On the inevitable intertwining of Systems and Software Requirements

Alistair Sutcliffe
Department of Computation,
UMIST.

The above title as many will recognise is a paraphrase of the Swartout and Balzer (1983) paper that argued cogently that specifications and designs are inevitably linked by a process of developing a shared understanding between users and developers about what is required and what is technically possible. I shall argue that this message needs to be heeded in RE today, and the intertwining becomes even more pertinent when we scale up our ambition up to address systems requirements engineering in the INCOSE sense. Some take the position that system requirements are merely a logical extension of software requirements- see Anthony Hall's position paper. I beg to disagree, furthermore I shall argue that careful thought about system requirements should cause us to change our conception of software requirements.

1. Systems Engineering Perspective

Systems engineers in the INCOSE community don't tend to think in terms of requirements as neat lists that can be managed in DOORS or Requisite Pro. Instead requirements are expressed in models. This view is also held by Requirements Engineers who see Requirements Specification as one expression of what is required of the to-be-designed artefact. The processes of requirements acquisition, analysis and modelling are similar in INCOSE and RE; however, when in comes to validation opinions diverge. Systems engineers talk about validation in terms of testing a specification with a necessary and sufficient set of scenarios rather than verification or proof. The problem in systems is one of complexity and scale. A frequently espoused view by systems engineers is that one can not specify systems in infinite detail, and even if you could by a top down decomposition, there are emergent properties which manifest themselves at higher levels that can not be detected from lower level models. Hence a scenario-based approach is necessary. I am not sure emergent properties exist as many would hold; however, they do illustrate the problem of interaction between components in complex systems and how predictable such interactions might be.

2. Requirements Analysis problems

The concept that requirements can ever be correct was questioned year ago by Manny Lehman. The moving world problem is disturbing for the formal view of requirements as ' a machine whose specification is S to satisfy requirements R when placed in a world described by domain properties D, the sequent D, S _ R must hold.' For this statement to be useful we have to address the closed and moving world problems:

- The moving world problem states that users can only effectively agree requirements by a process of reification in design, hence the inevitable intertwining of specification and design. The moving world problem is compounded by change in the system environment so that requirements at t0 are not longer sufficient when implemented at t1.
- The closed world problem states that to create a specification to satisfy requirements assumes we can model the domain as a closed world and know all the variables that might influence the requirements.

We can only assert that the relationship holds if we know that the world described by the domain properties is finite and predictable. When we scale our endeavor from small software related control problems to large socio technical systems the fragility of our assumptions become clear.

First we can not know the complete set of domain properties in the world because some of those properties will relate to people, whose behaviour is probabilistic in nature. Secondly many optative requirements are owned by human stakeholders who are fallible and inconsistent human beings. Once we are outside the realm of small scale embedded systems the relationship inevitably becomes probabilistic in nature. Furthermore when we scale up to deal with large components as addressed by systems engineers, then the world in which the machine S must satisfy requirements R become increasingly complex and uncertain.

System requirements in complex socio technical system involved people. We are creating requirements for people who are the consumers of the eventual products of RE (i.e. designs). In Michael Jackson's lexicon these are optative requirements. However in systems engineering, people are embedded in the designed machine as part of a socio-techncial system solution, hence they are phenomena that we must account for in requirements models. The problem with people is that they exhibit all sorts of messy, unpredictable behaviour that the formal modellers wish to sweep under the carpet. Unfortunately to do so is to abrogate our responsibility as engineers.

So maybe we can deal with complexity by divide and conquer? Alas no, just consider the problem of specifying requirements for a navy destroyer for the later half of the 21st century. The complexity and uncertainty starts at the strategic level of defining the ship's mission of ship. Unfortunately, politics and changes in the world mean that the mission is at best an informed guess. At the tactical level the ship has to perform various operations. Some of the domain properties at this level may be known, the weather for instance is subject to the laws of physics and we can anticipate the effect of global warming so specify more stormy seas. However, the ship as a system is competing in an evolutionary race with other designers who are improving their ships to make them stealthier, faster, etc. Where do we pitch our guess about the requirements here? Ask a domain expert of course, but the answer will naturally be a guess. Below the tactical level there are the operational level of ship subsystems (propulsion, navigation etc) involving human operators, hardware and software. Defining requirements at this level encounters more complexity. Human operators are error prone, even with the rigours of military training. While requirements might be specified as defenses to prevent human errors, the problem is how to anticipate the space of possible mistakes in the first place. Can anyone anticipate all the future states such a system will encounter?

The systems engineering problem is therefore considerably more complex than software engineering, although both fall along a continuum of complexity and determinism. The state space of toy software engineering problems (e.g. cruise control) is tractable in terms of formal reasoning. Larger-scale software engineering problems that have regular behaviour and are governed ultimately by the laws of physics (e.g. telecommunications switches) are also amenable to formal reasoning. Once humans enter the picture we are dealing with probabilistic phenomena. The problem is no longer finitely bound because no model of the user (in a cognitive sense) is ever included in a requirements engineering specification, although there have been attempts to create such formal models see Duke and Barnard's treatment of the ICS cognitive model in Modal action logic.

3. An Approach to Systems Requirements Engineering

In complex systems the problem space becomes vastly inflated with combinations of variables that may possibly affect human behaviour. Furthermore design of the machine's interface form one of set of those variables, so we are faced with circular problem of specifying requirements for human behaviour that may then be changed by the design. Anticipating unsafe human actions of course has been the goal of safety critical systems design, where approaches are still heuristic rather than formal in nature (e.g. Levenson 1995). Research is necessary to create better methods for understanding the problem space of complex systems. Scenario based methods for requirements analysis with critical incident analysis (Sutcliffe 1997, 1999) and obstacle driven approaches (Potts et al 1994) are promising directions to follow.

The customers of complex systems (e.g. UK MoD, US DoD) have been embarrassed many times when large complex systems have failed to deliver even basic required functionality within the expected budget and timescale. One response has been to shift the 'burden of proof' for validating requirements onto the equipment supplier, while at the same time being less specific about the initial requirements. Some requirements are given in performance criteria, e.g. the ship will have a speed of 33 knots in normal sea states; however, many requirements are stated in the form of operational and tactical scenarios, e.g. the ship will be able to land 150 fully equipped troops on a coast without harbour facilities.

In the EPSRC SIMP (System Integration for Major Projects) we are grappling with the systems requirements problem of how to elicit or create a necessary and sufficient set of scenarios that are used to both indicate missing requirements from a system design and validate the system has the appropriate capability to deal with scenario events. Finding the appropriate set of scenarios is a rerun of the program test data generation problem, unfortunately at the systems engineering level the set of variables and events that may form potential inputs is vast. Systems requirements evolve both top town and bottom up as new designs inform higher level strategic/tactical capabilities while the customer's brief at the strategic level has to be expanded into more detail by lower level operational scenarios and designs. We have adopt heuristic approach to bound the search space by employing several techniques, ranging from analysing histories of critical incidents with similar systems, to probabilistic modelling using BBNs (Bayesian Belief Nets) to identify critical parameters for task and system performance and carry out sensitivity analyses on the impact of environmental variables on performance. The process is systematic but still relies on human knowledge for critical insights and problem solving. A key question is how much knowledge can be embedded in BBN nets for diagnostic assessment of system requirements.

4. Conclusions

I believe requirements in systems engineering are not different from requirements in interactive systems. RE and systems engineering both need to adopt a probabilistic approach to dealing with phenomena in the world which are non-deterministic. Since people are involved in most systems we can only design to improve effectiveness and dependability but we can not honestly guarantee that specifications will meet requirements that can not be formally documented One answer may be to create families of templates (or problem frames) at differing levels of abstraction. Those dealing with people will have to be probabilistic. I do not deny the role of formal requirements specification, this should be encouraged for deterministic components embedded in complex socio technical systems. RE and system engineering have both adopted scenario based approaches but research needs to address how these approaches can be made more effective so customer can have more confidence that a system will meet their requirements and behave as dependably as can be expected within the constraints of human knowledge of the problem, time, and development budget.