# Designing for Architecture Evolvability : some conclusions from a MIS case study

Stephen Cook

Applied Software Engineering Research Group

University of Reading

S.C.Cook@reading.ac.uk

http://www.rdg.ac.uk/~sis99scc/

# Case Study Context

- Part of DESEL project
  - Designing for Ease of System Evolution
  - http://www.rdg.ac.uk/~sis99scc/desel/
- Partners include :
  - Rutherford Appleton Laboratory (IT Dept.)
  - University of Hertfordshire (Paul Wernick)
- Case studies continue…
  - conclusions so far are provisional
  - need to validate conclusions in wider context

# FRS Key Features

- In-house financial MIS for laboratory project managers
- Generates parameterised HTML reports from snapshots of Oracle Financials system, staff-time booking system etc.
- Data flows : simple
- Data semantics : complex and incompletely understood
- Long product-line history (17 years)
- Limited resources (2 staff-years p.a.)

# Evolvability as a Viewpoint

- Evolvability requires explicit attention :
  - system designers make different decisions if they consciously design for evolvability
  - project managers should control short-term pressures to defer evolvability issues
  - Do domain experts analyse requirements differently if they adopt an evolvability point of view?
- Is architecture evolvability a *definable* viewpoint (in the IEEE 1471 sense)?
  - further work in progress

# Use Patterns to Localise Evolution

- Example of a MIS architecture pattern :
  - 3-stage pipeline of Adapters (Gamma et al. p139) :
    - Adapter 1 : batches real-time data updates into snapshots
      - evolution is mostly technology-driven, so easier to manage
    - Adapter 2 : abstracts from atomic data using business rules
      - e.g. Distribute an Invoice Line across Projects
      - cleans implementation details of transactions
      - needs to be pluggable / swappable because rules evolve
      - issues of granularity, normalisation, under-specification of rules
      - Do most MIS evolution problems occur here?
    - Adapter 3 : marshals business objects into business process variants
      - e.g. Manage a project using CERN reporting conventions

# Reaffirm SE Principles

- Apply established software engineering principles to solve architectural problems, e.g. :
  - separation of concerns
  - abstraction / refinement
  - reusable architecture patterns
- Develop languages, notations and tools that incorporate SE principles
  - make poor designs more difficult to produce than better designs
  - enable designers to measure quality early in life-cycle