27

Development Framework for real-time control system design

J. M. Bass[†], *A. R. Browne*[†], *M. S. Hajji*[†], *P. R. Croll*[‡] *and P. J. Fleming*[†]
[†]*Dept. of Automatic Control and Systems Engineering,*
[‡]*Dept. of Computer Science, University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK. Tel. +44 (0)114 282 5236, Fax. + 44 (0)114 273 1729, E-mail: J.Bass@sheffield.ac.uk*

Abstract

The Development Framework provides a highly automated translation from a specification to a parallel implementation. The specification is in a popular graphical control engineering notation, typically representing a system with stringent dependability requirements and hard real-time constraints. An interface has been constructed between the Development Framework and the dependability modelling tool, SURF-2. The demonstration will illustrate the Development Framework design approach using a primary flight control Case Study. The example application consists of a three channel autopilot and airframe model. Dependability models of competing autopilot architectures will be contrasted in the demonstration.

Keywords

Computer-aided control system design, dependability modelling, computer-aided software engineering, real-time systems, distributed systems.

1 INTRODUCTION

The Development Framework, an environment to support the specification, design and implementation of real-time distributed computer control systems is described here. It is argued that both good design practice and fault-tolerance are required to ensure that stringent reliability targets are met. Distributed computer control systems have the advantage that redundant processing elements are available for use to provide fault-tolerance.

The Framework, provides support for three phases in the development of the system under design. The Specification Phase, described in Section 2, allows the designer to specify, analyse and simulate the control system under development. The Development Framework includes tools that automatically translate the control engineering representation into a software engineering representation. The Software Design Phase uses a software engineering notation, described in Section 3, to enable analysis and refinement of the system under development. One type of analysis available to the developer is the generation of stochastic Petri net dependability models, described in Section 3-1. Further Development Framework tools translate the software engineering representation into source code that can be compiled into executable code for a network of processors. The resulting parallel implementation is discussed in Section 4.

I. Jelly et al. (eds.), *Software Engineering for Parallel and Distributed Systems*
© IFIP International Federation for Information Processing 1996

The demonstration will use a Case Study to illustrate the Development Framework approach. The Case Study is not described in detail here, due to lack of space, but is introduced in Section 5. Conclusions are provided in Section 6. Further information regarding the Development Framework can be found in (Browne, 1994) and (Bass, 1994).

The Development Framework addresses a similar problem area to the ControlH/MetaH design environment (Vestal, 1994). In common with the Framework, the ControlH/MetaH tools use an application-specific graphical specification notation and an intermediate software engineering notation. However, the Framework integrates commercially available tools using translators, while ControlH/MetaH is implemented entirely using purpose built tools. Further, the ControlH/MetaH environment does not provide facilities for dependability modelling. Detailed dependability modelling, without the benefits of system specification, design and implementation support, can be performed using Markov chains or stochastic activity networks. The SAVE environment uses a textual system description to provide dependability measures (Blum, 1993). In contrast, the UltraSAN environment provides a graphical interface based on stochastic activity networks (Sanders, 1993). The dependability modelling tool selected for this work, SURF-2, is Markov-based using stochastic Petri nets.

The Development Framework approach encourages the designer to concentrate on the control engineering design aspects of the proposed system. This is achieved by providing a highly automated path from a control engineering specification to a distributed system implementation. Figure 1 shows the three phases supported by the Framework and the main benefits provided in each phase. The Development Framework provides an open architecture to encourage the designer to intervene at appropriate stages of the design lifecycle for the purposes of optimisation.

2 SPECIFICATION PHASE

The specification of software with the use of diagrams is seen as one of the main advantages of CASE systems. It is generally recognised that diagrams allow the representation of system structure in a much more accessible and natural form than written language or mathematics. Graphical notations have been developed that are appropriate for the specification of control systems and are used within the Development Framework. Therefore a control engineer should readily be able to understand the specification of a control system in such a notation. This would not generally be true if the design of the control system was in, for example, a

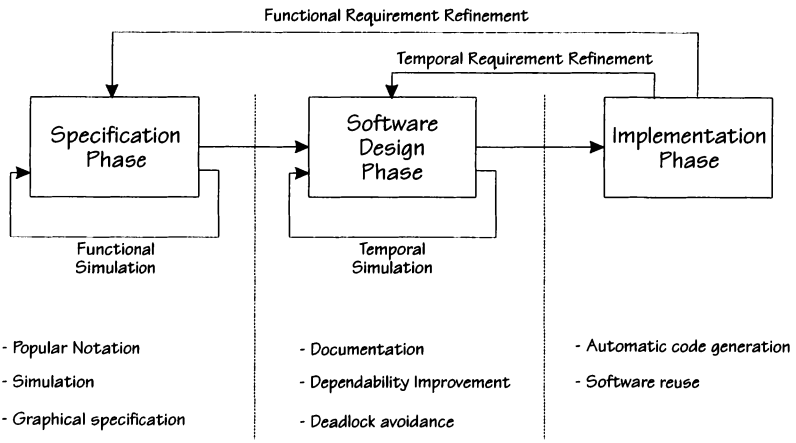

Figure 1 Development Framework overview.

software engineering notation. Simulink was selected for the specification of real-time control systems in the Development Framework because it: accommodates both continuous and discrete elements, and supports the hierarchical decomposition of diagrams enabling representation of complex control systems. Simulink supports modelling and simulation during control law design and is also used to provide a well documented mechanism for the specification of control systems. Simulation enables verification that the system meets requirements prior to implementation. The notation used by Simulink, in common with similar notations, was not designed to represent many of the features central to parallel and distributed systems, however. The Software Design Phase is, therefore, implemented to enable deadlock analysis, mapping and, if required, dependability analysis and the introduction of fault-tolerant mechanisms.

3 SOFTWARE DESIGN PHASE

The most novel and powerful feature of the Development Framework is the automatic translation of specifications, using an application-specific notation, into designs, using a generalised software engineering notation. An equivalent dataflow diagram is created for each Simulink diagram within a model, and a data structure diagram is created for every connection between blocks in each Simulink diagram. All functional blocks within the Simulink diagram (gains and transfer functions, for example) are converted into equivalent process symbols. Each Simulink inport/outport symbol is converted into an off-page connector, allowing processes and their decompositions to be linked. Thus, a complete description of the application system under design is maintained in the CASE tool. This complete description is required to allow the analysis, implementation and documentation of the proposed design.

The Framework draws on the CSP message passing paradigm (Hoare, 1985). The message-passing approach of CSP provides an elegant platform for the development of such distributed systems. Dataflow diagrams are used to model concurrent processes and message-passing channels. CSP-based processes and communication channels are, thus, conveniently modelled using CASE tools. The CASE tool environment, Software through Pictures (StP), was adopted for the Development Framework project because it: supports the well documented and widely known Yourdon methodology with Hatley/Pirbhai real-time extensions (Hatley, 1987); enables the generation and manipulation of diagrams with minimal user intervention; and has a flexible and extendible storage structure for specific information

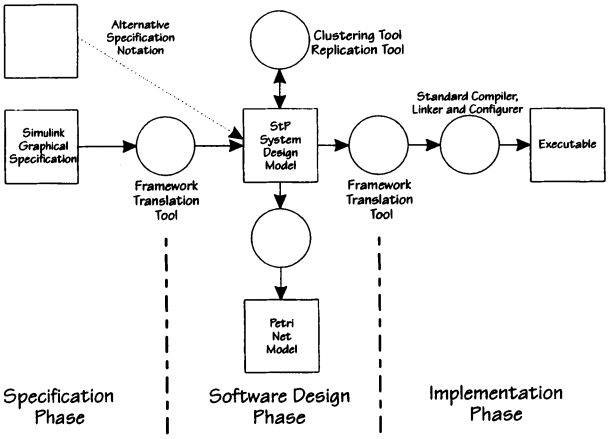

Figure 2 **Development Framework tools.**

about each object (diagram, process, data flow etc.) within the system.

Tools to perform replication of selected processes, generation of hierarchical coloured Petri nets and to cluster processes have been implemented. These allow analysis or perform optimisations on the distributed system under development in the software engineering domain. These optimisations can be performed with minimal intervention by the user. An approach to generating dependability models of the system under development is described below.

3-1 Stochastic Petri net tool

Generalised stochastic Petri nets enable the evaluation of system safety and reliability measures. The SURF-2 environment performs model processing based on graphical Petri net (or Markov chain) representations (Béounes, 1993). The SURF-2 Gateway, shown in Figure 3, supports automated generation of Petri nets from external software tools. The Framework stochastic Petri net tool analyses the system under development and translates the dataflow representation into a textual Petri net notation (Bass, 1995). Dependability models of selected fault-tolerant mechanisms are currently supported. Figure 4 shows typical translations for recovery block and n-version systems. The dependability models can be used to perform sensitivity analysis or contrast competing system architectures.

4 IMPLEMENTATION PHASE

A formalism is required in order to generate code from dataflow diagrams. Without this formalism there is no way of expressing the control of processes or the synchronisation of communications between them. The formalism represents each non-decomposed process symbol in the dataflow diagrams for a system as a separate process in the implementation. All these processes execute iteratively. In each iteration, the process: receives data from all input data flows; executes the functional code (transfer function or gain, for example); and sends data to all the output data flows. If a process has no input data flows the process waits for a signal from the process manager before executing the functional code. The process manager is a separate task responsible for the correct real-time operation of all the processes on a processor.

The formalism used limits the prototype Framework to the specification, design and im-

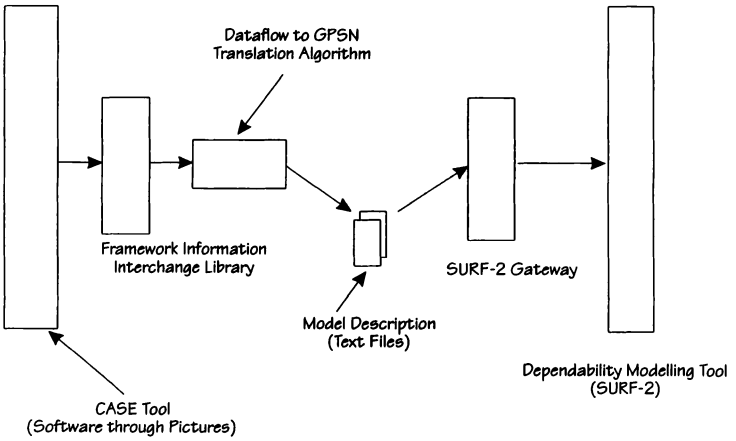

Figure 3 **Development Framework to SURF-2 interface.**

plementation of purely periodic systems. No concept of aperiodic tasks or events has yet been developed. All inter-process communication is strictly synchronous. The Framework currently produces source code in the language "C" for the Virtuoso real-time kernel executing on a network of Inmos Transputers. Transputers provide a convenient platform for the CSP model and have found numerous applications in real-time control (Irwin, 1992). The Virtuoso kernel includes a flexible, reconfigurable, synchronous message passing system and a rate-monotonic scheduler which makes it particularly suitable for the Framework.

The Framework code generator produces all the code required to compile, link and execute the system. For each process within the system two source code files are produced, a harness code file, and an application code file. The harness file contains code that manages inter-process communication and communication with the process manager. It is automatically generated to match the needs of the process. The application code file contains the code for the functional part of the process e.g. transfer function or gain. This code is an expansion of a template taken from a library of reusable source code modules. The development of such a library reduces both the implementation time, by automatically reusing existing code, and improves software reliability. The choice of a suitable library module for a process is performed automatically based on the number and type of input and output data flows and the type of routine (e.g. gain) that is required. This information is all stored within the CASE system when the control systems design is converted into data flow diagrams.

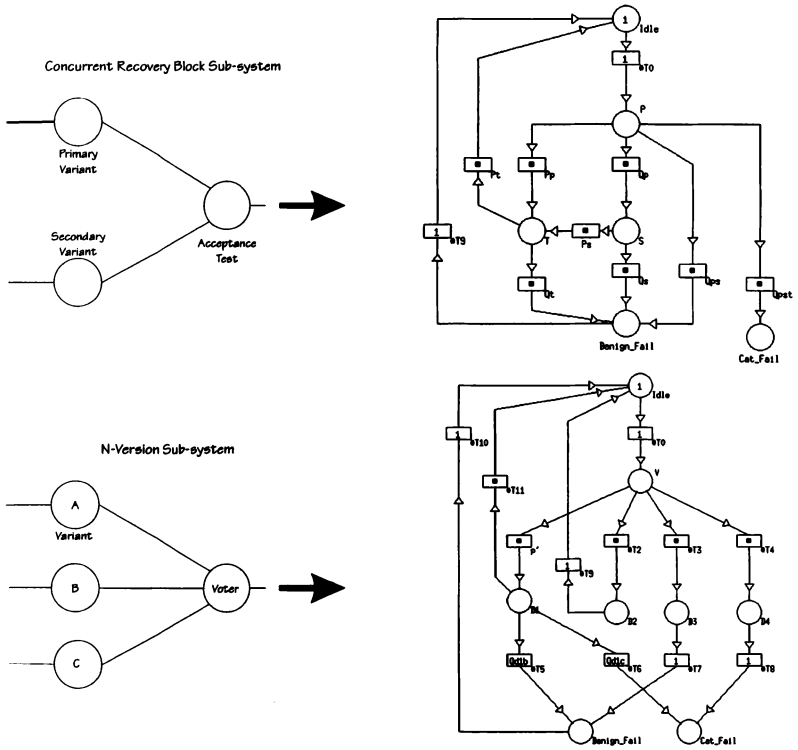

Figure 4 Typical dataflow to stochastic Petri net translations.

5 CASE STUDY

The software demonstration will use a primary flight control Case Study to illustrate the Development Framework design approach. The application consists of a generic three channel autopilot and airframe model.

6 CONCLUSIONS

The prototype Development Framework described here enables a highly automatic translation from an application-oriented system specification to an implementation executed on a parallel platform using a real-time kernel. In summary, the Framework approach offers a number of benefits. The system specification is in an application-oriented notation which can be simulated, to ensure correct functional behaviour, prior to implementation. Code re-use and automation of error-prone manual translations, reduce development time and increase confidence in implementation reliability. The open architecture provided by the Development Framework allows the addition of tools to address problems at different stages of the design lifecycle.

7 ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of UK EPSRC (under grant number GR/K64310) and Intelligent Systems International, suppliers of the Virtuoso kernel.

8 REFERENCES

Bass, J. M., A. R. Browne, M. S. Hajji, D. G. Marriott, P. R. Croll and P. J. Fleming (1994), "Automating the Development of Distributed Control Software", IEEE Parallel and Distributed Technology, Vol. 2, No. 4, Winter 1994, pp. 9-19.

Bass, J. M., S. Metge, P. R. Croll and P. J. Fleming (1995), "Dependability Modelling in a Prototype Development Framework", IEEE 25th Ann. Int. Symp. on Fault-Tolerant Computing Systems, Pasadena, June 1995, pp. 131-6.

Béounes, C., et al (1993), "SURF-2: A program for Dependability Evaluation of Complex Hardware and Software Systems", Digest of Papers, IEEE 23rd Ann. Int. Symp. on Fault-Tolerant Computing Systems, Toulouse, June 1993, pp. 668-73.

Blum A. M. et al, (1993), "System Availability Estimator (SAVE) Language Reference and User's Manual", Research Report RA219S, IBM Research Division, T. J. Watson Research Centre, Yorktown Heights, N. J., June 1993.

Browne, A. R., J. M. Bass, P. R. Croll and P. J. Fleming (1994), "A Prototype Framework of Design Tools for Computer-Aided Control Engineering", Joint IEEE/IFAC Symp. on Computer-Aided Control System Design, 1994, pp. 369-74.

Hatley, D. J. and I. A. Pirbhai (1987), "Strategies for Real-Time System Specification", Dorset House Publishing Co. Inc.

Hoare, C. A. R. (1985), "Communicating Sequential Processes", Prentice-Hall.

Irwin, G. W. and P. J. Fleming (eds.) (1992), "Transputers in Real-Time Control", Research Studies Press.

Sanders, W. H. and W. D. Obal II (1993), "Dependability Evaluation using UltraSAN", Digest of Papers, IEEE 23rd Ann. Int. Symp. on Fault-Tolerant Computing Systems, Toulouse, June 1993, pp. 674-79.

Vestal C., (1994), "Integrating Control and Software Views in a CACE/CASE Toolset", IEEE/IFAC Joint Symp. on Computer-Aided Control System Design, Tuscon, Arizona, March 1994, pp. 353-58.