CSci 658-01: Software Language Engineering
Introduction to Sofware Architecture

H. Conrad Cunningham

12 January 2018

Contents

Professor of Computer and Information Science
University of Mississippi
211 Weir Hall
P.O. Box 1848
University, MS 38677
(662) 915-5358

Acknowledgements: This material is based primarily on Chapter 1 of the textbook *Software Architecture: Perspectives on an Emerging Discipline* by Mary Shaw and David Garland (Prentice Hall, 1996).

I created these slides for the firstoffering of my Software Architecture special topics course in Spring 1998. Thanks to Wen Wen Xu for entering my handwritten slides as HTML so that I could prepare these notes more quickly.

I reformated the slides to use Pandoc Markdown in Spring 2018.

Advisory: The HTML version of this document may require use of a browser that supports the display of MathML. A good choice as of November 2017 is a recent version of Firefox from Mozilla

What is Engineering?

Many definitions include a phrase similar to:

- creating cost-effective solutions
- to practical problems
- by applying scientific knowledge
- building things
- in the service of mankind

Key approach: codify scientific knowledge about a problem domain in a form directly useful to practitioners (and hence extend the capability of the ordinary design talent)

Question: Is software development an engineering discipline?

Engineering Design

routine	<====>	innovative
familiar problems reuse prior solutions (handbooks, manuals)		unfamiliar problems novel solutions

Question: What is the situation with software design?

Software Reuse

Reuse of software

code	<===>	design

Problems in reuse:

- identifying reusable entities
- expressing them in useful form
- making them accessible
- adapting them to variants of the problem
- overcoming cultural tradition
- educating designers/programmers in reuse
- convincing management to invest in reuse

Figures from Shaw-Garlan Textbook

Examine Figure 1.2, *Evolution of an Engineering Discipline.*

Examine Figure 1.3, *Codification Cycle for Science and Engineering.*

Examine Figure 1.4, *Evolution of Software Engineering.*

Codification through Abstraction

Abstraction: emphasizing the essentials, ignoring the details

Computer science has progressed by increasing the level of abstraction---moving to larger conceptual building blocks.

(The following dates are approximate.)

1950's	machine language\
	symbolic assemblers\
	macro processors\
	simple compilers (e.g., early Fortran)\

1960's	data typing (Algol)\
	modularity\
	user-defined types\

1970's	theory of abstract data types\
	early OOP ideas\
	module interconnection languages (MIL)\

1980's	. . .\
	class libraries\

**1990\	CORBA\
**	application framework\
	patterns\
	. . .\

Aspects of the Emerging Science

- programming language abstraction (previous section)
- algorithm analysis
- automata/models
- language theory
- syntax

What is Software Architecture?

It involves description of:

- elements from which systems are constructed
- interactions among those elements
- patterns that guide their composition
- constraints on those patterns

Systems are defined in terms of:

- collection of components
- interactions among those components (connectors)

Example components:

- clients and servers
- databases
- filters
- layers in hierarchical system
- etc.

Example interactions:

- procedure calls
- shared variables
- piped streams
- client-server protocols
- database transactions
- event multicast
- etc.

System Design

There are many levels of design, each with own concerns.

Each level consists of:

- components (vocabulary)
- rules of composition (syntax)
- rules of behavior (semantics)

Software Design

1. **Architecture:** high-level system capability

   -- involves composing modules to form systems

2. **Code:** algorithms and data structures

   -- involves composing primitive language features to form modules

3. **Executable:** allocation of code and data to machine

   -- involves composing bit patterns to form language features

---

Research Areas in Software Architecture

- architectural description languages
- codification of architectural expertise
- architectural frameworks for specific domains
- formal foundations
- architecture design/analysis selection methods
- support tools
- architecture extraction, recovery, and reengineering (for legacy systems)
- etc.