Architectural Design

[Architectural Design (Chapter 17): Buxton & McDermid, 1993]
"Overview paper **or** *33 pages without saying anything"*

- introduction to concept of *"Architectural Design"*
- difficulty of balancing requirements and high-level design
- unquantifiable requirements:
  - Interface qualities: user friendliness, robustness, reliability
  - Behaviour qualities: maintainability, modifiability, extensibility, reusability
- almost no laws governing quality of software product
- functional, object-oriented, & process driven decomposition
- Myer's structure charts, dataflow notations, MASCOT (& sequels), data structure diagrams, Booch's HOOD(Hierarchical Object Oriented Design), De Marco's Structured Analysis/Structured Design (SA/SD), Jackson System Development (JSD), Nicholl's Structured System Analysis and Design Method (SSADM)

Foundations for the Study of Software Architecture

[ACM SIGSOFT: Perry & Wolf, 1992] *"The means justifies the end"*

- "The 1990s, we believe, will be the decade of *software architecture*."
- compares software architecture to network architecture (concludes nothing) and building architecture (proves that if you abstract enough you can compare anything)
- motivation: need another layer of abstraction for large systems
- introduces concepts of *architectural erosion* and *architectural drift*
- want to accomplish:
  - prescribe the architectural constraints to the desired level
  - separate aesthetics from engineering
  - express different aspects of the architecture in an appropriate manner
  - perform dependency and consistency analysis
- describes some very basic architectural styles:

A Field Guide to Boxology

[CMU: Shaw & Clements, 1996]
"Pretty box, awk, pretty box"

- extension of Garlan & Shaw's previous ground breaking work
- two-page table listing various architecture patterns, including cross-referencing and sub-classing
- control issues: topology, synchronicity, binding time
- data issues: topology, continuity, mode, binding time
- control & data issues: shape, directionality
- in-depth analysis of variations of two examples: pipe-and-filter and message passing

Architectural Styles, Design Patterns, and Objects

[IEEE Software: Monroe, Kompanek, Melton & Garlan, 1997]
"Buzz-words, get your red hot buzz-words here"

- explains each buzz word
- uses the example of a pipe and filter, attempting to restrict visibility using each concept
- discusses patterns and styles
- disturbing mix of terms "object" and "design pattern"

Programming-in-the-Large Versus Programming-in-the-Small

[IEEE Tran. on SE: DeRemer and Kron, 1976]
"Time does tell"

- MIL: Module Interconnection Language – "provide a means for the programmer(s) of a large system to express their intent regarding the overall program structure in a concise, precise, and checkable form"
- goals:
  - project management tool
  - design tool – "means of establishing overall program structure"
  - means of communication amongst designers
  - means of documenting
- introduces MIL75

ISF: A Visual Formalism for Specifying Interconnection Styles for Software Design

[IJSEKE: Mancoridis, 1998]
"nice title on ya buddy"

- ISF – Interconnection Style Formalism
- hierarchical, graphic notation for specifying a pattern
- dictates allowed interactions between modules
- rules are stored in Datalog so notation can be compiled and checked
- uses concepts of "define" and "permit" to dictate the style
- unclear on how this can interface with real code, or whether you would want it to

Binary Relational Algebra Applied to Software Architecture

[CSRI Tech Report: Holt, 1996]
"but why not babble fish?"

- introduces Grok tool for rudimentary relational algebra using Rigi Standard Format (RSF) tuple relationships
- simple set theory functions
- subsumption through cross-product
- although tool is buggy, it is essentially a good idea
- replaces miles of perl scripts
- why not just use a data base?

Software Architecture in Industrial Applications

[IEEE 17th Conference on SE: Soni, Nord, Hofmeister, 1995]
"if I told you, I would have to kill you"

- four architectural views:
  - conceptual – major design elements & connections
  - module – functional decomposition and layers
  - code – source organization
  - execution – dynamic structure
- use many examples from industry, all of which were proprietary and had to be obfuscated to the point of being useless before they could be included in the paper

Architectural Mismatch or Why it's hard to build systems out of existing parts

[IEEE 17th Conf. SE: Garlan, Allen, Ockerbloom, 1995]
"there is a sucker born every minute"

- building Aesop: wanted to mix an OO-database, toolkit for GUI, event-based integration mechanism, and RPC mechanism
- problems: excessive code size, poor performance, need to modify external packages, re-invent existing functions, unnecessarily complicated tools (3 minute startup time – insist that it wasn't just bad coders)
- assumptions about: nature of components, nature of connectors, global architectural structure, and construction process
- solution:
  - make architectural assumptions explicit
  - construct large piece of s/w using orthogonal sub-components
  - provide techniques for bridging mismatches
  - develop sources of architectural design guidance

An Architectural Analysis Case Study: Internet Information Systems

[WWW: Kazman, Bass, Abowd, Clements, 1997]
"the night of the living dead **or** *SAAM II"*

- quantifying intangible qualities is difficult
- consider "task scenarios" which are tangible
- use common architectural notation to compare candidate architectures
- identify task scenarios which correspond to functional components; assign a + if it is just one, else assign a -
- weight task scenarios as to their importance
- moves the argument?

The 4+1 View Model of Architecture

[IEEE Software: Kruchten, 1995]
"how many fingers do you need?"

- introduces five views
  - logical view – object model
  - process view – concurrency and synchronization
  - physical view- – mapping of software onto hardware; especially for large distributed systems
  - development view – static organization in development environment
  - scenarios
- correlates with Soni
- examples of PBX and flight simulation software

Software Reflexion Models: Bridging the Gap Between Source and High-Level Models

[SIGSOFT: Murphy, Notkin and Sullivan, 1995]
"Zed's dead baby, Zed's dead"

- compares abstract and concrete architecture
  - convergence – the good
  - divergence – the bad
  - absence – the gratuitous
- Z-notation to describe these concepts
- correspondence between concept and implementation

A Reverse-engineering Approach to Subsystem Structure Identification

[Software Maintenance: Muller, Orgun, Tilley, and Uhl, 1993]
"your smart-ass comment here"

- extraction of architecture from source
- Rigi is your friend
  - it parses
  - it relates
  - it draws
  - it filters
  - it does none of these things correctly

Assessing Modular Structure of Legacy Code Based on Mathematical Concept Analysis

[ICSE: Lindig and Snelting, 1997]

"how many elephants must die for an ivory tower of this size to be built?"

- attempts to mathematically formalize the concept of a module
- here a Greek letter, there a Greek letter
- coupling of modules based on variables used by a set of procedures
- maximal cohesion is used to define a module
- every procedure must use every variable to be in the module
- what is the point?

The Future

- very high level languages for architecture
- integrated development and planning tools
- reverse engineering tools – fully automated extraction
- utilization of coupling and cohesion measures
- pattern catalogues?
- better pattern matching techniques will lead to better extraction