AIDA II Progress Report

Towards a Co-simulation Environment for Computer Control Systems

Jad El-khoury

KTH, Department of Machine design,
Mechatronics lab, 100 44 Stockholm
jad@md.kth.se

ABSTRACT

Modern machinery, such as automobiles, trains and aircraft, are equipped with embedded distributed computer control systems where the software implemented functionality is steadily increasing. The development of such systems, however, is still a relatively new and young discipline. There is consequently a lack of tools, methods and models to support, in particular, the early architectural design stages.

Closely associated with the design of a distributed control system are design decisions on the overall system structure, function triggering and synchronization, policies for scheduling, communication and error handling. These parameters to a large extent determine the resulting system timing and dependability, and hence, impact the control application performance and robustness.

The primary target for the research described here is the development of executable models to support architectural design. The report describes earlier efforts in this direction including case studies in the DICOSMOS and AIDA projects. Experiences from the studies are discussed including aspects on modelling to support interdisciplinary design, modeling abstraction and accuracy, and tool implementation.

A state of the art survey of related modelling efforts reveals that very little appears to have been done in terms of modelling and simulation that involves simulation of the environment with the computer control system. In addition, the treatment of distributed computer systems, redundant systems, and error scenarios in this context appear to be scarce.

The possibilities for extending the simulation models to arrive at a ready to use library to support the design of distributed control systems are discussed. This work is already under way. Remaining work includes to verify the developed simulation models, and to evaluate the models and their usage in case studies. One related topic for further work includes integrating the simulation models with the AIDA modeling framework. A longer term aim is that the models and simulation features should form part of a larger toolset supporting also earlier and later development stages, thus providing more comprehensive analysis and synthesis support.

1. INTRODUCTION

This report is a summary of an internal report developed at the mechatronics lab in order to identify the short and long-term research work to be done in the AIDA II project. Please refer to (Törngren, 2001) for more detailed information.

1.1. Background: embedded control systems, trends and challenges

Machinery, such as automobiles, trains and aircraft, are increasingly being equipped with em-

bedded control systems that are based on distributed computer systems. In these systems the functionality is steadily increasing due to the possibilities enabled by software and networks for information exchange. The computer system is typically composed of a number of networks that connect the different nodes of the system. A node typically includes sensor and actuator interfaces, a microcontroller, and a communication interface to the broadcast bus. From a control function perspective, such machines can be said to be controlled by a hierarchical system, where subfunctions interact with each other and through the vehicle dynamics to provide the desired control performance.

However, a number of challenges face the developers of such machinery in order to really be able to benefit from the technology advances:

- *The need to cope with the dramatic increase in system (software) complexity*. Apart from the sheer amount of new functionality, complexity also arises from different types of interference between functions partly arising from their usage of shared computer system resources.
- *Managing and exploiting multidisciplinarity*. The development of embedded control systems requires knowledge in, and cooperation between, several different scientific and engineering disciplines.
- *Verification and validation of dependability requirements*. The main challenge here is that of developing mechatronic (software based) systems that are safer and more reliable than their mechanical counterparts. Albeit the advantages, software easily becomes very complex and is difficult to verify and validate. In addition, the distributed computer systems where the software is implemented add “new” failure modes.

1.2. Modelling and simulation in the context of architectural design

The development of embedded computer control systems is still a relatively new and young discipline. Consequently, there is a lack of tools, methods and models to support, in particular, the early so called architectural design stages. The primary target for the research described here is the development of models to support architectural design. In the context of designing a distributed computer control system, architectural design is here used to refer to decisions on the:

- Overall system structure, including both functional, software and computer structure (These issues include deciding on allocation and partitioning)
- Function triggering, synchronization, and policies for scheduling all resources
- Communication principles
- Policies for error handling, and the mechanisms used for error detection.

Choices of these “design parameters” to a large extent determine the resulting system reliability, safety, and timing, and impact the control application performance and robustness.

The very basic idea is to develop models that can be used to analyse different architectural proposals, simulation is the analysis approach taken in this work. In a longer term perspective, the aim is that the models and simulation features should form part of a larger toolset being developed at the mechatronics lab; a toolset that supports the design of embedded control systems in order the meet the main identified challenges: complexity, multidisciplinarity and dependability.

Some of the requirements on the models are as follows (the reader is referred to Törngren (1995) and Redell (1998) for more detail and background):

- The models should allow both functionality, to be implemented in a computer system, and the computer systems to be represented.

- The models should allow co-simulation of functionality, as implemented in a computer system, together with the controlled continuous time processes and the behavior of the computer system.
- The models should support interdisciplinary design, thus taking into account different supporting methods, modelling views, abstractions and accuracy, as required by control, system and computer engineers.
- The models should be useful also for a descriptive framework, visualising different aspects of the system, as well as being useful for other types of analysis such as scheduling analysis.

1.3. Why model based architectural design?

By model based design we primarily refer to models that are sufficiently formal to allow analysis to be carried out. Early architectural analysis allows the solution space to be explored, and at the same time provides a better opportunity for the early detection of erroneous requirements and design bugs. Clearly, such mistakes are very costly if detected late. Compared to traditional testing, the approach allows different failure modes to be approached and analysed one at a time, progressing the verification through the development in an iterative and incremental fashion.

A strong advantage of model based design used in the context of early analysis, is the ability to evaluate alternative architectures with very few restrictions. Compared to traditional integration testing, or hardware in-the-loop simulation where the complete environment of a node is simulated in real-time, model based design and analysis provides additional advantages including the ability to easily and with low cost change the design, and the possibility to instrument and analyse the internals of the distributed computer system. This is not to say that model based design can replace the others, but the tasks of for example the system integrator can be made much easier given that some aspects of the system, such as its timing behavior, have been analysed thoroughly earlier. A strong point of the modelling activity is that it is a very useful process that can reveal a number of aspects, such as missing requirements and design bugs.

The investigation of computer system models, at different levels of abstraction with different accuracy/complexity, is an educative process that we hope can provide additional understanding to support interdisciplinary design. The model and tool prototypes are also very important as part of the research because they enable different forms of feedback.

2. ACCOMPLISHMENTS IN MODELLING AND SIMULATION AT THE MECHATRONICS LAB

2.1. Early work on modelling and simulation in the DICOSMOS project

An early investigation of “timing problems” was carried out in the DICOSMOS project, see Wittenmark et al. (1995), Törngren (1995), Nilsson (1996). Here the effects of time-varying feedback delays, sampling period jitter and data loss in feedback control systems were investigated. Inspired by earlier work along the lines of Ray (1988), cosimulation was one approach taken in DICOSMOS towards analysing the timing problems. Special Simulink (www.mathworks.com) blocks were developed to model time-varying delays, sampling instant jitter, vacant sampling and sample rejection (data-loss through over-writing of data), Törngren (1995). A feedback control system, as modelled in Simulink, could then be instrumented with these blocks to come one step towards modelling a distributed computer implementation. The approach turned out to be useful for illustrating the effects of the timing problems and for analysing them, e.g. for comparing the effects of sampling period jitter vs. a time varying delay, or for analysing the sensitivity of a particular control design.

2.2. The AIDA modelling framework

The basis for developing the modelling framework was to determine the information needed in the context of the early stages of design of distributed control systems. The models were targeted towards motion control applications implying the need to be able to model time- and event-triggered, multirate, control systems with different modes of operation. These control systems are to be implemented on distributed heterogeneous hardware with both serial and parallel communication links interconnecting the processing elements. There is a need for modelling the various system specific overheads, scheduling policies, error handling etc. The derived requirements and the AIDA modelling framework are thoroughly described by Redell (1998), Törngren and Redell (2000).

2.3. Modelling and simulation of the SMART satellite fault-tolerant computer system

During very early design stages of the development of the distributed computer control system of the SMART satellite, to be launched in 2002 by the European Space Agency, the Swedish Space Corporation needed to analyse and verify the use of the Controller Area Network (CAN) in the on-board satellite computer control system. In particular the error handling of CAN in conjunction with the chosen design for a fault tolerant network was of concern. This work was partly carried out by the authors of this report, and included assessing the design by means of modelling and simulation. The main aim of the simulation was to verify that the given rules for redundancy are not conflicting, and that the system can recover from a single permanent fault. For more information on the simulation models the reader is referred to Törngren and Fredriksson (1999).

2.4. Cosimulation within the DICOSMOS2 project

Sanfridson (1999 & 2000) describes a co-simulation of a truck and semi-trailer using a CAN bus for the on-board distributed control system in order to investigate the performance of control applications and adjust control periods and message priorities on-line. The simulations have been implemented in Matlab/Simulink. The model of the CAN bus is fairly detailed which gives a realistic timely behaviour of the network communication. The major drawback is the amount of processor time required to carry out the simulation at this detailed level.

3. THE STATE OF THE ART

Various simulation tools have been developed to tackle some of the issues discussed in this report. A small survey has been performed in order to evaluate these tools, with an earlier complementing survey given by Redell (1998). What was common between these tools is the fact that they are developed with the aim of simulating real-time computer systems. However, each tool is focused on particular aspects of such systems.

The following tools were evaluated: (See Törngren, 2001 for further details)

- RT/CS Co-design: A Matlab Toolbox for Real-time and Control Systems Co-design
- DRTSS: A Simulation Framework for Complex Real-time Systems
- STRESS - A Simulator for Hard Real-time Systems
- HaRTS: Design and Simulation of Hard Real-time Applications

4. SOME ESSENTIAL ISSUES IN MODELLING AND SIMULATION

4.1. Interdisciplinary design: modeling purpose, abstraction and accuracy

Good cooperation and interfaces between the involved engineers is essential to maintain con-

sistency between specifications, design and implementation. When developing a motion control system for a machine, somehow the functions and elements thereof need to be allocated to the nodes. This principally means that an implementation independent functional design needs to be enhanced with new “system” functions that:

- Perform communication between parts of the control system.
- Perform scheduling of the computer system processors and networks.
- Perform additional error detection and handling.

This mapping will change the timing behavior of the functions due to effects such as delays and jitter.

4.2. Issues related to system development and tool implementation

While developing distributed control systems, it would be advantageous to have a simulation toolbox or library, in which the user can build the system based on prebuilt modules to define things such as the network protocols and the scheduling algorithms. With such a tool, the user can focus on the application details instead. Such a tool is possible since components like different types of schedulers and CAN network are well defined and standardised across applications. Such a tool will enforce a boundary between the application and the rest of the system. This will speed up the development process, and gives the developers extra flexibility in developing the application.

Also, to be useful and cost-effective for system development, the usage of the models and the simulation facilities need to be integrated into the development process.

5. FUTURE WORK

Investigating and extending the AIDA modeling framework. Some further ideas for developing the models are as follows:

- The developed simulation models should be integrated with the AIDA modelling framework. Ideally, the same basic models should be useful for static analysis and for simulation.
- The simulation models do describe both system structure and behavior but they are not always very descriptive since they sometimes lack graphical views. Some proposals in this direction have been made in the AIDA models (Redell, 1998).
- The semantics of pure simulation models (such as the ones in Simulink) inherently differ from the timing behaviour of real-time execution, (Törngren et al., 1997). How can this semantic gap be bridged?
- The simulation models and the AIDA framework need to be extended both upwards, to models used in earlier design stages, and downwards, towards the implementation. Basically, the motivation is given by the need for a holistic design framework that enables models (and work accomplished) to be reused, and used efficiently. Thus it would be desirable to be able to refine the models such that they can be used not just in early design. In addition, work carried out in configuring a computer node for example, should be reusable in later prototyping and implementation stages.
- The use of the Unified Modelling Language (UML, 1997) and CODARTS models (Gomaa, 1993, a development of structured analysis models) in conjunction with the modeling framework will be investigated. Key aspects here include assessing when and why to use object models vs. functional (structured analysis) models, and how they map to other entities such as tasks.

Developing a toolset for architectural analysis. This topic builds upon the ideas of the AIDA project. The idea is to develop a toolset that provides comprehensive analysis and modelling support for embedded control systems.

The following are some ideas for the implementation of a prototype toolset for research purposes:

- The envisioned toolset concept is based on a number of well established engineering tools that are complemented and extended with a number of functionalities. These functionalities include additional models to enable important analysis and synthesis to be carried out. They will also include necessary interfaces between tools. There are two strong reasons for this structure; given limited research efforts, energy should not be spent on reinventing the wheel. In addition, using available tools will make it easier to demonstrate and apply the toolset concept in a realistic setting. It will also facilitate the pinpointing of the opportunities provided by the complementing tools. As an example, there are tools around that can deal with modelling and analysis of reliability, safety, control system design, finite state machines, timing analysis, etc., but these tools are not integrated; and even if they were, can not provide the functionality required for appropriately supporting the design of distributed real-time control systems.
- The functionalities considered include support for overall system structuring in early design stages (where reliability, safety, resource load and costs are evaluated), hazard and safety analysis integrated with control system design, and control design complemented with support for distributed real-time system implementation where timing analysis is one important part.
- Model reuse must be enabled by the envisioned toolset. Reuse can come in different forms. One aspect of this is to be able to use a developed model throughout the life cycle of a product in order to support product maintenance including upgrades. As discussed earlier, this requires models to be refined during the development. Having developed a simulatable model, it would be valuable to reuse this information, for example the computer system configuration, in further prototyping and development work.

Many interesting issues exist and will arise in the development of this type of toolset.This includes how to manage the different types of models and how to use the toolset facilities in system development. The toolset should be complemented by an appropriate design methodology. One important piece of this methodology should be a verification framework that promotes the development of dependable embedded systems.

Extending the functionality of the toolset also requires the models to be extended (as partly discussed in the previous section). For example, the AIDA models currently do not include explicit fault models and attributes such as criticality.

6. REFERENCES

Gomaa (1993). Hassan Gomaa. *Software design methods for concurrent and real-time systems*. Addison-Wesley publishing company, 1993.

Nilsson (1996). Johan Nilsson. *Real-Time Control Systems with Delays*. PhD thesis. ISRN LUTFD2/TFRT--1049--SE, Lund Institute of Technology, Sweden.

Ray and Halevi (1988). Asok Ray and Y. Halevi. Integrated Communication and Control Systems: Part II - Design Considerations. *ASME Journal of Dynamic Systems, Measurements and Control*, Vol 110, Dec. 1998, pp 374-381.

Redell (1998). Ola Redell. *Modelling of Distributed Real-Time Control Systems, An Approach for Design and Early Analysis*, Licentiate Thesis, Department of Machine Design, KTH, 1998, TRITA-MMK 1998:9, ISSN 1400-1179, ISRN KTH/MMK--98/9--SE, Stockholm, Sweden.

Sanfridson (1999). Martin Sanfridson. QoS in Distributed Control of Safety-Critical Motion Systems. Work in progress paper at the *20th Real-time Systems Symposium*, Phoenix, December 1999.

Sanfridson (2000). Martin Sanfridson. *Timing problems in distributed control.* Licentiate Thesis, TRITA-MMK 2000:14, ISSN 1400-1179, ISRN KTH/MMK--00/14--SE, May 2000.

Törngren (1995). Martin Törngren. *Modelling and design of distributed real-time control applications*. Doctoral thesis, Department of Machine Design, KTH, TRITA-MMK 1995:7, ISSN1400-1179, ISRN KTH/MMK--95/7--SE.

Törngren and Fredriksson (1999). Martin Törngren and Peter Fredriksson. *SMART-1. CAN and Redundancy logic simulation of the SMART SU*. Swedish Space Corporation, Report S80-1-SRAPP-1.

Törngren and Redell (2000). Martin Törngren and Ola Redell. A Modelling Framework to support the design and analysis of distributed real-time control systems. *Journal of Microprocessors and Microsystems* 24 (2000) 81-93, Elsevier, special issue based on selected papers from the Mechatronics 98 proceedings.

Törngren (2001). Martin Törngren, Jad El-khoury, Martin Sanfridson and Ola Redell. *Modelling and Simulation of Embedded Computer Control Systems: Problem formulation*. Internal Report, Department of Machine Design, KTH, TRITA-MMK 2001:3, ISSN1400-1179, ISRN KTH/MMK--01/3--SE.

Törngren et al. (1997). Martin Törngren, Christer Eriksson, Kristian Sandström (1997). Real-time issues in the design and implementation of multirate sampled data systems. In Preprints of *SNART 97 -Swedish National Association on Real-Time Systems Conference*, Lund, 21-22 August 1997.

UML (1997). UML notation guide. Version 1.1. Sept. 1997. Object Management Group, doc. no. ad/97-08-05. http://www.rational.com/uml

Wittenmark et al. (1995). Björn Wittenmark, Johan Nilsson, and Martin Törngren. Timing Problems in Real-time Control Systems. P*roceedings of the 1995 American Control Conference*, Seattle, WA, USA.