Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

Programming dependable systems (slides)

Owen Rees

Abstract

Slides for a presentation to the ANSA TC.

This presentation is a snapshot of the work being documented in APM.1122.

APM.1155.00.03 **Draft** 24 February 1994

Request for Comments (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

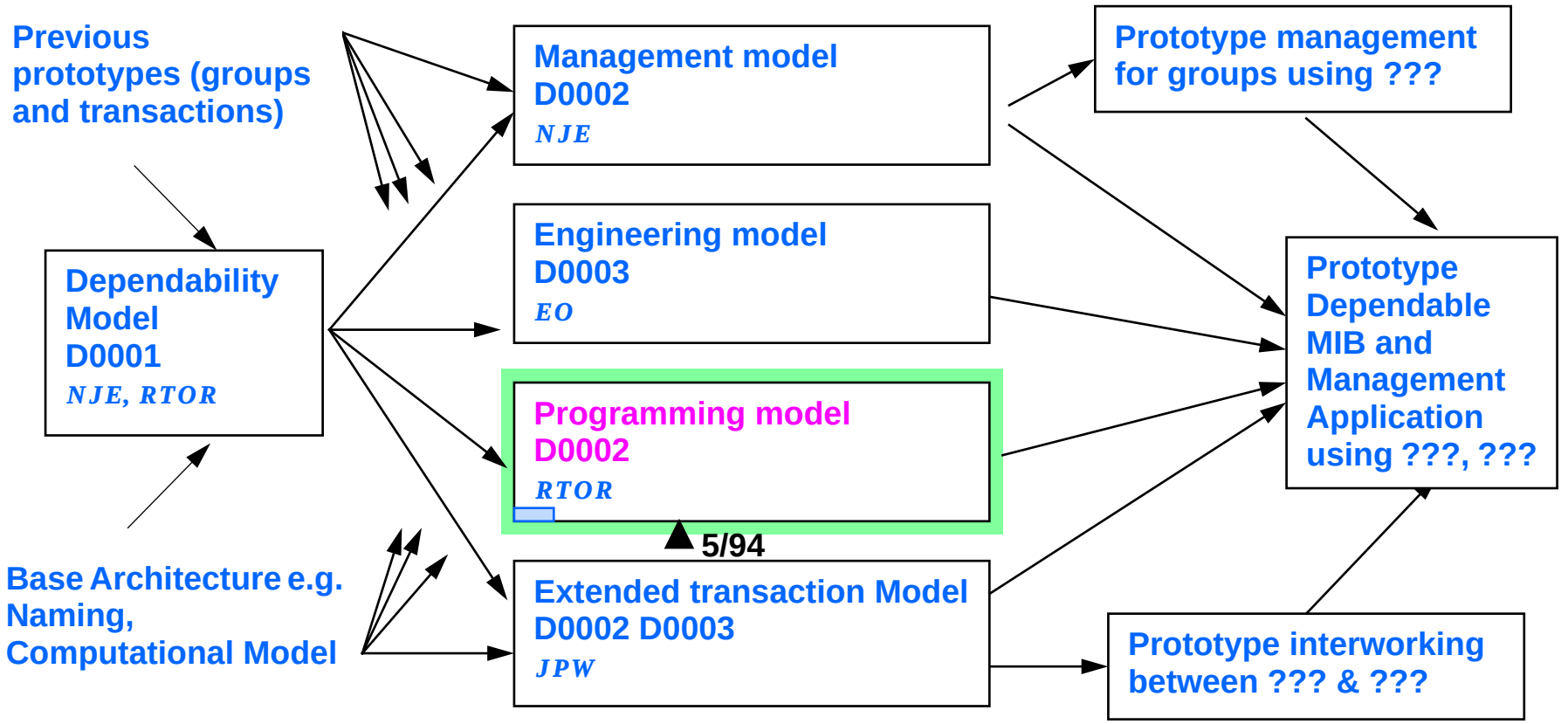
Dependability Programming Model

Work in Progress

Owen Rees

Dependability Group

Dependability outline plan



Objective of this talk

- Status of work on the programming model for dependability
- Objectives of the work
- Plan
  - timescales
  - outputs
- Progress
- Relevance of the work
  - Is it relevant to you?
  - What kind of output is most useful?

Objective of the work

- **Cheaper dependability by exploiting knowledge**
  - **mechanisms less expensive in resources**
  - **less programming effort**
- **Identify programming concepts that can be exploited**
- **Design and programming guidelines**
- **Propagating knowledge to decision points**
- **Infrastructure component requirements**
- **Potential for tool support**
  - **design, programming and configuration tools**
  - **language issues**

Programming model plan

- **Hypothesis: knowledge about use of mutable state can be exploited**
  - **Read access is distinguished from write access in transaction systems**

Feb 94	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Now	Dec 94	Jan 95
	▼			▼			▼			▼	
Identify benefits		Exploiting mutability analysis				dependability analysis		tool and language issues			

- **identify specific benefits that might be achieved**
- **identify how analysis of mutability might achieve those benefits**
- **analyse dependability characteristics of proposed mechanisms**
- **identify tool and language issues in realising these benefits**

Identify benefits

- **Hypothesis: knowledge of use of mutable state can be exploited**

- **identify various mechanisms (replication, transactions)**
- **cost in storage, processing, comms of each mechanism**
- **dependability gain for mechanism (faults detected, tolerated etc.)**
- **significance of mutable state in the mechanism**

- **Savings possible given knowledge**

Exploiting knowledge of mutability

- How can knowledge of mutability be exploited in practice

- identify infrastructure components
  - which components permit the range of mechanisms
- how to choose infrastructure components
  - configuration guided by knowledge
- guidelines on structuring the application
  - minimise requirement for expensive mechanisms
- how knowledge of mutability can be propagated
  - type, trading and binding issues
- prototype to demonstrate feasibility

Failure analysis

- Apply failure model to components and configurations
- Explore fault/error propagation characteristics
- Identify any stability problems

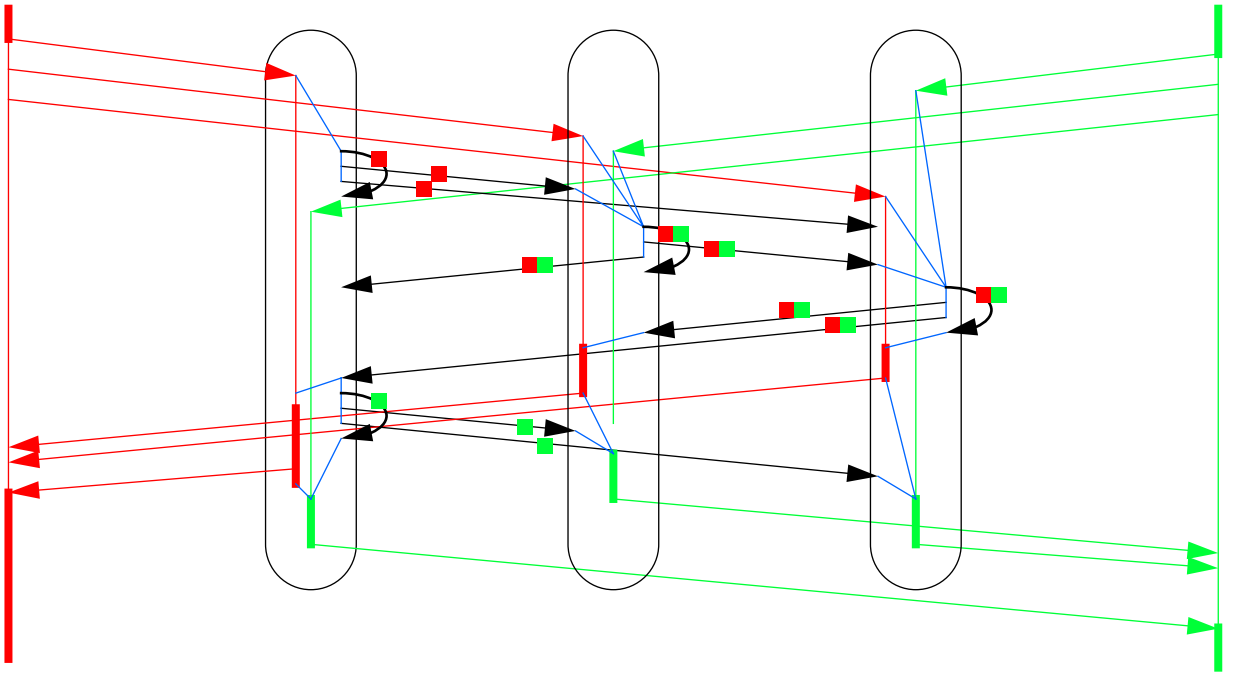
- Does the cheaper mechanism do the same job?
- If not, what is the difference?

- Designer has better cost/benefit data

Tool support and language issues

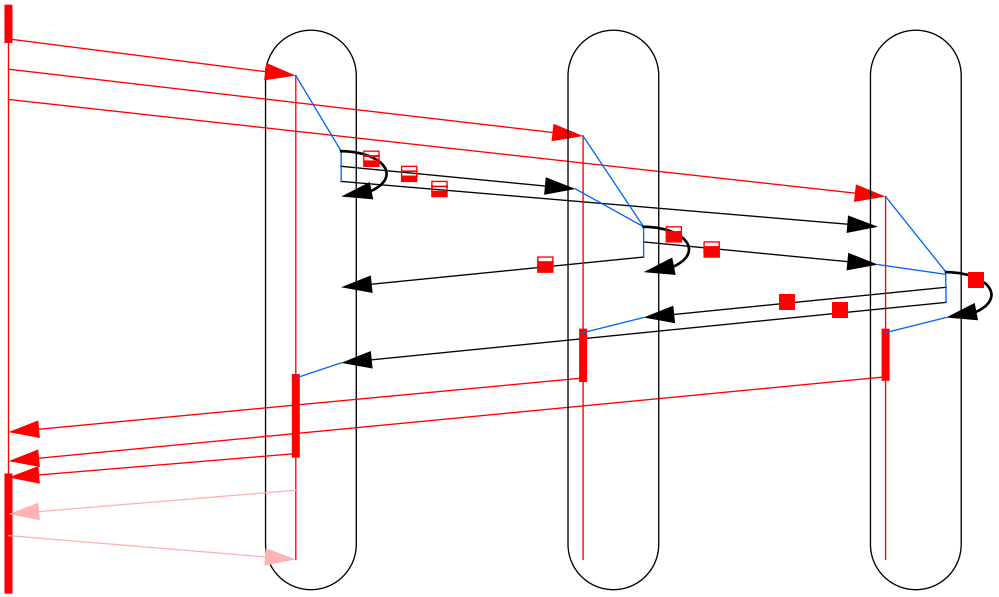
- **Identify language features that enable tool support for dependability**
  - **Given a language, how can exploitable knowledge be extracted?**
- **Prototype some dependability support tools?**

GEX example

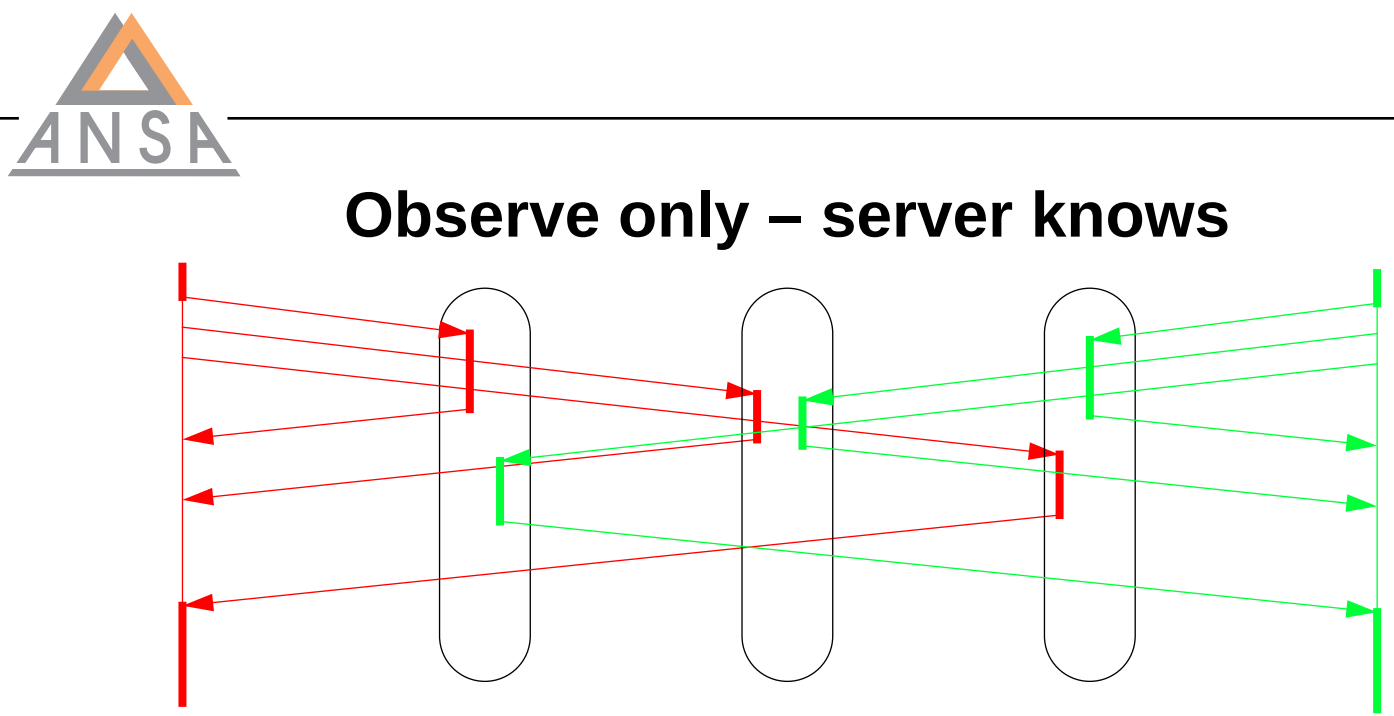


- 3 member group with concurrent invocations from singleton clients
- Protocol ensures consistent update

Cost of an invocation



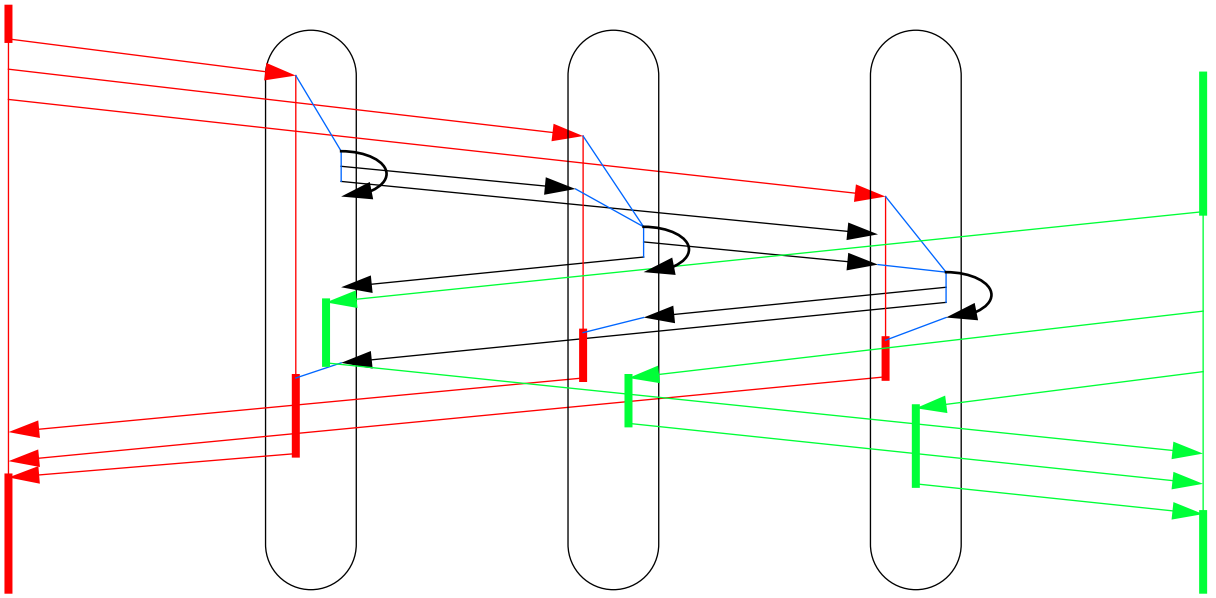
- 15 messages, 3 evaluations, 5 message latency
- Necessary for update because concurrent invocation is possible
- What if it only observes the state without updating?



Observe only – server knows

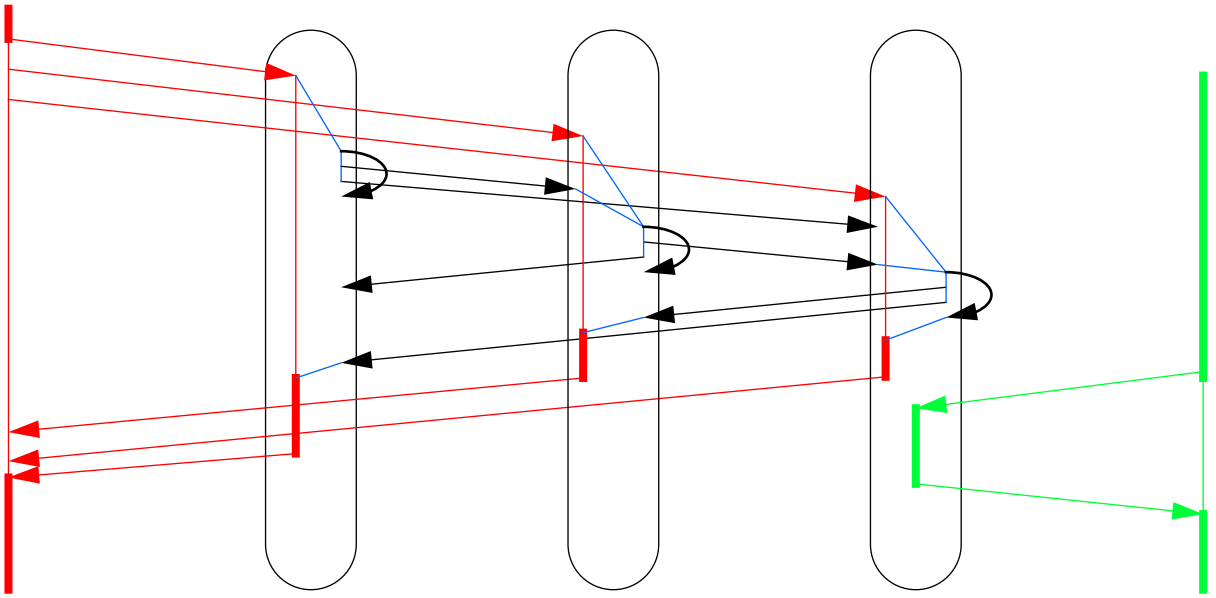
- Arbitrary order, or concurrent evaluations
- 6 messages, 3 evaluations, 2 message latency (down from 15, 3, 5)
- Observing mutable state implies update is possible – what problems?

Conflict between observe and update



- Values before and after update differ – collation may fail (policy dependent)
- Some collation policies succeed – design and management choice

Client and server know operation is observe only



- Client may invoke only one member – costs 2 messages, 1 eval, 2 latency
- Server must be expecting this
- Different failure properties – significance to be studied

Current status

- **Observe only operations can be much cheaper**
  - especially if this knowlege is available to both client and server
  - but fault detection and tolerance properties change
  - starting to quantify dependability trade-offs
- **Potential benefits are significant**
- **Protocols need some refinement**
- **Knowledge propagation is important**
  - related to types of operations in interfaces
  - link to trading and binding needs to be explored
- **Promising on paper – is it implementable?**
  - prototyping needed to develop ideas
  - and to demonstrate them

Relevance of this work

- **Is this work relevant to your company?**
- **Which aspects should be emphasised?**
- **How should the results be presented?**
  - **reports (APM.1122 is in preparation)**
  - **presentations**
  - **prototypes demonstrated at APM**

Summary

- **Objectives of the work**
  - Cheaper dependability by exploiting knowledge
- **Plan**
  - identify benefits
  - study how knowledge can be exploited
  - analyse dependability characteristics
  - identify tool and language issues
- **Progress**
  - benefits for replication identified
- **Relevance of the work**
  - Is it relevant to you?
  - What kind of output is most useful?