# Approaching the Formal Design and Development of Complex Systems: The Retrenchment Position

Richard Banach, Czeslaw Jeske, Simon Fraser, Richard Cross,
Michael Poppleton, Susan Stepney, and Steven King

Computer Science Dept., University of Manchester, Manchester, M13 9PL, UK
{banach,cjeske,sfraser,rcross}@cs.man.ac.uk
ECS, University of Southampton, Highfield, Southampton, SO17 1BJ, UK
mrp@ecs.soton.ac.uk
Computer Science Dept., University of York, Heslington, York, YO10 5DD, UK
{susan,steve.king}@cs.york.ac.uk

**Abstract.** The arguments for using formal techniques in the construction of complex systems are reviewed, and the refinement and retrenchment techniques in particular are summarised. Coarse grained retrenchment, with its capacity to analyse and express system properties at various levels of granularity, is promoted as a laboratory for understanding emergent behaviour of complex systems.

## 1 Introduction

Complex systems of many interacting parts with possibly explainable but unpredictable global behavior are an increasingly evident part of the technologically advanced world we live in these days. In the past, the fact that interaction between systems was via mechanical or analogue electrical couplings, meant that assembling truly vast communities of systems that interacted in nontrivial ways, was all but impossible. These days though, we have software, which can express arbitrarily complex interactions between systems with ease. Equally importantly, we have processor hardware on which to run the software, which has, in effect, negligible cost. All of this has meant that vast communities of interacting systems are not only conceivable, but actually arise in practice. See eg. [2] (and previous conferences in this series), [19], [16], [33], [1], [30].

Some such systems arise as a result of explicit and intentional design. The most obvious example is the international telecommunications system, which has evolved through the deliberate integration of many already large and relatively autonomous but much more functionally focused systems. Others, and the internet is the best known example, have arisen in a more or less ad hoc and unplanned manner, through the individual actions of many people, acting more or less independently. It is interesting that the more anarchic internet has arisen on top of the more disciplined but still not completely predictable international telecoms network.

To the extent that software is pervasive in such arenas, software platforms should be able to provide flexible, robust, adaptable and possibly evolvable architectures and methodologies for the development of such systems. There are a number of possible approaches to such development that one could consider. One could adopt one of the many existing informal software development techniques and attempt to model the evolving complex system using it. However the probable outcome of this is that the most important aspects of emerging complexity, are quite likely to be overlooked by the methodology during the development process. The reason is that due to the relatively informal approach, the details that ultimately give rise to the complexity, may well be relatively innocuous features of the system that the informal technique simply overlooks:- since informal techniques are not characterised by the unforgiving and unblinking eye that ensures completeness of coverage, in the way that formal techniques are.

The above observations, while conceding that informal techniques may lead to a 'product' more quickly, indicate that the use of informal techniques may nevertheless lead to increased costs further down the line, when unforseen but unacceptable aspects of emergent behaviour make their presence felt as a large system is assembled in anger. Of course such observations are not new as regards the inadequacies of informal techniques. There is by now plenty of evidence in the simpler arena of the development of stand-alone systems, that informal techniques alone do not routinely lead to the highest levels of dependability in the systems produced [13], [14].

So the argument leads inexorably to the espousal of formal techniques. These are founded on the idea that from a precise and complete mathematical model of the desired system, an implementation can ultimately be developed whose properties include the properties built into the mathematical description of the original model. Following the precepts of the formal mantra, the passage from the original model to the implementation is normally via some variety of refinement. The theory of the specific method being followed generates sound proof rules for refinement, that guarantee that if they are adhered to precisely, then the implementation that emerges will conform to the original model as required.

That at least is the ideal story. Already though, in the world of stand-alone systems, there is plenty of evidence that following the precepts of some particular refinement methodology with complete precision can be a daunting challenge [32], [23], [15], [8]. Indeed one of the authors has had extensive experience of industrial scale refinement projects, and knows at first hand the problems that rigorous insistence on refinement can bring [31]. Many aspects of real life development conspire to make the adoption of a naive refinement methodology often run ultimately into the sand. It is often for example the case that the mathematical models used near the top of the design hierarchy are of a different character than the ones used further down. For example, the former may be expressed using continuous mathematics and the latter using discrete mathematics. It turns out that for more or less all cases of interest, a sensible refinement cannot be established between relevant models in these two domains: the refinement goal is simply unachievable. For another example, the sheer complexity of real life applications may make unswerving loyalty to the exacting demands of the refinement methodology used, prohibitively expensive on the one hand, and frustratingly obfuscating on the other. The frustrating obfuscation can arise from the way logic treats all facts, whether major structural features or minor details, on the same footing, making the manipulation of a system description rise steeply with the size of the description itself, often at an exponential rate. This in turn rapidly leads to the prohibitive expense.

As a response to these phenomenological observations on refinement in practice, and recognising completely the internal solidity of refinement in theory, retrenchment was introduced to allow high level abstractions of system behaviour to evolve. The generality of the retrenchment formalism means that many kinds of system evolution can be contemplated. The most obvious kind is the evolution of high level abstractions towards low level implementations; this is an aim clearly intended to complement (we emphasise *to complement*, not to replace) the original remit of refinement. Another kind of evolution is the evolution of high level abstract models in response to changes in requirements, which (because they arise from real world considerations) are not a priori constrained or guaranteed to fit the refinement technical game plan. Yet another is the evolution of partial but internally consistent component submodels into into a larger complete design, where designing the component submodels in such a way that they composed seamlessly and without change into the larger whole, would make them internally incomplete and/or unconvincing.

Now we outline the rest of the paper. We review the refinement and retrenchment techniques in the next section. Then we introduce the retrenchment simulation relation in the one that follows. The section after that introduces coarse grained retrenchment, with its ability to give descriptions of the relationship between two systems at more than one level of granularity, and it is argued that this provides a good environment in which to explore emergent properties of complex systems.

## 2 Refinement and Retrenchment

Retrenchment may be viewed as a more generously parameterised variation of refinement. See [34], [17], [18] for treatments of model oriented refinement in general and [20], [26], [29], [35], for more details of the Z methodology, or [3], [27], [27] [28], for more details on B; both Z and B being specific incarnations of the model oriented approach to refinement. In particular retrenchment is defined by a more generously parameterised variation of the archetypical refinement proof obligation. Let us see this in a little more detail now. Refinement, in its most frequently encountered model oriented forward simulation form, relies on the following proof obligation (PO):

$$\begin{aligned} & G(u,v) \land Op_{\mathrm{C}}(v,j,v',p) \\ & \Rightarrow\ \ (\exists\, u',i,o \bullet Op_{\mathrm{A}}(u,i,u',o) \land G(u',v') \land (i=j) \land (o=p)) \end{aligned}$$

In this $G(u, v)$ is the glueing or retrieve relation, which relates abstract state values $u$ to concrete state values $v$.[1] $Op_{\mathrm{C}}(v, j, v', p)$ is the relation that describes a particular operation $Op$ at the concrete level, where $(v, j, v', p)$ is the tuple of before-state, input value, after-state, output value, for the transition. With the hypotheses of $G$ and $Op_{\mathrm{C}}$ we infer that there is an abstract after-state, input value, output value, such that the operation $Op$ at the abstract level has the transition $Op_{\mathrm{A}}(u, i, u', o)$ and that furthermore $G(u', v')$ and $(i = j)$ and $(o = p)$ hold.[2] The reestablishment of $G$ for the after-states enables an induction to be set up that says that a concrete execution sequence can be simulated by an abstract one. If $G$ is the vehicle by which the concrete states are interpreted to correspond to abstract ones, a user of the abstract system may indeed be fooled into thinking that he is using the abstract system when it is in fact the concrete system doing the work.

Unfortunately, the PO above turns out to be extremely demanding when confronted with many real world scenarios. Often it is extremely hard to find a $G$ such that the *naturally occuring* operation models at abstract and concrete levels are actually related by it. The most extreme such examples arise when the abstract world is described using continuous mathematics and the concrete world is described using discrete mathematics. Doubly unfortunately, this is precisely the world of many safety critical developments, where the target discrete code has to control or influence continuous physical apparatus, and the assurance that formal techniques can offer is sorely missed.

In many discrete real world cases also, the sheer complexity of the real world situation prohibits its complete modelling to the satisfaction of the refinement PO, a situation that leads to various compromises, whether in the modelling itself or in the degree to which the PO is faithfully discharged. In extremis, even if one strives hard and succeeds in constructing a refinement that rigorously conforms to the refinement PO, and such that at least one of the models would be of interest from an engineering vantage point, it is often the case that the other model is not relevant from an engineering vantage point, and thus the whole refinement activity, while perhaps of some interest as an academic exercise, fails to be of any real value from an engineering point of view, (other than that of having provided a demanding intellectual challenge for the humans involved, which will undoubtedly deepen their understanding of the system, whatever the appropriateness of the end result).

Retrenchment arose as a response to this state of affairs. See [7], [8], [9], [10], [11], [12] for the basic framework. The idea is to weaken the specific structure of the PO, in order to allow it to express relationships that actually hold between

[1] The 'abstract' and 'concrete' terminology is standard, if slightly inappropriate in contexts where the main goal is other than to move directly towards runnable code, such as indicated for retrenchment in the preceding paragraph. 'Source' and 'target' systems might have been better nomenclature, but we will stick with conventional usage here.

[2] There is also an initialisation PO that says that for each concrete initial state there is an abstract intial state related to it by $G$, but since this proof obligation remains unchanged in retrenchment, we will not discuss it further here.

the abstract and concrete models in practice, rather than have it remain a structure that though appealing, does not have content (or has too exacting content) in many practical cases. The retrenchment PO corresponding to the above is:

$$\begin{aligned}&G(u,v) \land P_{Op}(i,j,u,v) \land Op_C(v,j,v',p)\\ &\Rightarrow\ (\exists\ u',o \bullet Op_A(u,i,u',o) \land ((G(u',v') \land O_{Op}(o,p;u',v',u,v,i,j))\\ &\qquad \lor\ C_{Op}(u',v',o,p;u,v,i,j)))\end{aligned}$$

Here we see ingredients familiar from the refinement PO, but the whole statement is littered with a number of escape clauses which leaven the rigidity with which it might be applied. Thus the antecedent is polluted by the within relation $P_{Op}(i,j,u,v)$ which limits the scope to which the relationship between the models needs to be established; it also permits the inclusion of changes of a general sort between inputs and before-state values should these prove to be convenient in the evolution from abstract to concrete model. The consequent of the PO features a corresponding output relation $O_{Op}(o,p;u',v',u,v,i,j)$, which is required to hold in those fortunate cases when the retrieve relation $G$ can be reestablished for the after-state values, and which not only describes how the output values are related, but also features all the other variables occurring in the formula, allowing *any other property of interest* to be mentioned when the two systems' behaviour falls into this benign scenario.

Most dramatically however, the consequent of the PO features a disjunction, at the top level, with the concedes relation $C_{Op}(u',v',o,p;u,v,i,j)$. Now, whereas slightly strengthening the antecedent and the $G(u',v')$ part of the consequent, are both relatively mild modifications of the PO and most of existing refinement theory could be carried through for such modifications without the theory suffering extensive damage, the introduction of the disjunction has a devastating effect on the theory. The behaviour of the two systems can now depart from the ideal $G(u',v')$ in the after-states in an arbitrary way. Certainly any induction that relies on the reestablishment of the PO for the after-states collapses immediately.

The concedes relation is the most characteristic feature of retrenchment. It is the thing that differentiates retrenchment most markedly from any form of refinement to be found in the literature. It also makes retrenchment supremely expressive as a relationship between abstract and concrete models at the single step level, since the occurrence of all the variables of the system in the concedes relation, allows any relationship whatsoever to be incorporated in it, just as for the output relation, but (unlike for the output relation) this time in a completely unconstrained manner. There is no $G'$ (or anything else for that matter) to limit when $C$ may or may not apply.

This can be seen as a weakness of the theoretical framework, but it is better understood as a transfer of responsibility from the theoretical framework to the designers themselves. Since, for dealing with the kind of examples referred to above, something new and different from refinement was required, rather than risk limiting the applicability of any new framework by imposing ad hoc constraints that were difficult to justify a priori, and that might prove to be an obstacle with the wisdom of hindsight, the most general possible statement was chosen, that defaulted to the refinement PO when its additional components were suitably constrained. It is clear that the retrenchment PO does indeed reduce to the refinement PO when $P$ and $O$ are suitable identities and $C$ is $\mathsf{false}$.

Of course there is the opposite default. If $P$ is $\mathsf{false}$ and/or $C$ is $\mathsf{true}$, then the statement holds vacuously, which is often viewed as a serious shortcoming of retrenchment. But in fact it is no more of a problem, than the observation that one can assemble an arbitrary sequence of elements from the English lexicon and thereby obtain rubbish. The distiction is like the one between mathematical logic and system development.

In mathematical logic one draws a rigid boundary between what one is allowed to discuss and what one is not allowed to discuss. The former is delineated using a formal laguage, and the latter is simply not expressible. In such an environment, ad extremis, just about the only question one is permitted to consider about some procedure that pertains to the formal system, is whether it is trivial or not; i.e. whether, being trivial, all statements of a suitable form in the formal system can be related to each other using the procedure (or perhaps none can), or whether, being nontrivial, only some proper subset of them can be thus related. However formal system development is not a branch of mathematical logic, it is an application of it. In such an environment, there are many issues beyond those inherent in the formal system to consider, and these influence and often override the purely formal considerations. Thus the fact that retrenchment is capable of making vacuous statements about pairs of systems should not be viewed as any more of a weakness than the fact that the English language is capable of enunciating nonsensical utterances if the speaker so chooses. In both system development and English speech, one embarks on the activity with some purpose in mind, and not at random. In formal system development one wants to end up with a system as close as possible to the requirements one is aware of, and one wants to describe the activity of constructing that system in the most precise terms possible. In English speech one has some intention or information to impart before one opens one's mouth.

The interplay between the formal structures and the meta- or environmental considerations is obviously quite different with retrenchment as compared with refinement. In retrenchment one has a much more intimate interplay between the formal evolution step as described through the retrenchment relations, $G$, $P$, $O$, $C$, than is the case in refinement. The flexibility of retrenchment means that much more thought has to go into the choice of these relations than into the corresponding choice of just $G$ in refinement. This is no bad thing on reflection. Ultimately human beings have to take responsibility for designs. They cannot blame the formal systems they used for the errors they made. In refinement, there tends to be a much sharper boundary between the human design decisions that go into the abstract model, and the formal development that can then take the implementation forward, often in a close-to-black-box manner because of the strength of the guarantees offered by refinement. There are no such guarantees with retrenchment, so human beings must be much more vigilant in its use. However even though retrenchment does not responsibly permit a black box approach, it still demands that its PO be provable, and the proving activity injects valuable feedback into the design process. Thus the injection of mathematical discipline into what was hitherto an exclusively human process taking place beyond the confines of where refinement might enforce its discipline, can only be a good thing.

## 3 The Retrenchment Simulation Relation

Allied to the implicational proof obligation of retrenchment is its conjunctive counterpart, the retrenchment simulation relation. This is given by:

$$\begin{aligned}&G(u,v) \land P_{Op}(i,j,u,v) \land Op_C(v,j,v',p) \land Op_A(u,i,u',o) \land \\ &((G(u',v') \land O_{Op}(o,p;u',v',u,v,i,j)) \lor C_{Op}(u',v',o,p;u,v,i,j))\end{aligned}$$

This contains all the information relating a pair of nontrivial abstract and concrete steps that the PO speaks about. Unlike the corresponding refinement simulation relation:

$$G(u,v) \land (i=j) \land Op_C(v,j,v',p) \land Op_A(u,i,u',o) \land G(u',v') \land (o=p)$$

which, because of the paucity of data in $G$, merely says that ‘these two steps are in simulation’, the retrenchment simulation relation potentially contains a wealth of all sorts of data regarding the two systems, because, to put it simply, there are many more containers (i.e. $P$, $O$, $C$) to put such data into. The fact that $P$, $O$, $C$, occur only linearly in the PO is a boon here, since one does not have to struggle to ensure that various occurrences of any of them at different places in the PO inadvertently threaten the overall provability of the PO, as might happen if $P$, $O$, $C$, were to occur in multiple places. These remarks apply with much less force to $G$ since the relationship between the state spaces of the two systems is usually well understood and offers little choice.[3]

The data in the simulation relation may be tailored and exploited in many ways. An example concerns the mechanical extraction of fault trees from a retrenchment relationship with purposely designed $P$, $O$, $C$, relations [6]. Another is the general treatment of the translation of system properties (as expressed via ‘sets of traces’ of a certain kind) through the retrenchment process [4]. Unlike the refinement case, the retrenchment simulation relation is a partial relation on traces in more senses than one. Not only does it not need to hold for all (abstract or concrete) traces, it need not hold on the whole of any such trace. These aspects introduce an element of novelty not present in refinement.

Yet another area where the simulation relation plays a key role is in the algebraic integration of retrenchment and refinement [5], [21], [22]. It has been

[3] The nonlinearity in $G$ of the refinement PO, with $G$ required to be present in both the antecedent and consequent, and with no extra elbow room ‘to take up the slack’, is largely responsible for the much more highly constrained possibilities that are characteristic of refinement.

argued above that retrenchment and refinement offer differing yet complementary technical tools for the formal development of systems. To gain maximum benefit from them, it is therefore important to establish how they can interwork in a mutually supportive manner. A wealth of questions may be asked regarding such interworking, and the corresponding theory frequently relies, directly or indirectly, on the properties of the retrenchment simulation relation.

## 4 Coarse Grained Retrenchments and Complex Systems

The above discussion has focused exclusively on the retrenchment between individual steps of corresponding operations at two adjacent levels of abstraction, but while that environment has provided a suitable starting point for the early investigation and exploitation of the potential of retrenchment, it barely scratches the surface of what the technique might prove itself useful for if cast into a wider context, especially so as regards complex systems.

The big payoff for retrenchment will be in the exploitation of coarse grained retrenchments, that is, retrenchments between *collections* of atomic steps of abstract and concrete models. Given the nature of retrenchment that we have seen earlier, what will such a coarse grained retrenchment look like? Single step retrenchment is a very partial notion, as both $G$ and $P$ have to hold before a retrenchment relationship can be asserted. In the case that one wishes to relate collections of atomic steps via a retrenchment relation, some design decisions need to be taken in order to tailor the notion of coarse grained retrenchment arrived at, so that it meets application needs, applies to the widest possible set of scenarios, and fits smoothly with the single step retrenchment theory.

It turns out that an event structure formulation works best. At a given level of abstraction, the collections of atomic steps of interest are packaged up into suitable events in a prime event structure. This choice gives the following possibilities:

- The possibility of abstracting away from global state descriptions to local state ones.
- The possibility of making the theory equally applicable to distributed and single processor environments.
- The possibility of abstracting away from the detailed interleavings of independent actions of concurrent processes.
- The possibility of abstracting away from the scheduling mechanism that schedules the constituent steps of the event structure.
- The possibility of exploiting elementary conflict relations to most expressively describe the variety of outcomes of the event structure as a whole (whether ‘good’ and reestablishing $G' \wedge O$, or ‘bad’ and merely establishing $C$).
- The possibility of replacing the standard notion of conflict by some more sophisticated notion better suited to the case at hand.

How does retrenchment work between such entities? We just summarise the essentials in order to avoid the somewhat detailed notations which are needed to express things properly. Essentially the prime event structures used are finite, and thus have a collection of root and leaf nodes for the causality partial order. The values of the local variables at the root nodes must be consistent (according to event structure semantics) and any extension of these to a global state value defines a state value to which $G$ and $P$ and the before-values of the constituent concrete steps must apply. Likewise the leaf nodes give definitions for the after-values to be used in the $((G' \wedge O) \vee C)$ part, except that there is a complication arising from the incompatible flows through the event structure that ensue when there is a nontrivial conflict relation. Consistency must be maintained so that the same set of local variables can be deemed to have received a value via any execution of the event structure. By these means, retrenchments between event structures can be set up. Executions of the abstract and concrete system exhibit *deployments* of retrenchments between such event structures if suitable portions of the executions, enjoying appropriate atomicity properties, correspond to executions of the event structures themselves.

The framework just sketched offers a plethora of possibilities for describing the structure of complex systems. Firstly there is the interplay between fine grained and coarse grained descriptions of an individual system. Presumably the system endowed with such a pair of descriptions has particular properties that bear emphasis at one or other grain of description, and these can therefore be analysed in the interplay between the two descriptions. Then there is the possibility, nay likelihood, that one can formulate retrenchments between the two systems at both fine grained and coarse grained levels. What then is the relationship between these two kinds of retrenchment? To answer the question one must examine how retrenchment relationships compose. Compositionality of retrenchments is in general an extremely thorny issue. This is hardly surprising given how flexible the intrinsic retrenchment notion is. However, in reasonably benign circumstances, retrenchments can be composed to give reasonable results. In cases of interest, one can calculate a composed retrenchment from the individual retrenchments relating the component operations [6]. One can then relate the calculated coarse grained retrenchment to the retrenchment given for the coarse grained description posited ab initio. The converse problem also presents itself, that of decomposition [24], [25]. Given a coarse grained retrenchment posited ab initio for a coarse grained description of two systems, one can look to decompose it to give finer grained retrenchments for the components, and then compare the calculated retrenchments to the retrenchments given a priori for the individual fine grained components. In this manner one can gain a more deep appreciation of the properties possessed by the two systems as a whole. Normally one would expect the properties of the composition and decomposition approaches to be complementary, for example they could be adjoint to one another.

In general there are of course many details regarding the two systems being examined, that can be included in the data of a retrenchment. One of the challenges in making the best use of retrenchment is to decide what data to put into the $P$, $O$, $C$, relations, an issue substantially affected by the use to which one wishes to put the retrenchment. In the case of complex systems, the most desirable goal is to explain, and even better to predict, the complex system's emergent properties. The retrenchment framework furnishes a fertile infrastructure for such work. In the ideal case, the structures provided by retrenchment will allow the desired emergent properties to be smoothly extracted.

To be honest, actually predicting novel emergent behaviour is likely to be a tall order. Emergent properties are so called, precisely because their appearance on the scene is largely unexpected, or at least not easily predictable from the more selfevident properties of the component systems. Retrenchment though, provides a sensible laboratory in which to explore this landscape, even if it is to the more limited extent of trying to analyse observed emergent behaviour, in order to understand it on the basis of the properties of the component systems. One can start with a much simplified system; simplified in terms of utilising simpler components, or of simplified interactions between them, or both. If the starting point is simple enough, the system's behaviour should be amenable to analysis and be understandable in a fairly complete way. One can then gradually introduce complexity, each such increment being captured within an appropriate retrenchment. Reanalysis of the incremented systems in terms of the new properties introduced, can (ideally) predict emergent behaviour, or (realistically) provide the basis of a post hoc analysis. Done right, the series of retenchments involved in building up the system, will have structured the introduced complexities in such a way that the cause of unexpected and emergent behaviours will be easier to ascertain.

## 5 Conclusions

In the preceding sections, we have reviewed the arguments that lead to the espousal of formal techniques in system construction, and reexamined refinement and retrenchment in particular. Some of the methodological contrasts between refinement and retrenchment were reviewed, and this led to an outline of the richer world of coarse grained retrenchment, and the highly structured environment for analysis of system behaviour that it in turn led to. This gave rise to the suggestion that building up a complex system from simple constituents, with the increments in complexity being captured via retrenchments, was a viable route to a clearer understanding of emergent behaviour.

## References

1. *Developing Quality Complex Database Systems: Practices, Techniques and Technologies.* Idea Group Inc., 2001.
2. *Proc. IEEE Conference on Engineering of Complex Computer Systems.* 2004.
3. J R Abrial. *The B-Book: Assigning Programs to Meanings.* Cambridge University Press, 1996.
4. R Banach. Retrenchment and system properties. Submitted.

5. R Banach. Maximally abstract retrenchments. *Proc. IEEE ICFEM-00*, pages 133–142, 2000.
6. R Banach and R Cross. Safety requirements and fault trees using retrenchment. 2004. Submitted.
7. R Banach and C Jeske. Output retrenchments, defaults, stronger compositions, feature engineering. Submitted.
8. R Banach and M Poppleton. Engineering and theoretical underpinnings of retrenchment. Submitted.
9. R Banach and M Poppleton. Retrenchment: An engineering variation on refinement. *B'98: Recent Advances in the Development and Use of the B Method: Second International B Conference, Montpellier, France, LNCS*, 1393:129–147, 1998.
10. R Banach and M Poppleton. Retrenchment and punctured simulation. *Proc. IFM-99, Springer*, Araki, Gallway, Taguchi (eds.):457–476, 1999.
11. R Banach and M Poppleton. Sharp retrenchment, modulated refinement and punctured simulation. *Form. Asp. Comp.*, 11:498–540, 1999.
12. R Banach and M Poppleton. Retrenching partial requirements into system definitions: A simple feature interaction case study. *Requirements Engineering Journal*, 8:266–288, 2003.
13. J Bowen and M Hinchey. *High-Integrity System Specification and Design.* Springer-Verlag, 1999.
14. J Bowen and M Hinchey. *Industrial-Strength Formal Methods in Practice.* Springer-Verlag, 1999.
15. J P Bowen and S Stavridou. Formal methods and software safety. In H. H. Frey, editor, *Safety of Computer Control Systems (SAFECOMP)*, pages 93–98. Pergamon Press, October 1992. Proc. IFAC Symposium, Zurich, Switzerland.
16. M C Calzarossa and S Tucci. *Performance Evaluation of Complex Systems.* Springer-Verlag, 2002.
17. W P de Roever and K Engelhardt. *Data Refinement Model-Oriented Proof methods and their Comparison.* Cambridge University Press, 1998.
18. J Derrick and E Boiten. *Refinement in Z and Object-Z: Foundations and Advanced Applications.* Springer-Verlag UK, 2001.
19. D G Green and T Bossomaier. *Complex Systems: from Biology to Computation.* IOS Press, 1993.
20. J Jacky. *The Way of Z.* Cambridge University Press, 1997.
21. C Jeske and R Banach. Minimally and maximally abstract retrenchments. *Proc. IFM-02, LNCS*, 2335:380–399, 2002.
22. C Jeske and R Banach. Reconciling retrenchments and refinements. 2004. Submitted.
23. S Liu and R Adams. Limitations of formal methods and an approach to improvement. *Proc. 1995 Asia-Pacific Software Engineering Conference (APSEC'95), IEEE Computer Society Press, Brisbane, Australia*, pages 498–507, December 1995.
24. M Poppleton and R Banach. Structuring retrenchments in B by decomposition. *International Symposium of Formal Methods Europe (FME 2003), Pisa, Italy, LNCS*, 2805:814–833, September 2003.
25. M Poppleton and R Banach. Requirements validation by lifting retrenchments in B. In *Proc. 9th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS-04), Florence, Italy.* IEEE Computer Society Press, 2004. to appear.
26. B Potter, J Sinclair, and D Till. *An Introduction to Formal Specification and Z.* Prentice Hall, second edition, 1996.

27. S Schneider. *The B-Method: An Introduction.* PALGRAVE, 2001.
28. E Sekerinski and K Sere. *Program Development by Refinement: Case Studies Using the B-Method.* Springer, 1998.
29. J M Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall, 1989.
30. Susan Stepney. Critical critical systems. In A Abdallah, P Ryan, and S Schenider, editors, *Formal Aspects of Security: FASec 2002, Royal Holloway, London*, pages 62–70.
31. Susan Stepney and David Cooper. Formal methods for industrial products. In J P Bowen, S Dunne, A Galloway, and S King, editors, *ZB2000: First International Conference of B and Z Users, York, UK, August 2000*, volume 1878 of *LNCS*, pages 374–393. Springer, 2000.
32. D Stidolph and J Whitehead. Managerial issues for the consideration and use of formal methods. *Proc. FME-03, LNCS*, 2805:170–186, 2003.
33. J A Wise, V D Hopkin, and P Stager. *Verification and Validation of Complex Systems.* Springer-Verlag, 1992.
34. J Woodcock and J Davies. *Using Z, Specification, Refinement and Proof.* Prentice Hall, 1996.
35. J C P Woodcock and C C Morgan. Refinement of state-based concurrent systems. *Formal Methods in Software Development, LNCS*, 428, 1990.