Three Patterns
that help explain the development of
Software Engineering

Mary Shaw
Carnegie Mellon University

Vitruvius

1

Three Patterns

- **Evolution of Engineering Disciplines**
  - > **Technologies evolve from craft through commercial practice before they integrate scientific knowledge and become true engineering disciplines.**
- **Abstraction and its Coupling to Specifications**
  - > **Abstraction granularity increases over time**
  - > **Formal specification progress is linked to utility of the specifications**
- **Progressive Codification**
  - > **Specification techniques evolve in parallel with our understanding of the phenomena they specify**

Vitruvius

2

Definition: What is "Engineering"?

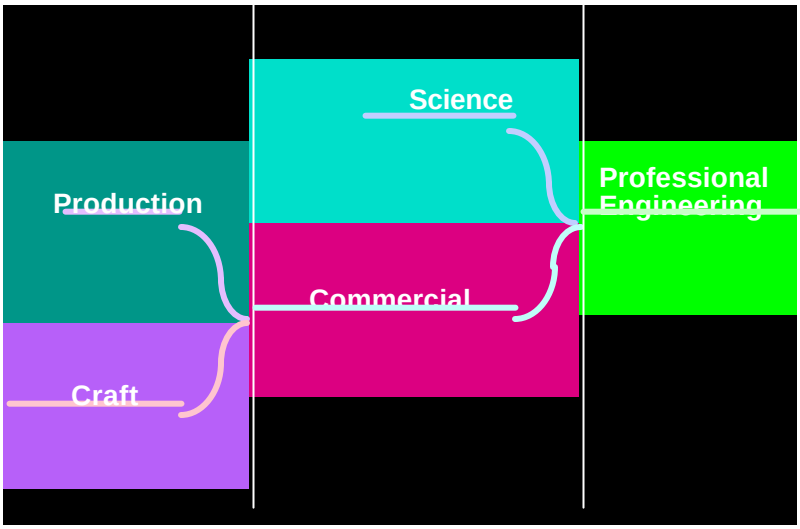
Definitions abound. They have in common:

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

Engineering enables ordinary people to do things that formerly required virtuosos

Vitruvius

3



Craft	Commercial	Professional Engineering
Virtuosos & talented amateurs	Skilled craftsmen	Educated professionals
Intuition & brute force	Established procedure	Analysis & theory
Haphazard progress	Pragmatic refinement	Progress relies on science
Casual transmission	Training in mechanics	Educated professional class
Extravagant use of available materials	Economic concern for cost & supply of materials	Enabling new applications through analysis
Manufacture for use rather than sale	Manufacture for sale	Market segmentation by product variety

"Software Engineering"

Rallying Cry:

Phrase coined in 1968 to draw attention to software problems

Aspiration, not description

Requirement:

Cost-effective solutions ...

... to practical problems ...

... of increasing scale

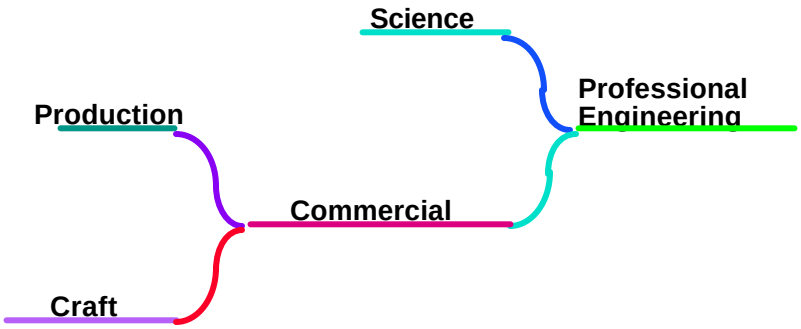
Technical basis

Computer science

Vitruvius

6

Where is Software Engineering ?



Craft **, becoming** ***Commercial***
A ***Profession*** **only in isolated instances**

Vitruvius

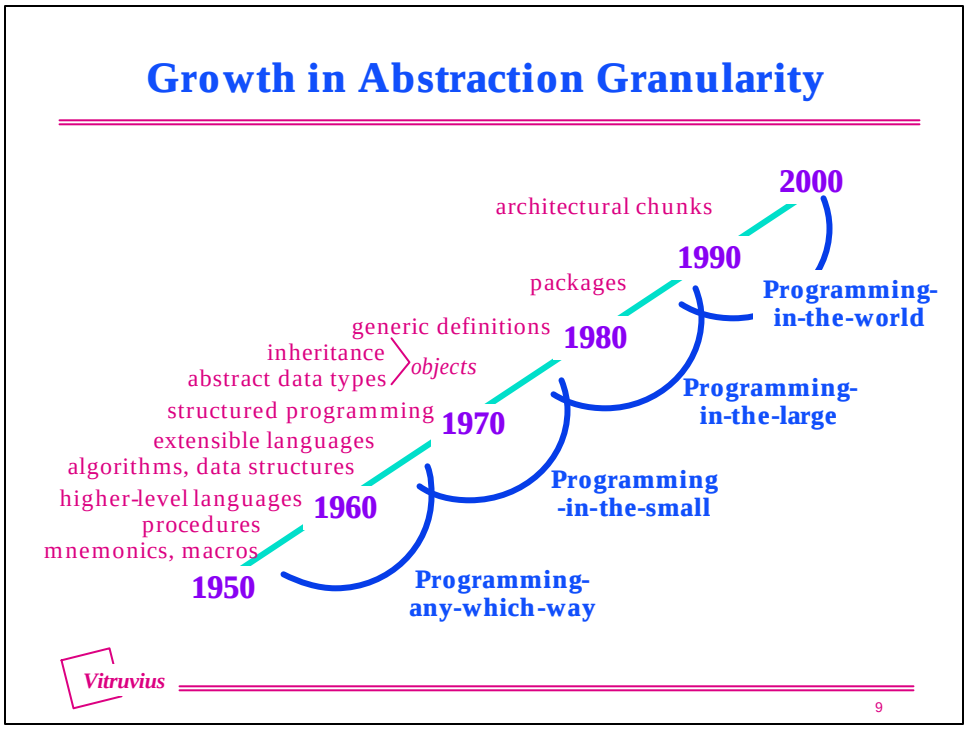
7

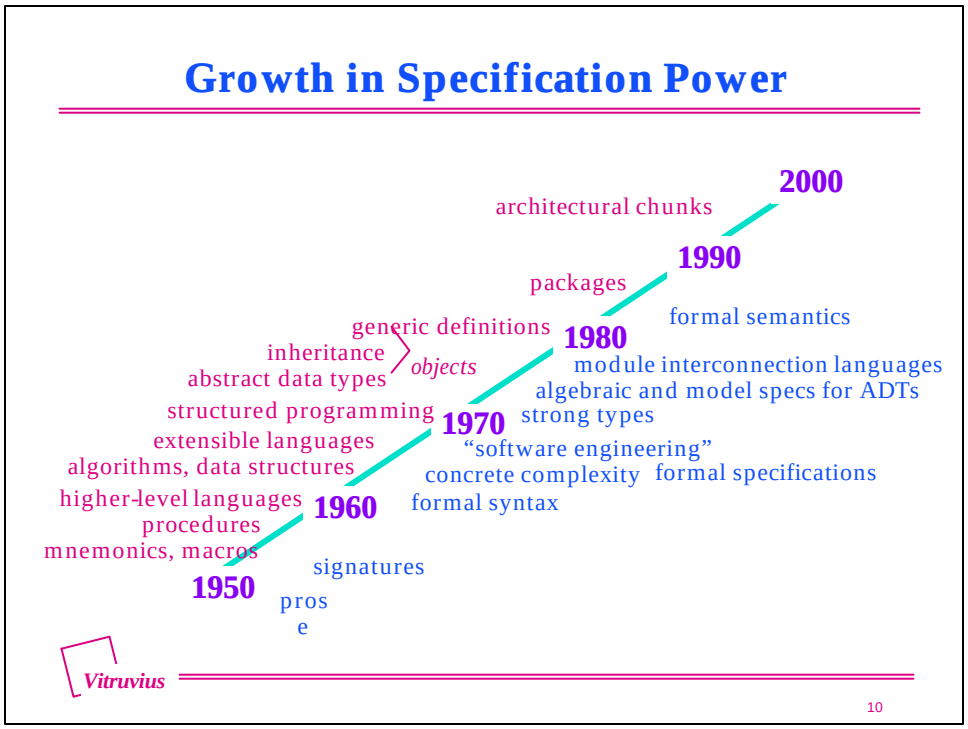
Quick History of Software Engineering

1960 ± 5 Programming-any-which-way	*1970 ± 5 Programming-in-the-small*	*1980 ± 5 Programming-in-the-large*
Mnemonics, precise use of prose	Simple input-output specifications	Systems with complex specifications
Emphasis on small programs	Emphasis on algorithms	Emphasis on system structure, management
Representing structure, symbolic information	Data structures and types	Long-lived databases
Elementary understanding of control flow	Programs execute once and terminate	Program assemblies execute continually

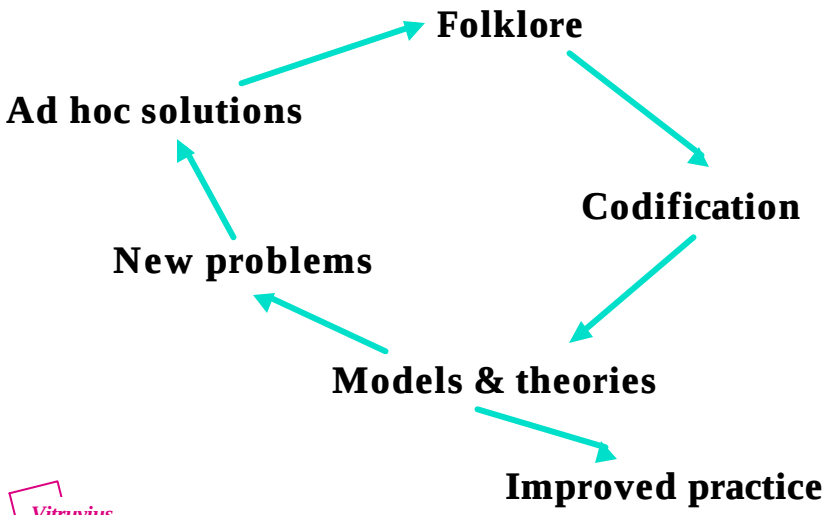
Vitruvius

8


Growth in Abstraction Granularity
2000
architectural chunks
1990
packages
Programming-
in-the-world
generic definitions
1980
inheritance
abstract data types
objects
Programming-
in-the-large
structured programming
1970
extensible languages
algorithms, data structures
Programming
-in-the-small
higher-level languages
1960
procedures
mnemonics, macros
1950
Programming-
any-which-way
Vitruvius
9


Growth in Specification Power
2000
architectural chunks
1990
packages
formal semantics
generic definitions
1980
inheritance
abstract data types
objects
module interconnection languages
algebraic and model specs for ADTs
structured programming
1970
strong types
extensible languages
"software engineering"
algorithms, data structures
concrete complexity
formal specifications
higher-level languages
1960
formal syntax
procedures
mnemonics, macros
signatures
1950
pros
e
Vitruvius
10

Good Science Feeds Good Engineering



Vitruvius

11

Progressive Codification

Software specification techniques often evolve in parallel with our understanding of the phenomena they specify

Sample progression:

- > **Ad hoc:** implement any way you can
- > **Capture:** simply retain and explain a definition
- > **Construction:** explain how to build instance from parts
- > **Composition:** explain how to compose parts & specs
- > **Selection:** guide designer's choice among alternatives
- > **Verification:** determine whether impl meets spec
- > **Automation:** construct instance from external spec

Vitruvius

12

Brooks on kinds of results

Brooks proposes recognizing three kinds of results:

- ***findings:*** **well-established scientific truths**
- ***observations:*** **reports on actual phenomena**
- ***rules-of-thumb:*** **generalizations, signed by an author but perhaps not fully supported by data**

Criteria for judging quality

- ***findings:*** **truthfulness and rigor**
- ***observations:*** **interestingness**
- ***rules-of-thumb:*** **usefulness**

and freshness for all three

Frederick P. Brooks, Jr. Grasping Reality Through Illusion -- Interactive Graphics Serving Science. *Proceedings of the ACM SIGCHI Human Factors in Computer Systems Conference*, May 1988, pp. 1-11.

Vitruvius

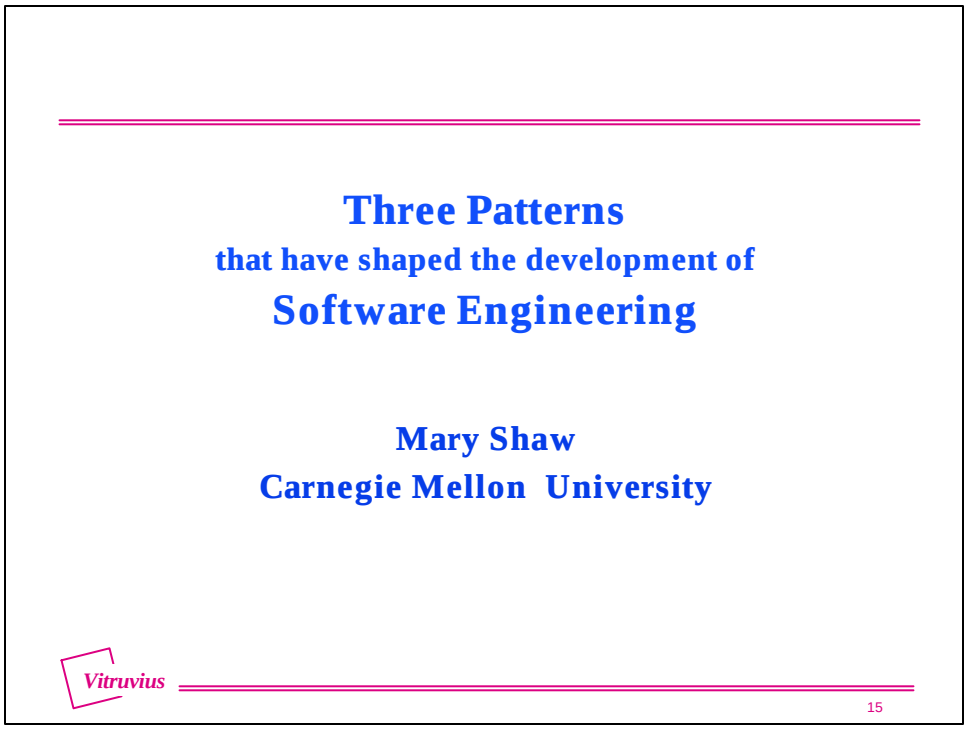
13

Citations for Three Patterns

- **Evolution of Engineering Disciplines**
  - **Mary Shaw. Prospects for an Engineering Discipline of Software. *IEEE Software*, November 1990.**
- **Coupling Between Formalism and Utility**
  - **Mary Shaw. The Impact of Abstraction Concerns on Modern Programming Languages. *Proc IEEE*, September 1980, also *IEEE Software* Oct 1984.**
- **Progressive Codification**
  - **Mary Shaw and David Garlan. Formulations and Formalisms in Software Architecture. Computer Science Today (LNCS 1000), Jan van Leeuwen (ed), Springer-Verlag 1995.**

Vitruvius

14


Three Patterns
that have shaped the development of
Software Engineering
Mary Shaw
Carnegie Mellon University
Vitruvius
15