerongen, "Modelling, Simulation and Controller Design for Mechatronic Systems with 20-Sim 3.0," presented at 1st IFAC conference on Mechatronic Systems, Darmstadt, Germany, 2000.

[5] G. H. Hilderink, A. W. P. Bakkers, and J. F. Broenink, "A distributed Real-Time Java system based on CSP," presented at Proc. Third IEEE Int. Symp. On Object Oriented Real-Time Distributed Computing ISORC'2000, Newport Beach, CA, USA, 2000.

[6] P. C. Breedveld, "Multibond-graph elements in physical systems theory," *Journal of the Franklin Institute*, vol. 319, pp. 1-36, 1985.

[7] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System dynamics, a unified approach*, 2nd ed. New York, NY: J Wiley, 1990.