A REFINEMENT BASED APPROACH TO CALCULATING A FAULT-TOLERANT RAILWAY SIGNAL DEVICE

Alistair A. McEwan & J. C. P. Woodcock

Computing Laboratory, University of Kent at Canterbury, UK

A.A.McEwan@kent.ac.uk, J.C.P.Woodcock@kent.ac.uk

Abstract In this extended abstract, we present a technique whereby a new, fail-safe and fault-tolerant architecture for an existing device is calculated, and verified, from an existing abstract specification. The inspiration for the architecture draws from the Byzantine Generals problem. The model is presented in Hoare's CSP, safety properties are proved using the model checker FDR, and development is guided by the laws of *Circus*. The case study is Montigel's Dwarf Signal.

Keywords: Safety, fault-tolerance, verification, CSP, *Circus*, Montigel's Dwarf Signal

1. Introduction

In this extended abstract, we show how safety critical devices employing distributed consensus approaches to compositional safety can be designed and calculated from an abstract specification, using Communicating Sequential Processes (CSP) [3, 9] and *Circus* [12]. A new implementation of an existing device is calculated from an existing abstract specification, appealing to a well-known distributed consensus algorithm. The result is an architecture that can be shown to preserve all the safety properties of the original, and offer more fault-tolerance. Proofs are conducted using a model-checking tool. The case study used is Montigel's Dwarf Signal [7], a track-side railway signalling device in use in the Austrian railway system.

We begin by presenting relevant background. Section 2 presents a CSP model of a simple distributed consensus algorithm, followed by a description in section 3 of how a new architecture can be calculated, developed, and verified; appealing to the distributed consensus algorithm. Finally, in section 4 we suggest where further improvements to the architecture may be possible, and analyse the success of the development strategy.

CSP and *Circus*

The process algebra CSP is a mathematical approach to the study of concurrency and communication. It is suited to the specification, design, and implementation of state-based system that are characterised by the events in which they participate. *Failures Divergences Refinement* (FDR) [5] is a tool for model-checking networks of CSP processes, checking the containment of processes, and allowing the proving or refuting of assertions about those processes.

The development of a Unifying Theory of Programming [4] has fuelled interest in integrating different languages and notations to obtain new formalisms. *Circus* is an integration of CSP, Z[15], the Guarded Command Language[2] and the Z Refinement Calculus[1]. The semantics of the language have been defined [13]; and a development calculus is presented in [12]. The reasons for choosing to conduct this case study in CSP and *Circus* are three-fold. Firstly, existing techniques in the area are presented in CSP. Secondly, the laws of *Circus* can be used to guide development. Thirdly, verification of each design step can be performed automatically using FDR.

[14] presents a refinement-based technique for verifying safety properties in CSP by building *safety specifications*. The technique involves a set of predicates, implemented as CSP processes, restricting observable states and transitions in a system. Furthermore, a model describing the safe/failed life cycle of a process is presented. This model is used to calculate the fault-tolerance of a system—the failures that can be tolerated before functionality is lost.

Development of the new device is done in a top-down manner. An existing abstract specification of the device is refined, guided by design decisions, into models where sections of the implementation are exposed gradually. It is our belief that, by taking an abstract specification, and refining it into a concrete model by applying a design calculus, greater confidence can be gained in the verification of the implementation, combined with the ability to experiment with design patterns.

Montigel's Dwarf Signal

Montigel's Dwarf Signal is a track-side railway signalling device produced by Alcatel, in use in the Austrian Railway system. The signal advises train drivers how to proceed on a section of track, indicating one of three options—*stop*, *go*, or *proceed with caution*. It is implemented using three lamp assemblies. A lamp assembly consists of the lamp, sensors reporting the state of the lamp, channels calculating required changes of state, actors effecting those changes and an interlock ensuring behaviour is agreed between actors. A further control system accepts commands from a signal-man—for instance, *set to go*—and translates these into concrete commands in each lamp assembly.

Figure 1. The Dwarf Signal

Previous work includes [11], demonstrating that CSP can be used to accurately model the Dwarf Signal. Furthermore, [14] constructs a safety specification for the device, and its behaviour is verified correct with respect to this safety specification. A further result is that it is shown that only one failure may be tolerated before all further functionality is lost, even though the device is in a safe state.

2. A CSP model of a distributed consensus algorithm

Frequently, distributed consensus algorithms are used to identify, and mitigate, failures in a system—the most general failure considered is termed a *Byzantine failure*, where a process may start to behave arbitrarily. Many authors have considered this problem, and a number of common problem areas and well understood algorithms is given in [6]. [8] introduces the constraint that a network containing unauthenticated signatures requires $3n + 1$ processes to reach consensus in the presence of up to n failures, and that where messages are more reliable—i.e, authenticated, $2n + 1$ processes is sufficient. An algorithm for achieving consensus, and for identifying suspect processes is given.

The algorithm consists of two rounds of communication between participants. In the first round, each process informs every other of its intent, and in the second, informs every other process of information collected in the first round. A faulty process may lie at any stage, and the algorithm allows processes to detect this.

Previous experience of the case study permits optimisations of the algorithm. When a channel fails, it informs other channels—therefore other processes in the system may ignore it in all further negotiations. The other possibility of failure is in a sensor—in this event, a channel will observe in round one of the protocol that it disagrees with the majority, and take appropriate

$$\begin{aligned}
& \text{Inform}_{v,a,P,Q} \triangleq \\
& \quad P \neq \emptyset \ \& \ \Box x : P \bullet \\
& \quad \quad \text{notify}.x?y \rightarrow \text{if } y = v \text{ then } \text{Inform}_{v,a \cup \{x\}, P \setminus \{x\}, Q} \\
& \quad \quad \quad \text{else } \text{Inform}_{v,a,P \setminus \{x\}, Q} \\
& \quad \Box \\
& \quad \neg Q \ \& \ \text{notify}.i.v \rightarrow \text{Inform}_{v,a \cup \{v\}, P, \text{true}} \\
& \quad \Box \\
& \quad P = \emptyset \wedge Q \ \& \ \text{if } \#a \leq 2 \text{ then } \text{Minority}'_{\text{Chan}} \text{ else } \text{Consent}_v \\
\\
& \text{Consent}_v \triangleq \\
& \quad \text{actuate}.i.v \rightarrow \text{Inform}_{v,\{i\}, \text{Chan} \setminus \{i\}, \text{false}} \\
\\
& \text{Minority}_P \triangleq \\
& \quad ((\parallel x : P \setminus \{i\} \bullet \text{notify}.x?y \rightarrow \text{SKIP}) \parallel \text{notify}.i.gs \rightarrow \text{SKIP}); \\
& \quad \text{Minority}'_P \\
\\
& \text{Minority}'_P \triangleq \\
& \quad (\parallel x : P \bullet \text{confirm}.x?y \rightarrow \text{SKIP}); \text{actuate}.i.gs \rightarrow \text{Minority}_P
\end{aligned}$$

Figure 2. An optimised protocol for a channel

action. Therefore any one channel has no need to suspect that another has failed, leading to an optimised, single voting round version of the protocol, as in figure 2.

3. Development strategy

Figure 3 presents the development strategy for the new design, and compares it to a suggested strategy for the existing design. This strategy is divided into four identifiable stages, one of which is common to both designs.

In the first stage, the existing abstract specification is refined into one where a lamp, and an abstract model of the control system is revealed. In the second stage, the decision to adopt a distributed consensus algorithm is taken, and consequently, the refined abstract control system reveals interlocks and an abstraction of a four-way lane. Verification of both of these stages is performed automatically using FDR, and calculated using a test-refute cycle.

In the third stage, externally observable state in each lane is revealed, and the actions are partitioned into four concurrent processes using the laws of *Circus*. Typically, a development step such as this in a *Circus* specification involves

Figure 3. Development strategy for 2 and 4 lane architectures

proof obligations; however, in this case study, we verify the correctness of the design step using FDR.

The final stage is to calculate the specification of a channel from the previous phase. This is a weakest pre-specification style calculation: the behaviour sought is that of the abstract specification minus the known behaviour of the actor and sensor. The implementation of this weakest pre-specification must use the distributed consensus model, and the correctness is again verified using FDR.

Figure 4 shows the final architecture. Lamps, sensors, actors and interlocks are all taken from [11]. The interlock system consists of three interlocks, allowing four inputs. A lane consists of a sensor, an actor, and a channel. The lamp reports to all of the sensors, each of which in turn reports to its channel, which, in turn, drives its actor. Each channel is driven by the outside world through its *set* command, and communicated its intentions, or failure, via the events *inform*, and *die*. A channel drives its actor via a *change* event, which, in turn, attempts to drive its interlock via a *switch* command. Failure, negotiations, and driving the actor are internal to the lane and not externally visible.

4. Conclusions and future work

Previous work showed the abstract specification to be fail-safe; transitivity of refinement guarantees that the new architecture is also. Additionally, existing techniques may be applied to the new architecture to investigate the hypothesis that it is more fault-tolerant than the existing design.

[8] introduced the result that when messages are authenticated, $2n + 1$ processes are sufficient to achieve safety in the presence of n failures. In this case study, messages are assumed authenticated and correct. If this assumption holds in the physical device, three lanes would be sufficient to tolerate the failure of a single lane; investigation of this is an item we leave for future work.

Figure 4. A 4-lane distributed consensus design

A second possible optimisation concerns the interlock. An interlock drives the lamp to ground state if any of the participants request it: this may happen in the case of a single actor failure. However, in this architecture, a channel only requests to leave ground state if it is correct to do so—so an interlock that effected a change out of ground state if *any* actor requests it may tolerate more actor failure. Design of such an interlock is an area we leave for future investigation.

A limitation of this model is the assumption that the *set* commands received by a lamp assembly are synchronous and reliable. This assumption may be too strong if the control system driving the three lamps has failure modes where this is not the case. Investigation of a more asynchronous model of driving the channels is another item we leave for future investigation.

In conclusion, in this abstract we have demonstrated how a new design for an existing device may be developed using CSP and refinement laws for *Circus*, and the development steps verified using FDR. By designing the new architecture to use existing distributed consensus results, a more fault-tolerant system has been built.

References

- [1] A. L. C. Cavalcanti. *A Refinement Calculus for Z*. DPhil thesis, The University of Oxford, 1997.
- [2] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.
- [3] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International Series in Computer Science. Prentice-Hall, 1985.
- [4] C. A. R. Hoare and Jifeng He. *Unifying Theories of Programming*. Prentice-Hall Series in Computer Science. Prentice-Hall, 1998.
- [5] Formal Systems (Europe) Ltd. FDR: User manual and tutorial, version 2.28. Technical report, Formal Systems (Europe) Ltd., 1999.
- [6] Nancy A. Lynch. *Distributed Algorithms*. Morgan-Kaufmann, 1996.
- [7] Markus Montigel. Specification of the control processes for a Dwarf Signal. In *Proceedings of the 5th FMERail Workshop*, Toulouse, France, 1999.
- [8] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the Association for Computing Machinery*, 27, 1980.
- [9] A. W. Roscoe. *The theory and practice of concurrency*. Prentice Hall Series in Computer Science. Prentice Hall, 1998.
- [10] Andrew Clive Simpson. *Safety through security*. DPhil thesis, The University of Oxford, 1996.
- [11] J. C. P. Woodcock. Montigel's Dwarf, a treatment of the Dwarf Signal problem using CSP/FDR. In *Proceedings of the 5th FMERail Workshop*, Toulouse, France, 1999.
- [12] J. C. P. Woodcock and A. L. C. Cavalcanti. A concurrent language for refinement. In *5th Irish Workshop on Formal Methods*, 2001.
- [13] J. C. P. Woodcock and A. L. C. Cavalcanti. The semantics of *Circus*. In Didier Ber, Jonathan P. Bowen, Martin C. Henson, and Ken Robinson, editors, *Formal Specification and Development in Z and B*, pages 184–203. ZB 2002, Springer-Verlag, 2002.
- [14] J. C. P. Woodcock and Alistair A. McEwan. Verifying the safety of a railway signalling device. In H. Ehrig, B. J. Kramer, and A. Ertas, editors, *Proceedings of IDPT 2002*, volume 1. The 6th Biennial World Conference on Integrated Design and Process Technology, Society for Design and Process Science, 2002. Winner of the best paper award.
- [15] Jim Woodcock and Jim Davies. *Using Z: Specification, Refinement, and Proof*. International Series in Computer Science. Prentice-Hall, 1996.