A case study: Implementation of Control Systems Using B-Action Systems[1]

Pontus Boström and Marina Waldén

Åbo Akademi University, Department of Computer Science
Turku Centre for Computer Science (TUCS)
Lemminkäisenkatu 14 A, 20520 Turku, Finland
E-mail: {Pontus.Bostrom, Marina.Walden}@abo.fi

Introduction

We present a case study where we apply a methodology for formal derivation and implementation of control systems of industrial size. We use B Action Systems [WaSe98] as our theoretical framework for developing reliable and correct control systems in a stepwise manner. In the case study we develop part of a microplate liquid handling workstation [PE01] manufactured at Wallac, a division of Perkin Elmer Lifesciences. Previously, we have concentrated on modelling control systems of industrial size [MATISSE03], but here we focus on their implementation (see also [Boström03]).

Formal development of control systems

Discrete control systems are used in many safety-critical systems such as, e.g., cars and airplanes. These systems need to be safe and very reliable. Control systems usually consist of an environment and software, here referred to as *plant* and *controller*. The plant and controller can communicate with sensors and actuators. The controller reads the sensors in order to obtain an updated view of the state of the plant. The controller then produces an output to the actuators in response to the values read from the sensors. Systems working in this way are said to be reactive.

In order to achieve safe and reliable control systems we need formal software construction techniques. With a formal analysis tool the confidence in the formal development can be increased. Here we use the B Action Systems [WaSe98] formalism as our formal framework for developing distributed control systems. Since B Action Systems are Action Systems [BaKu83, BaSe96] applied in the B Method [Abrial96], we can benefit from the useful formalism for reasoning about distributed systems given by Action Systems and from the tool support in B. The development within B Action Systems is performed in a stepwise manner from abstract specification to concrete implementation using superposition refinement [BaKu83]. The correctness of each step should be proved to achieve a reliable system. The B tool assists the development process by generating the proof obligations needed. Some extra machine constructs in B need to be created to be able to generate all B Action Systems specific proof obligations [BuWa96]. These proof obligations can then be proved with the automatic or the interactive prover of the tool. Moreover, the B tool has a translator for automatically transforming concrete B specifications into the programming languages C, C++ or Ada.

The development process starts by capturing the requirements of the system. The different components of the system are then identified. From the requirements we create an abstract B Action System. We model the whole system as one entity in order to make it possible to state properties about the entire system. The system is then decomposed into the earlier identified components [MATISSE03]. Each component is refined in a stepwise

[1] Work done within the MATISSE-project, IST-1999-11435, *http://www.matisse.qinetiq.com*

manner adding new implementation details to the component specification in each step. By incorporating safety analysis in the development we can gradually add safety properties to the system [Troubitsyna00]. After having introduced all required implementation details the components are decomposed into plant, controller, sensors and actuators. The controller in turn is divided into two parts; an interrupt handler and a part containing the actual control procedures. At this point the control procedures can be automatically translated into a programming language. The actions of the interrupt handler are merged into a single action that receives messages in form of interrupts from the environment and calls the corresponding controller procedures. This merging process still lacks tool support. When also the interrupt handler has been translated into a programming language, the implemented components of the control system are interfaced with the hardware or the operating system.

The Fillwell case study

In our case study we used the methodology described above to develop part of a microplate liquid handling workstation, Fillwell [PE01], for discovering new drugs. The Fillwell workstation consists of a dispense head, a gantry and a processing table with microplates. The dispense head dispenses liquid into the plates on the processing table. The gantry moves the dispense head from one plate to another on the table. Both the dispensing and moving have to be performed with very high precision.

Here we concentrate on implementing the controller of the gantry [Boström03]. The gantry moves the dispense head horizontally (in the x- and y-directions) with two linear motors and vertically (in the z-direction) with a stepper motor. Three position encoders, one for each direction, give the position of the dispense head. Moreover, the system has an emergency stop button. When it is pressed the system should shut down safely. We can identify five components of the gantry: three components for handling the movement in the x-, y- and z-direction; a component for coordinating these components; and a component for handling the communication between the other components.

All the components of the gantry were independently refined. We used the tool Atelier B [ClearSy03] to assist in the development. In the components for the x-, y- and z-movement we added features for handling interrupts and measuring the position of the dispense head. Also the coordinating component was refined to take into account the changes in the other components. When these features had been added all the five components are partitioned into plant, controller (with procedures and interrupt handlers), sensors and actuators. For the gantry development 3109 proof obligations were generated. The automatic prover of Atelier B proved most of them (94%), the rest were proved with the interactive prover.

Finally, the controllers of the five components were translated into C with Atelier B. For each controller the actions in the interrupt handler were manually merged to form a single action and these transformed interrupt handlers were then automatically translated into C. The program code has not been used on the real Fillwell workstation, but a simulator has been constructed to interface the implemented components.

Conclusions

Our method provides a way to model, specify and design the complete control system with both the environment and the software controller taken into account. By using the B Action System framework and the Atelier B tool, safety properties can be proved about the entire system. The software for the controller can be automatically generated from the specification. Hence, we have a method for correct implementation of control systems. Though, it is only possible to prove that a system is correct in respect to the specification. If the specification is wrong, they system will not work correctly.

There are some limitations with this method. These limitations are mainly due to limitations in the B Method. For example, it is not possible to use sequential composition in the controller. Moreover, continuous behaviour of the environment cannot be modelled, nor does the B Method support real or rational numbers, which are necessary when modelling dynamic systems. Despite these limitations the method using the B Action Systems formalism seems to be quite well suited for these kinds of problems. The method is relatively easy to apply and the case study has provided some insight in the design of larger systems using this method [Boström03].

B Action Systems have been used previously by Sekerinski [Sek98] for designing control systems. However, he has used a bottom-up approach were the system is modelled by several small machines that are later merged into one. Lano presents a method for implementing discrete event systems using the B Method [Lano00]. However, he does not model the environment in his approach and can, therefore, not prove properties about the system as a whole.

References

[Abrial96] J. R. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[BaKu83] R. J. R. Back and R. Kurki-Suonio. Decentralization of Process Nets with Centralized Control. *Proceedings of 2nd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, Montreal, Canada, August 1983, pp. 131-142.

[BaSe96] R.J.R. Back and K. Sere. From modular systems to action systems. *Software -- Concepts and Tools 17*, pp. 26-39, 1996.

[Boström03] P. Boström. *Formal Development of Control Systems*. M. Sc. Thesis, Department of computer science, Åbo Akademi University, Finland, September 2003.

[BuWa96] M. Butler and M. Waldén. Distributed system development in B. *Proceedings of the 1st Conference on the B Method*, Nantes, France, pp. 155-168, November 1996.

[ClearSy03] *Atelier B*, ClearSy, http://www.atelierb.societe.com/, latest accessed 15.09.2003.

[Lano00] K. Lano, J. Bicarregui and P. Kan. Experiences of Using Formal Methods for Chemical Process Control Specification. *Control Engineering Practice*. Vol. 8. Elsevier Science, 2000.

[MATISSE03] *MATISSE Handbook for Correct Systems Construction*. EU-project MATISSE: Methodologies and Technologies for Industrial Strength Systems Engineering, IST-1999-11345, 2003. http://www.esil.univ-mrs.fr/~spc/matisse/Handbook

[PE01] Perkin-Elmer Life Sciences, Fillwell™2002 – Features Guide, 2001. *http:/www.abo.fi/~marina.walden/fillwell.pdf.*

[Sek98] E. Sekerinski. Production Cell. Chapter 6 in E. Sekerinski and K. Sere (editors.), *Program Development by Refinement – Case Studies Using the B Method*, pp. 197-254. Springer-Verlag, London, 1998.

[Troubitsyna00] E. Troubitsyna. Stepwise Development of Dependable Systems, Turku Centre for Computer Science, TUCS, Ph. D. thesis No. 29, June 2000.

[WaSe98] M. Waldén and K. Sere. Reasoning About Action Systems Using the B Method. *Formal Methods in Systems Design 13(5-35)*. Kluwer Academic Publishers, 1998.