Specification before satisfaction: 
the case for research into obtaining the right specification

EXTENDED ABSTRACT

Cliff B Jones
University of Newcastle upon Tyne
Newcastle, NE1 7RU, UK
cliff.jones@ncl.ac.uk

Model-oriented specification techniques like VDM [Jon80,Jon90], Z [Hay93] 
and B [Abr96] have an enormous amount in common (cf. [Hay92,HIJ94]). Among 
other things that this formal methods community shares is the view that one 
can start with a formal specification and show that a design/implementation 
satisfies that specification. It is however obvious that, if a specification does not 
actually reflect the real need, proving a program correct with respect to it is 
somewhat pointless.

As computers have become more powerful and less expensive, they have be-come ever more deeply embedded in the way nearly everyone works. In their 
short sixty year history, computers have moved from batch processors in their 
own buildings to work tools on every desk (or lap); essential components of ad-
ministration, retail trade, banking and vehicles; and are on their way to becoming 
invisible dust sprinkled on who-knows-what. This, in itself, has changed the task 
of understanding the requirements of a system. Above all, the close interaction 
of people with computer systems makes it essential that designers consider the 
whole system when formulating a specification of the technical parts.

It is often easiest to make the point by looking at accidents. Donald MacKen-
ze in [Mac94,Mac01] has traced the cause of just over 1100 deaths where com-
puter systems appear to be implicated (up to 1994). Three percent of the lives 
lost appear to be attributed to bugs! Far more common causes of accidents ap-
pear to be where humans misunderstand what is going on in a control system or 
the object being controlled. This is a much deeper issue than the details of an 
interface; in many cases it is a fundamental question of the allocation of tasks 
between person and machine. Key questions include the visibility of the state 
of the system being controlled and the extent to which operations the user can 
perform are clumped together.

Although accidents are shocking and thus grab attention, there is also a 
significant penalty in the deployment of systems which make their users' lives 
more difficult than they need be. The enormous cost of systems which are so 
unsuitable that they are not even deployed is reported weekly in newspapers.

Of course, we should use formal specification techniques and we still need re-
search to make them more widely usable. But it would appear to be worthwhile
to see whether there is also a technical response to the question of how one arrives at a specification which does reflect the needs of the environment in which a system will be embedded. Does the formal methods community have a contribution to make here? I believe so. Dines Björner’s forthcoming books [Bjö05] tackle “domain modelling”. This paper sets out some further research challenges to which we might be able to offer useful responses.

This invited talk will review some suggestions which have arisen in the six year “Interdisciplinary Research Collaboration on Dependability” (DIRC) — see the WWW pages at [WWW04] for details. DIRC is focusing its research on how to design Dependable1 computer-based systems. The phrase “computer-based systems” is intended to emphasize that most computer systems today are deeply embedded into an environment which also involves people. For example, the requirement in a hospital is for dependability of the overall system. Sometimes, humans will use a computer system to achieve objectives even where they know that it delivers less than perfect information; on other occasions, computers can be programmed to warn when errors are made by humans. People are less good than computers at narrowly specified repetitive tasks but are much better at recognising and reacting to exceptional situations. To achieve overall system dependability, both humans and programs must be properly deployed.

Some insights from the DIRC project include:

- An approach being worked on with Ian Hayes and Michael Jackson [HJJ03] looks at determining the specification of, say, a control system by first specifying a wider system including the phenomena of the physical world which are to be influenced. To avoid having to build a model of the behaviour of all physical components, assumptions about their behaviour are recorded using rely conditions (cf. [Jon83]). This leaves a clear record of assumptions which need to be considered before the control system is deployed. Development from the derived specification of the control system is conducted in the standard (formal) way.

- The design of boundaries that limit the propagation of failures is better articulated for technical systems than for the human part of computer-based systems. This is odd because the intuition about limiting, say, accounting errors by auditors is long established. Many examples can be cited to suggest that most human systems are “debugged” rather than designed. The motivation for where to place containment boundaries ought come from an analysis of the frequency of minor faults and the the danger of their affecting a wider system. This analysis ought precede the allocation of tasks to computers which, in turn of course, must be done prior to their specifications being frozen.

---

1 The classic text on the terminology of dependability is [Lap92]; see also [Ran00]; an attempt to formalise the useful trichotomy between faults, errors and failures is given in [Jon03].
- A major cause of near or actual accidents is a “cognitive mismatch”\(^2\) between an operator’s view of what is going on and the actual state of affairs in the system they are trying to control. This was a significant factor in the “Three Mile Island” reactor incident. John Rushby [Rus99] has looked at pilot errors on the MD-88: in simulators, they frequently breach the required altitude ceiling. Rushby’s careful formal analysis builds a state model of the pilot’s understanding of the system and explores its interaction with a model of the aircraft systems. It would be informative to compare this approach with rely conditions.

- The general way in which processes (or procedures) are used in the human parts of computer-based systems is interesting. If one contrasts a traditional car production line with the depiction in the film “Apollo-13” of the search for a solution to the need to improvise CO\(_2\) scrubbers in the damaged capsule, one sees that processes both limit action and reduce the need for information. Designing processes which cope with all exceptions is in many cases impossible and one argument for relying on humans in computer-based systems is precisely that they notice when it is safer to violate a procedure than to slavishly follow one that does not cover an exceptional case. Clearly, either following an inappropriate process or deviating from a correct process can both lead to system failure. But it is absolutely mandatory that thought is given to processes in the design of a computer-based system. Interestingly, one can spot errors in legislation where an algorithmic rule is frozen into law: there have been several cases in financial legislation where a well-intentioned trigger has had (or nearly had) counter-productive effects.

- Within DfRC, the role of advisory systems has received particular attention: [SPA03] studies an image analysis prompter used in the analysis of mammograms. Surprising conclusions include statistically significant evidence that under the tested conditions the most accurate operators can offer less accurate conclusions with the help of the advisory system than without its use. It is clear that the role of such advisory systems has to be considered far more widely than just by looking at their technical specifications. In fact, even pure safety limiters (where one would believe they can only increase safety) have been used by operators in a way which supplants their normal judgment.

- Systems can create other things whose dependability is the goal. In the simplest case, a production line might manufacture silicon chips and faults in the manufacturing process might result in faulty components for computers. A software example is a compiler that, if faulty, could translate a perfect program into machine code which does not respect the formal semantics of the source language. In many cases, the creation process is human and, for example, a designer of a bridge which fails to withstand expected forces is

---

\(^2\) Both of James Reason’s books [Rea90,Rea97] look at relevant issues: the earlier reference looks at a division of the sort of errors that humans make; the second has insightful analyses of many system failures. Perrow in [Per99] talks of “Normal accidents”.

at fault. The creation of computer software is just such a process and is not always fault free! DIRC has provided an opportunity to look at Gerry Weinberg’s conjectures in [Wei71] that different psychological types might be more or less adept at different sub-tasks within the broad area known as programming. The implications of this research for building dependable systems might include steering people toward the tasks at which they are likely to perform best (and probably be most content).

- If the above list were not daunting enough (and it is far from complete even with respect to DIRC’s findings) there is another overriding concern. The sort of computer-based system we have been studying will always evolve. Designing a system which can be modified in reaction to a reasonable class of evolutions in the environment is extremely challenging. One class of system which has been studied within the DIRC project is generic systems. The justification of this sort of system is that it can be instantiated for a range of applications characterising this range is itself a technical problem. It is clear that issues around evolution will have a long-term impact on dependability.

There are related questions of how data survives such evolution which are equally challenging.

DIRC has identified far more than the above set of issues; the selection here has been based on the ease with which this one member of a project (involving more than fifty researchers) could pull together the information.

One key experience from the first three quarters of the project is the invaluable role of interdisciplinarity. Looking at experiments on psychological type and debugging performance required wholehearted collaboration of psychologists and computer scientists; tackling the mammography advisory system involved interaction between statisticians, sociologists and psychologists. DIRC could list many more examples of how our combination of psychologists, statisticians, sociologists and computer scientists has made real progress that no one of these disciplines could have accomplished.

My own disposition is to seek technical approaches to problems and I hope that the list above indicates that this is a viable challenge. But the DIRC project has been a superb example of collaboration and if faced with a complex application area, I would now know how to call on the expertise of other disciplines. In particular, the painstaking gathering of observational data needs sociologists.

We have learned two general things in the DIRC project which are worth passing on to others who might wish to follow such a wide interdisciplinary approach. Collaboration has to be based on respect for the disciplines of other researchers: values differ and publication strategies vary between disciplines but if it is good research by the standards of the other discipline one should not—for example—argue that it is not presented in the style of one’s own discipline. The other message is to tackle application problems together as a team. With an “Operations Research” (OR) like team representing several disciplines terminology problems disappear, contributions become understood and something is achieved which no single discipline could have envisaged.
Acknowledgments

My research acknowledgment is to the many colleagues involved in DIRC; it is a privilege to lead such an exciting project.

We are all grateful to EPSRC for the six year funding window which we feel was essential to foster such a wide interdisciplinary span.

References


