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

R. Banach, C. Jeske, S. Fraser, R. Cross
Computer Science Dept., University of Manchester, UK

M. Poppleton
ECS, University of Southampton, UK

S. Stepney, S. King
Computer Science Dept., University of York, UK
Contents:

1. Informal and formal development.
2. Model oriented refinement and some issues.
3. Retrenchment.
4. Retrenchment and complex systems.
1. Informal and formal development.

Complex systems are increasingly ubiquitous.

- The usual well-known examples from the ‘hard sciences’.
- Examples from biology.
- Increasingly, human engineered systems:
  - Telecoms network
  - The internet
- Understanding such systems is hard.
- With human engineered systems, there is the possibility of engineering the complexity incrementally, leading to better understanding of emergent behaviour.
- Informal techniques are ‘quicker’ but usually prone to imprecision.
- Formal techniques hold out the hope of better understanding, but are more challenging.
2. Model oriented refinement and some issues.

Much of formal development is built around refinement.

Refinement is a word used to mean a large number of different things in different contexts.

In model oriented refinement, we build models of the system, by specifying:

- the state (and I/O) space of a model
- the operations (or events) of a model: via eg.
  - transition systems,
  - programming notations,
  - predicate transformers,
  - etc.

Models can then be related pairwise by REFINEMENT. This usually involves a notion of substitutivity, of some concrete system behaviours for some abstract system behaviours, intended to help move closer to an implementation, and leading to sufficient conditions for refinement.
Refinement can work wonderfully well in certain circumstances. Its paradigm of building models of the system at various levels of abstraction fits very naturally with the desire of designers to manipulate ‘solid’ representations of the system.

However, there are aspects of the real world design activity that can get into tension with refinement.
3. Retrenchment.

The ferocity of the refinement POs is what restricts their application in many areas. What can we do about the ferocity of the refinement POs? We can attempt to judiciously weaken them.

This of course would have consequences ... Refinement is derived from the prior assumption of substitutivity of concrete for abstract. Any interference with the refinement POs can fatally wound this link with substitutivity properties.

We are prepared to forgo this highly desirable aspect for the sake of:

- Being able to address more (and more of) applications contexts formally.
- Being able to live with real world and management constraints.
The refinement PO

\[ G(u,v) \land (i = j) \land Op_A(u,i,u',o) \Rightarrow (\exists u',o \cdot Op_A(u,i,u',o) \land G(u',v') \land (o = p)) \]
The retrenchment PO

\[
G(u, v) \land P_{Op}(i, j, u, v) \quad \Rightarrow \\
(\exists u', o \cdot Op_A(u, i, u', o) \land ((G(u', v') \land O_{Op}(o, p; u', v', u, v, i, j)) \\
\lor C_{Op}(u', v', o, p; u, v, i, j)))
\]
Definition of a retrenchment between Abs and Conc

A retrenchment is defined by the following data:

\[ \text{Ops}_A \subseteq \text{Ops}_C \]  
(the inclusion of operation names can be proper)

For each abstract operation name a relation  
\[ O_{P_A} : U \times I_{O_{P_A}} \leftrightarrow U \times O_{O_{P_A}} \]

For each concrete operation name a relation  
\[ O_{P_C} : V \times J_{O_{P_C}} \leftrightarrow V \times P_{O_{P_C}} \]

\[ G(u,v) \]  
(the retrieve (or glueing) relation)

\[ P_{Op}(i,j,u,v) \]  
(the within (or provided) relation)

\[ O_{Op}(o,p;u',v',u,v,i,j) \]  
(the output relation)

\[ C_{Op}(u',v',o,p;u,v,i,j) \]  
(the concedes relation)

\[ \text{Initialisation PO:} \quad \text{Init}_{C}(v') \Rightarrow (\exists u' \cdot \text{Init}_{A}(u') \land (G(u',v'))) \]  
(as for refinement)

\[ \text{Operation PO:} \quad (\text{per } Op \in \text{Ops}_A) \]
\[ G(u,v) \land P_{Op}(i,j,u,v) \land O_{P_C}(v,j,v',p) \Rightarrow \]
\[ (\exists u',o \cdot O_{P_A}(u,i,u',o) \land ((G(u',v') \land O_{Op}(o,p;u',v',u,v,i,j)) \]
\[ \lor C_{Op}(u',v',o,p;u,v,i,j))) \]
### Refinement and retrenchment compared

<table>
<thead>
<tr>
<th>Refinement</th>
<th>Retrenchment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forces designer to discharge PO; PO is highly constrained; in particular it is nonlinear in $G$.</td>
<td>Forces designer to discharge PO; PO is very liberal; in particular, despite being nonlinear in $G$, it is linear in $P_{Op}, O_{Op}, C_{Op}$.</td>
</tr>
<tr>
<td>Refinement is a limit of retrenchment, and so has a stronger theory than retrenchment.</td>
<td>Retrenchment is a generalisation of refinement, and so has wider applicability than refinement.</td>
</tr>
<tr>
<td>Fewer situations are describable using refinement.</td>
<td>Retrenchment has a weaker theory than refinement.</td>
</tr>
</tbody>
</table>
4. Retrenchment and complex systems.

Straightforward one-step retrenchment will not do that much to conquer the complex systems problem space.

One-step retrenchment has design flexibility but is limited in scope.

Coarse grained retrenchment offers the promise of enlarging the scope.
Coarse grained retrenchment

When a collection of individual steps is to be regarded as a whole, in effect the only issue is atomicity. (Without atomicity, what’s the point?)

In coarse grained retrenchment atomicity is assumed.

In coarse grained retrenchment a partial states picture is adopted, the syntactic role is taken by relevant collections of individual steps which are encapsulated in prime event structures (ESs) $E = (E, \leq, \#)$.

Variable values at root and leaf events of $E$ extend to the before- and after- values of the global state.
An execution structure is the corresponding (conflict free) runtime notion. It yields an execution fragment via serialisation. Fragments thus yield deployments of ESs.

Variable values at root and leaf events of $E$ extend to before- and after- values of the global state.

Assumption: No external access here to variables utilised in $E$.

These global state values appear in the retrenchment PO for corresponding abstract and concrete ESs.
Interaction of coarse grained retrenchment with composition/decomposition mechanisms yields a fertile infrastructure for sophisticated analyses of system behaviour.
Complex Systems

• Identify basic components of the complex system.
• Are they already too complicated to understand properly? If so simplify them.
• Determine the assembly of basic components to make up the global system, describe it using sequential/parallel/dataflow composition to get top level operations of the global system.
• Capture the incorporation of more of the properties of the basic components into the global model using successive coarse grained retrenchments. Can emerging properties be identified in the retrenchment analysis?
• Compare calculated retrenchment for the composed system with any ad hoc retrenchment written for it. Compare calculated retrenchments for the decomposed components with ad hoc retrenchments describing increments of basic component properties.