1. The nature of software design

- *a priori...*
  - proactive, master planned, top-down, methodological
- ...vs *ad hoc*
  - reactive, piecemeal, incremental, recursive, informal, improvised
- *In architecture and design theory*
  - Thackara (beyond the object), Jencks (adhocism), Alexander (the pattern language)

1.1 Master-planned design

1.2 Piecemeal design

1.3 Software design as craft

- *Vernacular processes fit software practice*
  - hand tools and fabric dictates design practice
  - design is ‘intimately’ iterative
  - roles are unencumbered
- *Vernacular knowledge and skills transfer*
  - designed objects articulate their history
- *Reality of software design is very craft-like*
  - methodology’s view of ‘design as transformation’
  - eg. eXtreme programming.
1.4 Summary - software design knowledge

- Software designers control their product
  - in practice, processes can often be orthogonal to the 'real work'
  - design is a continuous, personal, craft-like
- Software designers rationalise aspiration and application
  - interpretation, judgement, values
- Software designers must *satisfice*
  - optimise rather than perfect (perfection is often prohibitively expensive)

2 Applicability of the evolutionary model

- The tenets of evolution
  - Classification - what constitutes the evolving 'unit' of design, and where do we look to see the evolutionary history?
  - Variation - how does an instance of a unit of design vary?
  - Inheritance - are these variations passed on?
  - Selection - does the environment select the fittest?

2.1 Classification - what defines a 'species'??

- Problem with the gene/species analogs
  - if gene is an individual idea (meme), what defines the species?
  - what defines/constrains the meme anyway?
- The 'design meme'
  - similar design solutions are repeatedly rediscovered
    - at high cost to individual designers, projects and the global industry.

2.2 Variation - do instances of the 'species' vary?

- The 'design meme' varies constantly
  - complexity and context mean that 'design memes' must be documented to survive
  - faithful copying depends upon the copier's understanding
- Variation is assured
  - however, complexity and inexperience sometimes masks its effects.

2.3 Inheritance - are variations passed on through 'generations'?

- Generations are bounded by contexts
  - knowledge of a design solution truly gets reused (re-instantiated) only over project boundaries
- The 'carrier' is the designer
  - the consultant/contract designer has long been identified as the most effective design knowledge carrier (Allen 77)

2.4 Selection - does the environment select?

- Selection is realised by designers
  - solutions that worked are reused
    - those that didn't are forgotten (!)
  - solutions that are discovered are adopted
- Code realises design knowledge transfer
  - unlike other forms of expertise, design memes are expressed in code
  - the effectiveness of this form of transfer is low because justification/explanation is often weak.
2.5 Summary - for design evolution

- Devolved design control
  - Knowledge develops from experience, not dictation or instruction

- Evolution (unlike revolution) is efficient
  - Knowledge about complex software designs is built incrementally

- Design knowledge transfer is vernacular
  - Actual knowledge transfer frequently occurs in craft-like ways.

2.6 Summary - against design evolution

- Gene analog makes species definition weak
  - Same problem that the memeticists face

- Variation is not systematic
  - Imperfect human memory, differing complexity, changing contexts

- Stable context required
  - Changes to context weaken the basis of selection judgement.

3. Knowledge management implications

- How should we think about design?
  - Need to manage engineering and craft knowledge differently
    - Current methodologies sit at these extremes
    - Current methodologies emphasise process over product and knowledge exchange
  - Must look for tacit-to-explicit knowledge mechanisms
    - Patterns provide an exemplar.

3.1 Implications

- Need to anticipate good replicators
  - Need research on the characteristics of strongly replicating design memes
    - Simplicity over complexity
    - Simple ‘hooks’ in complex design patterns
    - The simple, self-explanatory idioms are most widely coded.

3.2 Implications

- Speeding up evolution
  - Being a ‘reflective practitioner’ is essential
  - The patterns community exemplify one process
  - Highly iterative development
  - Highly interactive (craft-like) transfer.

4 Conclusions

- Software design is like non-software design
  - Rational macro-process, irrational micro-process

- Software design knowledge is memetic
  - Knowledge reified in literate form (patterns, code, narratives)

- Methodology must admit design’s inherent informality before it can incorporate useful Knowledge Management techniques.