Dependability of Computer-Based Systems

Cliff B Jones

Department of Computing Science,
University of Newcastle
NE1 7RU, UK
e-mail: cliff.jones@ncl.ac.uk

Abstract. This paper sets out a programme of work in the area of dependability. The research is to be pursued under the aegis of a six-year *Inter-Disciplinary Research Collaboration* funded by the UK Engineering and Physical Sciences Research Council. The research considers computer-based systems which comprise humans as well as hardware and software. The aim here is to indicate how formal methods ideas, coupled with structuring proposals, can help address a problem which clearly also requires social science input.

EXTENDED ABSTRACT

Reasoning about interference

This section summarises earlier work on formal development methods for concurrent systems.

The essence of concurrency is interference: shared-variable programs must be designed so as to tolerate state changes; communication-based concurrency shifts the interference to that from messages. One possible way of specifying interference is to use rely/guarantee-conditions (see [Jon83, Sti88, Stø90, Xu92, Col94, Din99]).

Programming language designers have proposed a series of increasingly sophisticated constructs to control interference; the case for using object-oriented constructs is set out in [Jon93].

Faults as interference

The essence of this section is to argue that faults can be viewed as interference in the same way that concurrent processes bring about changes beyond the control of the process whose specification and design are being considered. Without yet proposing notation for each case, a range of motivating examples are considered.

The first example is one that re-awakened this author's interest in considering faults as interference. Faced with the task of specifying a traffic light system, many computer scientists would confine themselves to the control system and specify the signals which must be emitted. Michael Jackson (see [Jac00]) considers the wider issues of the correct wiring of the control system to the physical

lights and the initial state of these lights units. One could widen the specification to address the overall light system (at one level the requirement is that at least one light must always be red) and record assumptions (as rely-conditions) which state that emitting a signal implies that the light unit changes state. Recording such a rely-condition does not itself result in a dependable system but it ensures that the assumptions are recorded and use of proof rules for development of concurrency should ensure that there are no further hidden assumptions. In fact, one could take this example further by specifying that the real requirement is to reduce the probability of a crash to a certain level and then record probabilities that drivers behave in certain ways when faced with red lights (see, for example, [MMS96] for ways of reasoning about probabilities in design).

A second -trivial- example should again illustrate the shift of view in documenting assumptions. Rather than specifying a control system in terms of the readings delivered by measuring devices, it might be preferable to specify the overall system in terms of the actual temperature etc. and provide a rely-condition which records the acceptable tolerance on the measuring device. Here again, the message is to expose the assumptions.

A more realistic example can be given in the same domain: it would be common for such sensors to be deployed using "triple modular redundancy". The viewpoint of recording the assumptions would suggest that a rely-condition should state that two roughly equal measurements are far less likely to be in error than one which is wildly different (or is perhaps some distinguished error value).

As well as the primary message that exposing assumptions will force their consideration, there is the clear advantage that checking that such rely-conditions are adequate to prove that a system will meet its overall specification will check for any missed assumptions.

Fault containment and recovery

Significant work has been done on designing architectures for fault-containment and recovery – see for example [Ran75, XRR+99].

Human errors and their containment

The work in the *Dependability Interdisciplinary Research Collaboration* on which we are embarking will address not just dependable computer systems but will also consider wider systems where the role of the humans involved is seen as critical to overall system dependability. The need for this is emphasized by [Mac94] which reports a large number of computer related accidents which resulted in death and notes that in the majority of cases the key problem related more to the interaction between people and computers than a specific hardware or software malfunction.

There are of course many examples of where a program tries to guard against inadvertent errors of its users: the check in many operating systems asking a

user to confirm the request to delete files or the need to retype a new password (being invisible there is no other check) are trivial instances. More interesting is the architecture of the overall system known as Pen& Pad [HRH+90] in which software is programmed to warn against possible misprescription of drugs by doctors: no attempt was made to automate prescription but the system would check against dangerous cocktails or specific drugs which might not be tolerable to some other condition that is indicated on the patient's record.

The logical extension of the work outlined in the two preceding sections on purely computer systems is to aim for a more systematic treatment of human errors. Fortunately the work of psychologists like Reason (see [Rea90]) in categorising human errors offers the hope of describing and reasoning about the sort of human errors against which a system is designed to guard. The objective would be to minimise the risk of the errors of the computer system and (groups of) humans "lining up" in the way indicated in [Rea97].

Further research

There are many further areas of research related to the themes above. For example:

- Both pre and rely-conditions can record assumptions but if they become complex they might be a warning that an interface has become too messy (cf. [CJ00]) – ways of evaluating interfaces and architectures are needed (see [SG96]).
- The idea of using rely-conditions to record failure assumptions occurred to the author in a connection with a control system some years ago. One reason for not describing the idea more publicly was that there often appears to be a mismatch of abstraction levels between the specification and the error inducing level. There needs to be more research on whether this can be avoided.
- The role of malicious attacks is being considered in the IST-funded MAFTIA project.
- A key area of system "misuse" is where the user has an incorrect model of what is going on inside the combined control/controlled system – minimizing this risk must be an objective.
- Progress in modelling the human mind (e.g. [CS94]) should be tracked.

Acknowledgements

There is of course related research and the work of Michael Harrison and his colleagues (who are within the IRC) and John Rushby (see [Rus99]) has influenced the thinking so far. One particular stimulus for this paper was the talk that Michael Jackson gave at the Munich meeting of IFIP's WG2.3 last year – more generally discussions at this working group on Programming Methodology have acted as a sounding board and encouragement to the author. This extended

abstract was published earlier in the proceedings of MPC'2000. Evolving details of the Dependability IRC can be found at www.dirc.org.uk.

References

[CJ00] Pierre Collette and Cliff B. Jones. Enhancing the tractability of rely/guarantee specifications in the development of interfering operations. In G. D. Plotkin, editor, *Proof, Language and Interaction*, chapter 10, pages 275–305. MIT Press, 2000.

[Col94] Pierre Collette. *Design of Compositional Proof Systems Based on Assumption-Commitment Specifications – Application to UNITY*. PhD thesis, Louvain-la-Neuve, June 1994.

[CS94] Patricia S Churchland and Terrance J Sejnowski. *The Computational Brain*. MIT Press, 1994.

[Din99] Jürgen Dingel. *Systematic Parallel Programming*. PhD thesis, Carnegie Mellon University, 1999.

[HRH+90] T J Howkins, A L Rector, C A Horan, A Nowlan, and A Wilson. An overview of PEN& PAD. *Lecture Notes in Medical Informatics*, 40:73–78, 1990.

[Jac00] Michael Jackson. *Problem Frames: Structring and Analysing Software Development Problems*. Addison-Wesley, 2000.

[Jon83] C. B. Jones. Specification and design of (parallel) programs. In *Proceedings of IFIP'83*, pages 321–332. North-Holland, 1983.

[Jon93] C. B. Jones. Constraining interference in an object-based design method. In M-C. Gaudel and J-P. Jouannaud, editors, *TAPSOFT'93*, volume 668 of *Lecture Notes in Computer Science*, pages 136–150. Springer-Verlag, 1993.

[Mac94] Donald MacKenzie. Computer-related accidental death: an empirical exploration. *Science and Public Policy*, 21:233–248, 1994.

[MMS96] Carroll Morgan, Annabelle McIver, and J W Sanders. Refinement-oriented probability for CSP. *Formal Aspects of Computing*, 8(6):617–647, 1996.

[Ran75] B. Randell. System structure for fault tolerance. *IEEE Transactionns on Software Engineering*, SE-1:220–232, 1975.

[Rea90] James Reason. *Human Error*. Cambridge University Press, 1990.

[Rea97] James Reason. *Managing the Risks of Organisational Accidents*. Ashgate Publishing Limited, 1997.

[Rus99] John Rushby. Using model checking to help discover mode confusions and other automation surprises. In *Proceedings of 3rd Workshop on Human Error*, pages 1–18. HESSD'99, 1999.

[SG96] Mary Shaw and David Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

[Sti88] C. Stirling. A generalisation of Owicki-Gries's Hoare logic for a concurrent while language. *TCS*, 58:347–359, 1988.

[Stø90] K. Stølen. *Development of Parallel Programs on Shared Data-Structures*. PhD thesis, Manchester University, 1990. available as UMCS-91-1-1.

[XRR+99] J. Xu, B. Randell, A. Romanovsky, R. J. Stroud, A. F. Zorzo, E. Canver, and F. von Henke. Rigorous development os a safety-critical system based on coordinated atomic actions. In *Proc. of 29th Int. Symp. Fault-Tollerant Computing*. IEEE Computer Society Press, 1999.

[Xu92] Qiwen Xu. *A Theory of State-based Parallel Programming*. PhD thesis, Oxford University, 1992.

This article was processed using the LaTeX macro package with LLNCS style