



Progress toward an *Engineering Discipline* of Software

Mary Shaw

Institute for Software Research
Carnegie Mellon University

What does it mean to have an engineering discipline for software?

How far has software engineering progressed toward that goal?

What are the next steps?

with examples from civil engineering
and software architecture

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*

What is “engineering”?

Definitions abound

They have in common:

- Creating cost-effective solutions ...

- ... to practical problems ...

- ... by applying **codified** knowledge ...

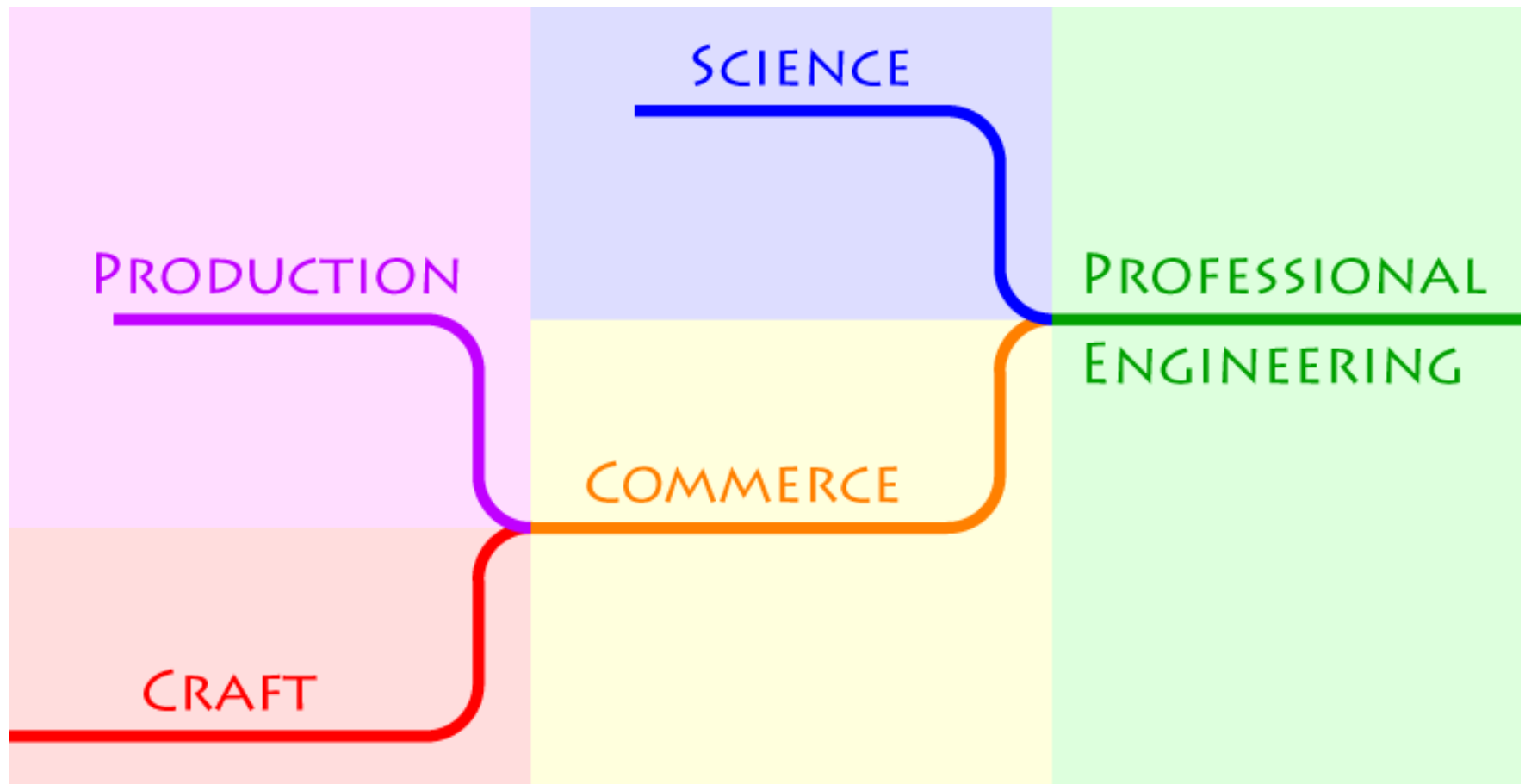
- ... building things ...

- ... in the service of mankind

*Engineering enables ordinary people
to do things that formerly required virtuosos*

Characteristics of engineering

- limited time, knowledge, and resources force decisions on tradeoffs
- best-codified knowledge, preferentially science, shapes design decisions
- reference materials make knowledge and experience available
- analysis of design predicts properties of implementation



Engineering evolves from craft and commerce; it requires scientific foundations, or at least systematically codified knowledge.

Exploiting technology requires both management and a body of systematic, scientific knowledge.

Science often arises from progressive codification of practice.

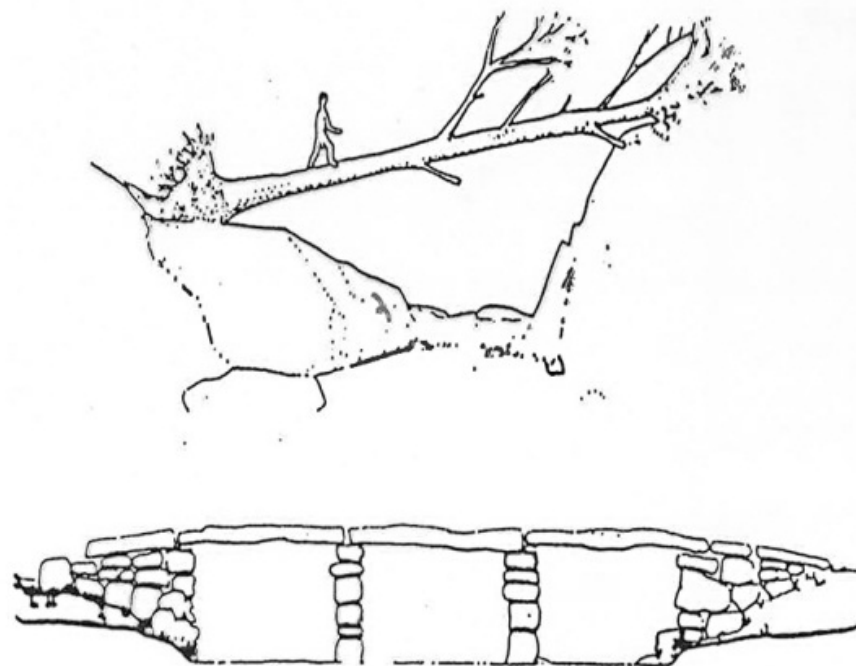
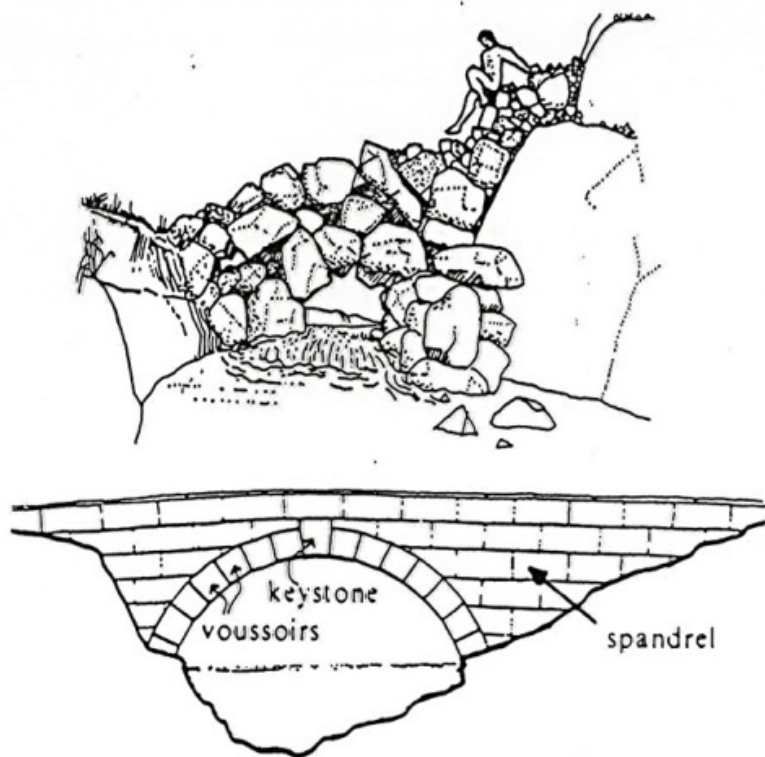
Civil Engineering as Model

A large steel arch bridge spans a deep valley. The bridge's structure is a complex lattice of steel beams forming a large arch. A skydiver with a colorful parachute is visible in the upper left sky. The valley below is filled with trees showing autumn foliage. The sky is blue with light clouds.

Civil Engineering

Example:

Bridges and Arches



Great Buildings of the World
Bridges, Derrick Beckett,
 Hamlyn Publishing Group, Ltd.,
 London, England, pp 10,12,16,19

1st Century CE

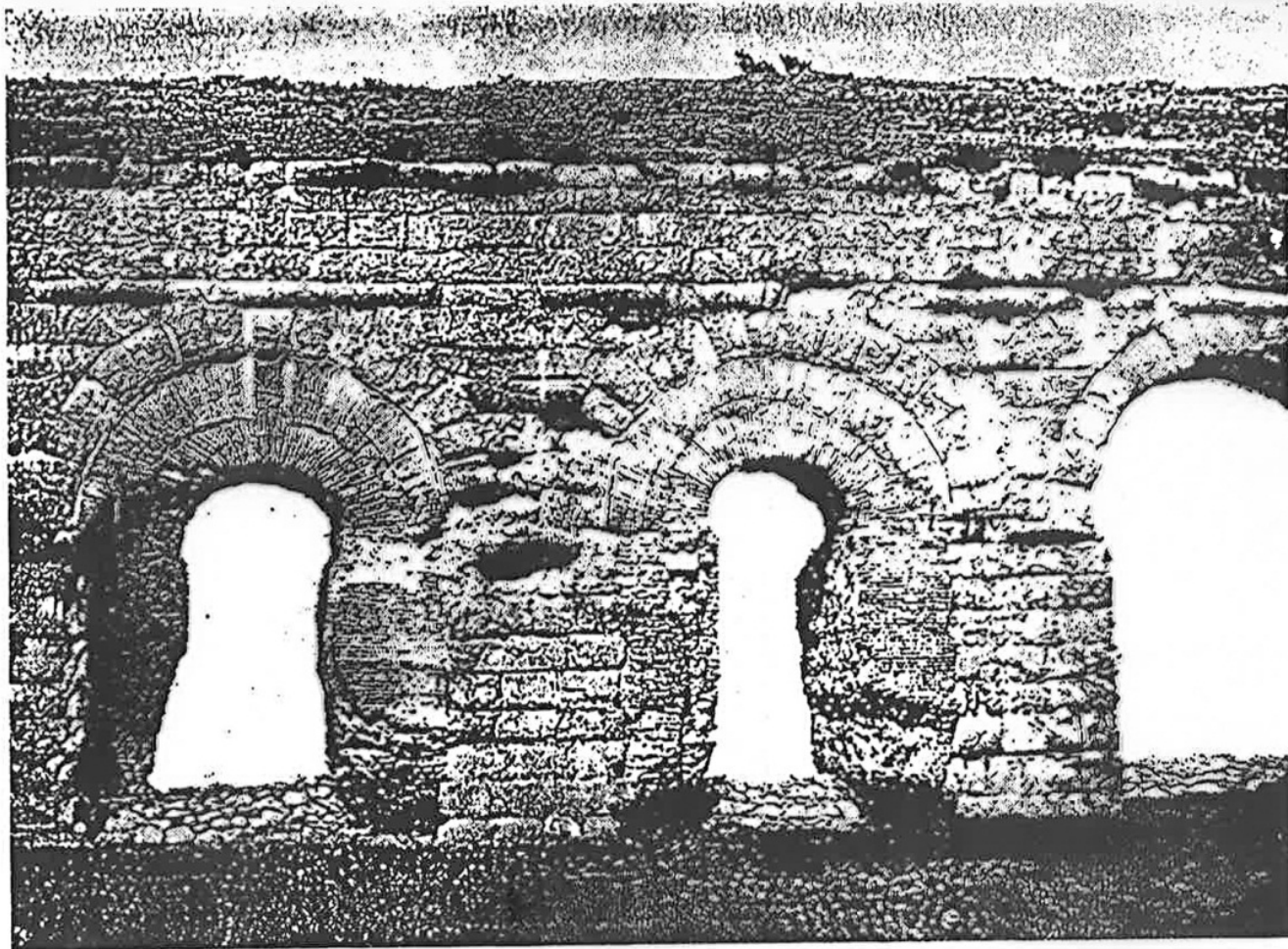
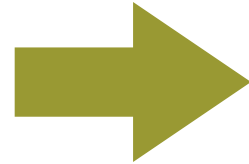


Figure 4.4 Two Roman aqueducts, Anio Novus built on Claudia (From Curt Merckel, *Die Ingenieurtechnik im Alterthum*, 1899; courtesy Julius Springer-Verlag)

Craft of bridges

Romans



Empirical progress via failure and repair

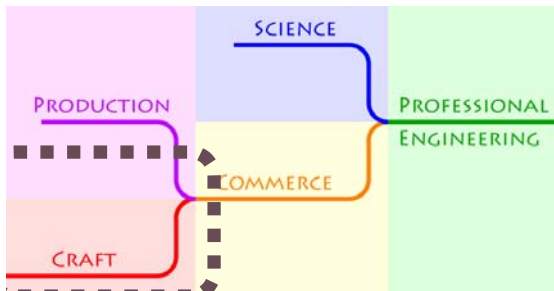
Renaissance
& Industrial
Revolution

No deliberate application of mathematics to determine size or shape

Scientific
Engineering

Little theory, but construction methods lasted until 19th century

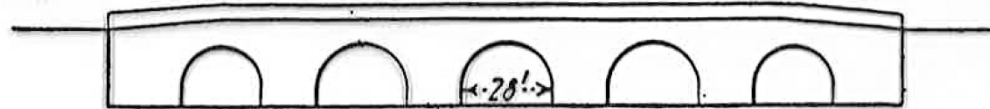
Vitruvius: *De Architectura* [about 25 BC]



Ponte di Augusto, Rimini

Waterway below floor level 35%

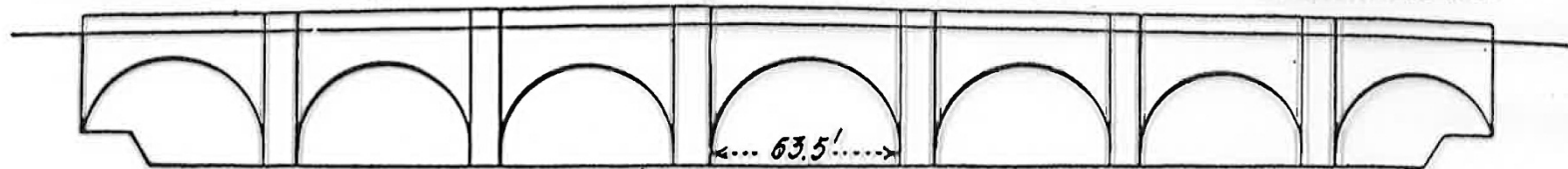
Roman 14 A.D.



Pont Neuf (North Section), Paris

Waterway below floor level 50%

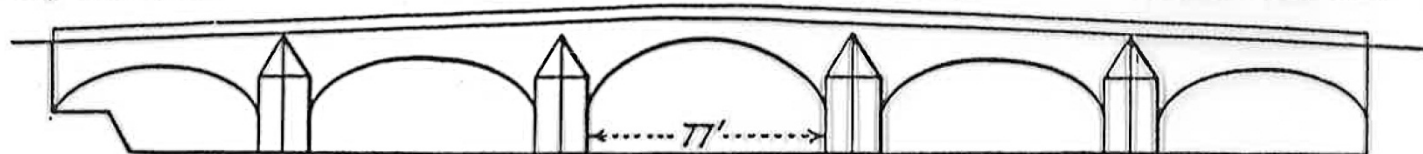
French 1578-1607



Pont Royal, Paris

Waterway below floor level 55%

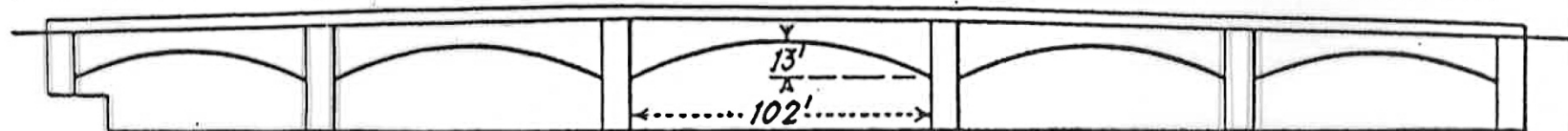
French 1685-1687



Pont de la Concorde, Paris

Water below floor level 65%

French 1787-1791



The Evolution of the Stone-arch Bridge

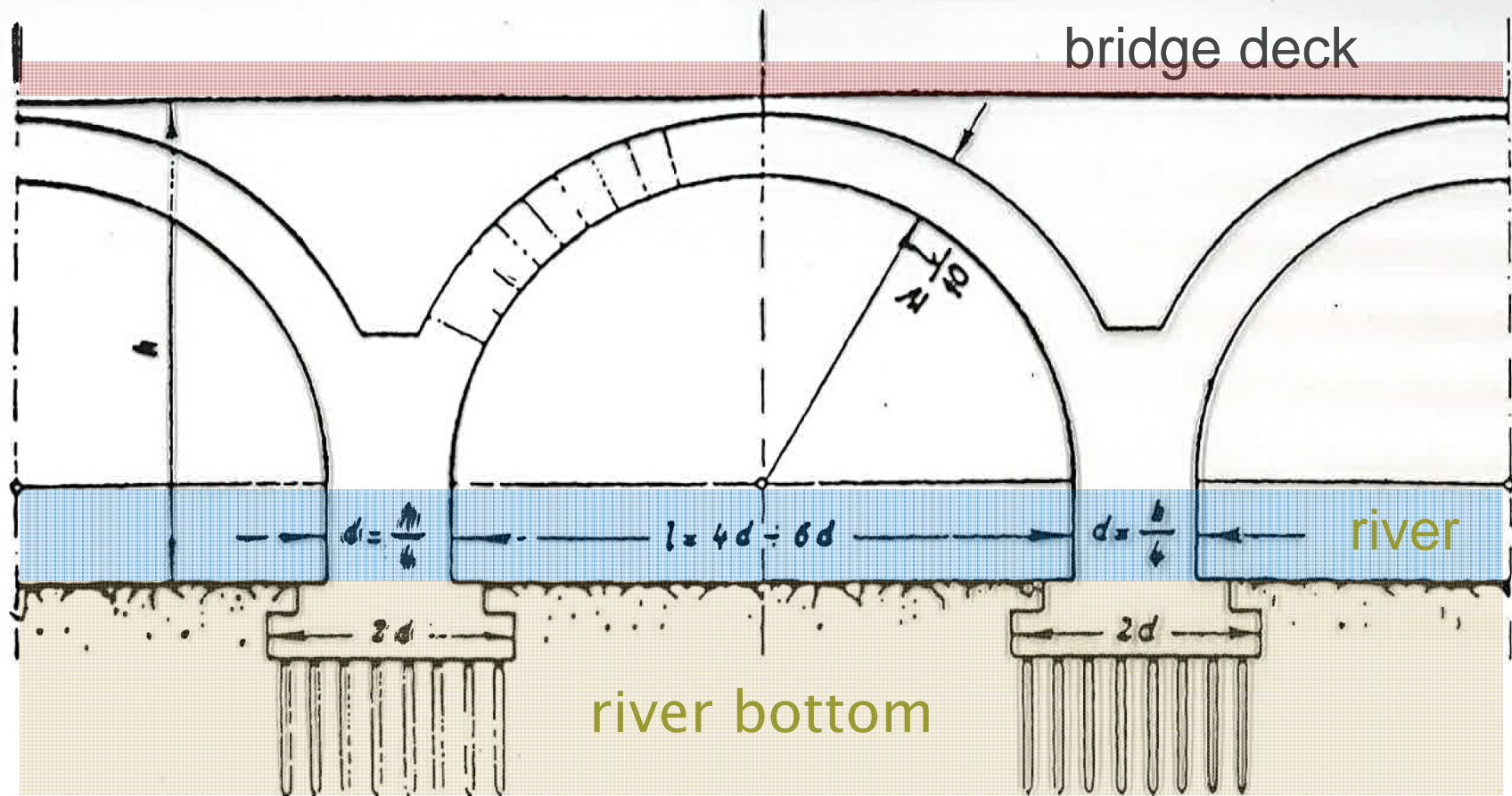
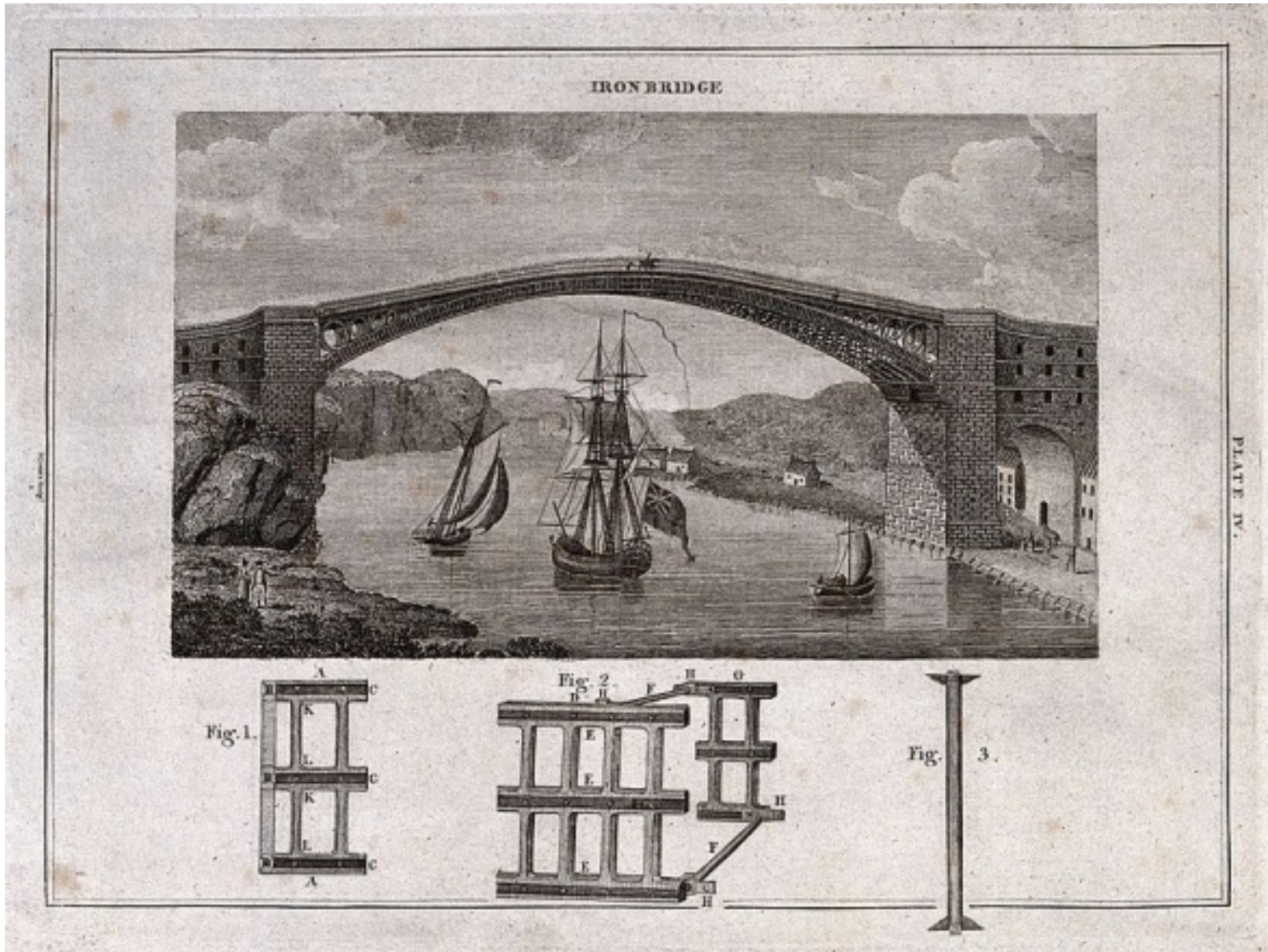


Fig. 28. Arch bridge, according to Leon Battista Alberti.

15th century

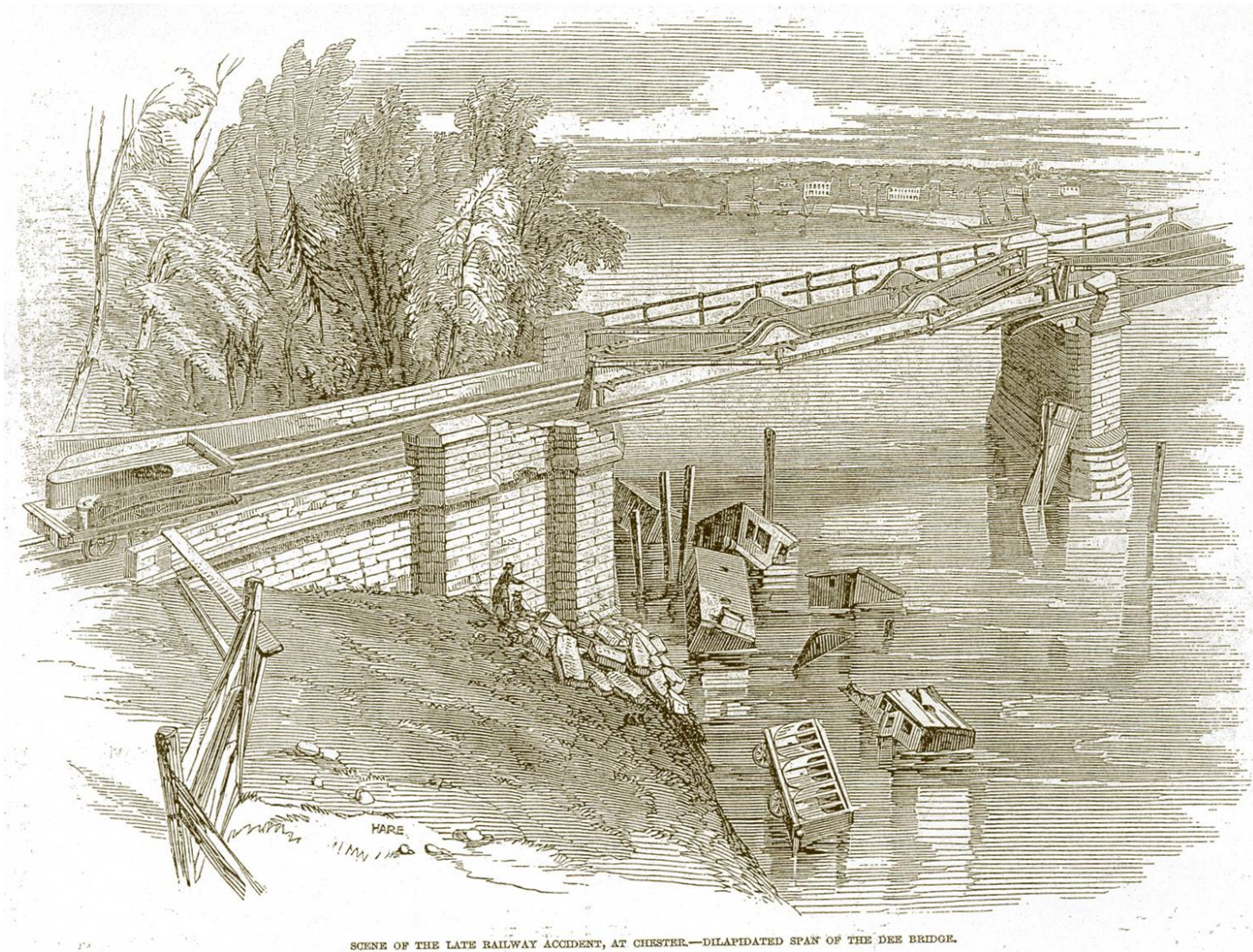
Ironbridge at Coalbrookdale, 1779





Mary Shaw

Dee Bridge disaster, 1847



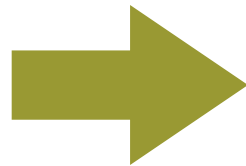
SCENE OF THE LATE RAILWAY ACCIDENT, AT CHESTER.—DILAPIDATED SPAN OF THE DEE BRIDGE.

Business of bridges

Romans

Increasingly long spans,
lighter structures

Renaissance
& Industrial
Revolution



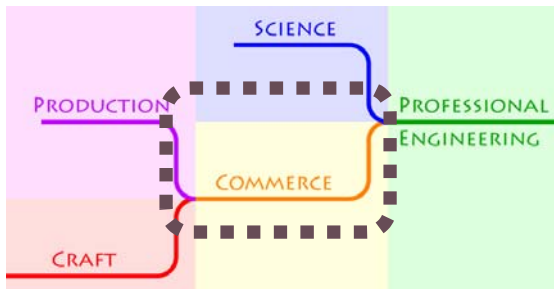
Rules of thumb about
proportions

Explanation of structures:

- Brunelleschi on arches
and domes 15th century
- Galileo on beams 17th century

Scientific
Engineering

Introduction of cast iron,
wrought iron, steel, and
reinforced concrete



Fundamental Problems

**Composition
of forces**

Bending

Theories that solved these problems

Statics

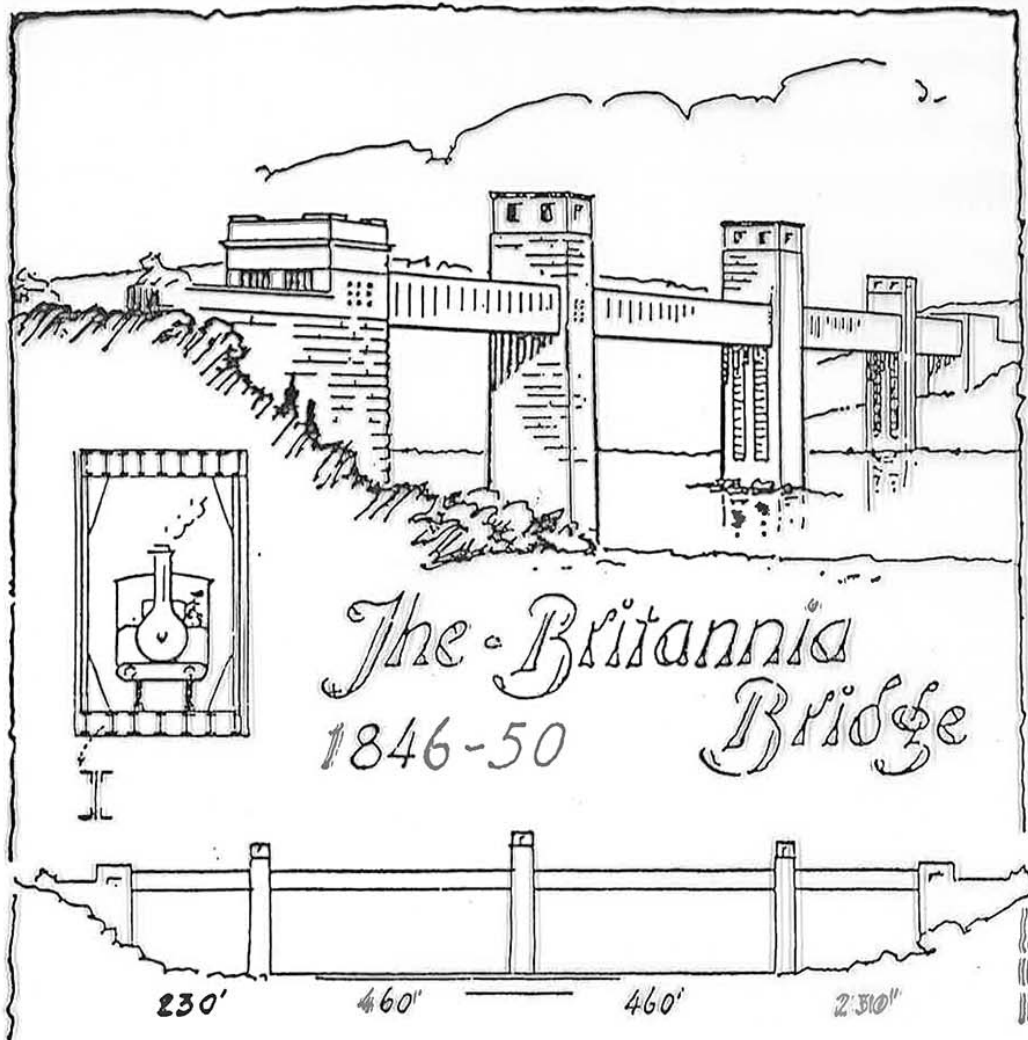
**Strength of
materials**

**Varignon & Newton
late 17th century**

**Coulomb & Navier
early 18th century**

Hardest problem was identifying the proper basic concepts, e.g. force.

New mathematics was needed (calculus).



Wikimedia: Velela

Story of Engineering, James Kip Fiac
eday & Co., Inc., Garden City, NY,
228

PROPERTIES OF VARIOUS SECTIONS

Sections	Area of Section A	Distance from Axis to Extremities of Section y and y_1	Moment of Inertia I	Section Modulus $S = \frac{I}{y}$	Radius of Gyration $r = \sqrt{\frac{I}{A}}$
	$bd - ah$	$y = \frac{d}{2}$	$\frac{1}{12} (bd^3 - ah^3)$	$\frac{bd^3 - ah^3}{6d}$	$\sqrt{\frac{bd^3 - ah^3}{12 (bd - ah)}}$
	$bd - ah$	$y = b - y_1$ $y_1 = \frac{2 b h m + h a^2}{2 A}$	$\frac{1}{3} (2 m b^3 + h a^3) - A y_1^3$	$\frac{I}{y}$	$\sqrt{\frac{I}{A}}$
	$bd - 2 ah$	$y = \frac{d}{2}$	$\frac{1}{12} (bd^3 - 2 ah^3)$	$\frac{bd^3 - 2 ah^3}{6d}$	$\sqrt{\frac{I}{A}}$
	$bd - 2 ah$	$y = \frac{b}{2}$	$\frac{1}{12} (2 m b^3 + h a^3)$	$\frac{2 m b^3 + h a^3}{6b}$	$\sqrt{\frac{I}{A}}$
	$bnt + ht$	$y = d - y_1$ $y_1 = \frac{d^3 t + m b^3 (b - t)}{2 A}$	$\frac{1}{3} (t y^3 + b y_1^3 - 2 a (y_1 - m)^3)$	$\frac{I}{y}$	$\sqrt{\frac{I}{A}}$
	$bnt + ht$	$y = \frac{b}{2}$	$\frac{1}{12} (m b^3 + h t^3)$	$\frac{m b^3 + h t^3}{6b}$	$\sqrt{\frac{I}{A}}$

PROPERTIES OF VARIOUS SECTIONS

Sections	Area of Section A	Distance from Axis to Extremities of Section y and y_1	Moment of Inertia I	Section Modulus $S = \frac{I}{y}$	Radius of Gyration $r = \sqrt{\frac{I}{A}}$
	$bd + a (m + n)$	$y = \frac{d}{2}$	$\frac{1}{12} \left[bd^3 - \frac{a}{4 (m + n)} (d^4 - h^4) \right]$	$\frac{2 I}{d}$	$\sqrt{\frac{I}{A}}$
	$bd + a (m + n)$	$y = b - y_1$ $y_1 = \frac{b m + \frac{a^2}{2} + \frac{a (m + n)}{2} (b + 2t)}{A}$	$\frac{1}{3} \left[2 m b^3 + h t^3 + \frac{m - n}{2 a} (b^4 - t^4) \right] - A y_1^3$	$\frac{I}{y}$	$\sqrt{\frac{I}{A}}$
	$bd + 2 a (m + n)$	$y = \frac{d}{2}$	$\frac{1}{12} \left[bd^3 - \frac{a}{4 (m + n)} (d^4 - h^4) \right]$	$\frac{2 I}{d}$	$\sqrt{\frac{I}{A}}$
	$bd + 2 a (m + n)$	$y = \frac{b}{2}$	$\frac{1}{12} \left[2 m b^3 + h t^3 + \frac{m - n}{4 a} (b^4 - t^4) \right]$	$\frac{2 I}{b}$	$\sqrt{\frac{I}{A}}$
	$\frac{d (t + u)}{2} + t m + a (m + n)$	$y = h - y_1$ $y_1 = \left[\frac{4 m b^3 + 2 a (m + n) (m + 2 n) + 3 t d^3}{2} - \frac{a (t + u) (3 d - a)}{2} \right] + 6 A$	$\frac{1}{12} \left[d^3 (2 m + t) + 4 b m^3 - 2 a (m + n)^3 \right] - A (y_1 - m)^3$	$\frac{I}{y}$	$\sqrt{\frac{I}{A}}$
	$\frac{d (t + u)}{2} + t m + a (m + n)$	$y = \frac{b}{2}$	$\frac{m b^3 + (m + n) t^3 + a t^3}{12} + \frac{a (m + n) [2 d^3 + (2 a + 3 t)^2]}{36} + \frac{a (t + u) [(t + u)^3 + 2 (t + 2u)^2]}{144}$	$\frac{2 I}{b}$	$\sqrt{\frac{I}{A}}$

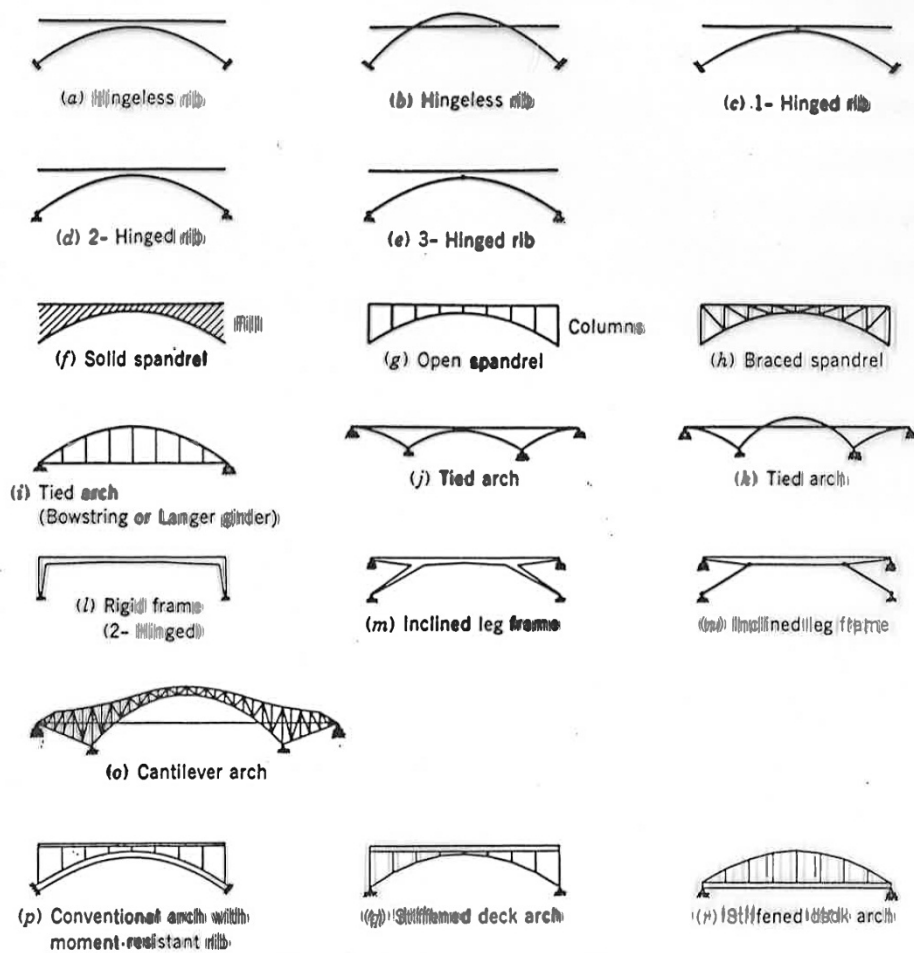


Figure 10.2 Types of arch bridge.

ign of Bridge Superstructures, Collins O'Connor,
y-Interscience, New York, NY, 1971, p. 489

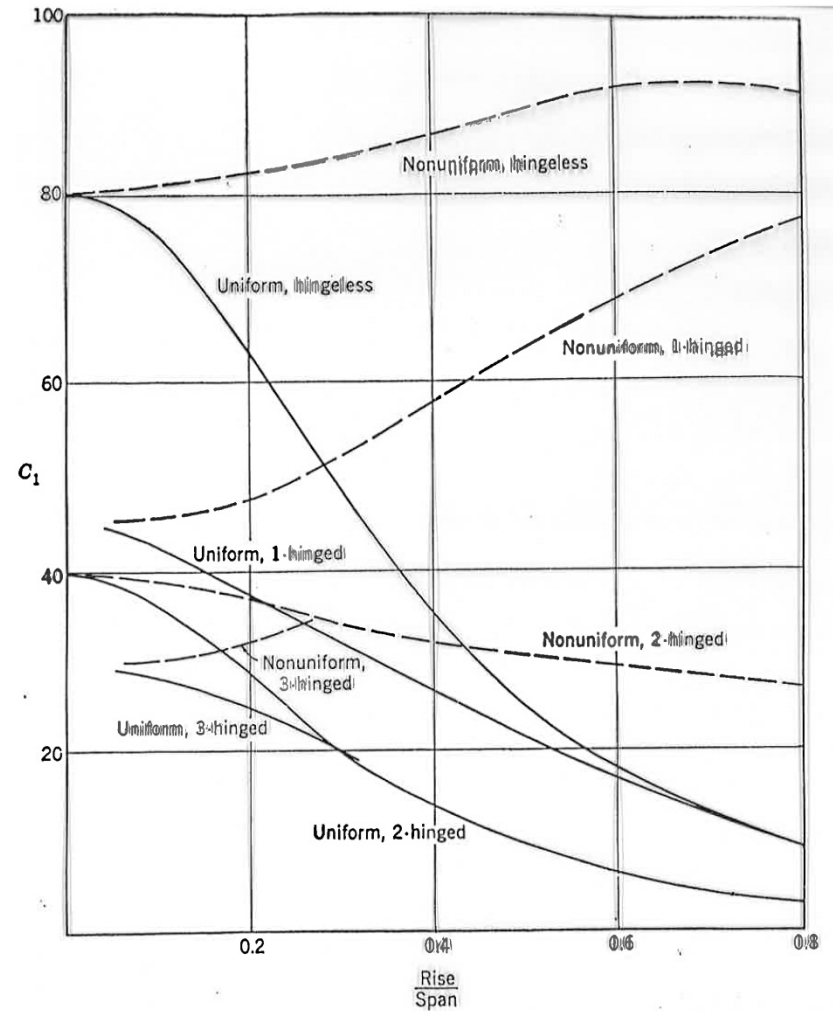


Figure 10.29 Coefficients for in-plane buckling of parabolic arch [59] $H_{cr} = C_1(EI/L^2)$.

ign of Bridge Superstructures, Collins O'Connor,
y-Interscience, New York, NY, 1971,

Engineering of bridges

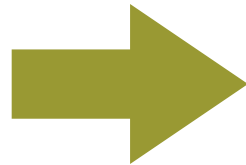
Romans

1700: good theories
(statics, strength
of materials)

Renaissance
& Industrial
Revolution

1750: tabulations of
properties of
materials

Scientific
Engineering

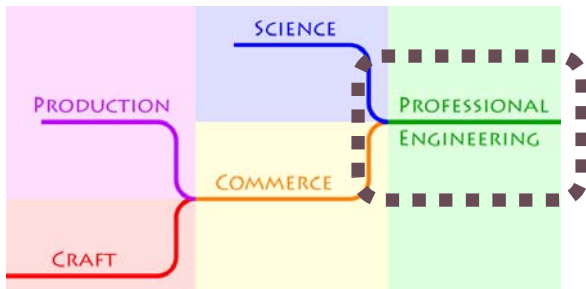


1850: formal analysis of
a bridge structure

1900: structural analysis
worked out

1950: systematic theory

2000: design automaton



21st century

PennDOT now requires use of its software for automated design of simple bridges

- PennDOT's Bridge Automated Design and Drafting Software (BRADD) automates bridge design from problem definition through CAD drawing.
- BRADD designs concrete, steel, and concrete bridges with spans of 18 feet to 200 feet.
- <http://bradd.engrprograms.com/home/>

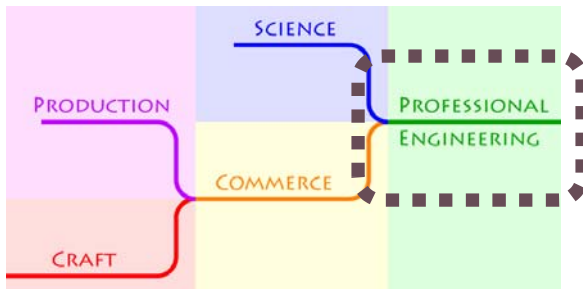


Table 2.3-2 Matrix of Abutment Types versus Superstructure Types

Superstructure Type	Abutment Type					
	Traditional			Integral	SuperOnly High/Stub/Wall	SuperOnly Integral
	High	Wall	Stub			
Prestressed Concrete Adjacent Box Beam		☒			☒	
Prestressed Concrete Spread Box Beam	☒	☒	☒	☒	☒	☒
Prestressed Concrete I-Beam	☒	☒	☒	☒	☒	☒
Steel Rolled Beam	☒	☒	☒	☒	☒	☒
Steel Plate Girder	☒	☒	☒	☒	☒	☒

- Automates production of scaled bridge contract drawings

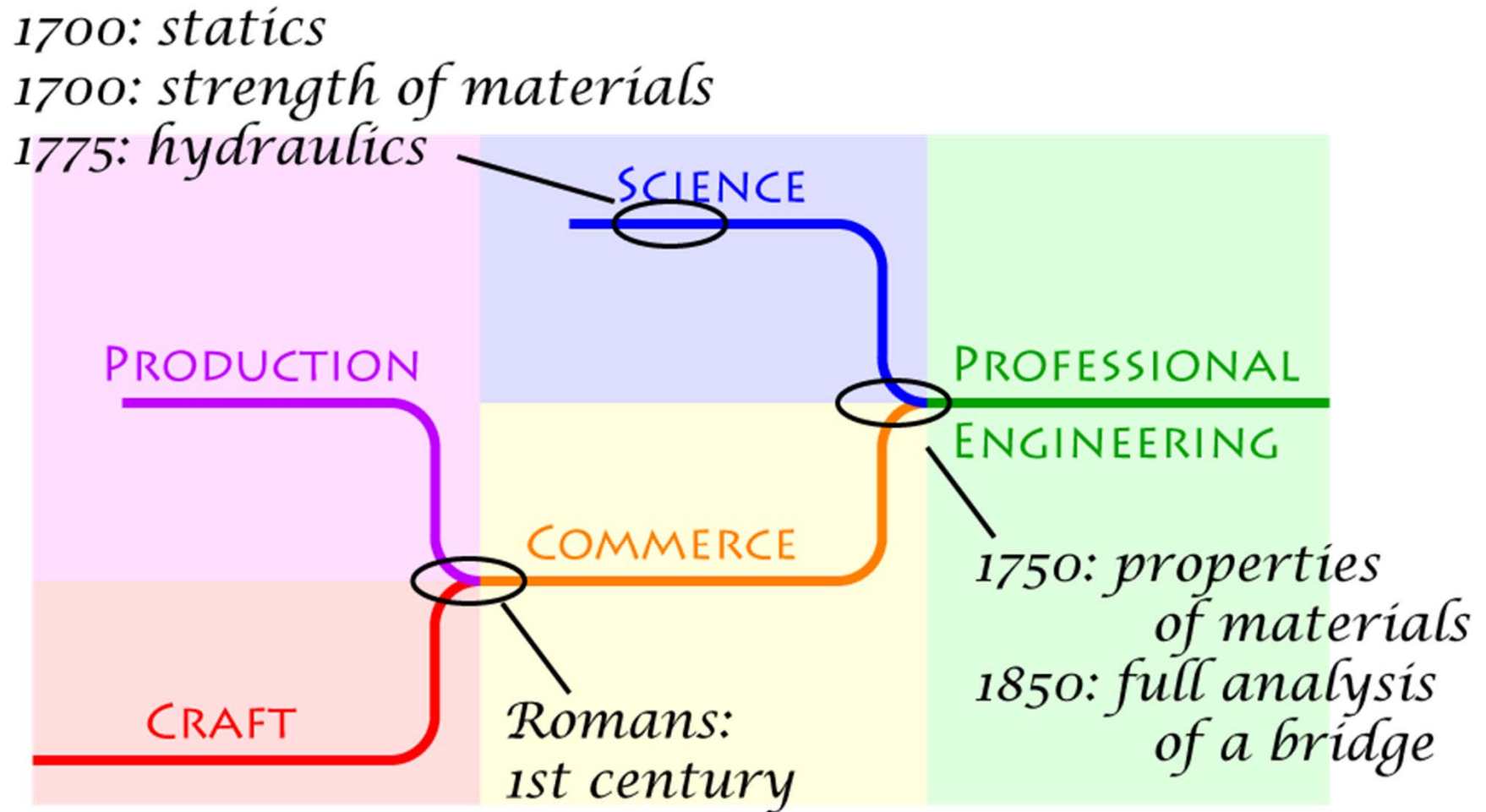
Engineering
Input

Design Drawings

BRADD



Evolution of civil engineering



Software Engineering

Software engineering as engineering

From the definition of engineering:

Creating cost-effective solutions ...

... to practical problems ...

... by applying codified knowledge ...

... building things ...

... in the service of mankind

Software engineering as engineering

From the definition of engineering:

The branch of computer science that ...

... creates cost-effective solutions ...

... to practical computing problems ...

... by applying codified knowledge ...

... developing software systems ...

... in the service of mankind

Software is design-intensive -- manufacturing costs are minor

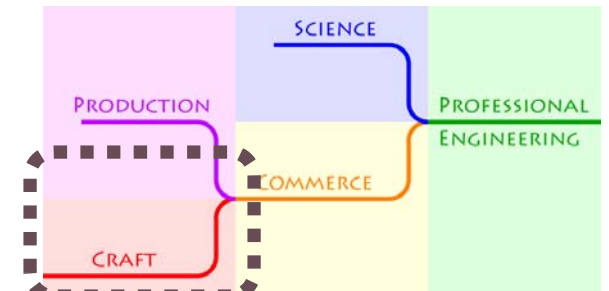
Software is symbolic, abstract, and constrained more by intellectual complexity than by fundamental physical laws

"Software Engineering"

Rallying Cry

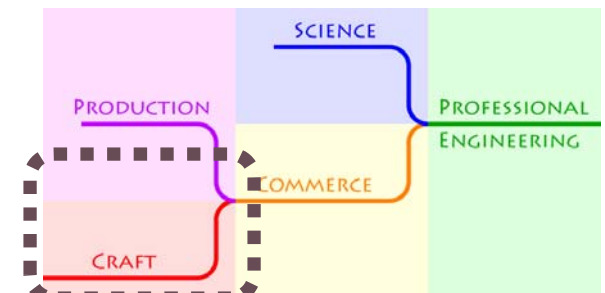
Phrase coined in 1968 to draw attention to software problems

Aspiration, not description



Craft practice, 1968

- Monolithic development, merging research, development, production
- Software fine in many areas, but not for life-critical applications
- Widening gap between ambitions and achievement, increasing risk
- Software is late, over cost estimate, doesn't meet specifications
- Too much revolution, not enough evolution



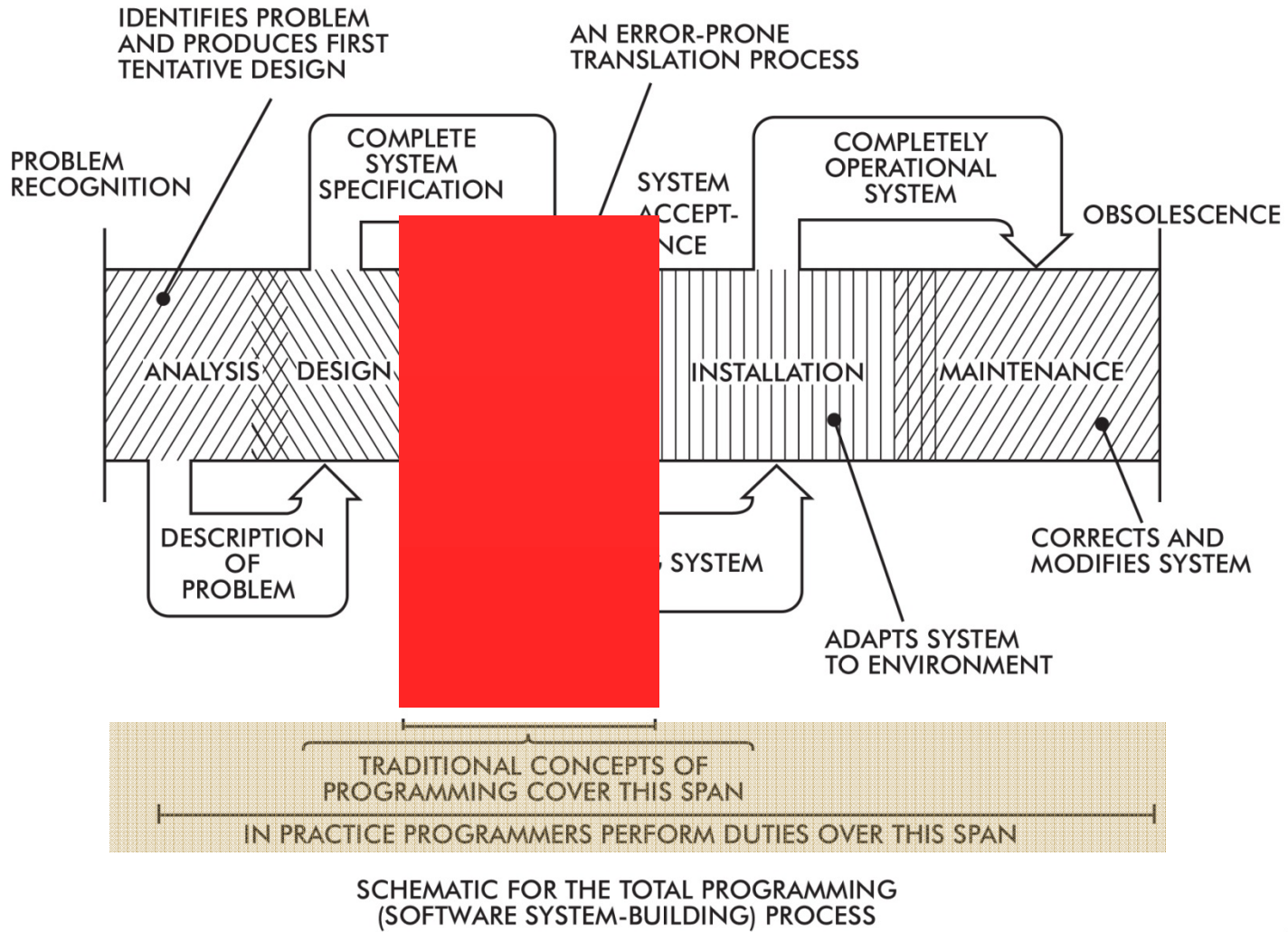


Figure 2. From Selig: Documentation for service and users. Originally due to Constantine.

Production techniques

Systematic **software development methods** bring order and predictability to projects via structure and project management (1970-1990s)

- Structured programming

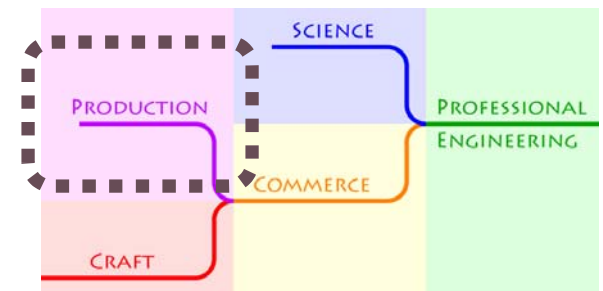
- Waterfall models

- Incremental and iterative development

- Cost/schedule estimation

- Process maturity

- Extreme, agile processes



Codified knowledge

Data structures, algorithms

Programming languages and semantics

Verification and model checking

Computability and computational models

Objects and abstract data types

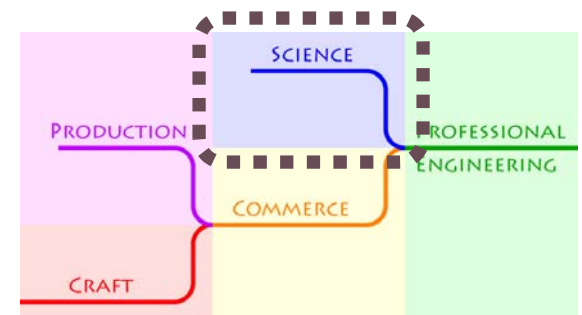
Canonical structures for many applications

Software architectures

Model-based engineering

Pattern languages

...



Fundamental ideas

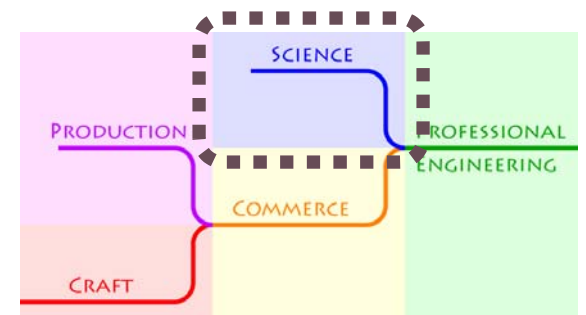
Abstraction enables control of complexity

Imposing **structure** on problems makes them more tractable; **canonical solutions** are available

Symbolic representations are necessary and sufficient for solving information-based problems

Precise models support **analysis and prediction**

Exponential growth creates opportunities and limits



Commerce drives science

Science is often stimulated by problems in commercial practice

safety-critical applications → safety analysis

large systems → architectural patterns

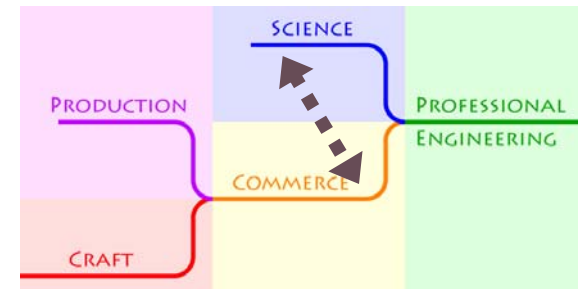
concurrency → parallel logics and languages

large state spaces → model checking

many versions → program families, inheritance

large-scale search → MapReduce

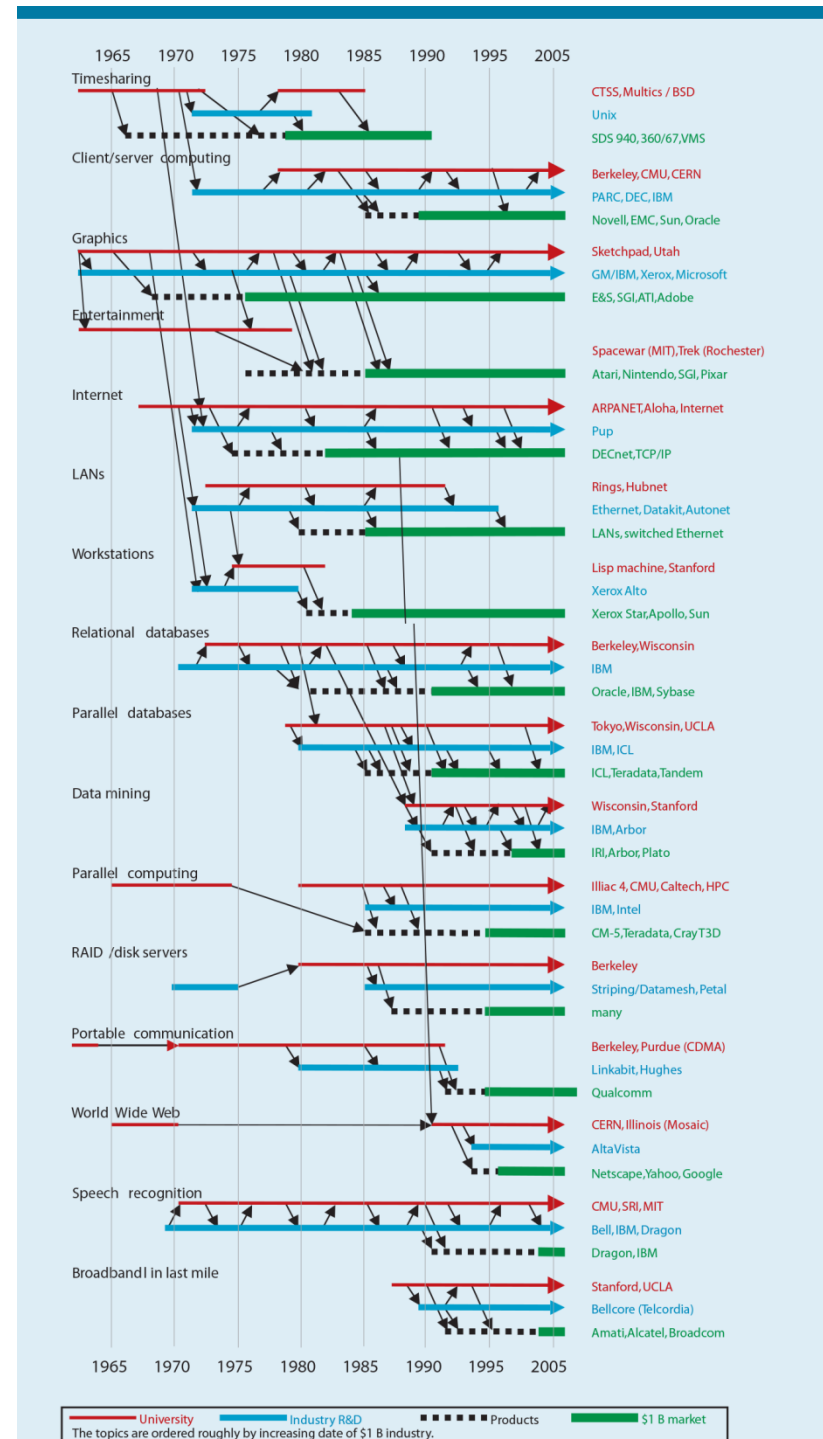
adaptive systems → MAPE

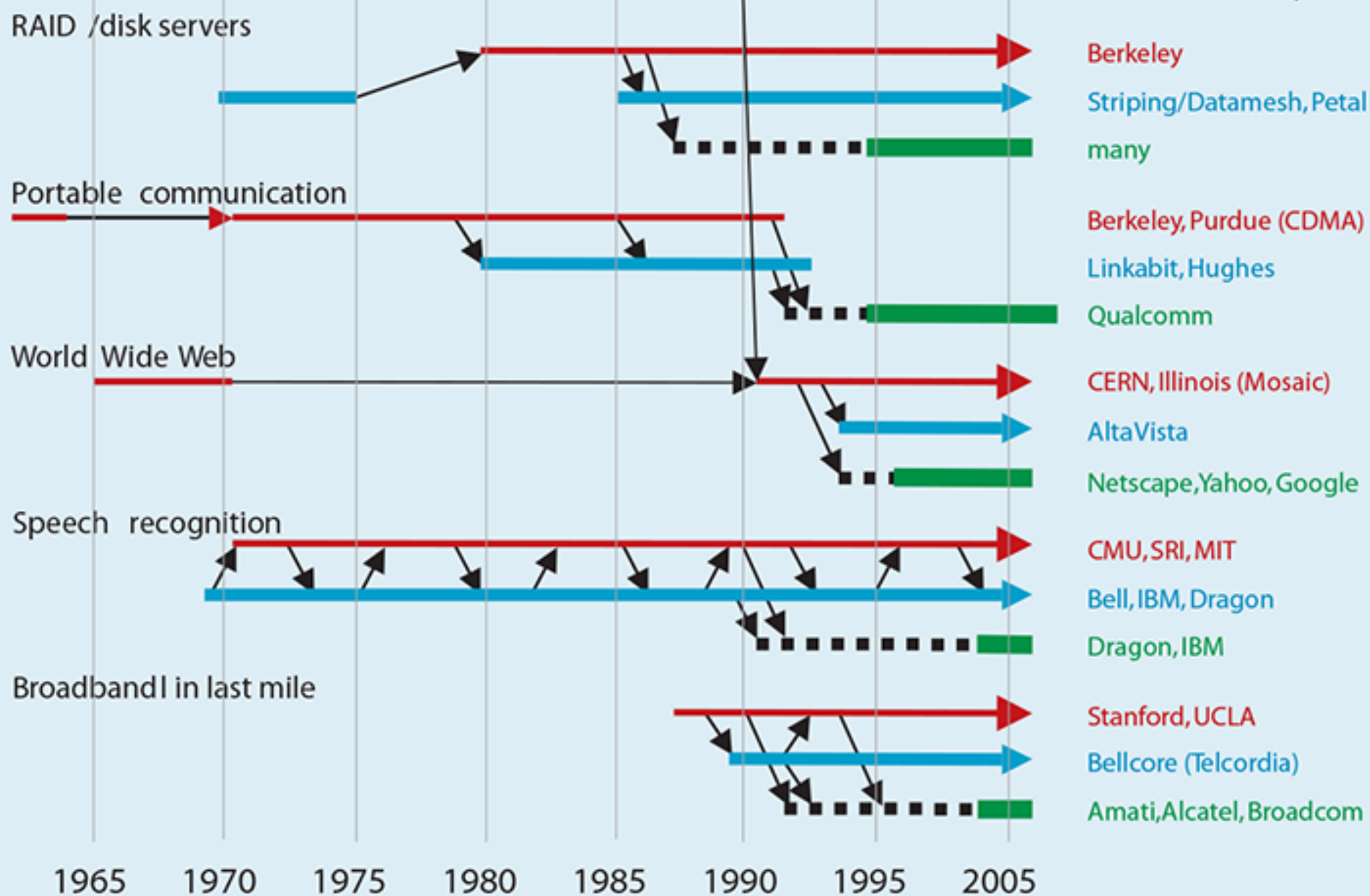


Research feeds practice

Research and development stimulates creation of innovative ideas and industries.

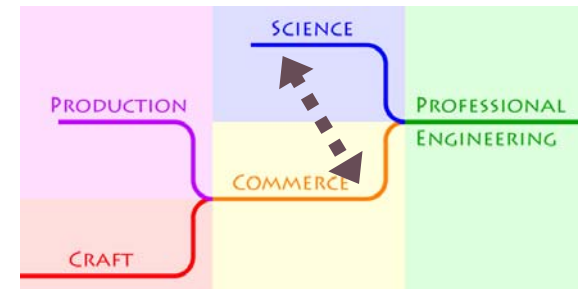
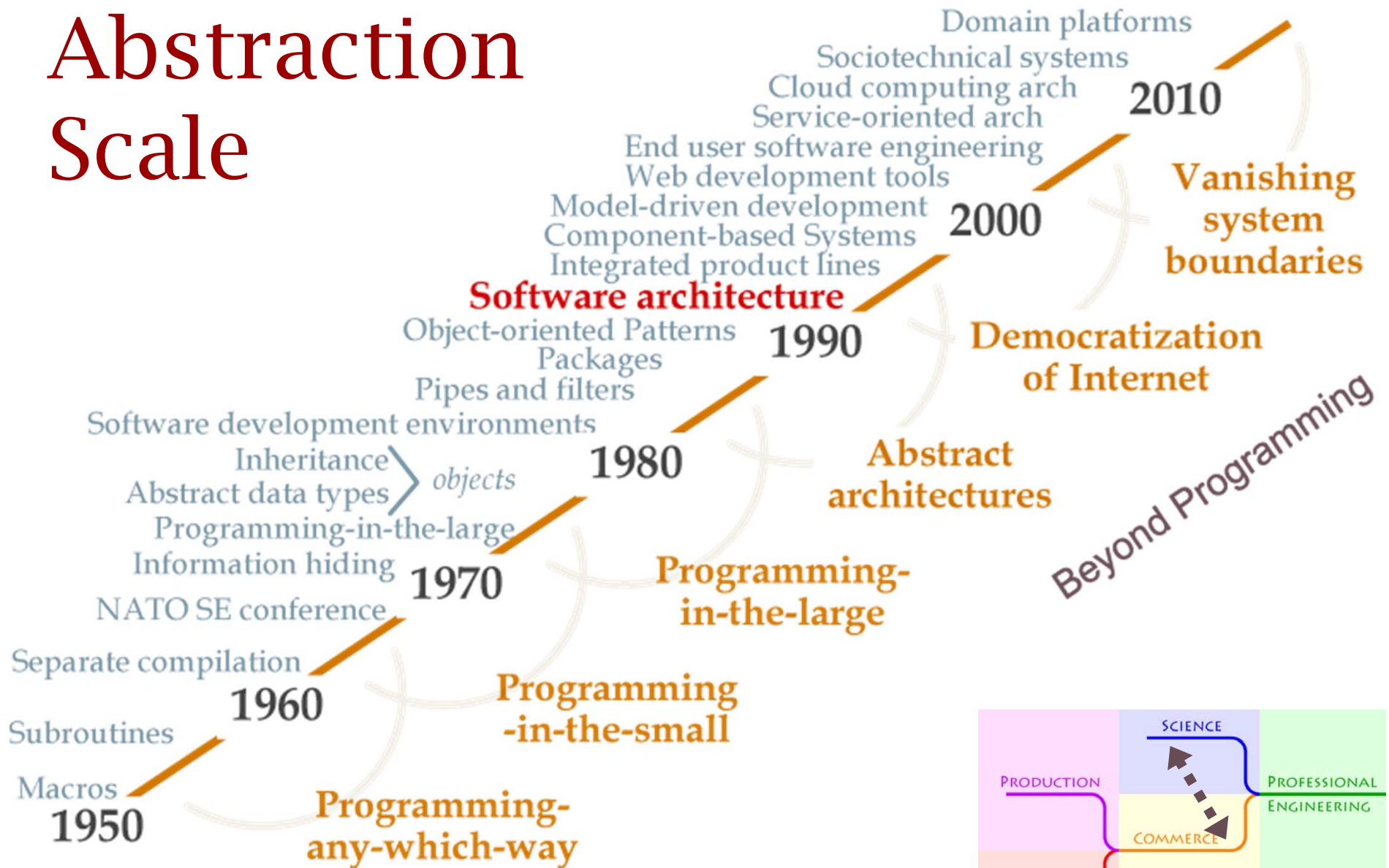
E. Lazowska.
Viewpoint.
CACM Aug 2008





The topics are ordered roughly by increasing date of \$1 B industry.

Increasing Abstraction Scale



Design guidance

Choosing among algorithms based on the problem setting

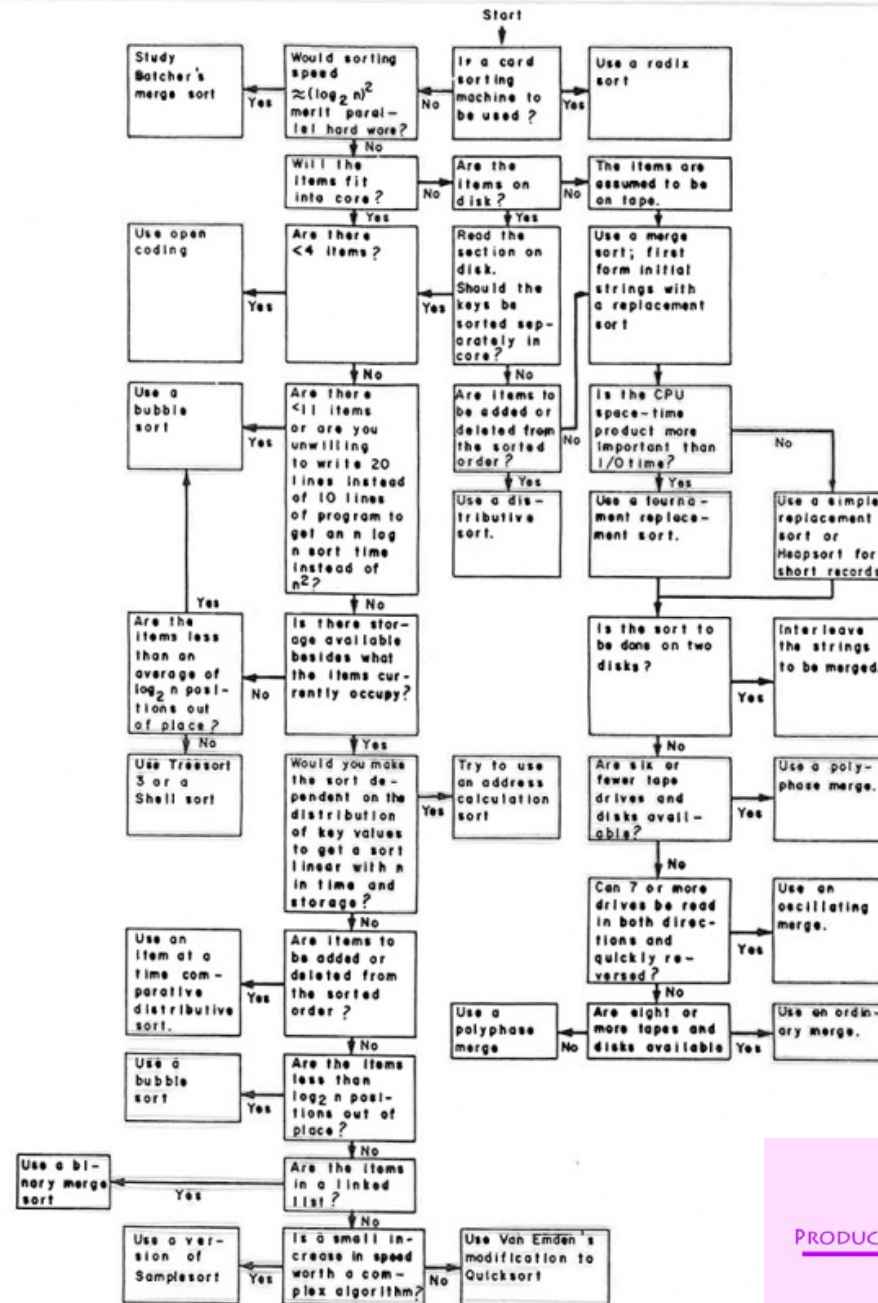
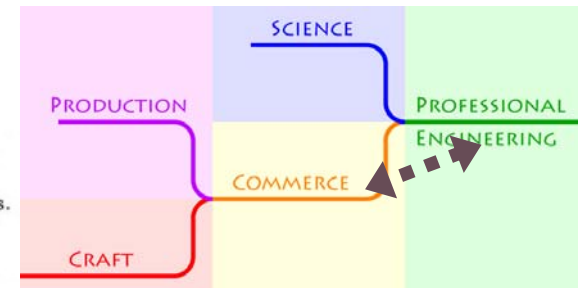
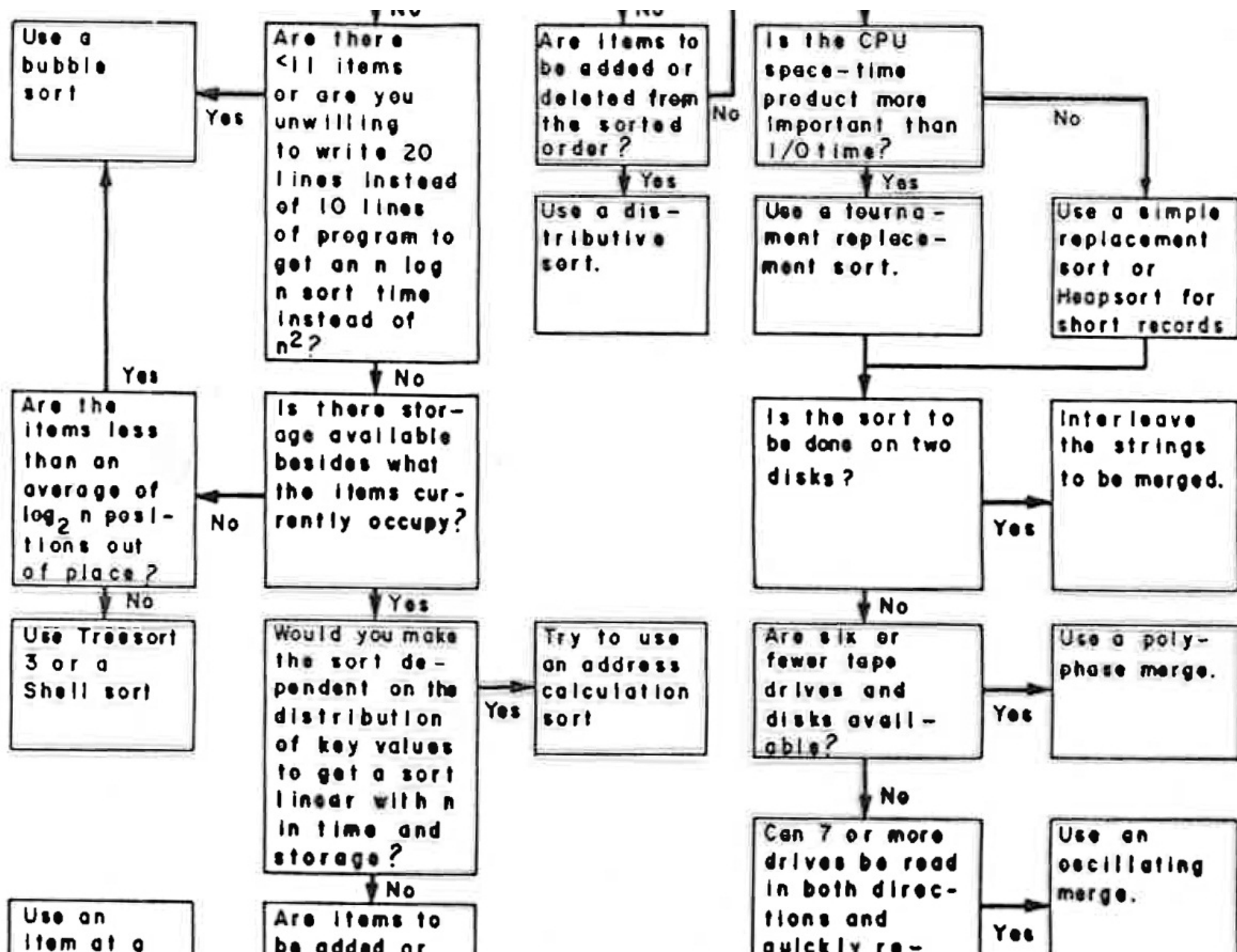


FIG. 21. Suggestions on the use of the various sorting techniques.





Software Architecture

Software architecture ...

- ... is principled understanding of the large-scale structure of software systems as collections of interacting elements
- ... emerged 1990s from informal roots
- ... codifies a vocabulary for software system structures based on types of components and connectors
- ... provides guidance for explicit design choices bridging requirements to code

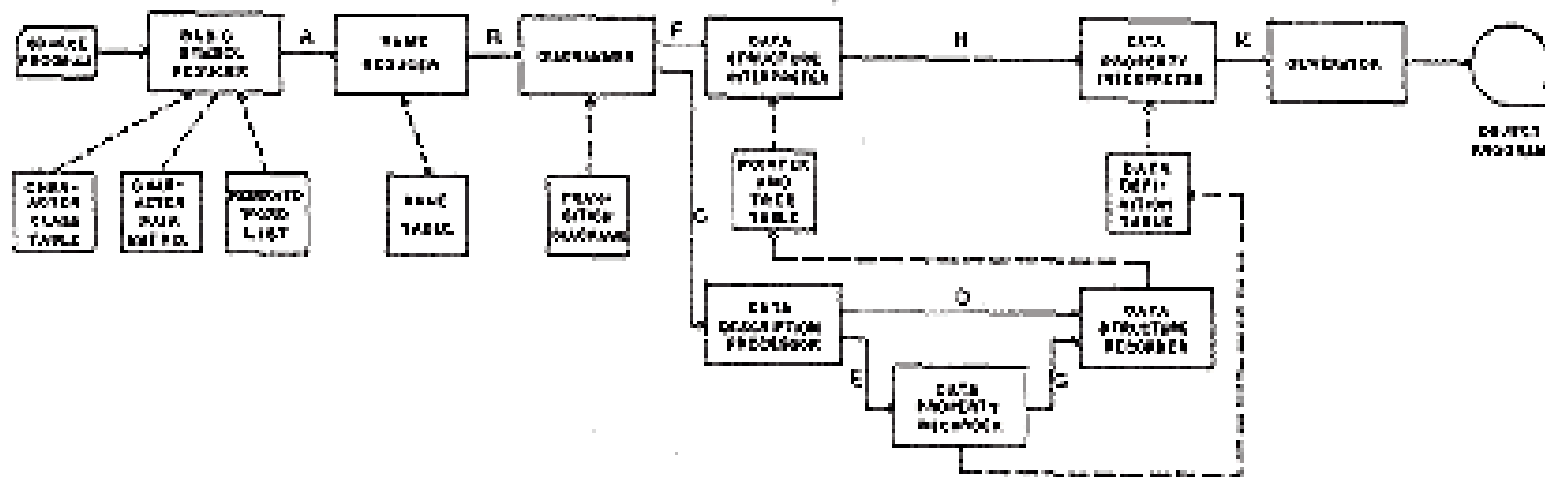


Fig. 4. COBOL Compiler Organization

with a program transformation

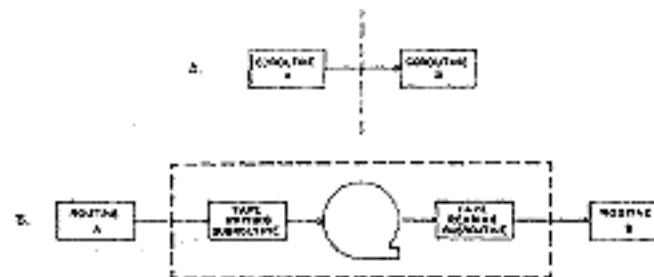
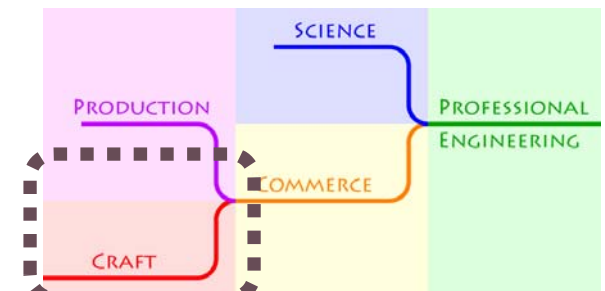


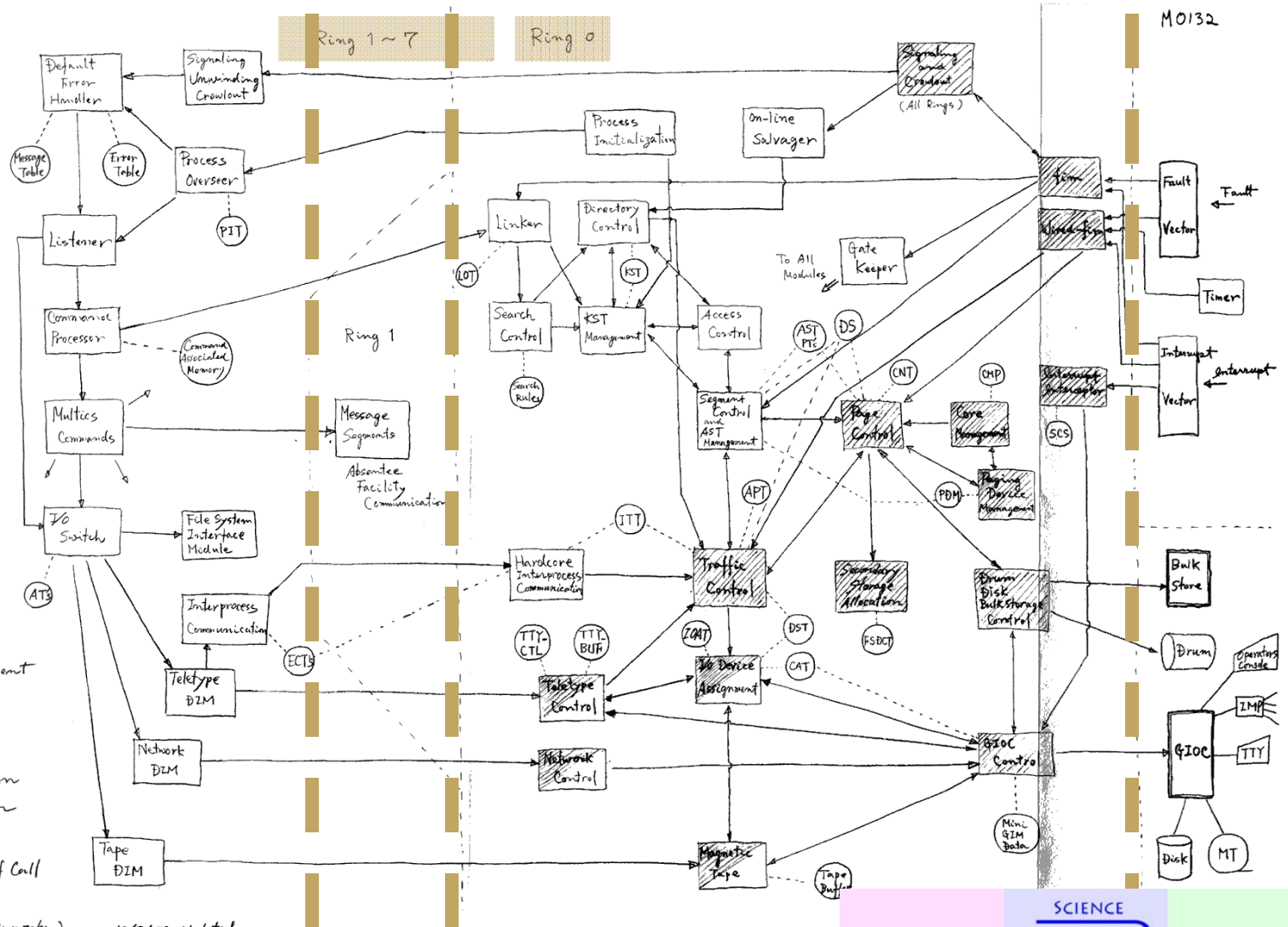
FIG. 3. Illustration of a property of separable programs.
A. A and B, linked as coroutines, communicate directly.
B. A writes its entire output before B reads anything.



A Multics Process (System 17.11a)

PIT: Process Initialization Table
 ATs: Attachment Tables
 ECTs: Event Channel Tables
 TTY-CTL: Teletype Control Driving Table
 TTY-BUF: Teletype Buffers
 LOT: Linkage Offset Table
 ITT: Interprocess Transmission Table
 KST: Known Segment Table
 APT: Active Process Table
 DST: Device Signal Table
 CAT: Channel Assignment Table
 AST: Active Segment Table
 FSDCT: File System Device Configuration Table
 CNT: Counters for Core Metering
 PDM: Paging Device Map
 PTs: Page Tables
 CMP: Core Map
 SCS: System Communication Segment
 IOAT: I/O Assignment Table
 DS: Descriptor Segment

Partially Wired-down
 Wired-down
 Data Base
 Direction of Call



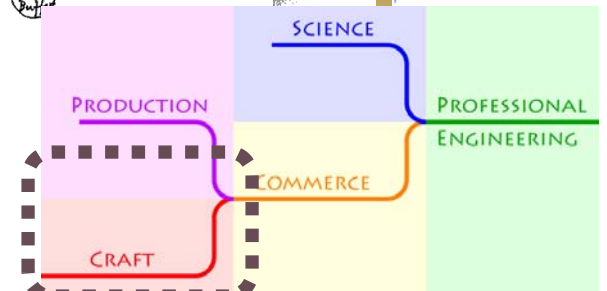
11/29/72

B. Greenberg (Drawn By M. Miyazaki)

12/7/72 Updated

A layered system !!

<http://www.multicians.org/architecture.html>



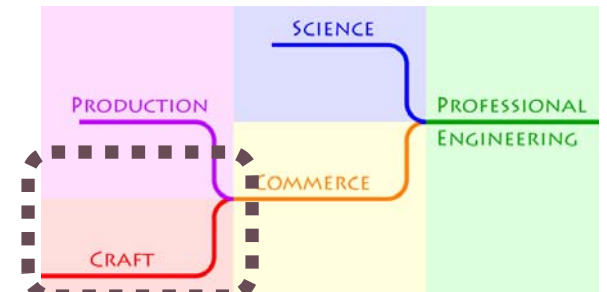
Craft practice

Software has always had structure

- Informal vocabulary
 - Objects, pipes/filters, interpreters, repositories ...
- Intuitions and folklore about fitness to task

Ancient examples (since NATO69) :

- Software bundled with hardware
- Compilers, layered operating systems
- Databases for accounting



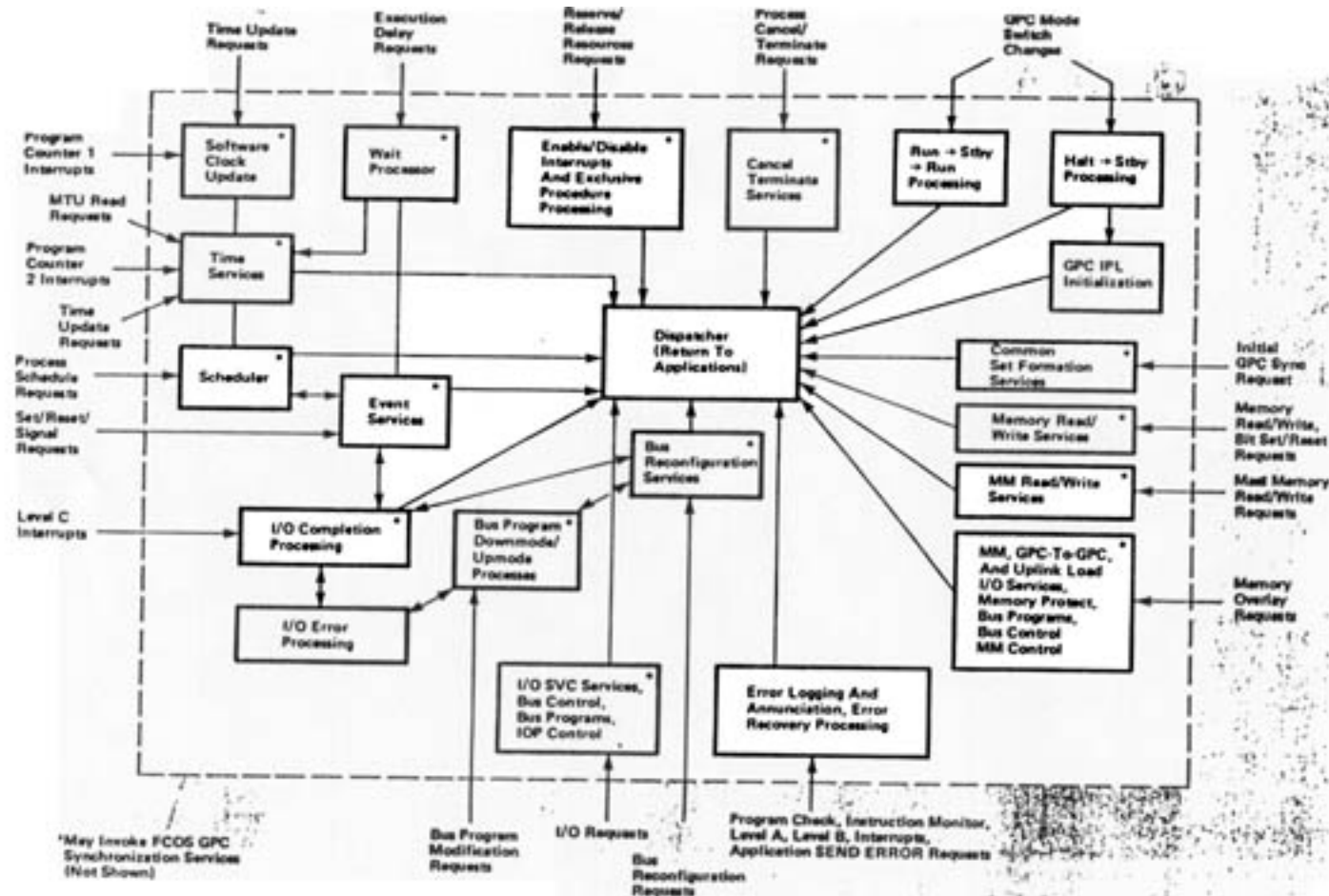
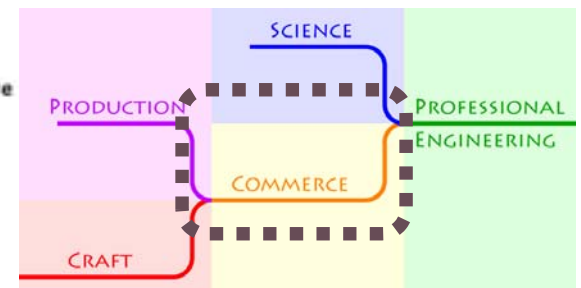


FIGURE 7. Flight Computer Operating System (The FCOS dispatcher coordinates and controls all work performed by the on-board computers.)

Communications of the ACM, "Architecture of the Space Shuttle Primary Avionics Software
September 1984, Vol. 27, No. 9, P. 933



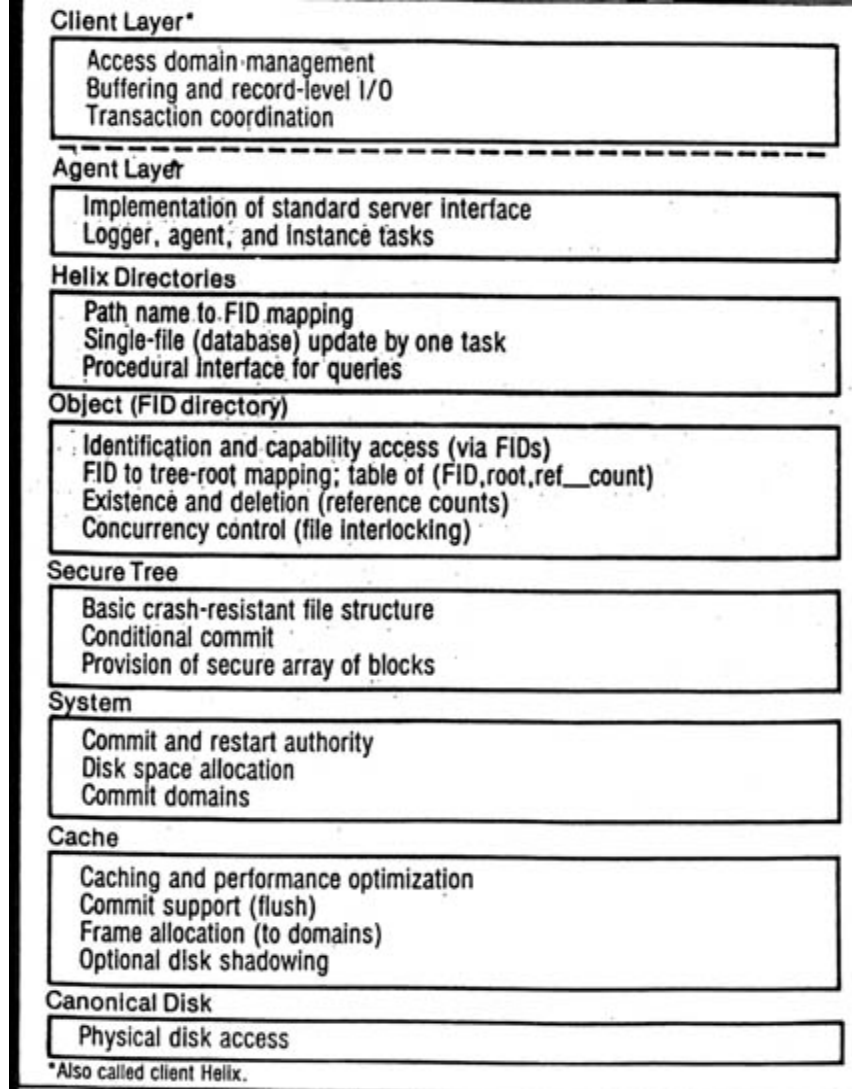
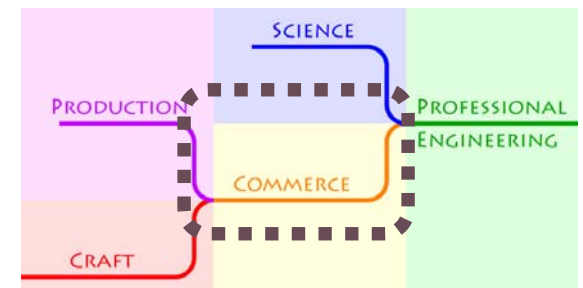


Figure 2. Abstraction layering.

IEEE Software, "Helix: The architecture of the XMS Distributed File System,"
Marek Fridrich and William Older, May 1985, Vol. 2, No. 3, P. 23



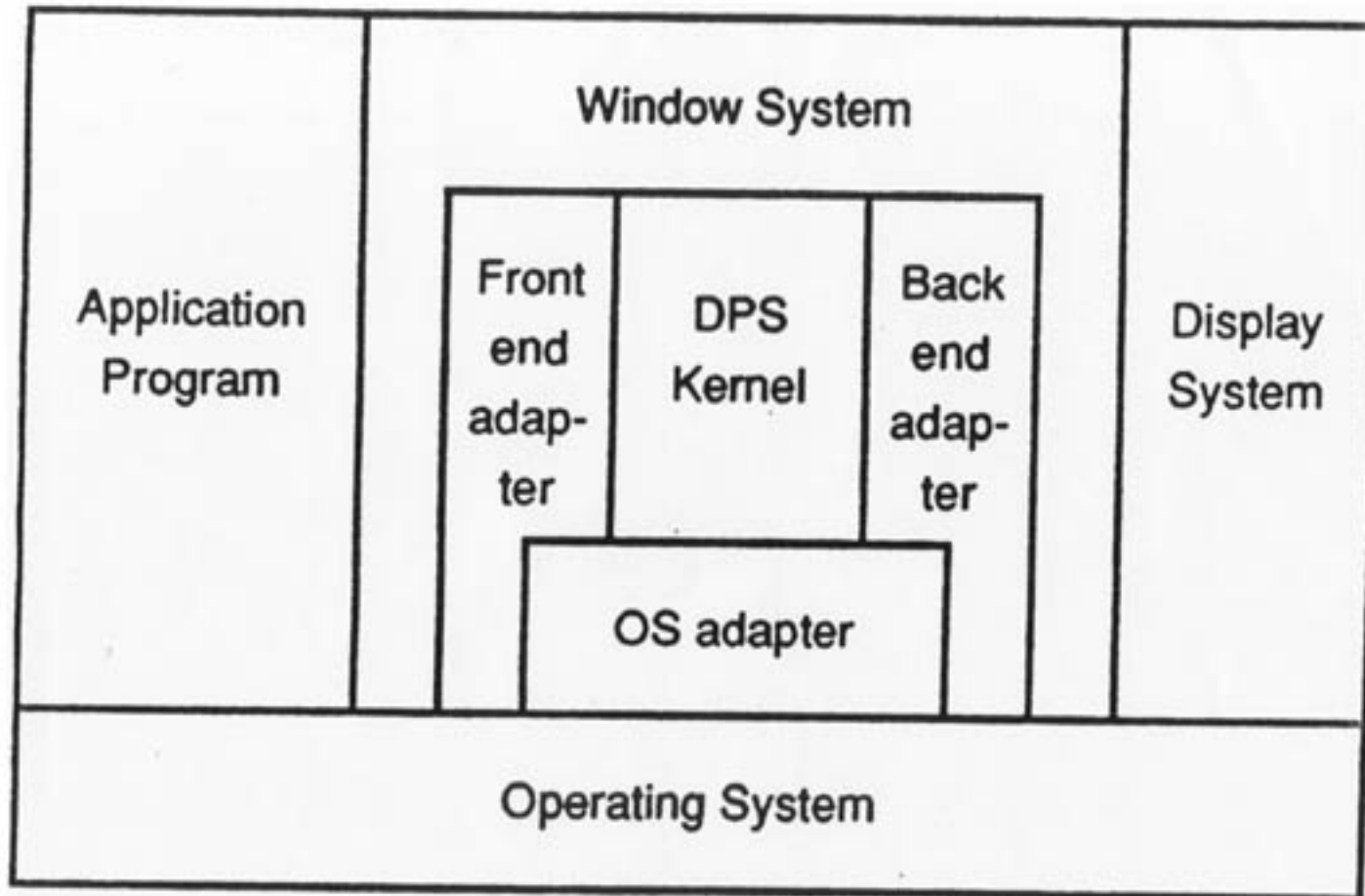
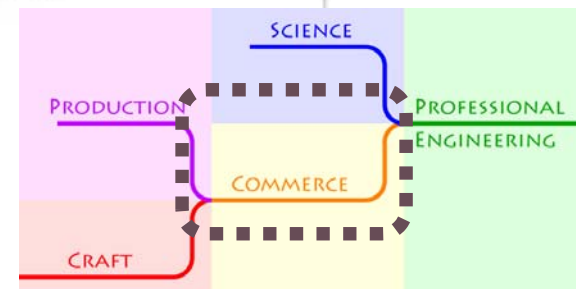
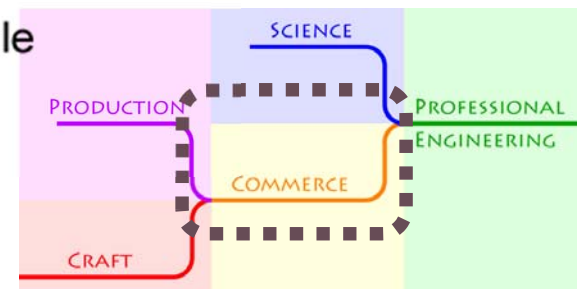
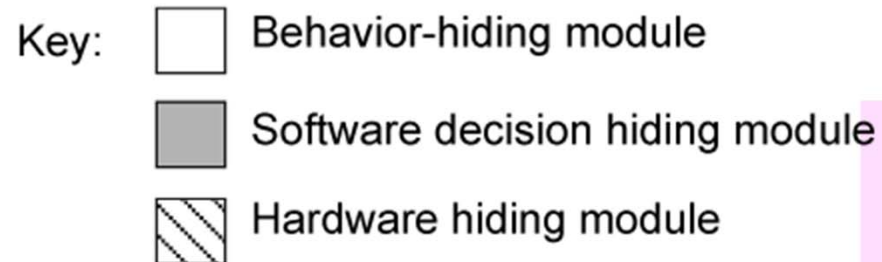
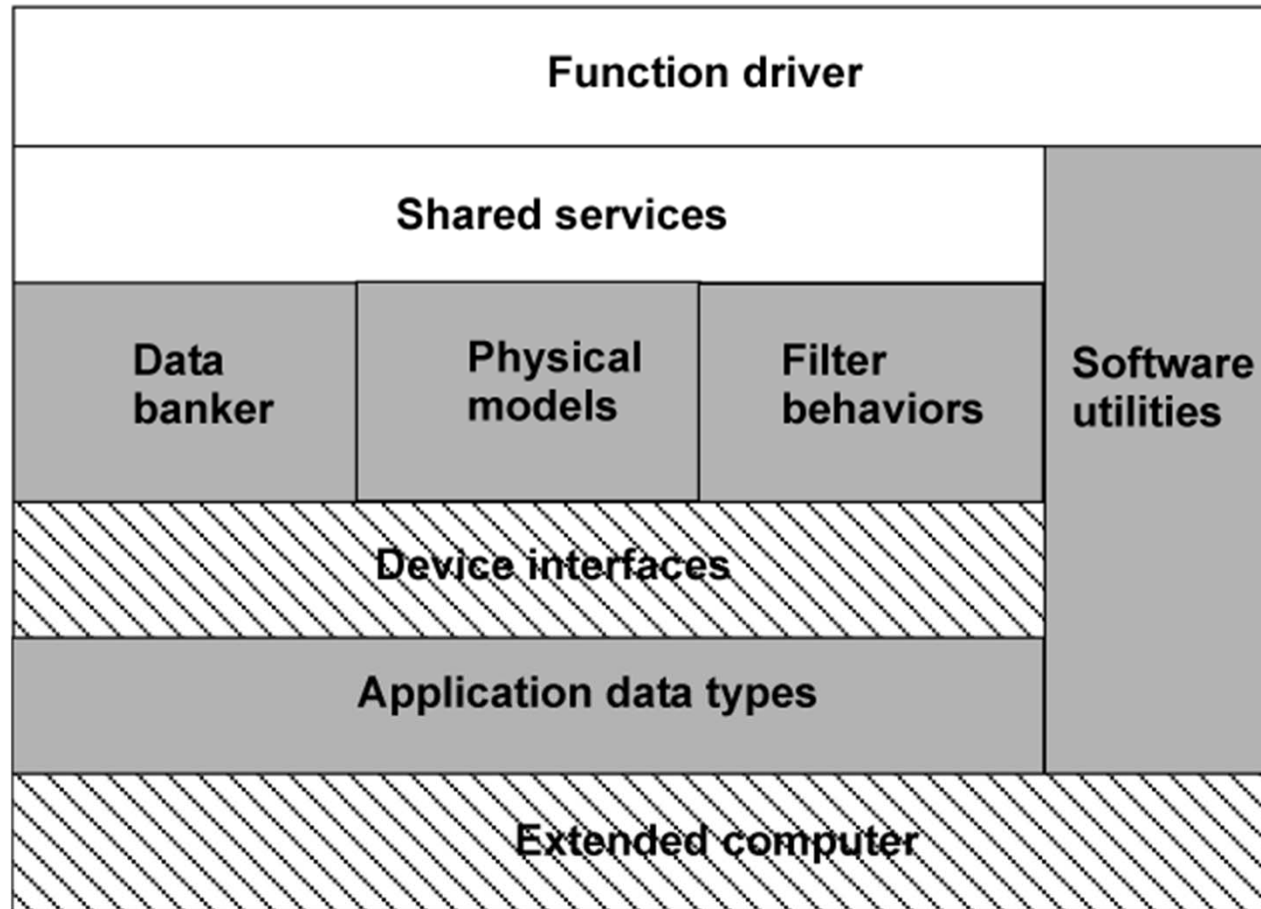


Figure 2. Display PostScript interpreter components.

An Overview of the DISPLAY POSTSCRIPT™ System, Adobe Systems Incorporated, March 16, 1988, P. 10

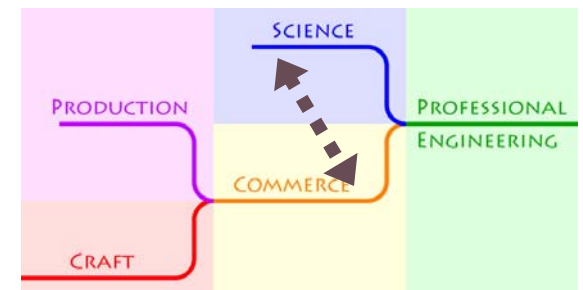




A7E avionics architecture, as shown in Bachman et al
Software Documentation in Practice, SEI 2000

Commerce stimulates science

informal vocabulary → styles/patterns
communication need → notations, ADLs
ad hoc analysis → style-specific analysis
multiple versions → product lines
maintenance issues →
architecture as high-level documentation



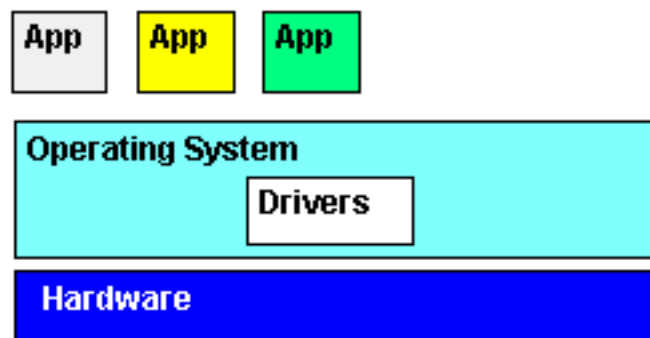
Sample idioms / styles / patterns

- layers
 - virtual machines <hierarchy of abstractions>
 - client-server systems <decomposition of function>
- data flow
 - batch sequential <indep. programs, batch data>
 - pipes and filters <transducers, data streams>
- interacting processes
 - communicating processes <processes, messages>
 - event systems <processes, implicit invocation>

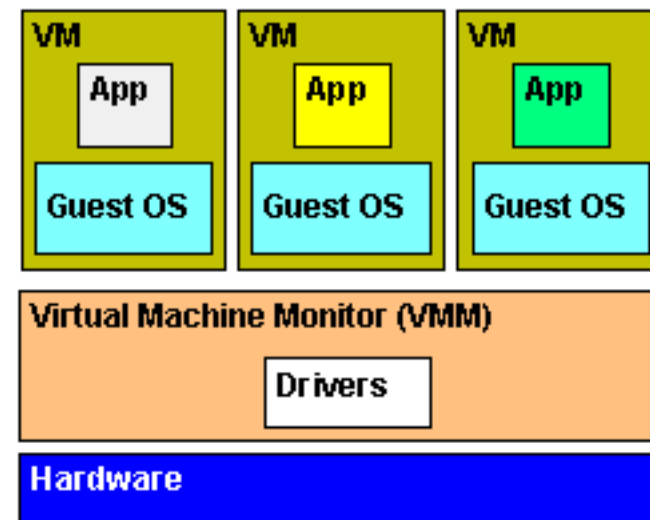
Explanations for practitioners

N-Tier architecture

Non-Virtualized Computer

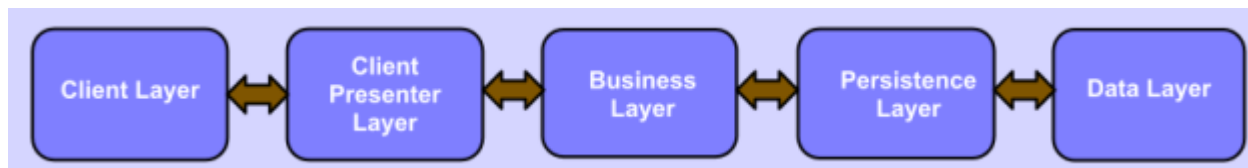


Virtualized Computer

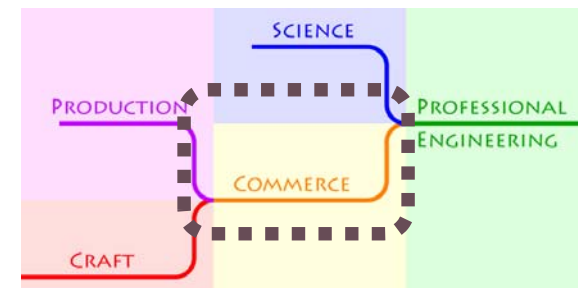


<http://www.pcmag.com/encyclopedia/term/53927/virtual-machine>

Virtual machine



<http://www.codeproject.com/Articles/430014/N-Tier-Architecture-and-Tips>



Commercial practice

1970s: batch processing

- modules and procedure calls, Cobol

1980s: informal “architecture” in papers

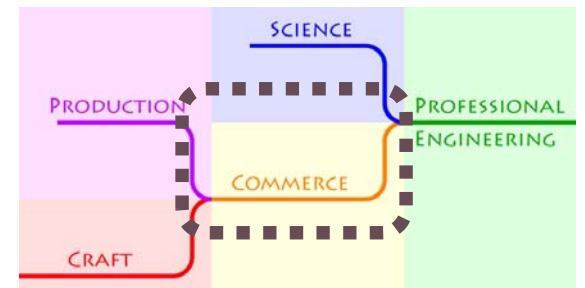
- colloquial use of architectural terms

1990s: early structure

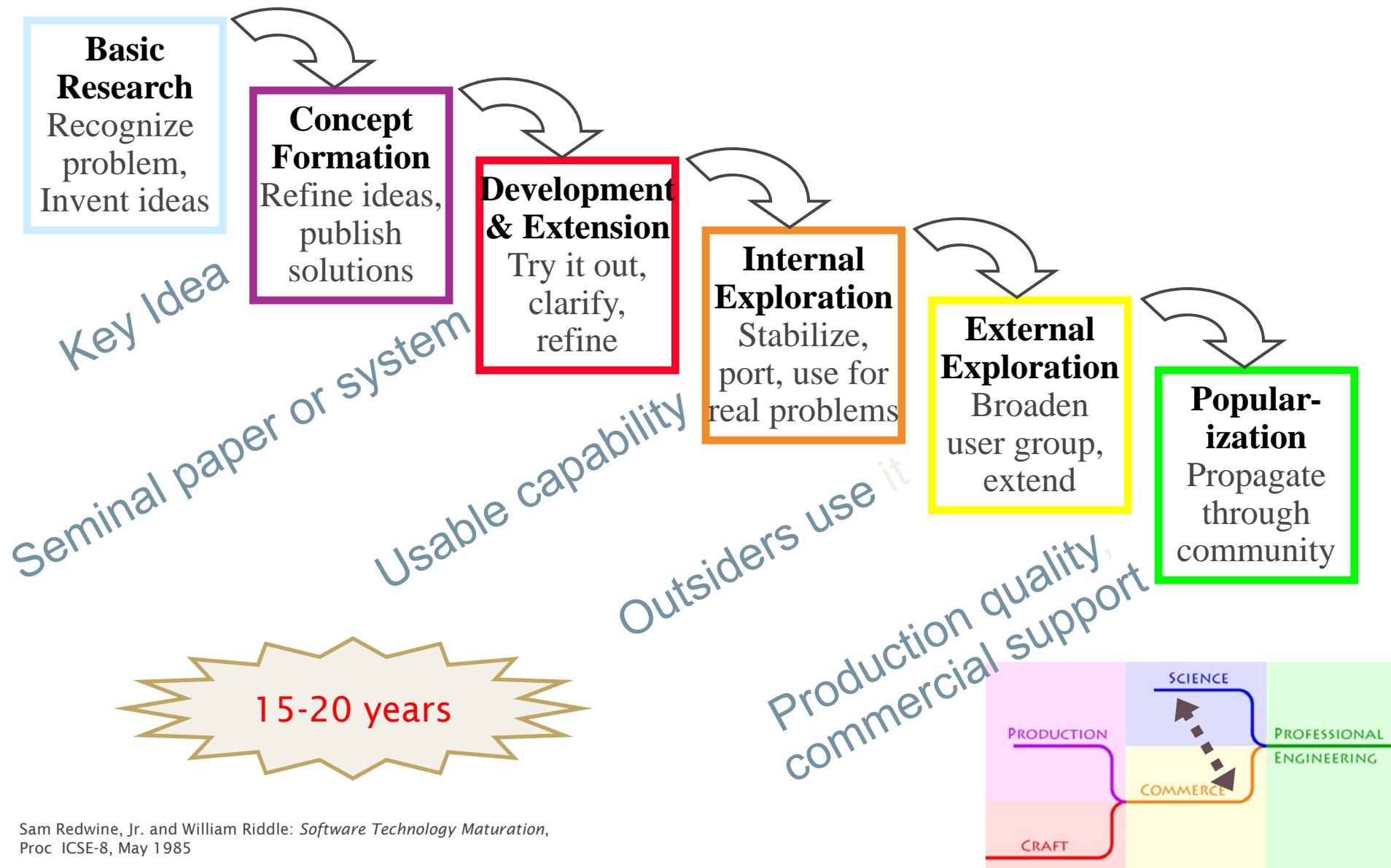
- software product lines

2000s: architecture research enters practice

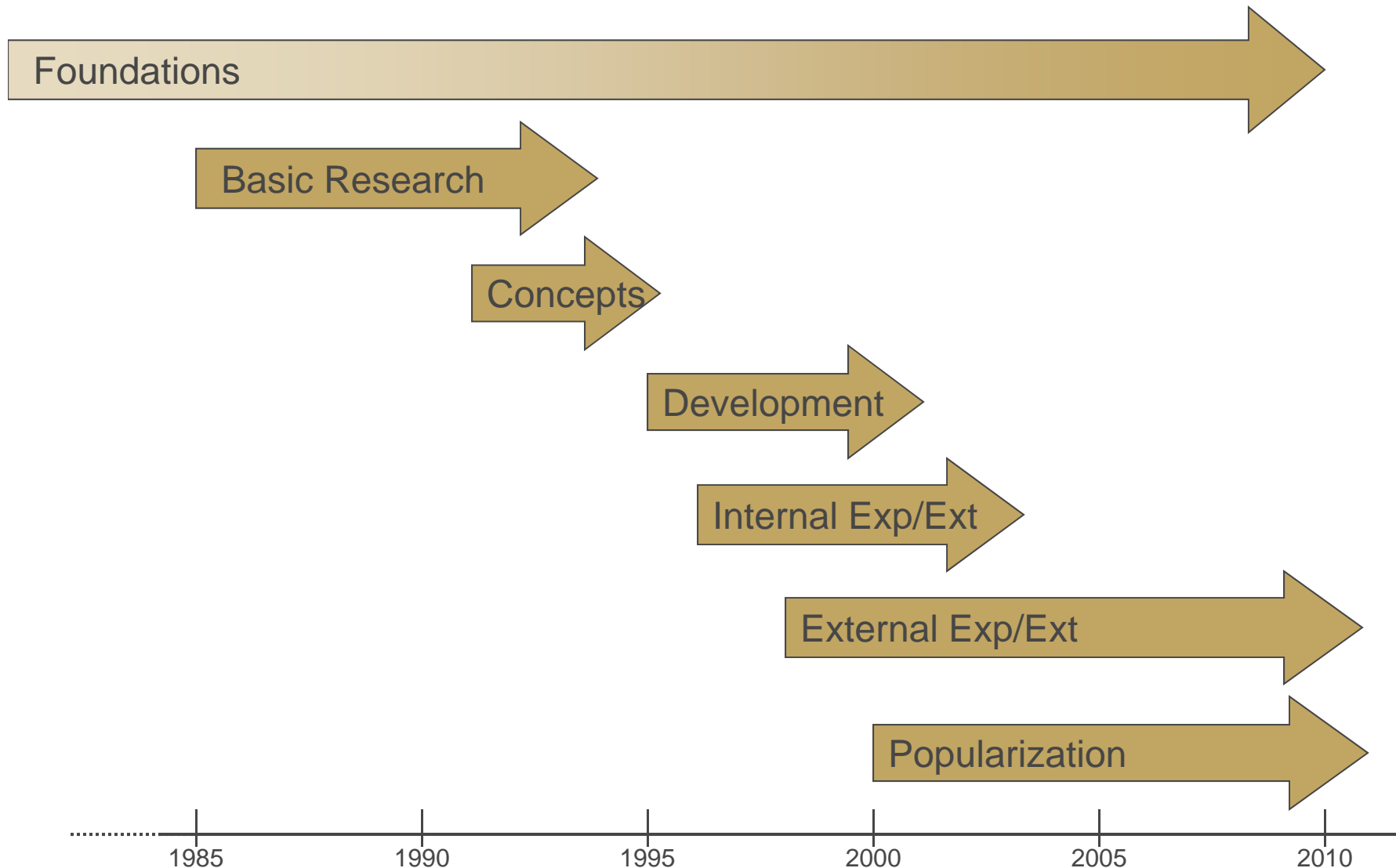
- company-specific overall architectures
- frameworks, UML
- objects everywhere



Maturation of scientific ideas



Maturation of software architecture



Foundations

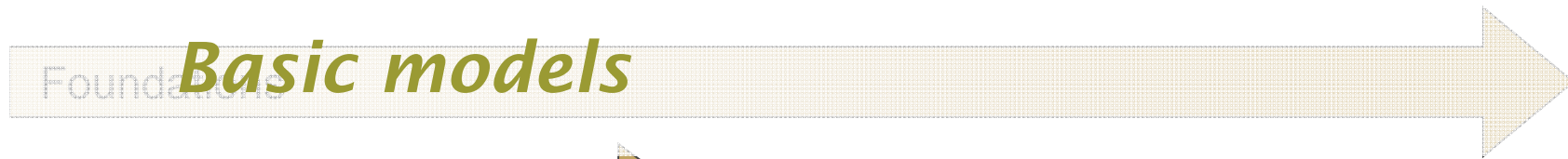
Foundations

information hiding, abstract data types, objects
layered systems
influence of structure on properties
model-driven approaches
computational models and logics
system implementation languages
...etc ...

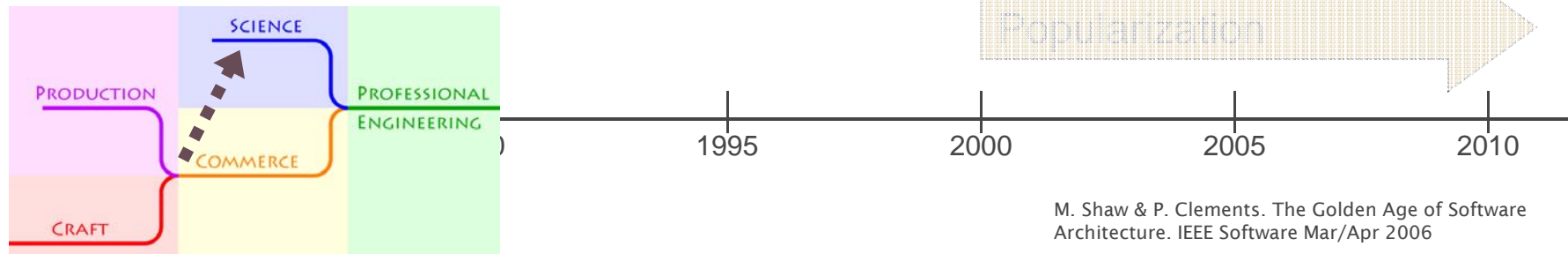
*These concepts evolved on their
own 15- to 20-year cycles;
related concepts continue to evolve*



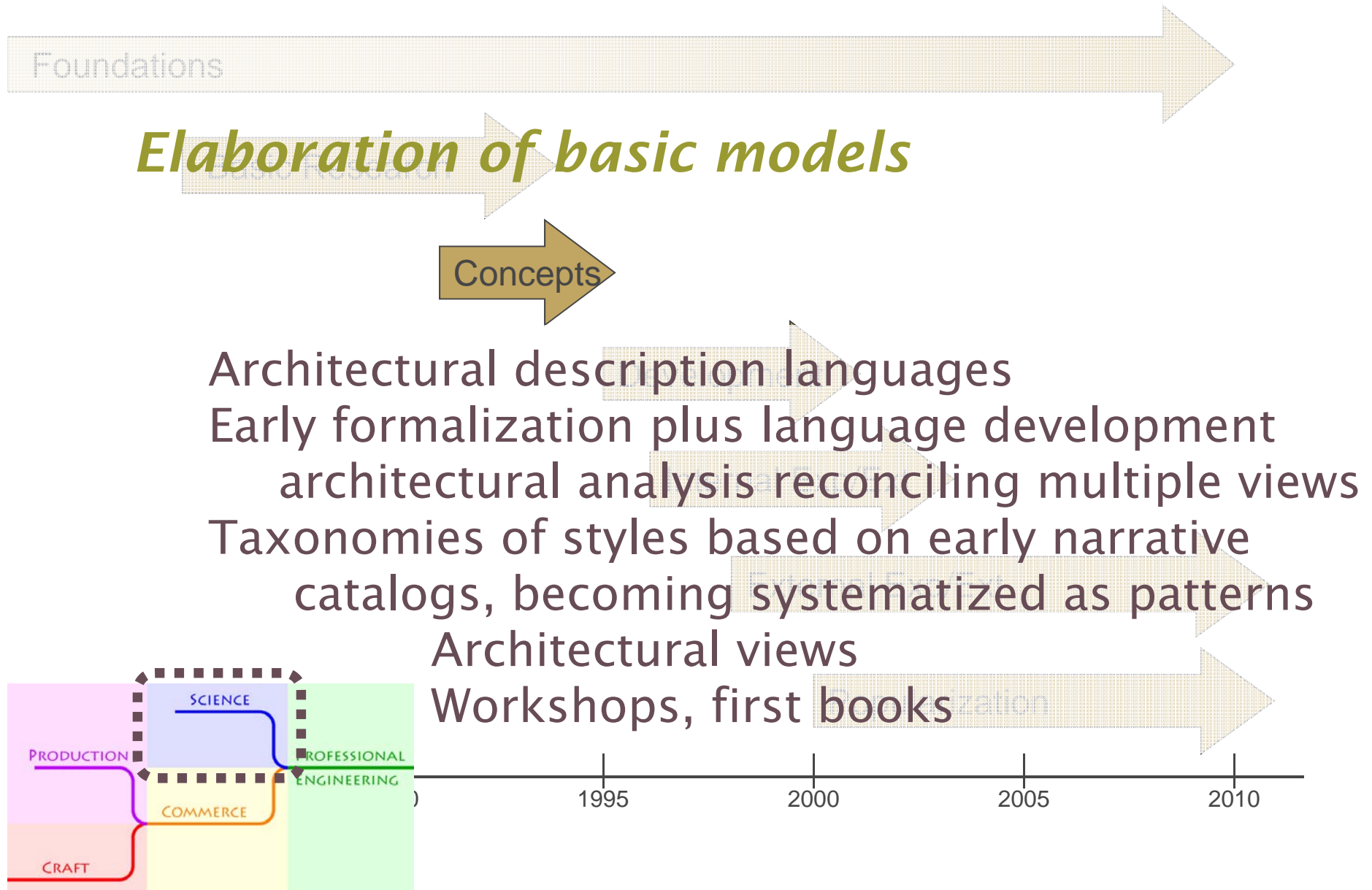
Basic research, 1985-1993



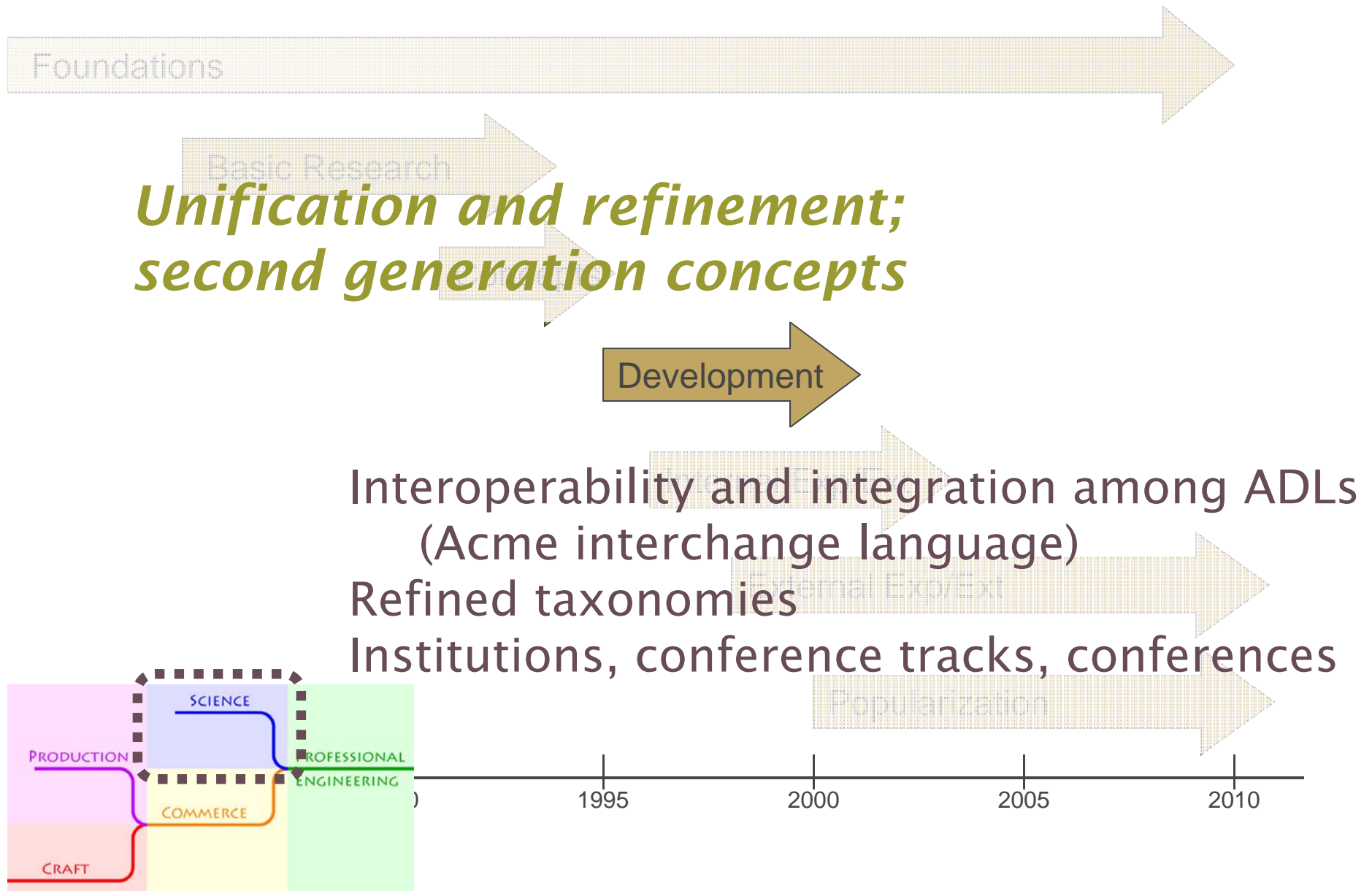
Deliberately designed structures for specific problems
Product line architectures for domains such as
 avionics, oscilloscopes, missile control
Catalogs of common idioms and architectural styles
 to support a design vocabulary
Balanced emphasis on components and connectors



Concept formation 1992-1996



Development & extension: 1995-2000



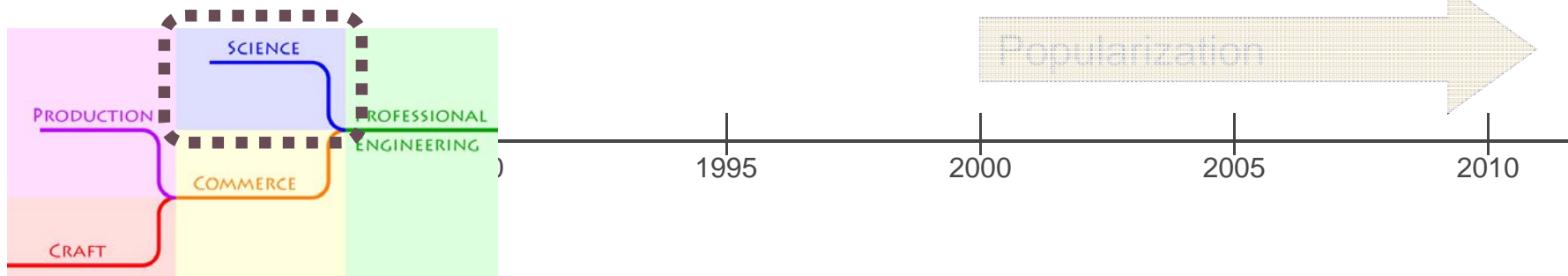
Internal exploration: 1996-2003

Foundations

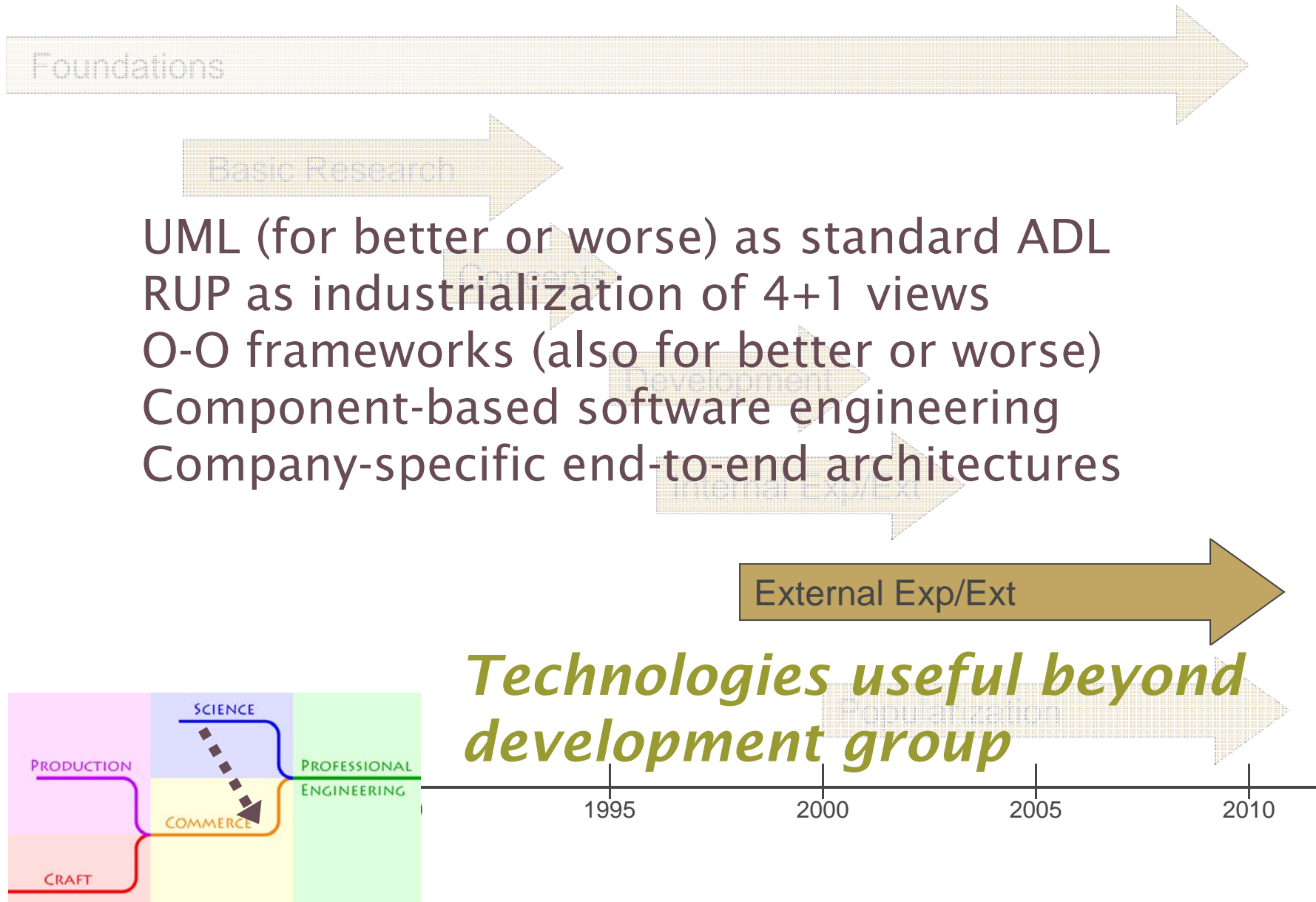
- Increased attention to architecture in design;
- informal use of styles/patterns to guide design
- Some designs formally analyzed
- HLA for distributed simulation
- Connecting architecture decisions to quality attributes
- Analysis and evaluation techniques, SAAM >> ATAM
- Books on practice and on specific aspects

Internal Exp/Ext

*Explicit attention to architecture in design;
architecture's role in quality attributes*



External exploration: 1998-present



Popularization: 2000-present

Production-quality, supported, commercialized marketed versions of the technology

Architectural patterns fueled by web

e.g. n-tier client-server, agent-based, service-oriented
with ecosystem of services, tools, platforms, training

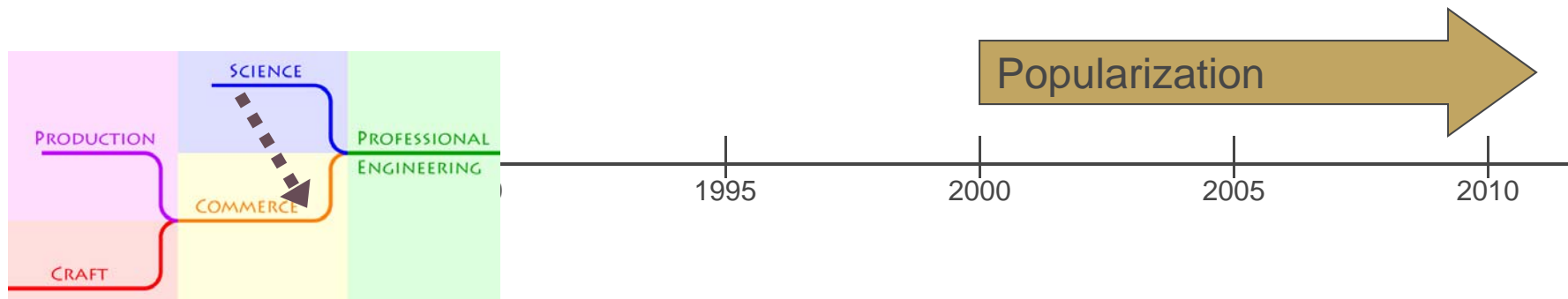
Frameworks and platforms

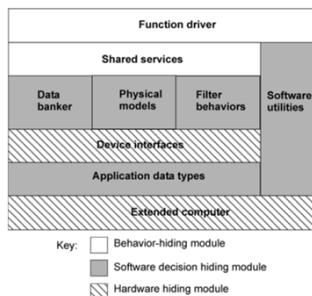
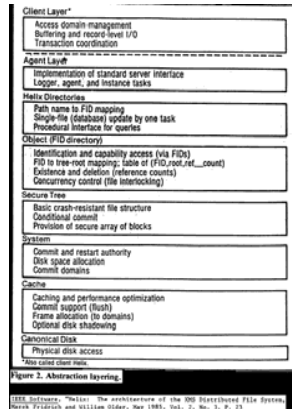
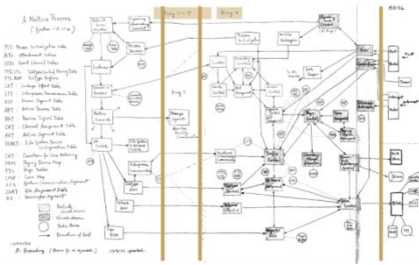
Standards for interfaces and system families

Architect as senior technical leader

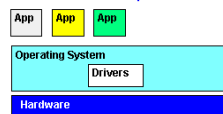
Courses and place in core curriculum

Conferences and organizations for practitioners

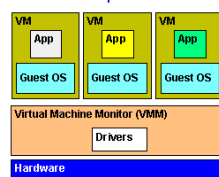




Non-Virtualized Computer

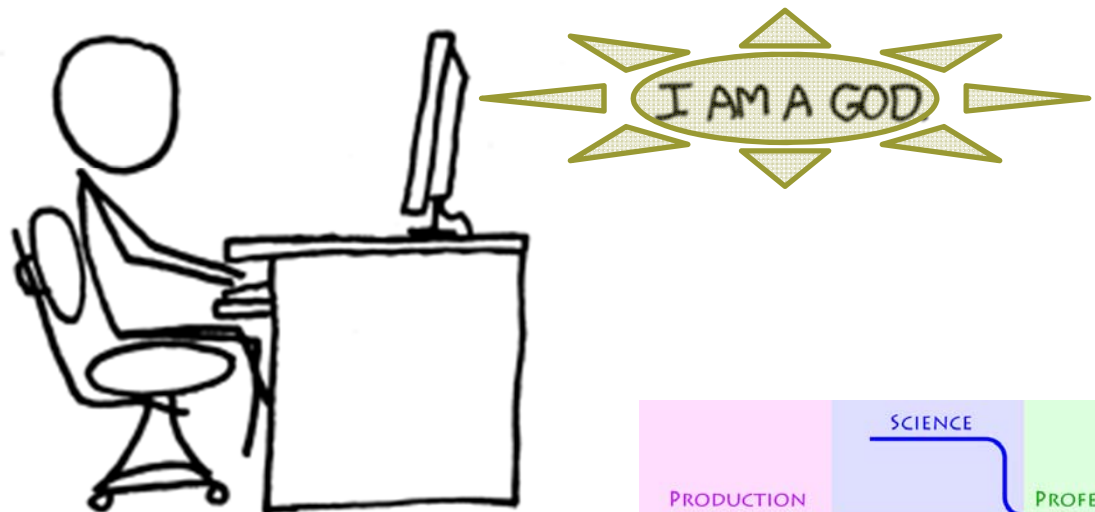


Virtualized Computer

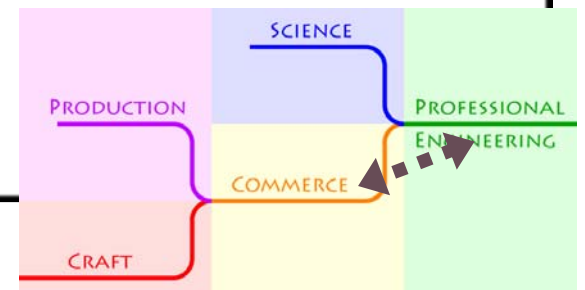


AN **x64 PROCESSOR** IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE **XNU KERNEL**, WHICH IS FRANTICALLY WORKING THROUGH ALL THE **POSIX-SPECIFIED** ABSTRACTION TO CREATE THE **DARWIN SYSTEM** UNDERLYING **OS X**, WHICH IN TURN IS STRAINING ITSELF TO RUN **FIREFOX** AND ITS **GECKO RENDERER**, WHICH CREATES A **FLASH OBJECT** WHICH RENDERS DOZENS OF **VIDEO FRAMES** EVERY SECOND

BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.

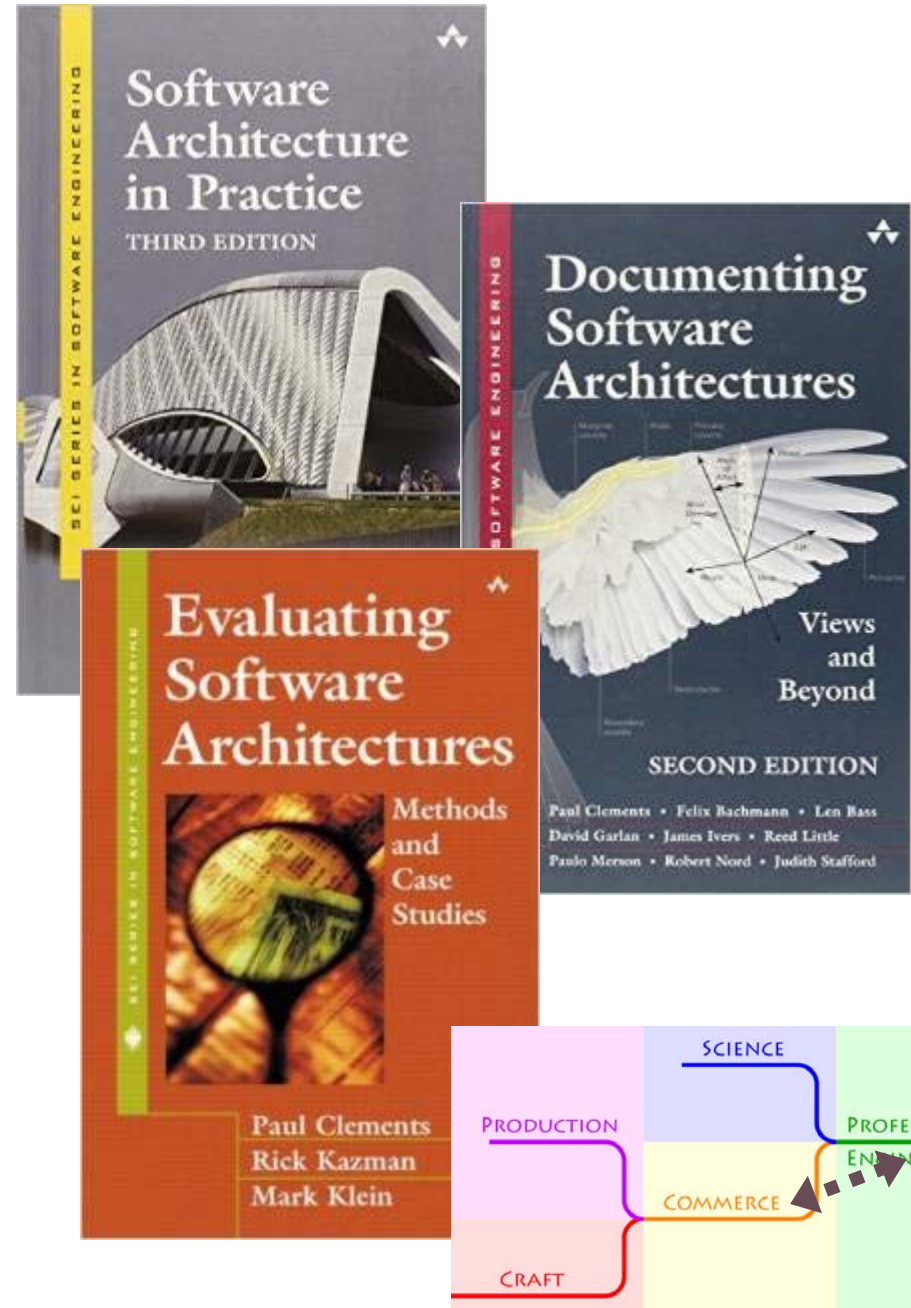


<http://xkcd.com/676/>



Systematically Organized Knowledge

SEI Series organizes
knowledge about
architecture and its
analysis



Architectural styles and reasoning

Style class	Characteristic	Reasoning
Data flow	Styles dominated by motion of data through the system, no “upstream” content control by recipient	Functional composition, latency
Closed loop control	Styles that adjust performance to achieve target	Control theory
Call-and-return	Styles dominated by order of computation, usually with single thread of control	Hierarchy (local reasoning)
Interacting processes	Styles dominated by communication patterns among independent, usually concurrent, processes	Nondeterminism
Data sharing styles	Styles dominated by direct sharing of data among components	Representation
Data-centered repositories	Styles dominated by a complex central data store, manipulated by independent computations	ACID properties, transaction rates, data integrity
Hierarchical	Styles dominated by reduced coupling, with resulting partition of the system into subsystems with limited interaction	Levels of service

Rules of thumb on *data flow*

If your problem is decomposed into sequential stages, consider *batch sequential* or *pipeline* architectures.

If each stage is incremental, so that later stages can begin before earlier stages finish, consider a *pipeline* architecture.

But avoid if there is a lot of concurrent access to shared data.

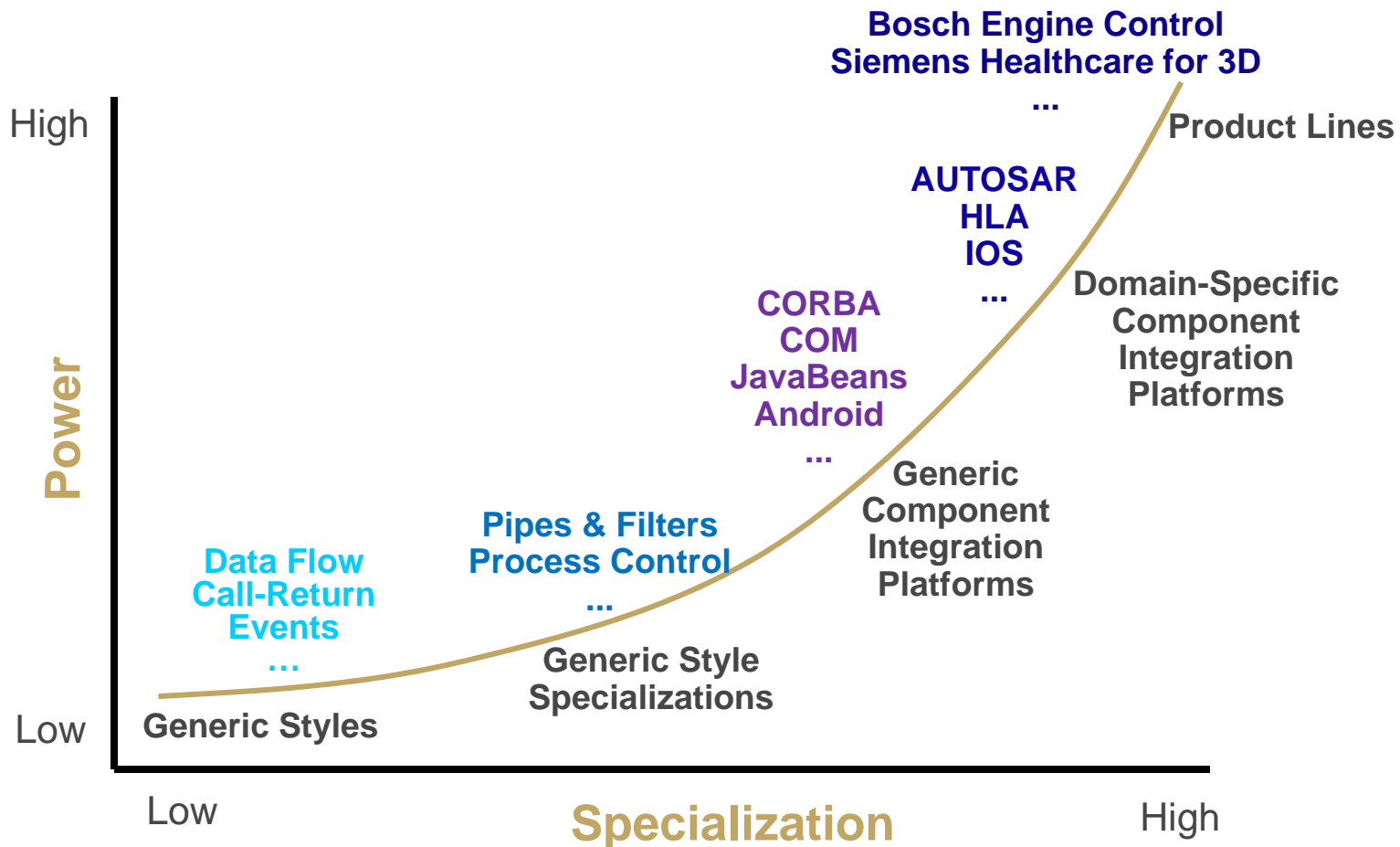
If your problem involves transformations on continuous streams of data (or on very long streams), consider a *pipeline* architecture.

However, if your problem involves passing rich data representations, avoid pipelines restricted to ASCII.

If your system involves controlling continuing action, is embedded in a physical system, and is subject to unpredictable external perturbation so that preset algorithms go awry, consider *closed loop* architectures.

Generality-power trades

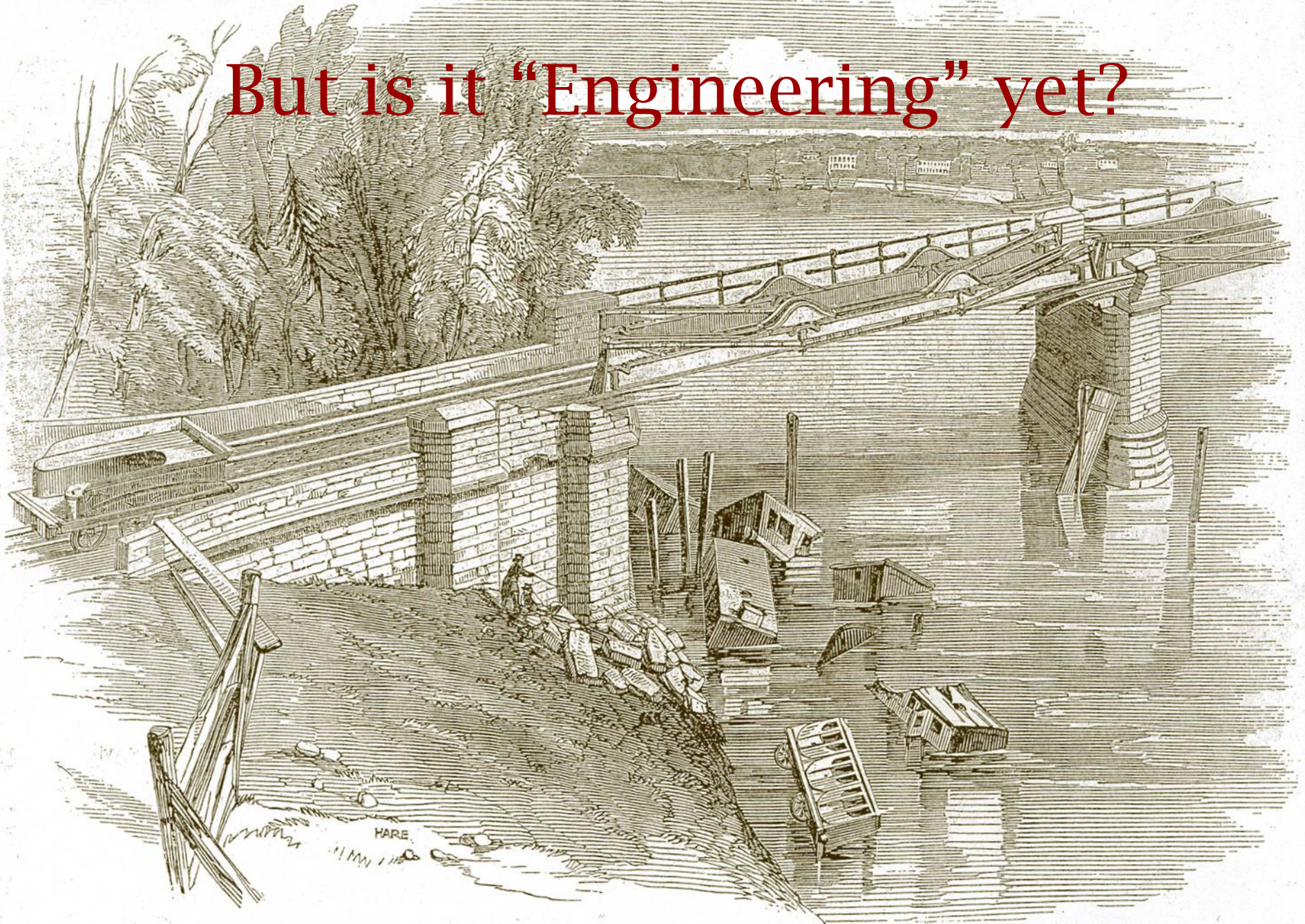
Styles, Platforms, and Product Lines



Status of results

- Not all results are established scientific truths
 - observations and generalizations can be useful
 - Brooks proposes recognizing three kinds of results, with individual criteria for quality:
 - *findings* -- well-established scientific truths -- judged by *truthfulness* and *rigor*
 - *observations* -- reports on actual phenomena -- judged by *interestingness*
 - *rules-of-thumb* -- generalizations, signed by an author but perhaps not fully supported by data) -- judged by *usefulness*
- with *freshness* as criterion for all

But is it “Engineering” yet?



SCENE OF THE LATE RAILWAY ACCIDENT, AT CHESTER.—DILAPIDATED SPAN OF THE DEE BRIDGE.

Illustrated London News, 1847

But is it “Engineering” yet?

“Engineering” is associated with a level of assurance that protects the public health, safety, and welfare.

Consider, though

- Toyota unexpected acceleration
- Retail data breaches (Home Depot, Staples,...)
- HealthCare.gov rollout
- Sony cyberattack
- TurboTax vulnerability
- . . .

Toyota unintended acceleration

- Throttle stuck open, driver couldn't stop car
 - Hundreds died/injured in 2002-2010 models
 - Toyota denied claims but settled for \$1.6++ Billion
- Electronic Throttle Control System (ETCS)
 - wide open throttle → brakes won't stop car
 - single-bit failure could kill critical subtask
- Software didn't follow known good practices
 - watchdog didn't detect major task failure
 - cyclomatic complexity often over 50
 - poor coding practice, ~10,000 global variables
 - recursion could cause uncaught stack overflow
 - poor development/testing process compliance



2014 Data Breach Category Summary

How is this report produced? What are the rules? See last page of report for details. Report Date: 1/5/2015

Page 1 of 1

Totals for Category: Banking/Credit/Financial	# of Breaches: 43 % of Breaches: 5.5%	# of Records: 1,198,492 %of Records: 1.4%
Totals for Category: Business	# of Breaches: 258 % of Breaches: 33.0	# of Records: 68,237,914 %of Records: 79.7%
Totals for Category: Educational	# of Breaches: 57 % of Breaches: 7.3%	# of Records: 1,247,812 %of Records: 1.5%
Totals for Category: Government/Military	# of Breaches: 92 % of Breaches: 11.7	# of Records: 6,649,319 %of Records: 7.8%
Totals for Category: Medical/Healthcare	# of Breaches: 333 % of Breaches: 42.5	# of Records: 8,277,991 %of Records: 9.7%
Totals for All Categories:	# of Breaches: 783 % of Breaches: 100.0	# of Records: 85,611,528 %of Records: 100.0%

2014 Breaches Identified by the ITRC as of: 1/5/2015

Total Breaches: 783
Records Exposed: 85,611,528

US Government Accountability Office reports on Healthcare.gov

Ineffective Planning and Oversight Practices Underscore
the Need for Improved Contract Management

GAO-14-694: Published: Jul 30, 2014.

Contract Planning and Oversight Practices Were Ineffective
Given the Challenges and Risks

GAO-14-824T: Published: Jul 31, 2014.

Actions Needed to Address Weaknesses in Information
Security and Privacy Controls

GAO-14-730: Published: Sep 16, 2014.





Information Security and Privacy Controls Should Be
Enhanced to Address Weaknesses

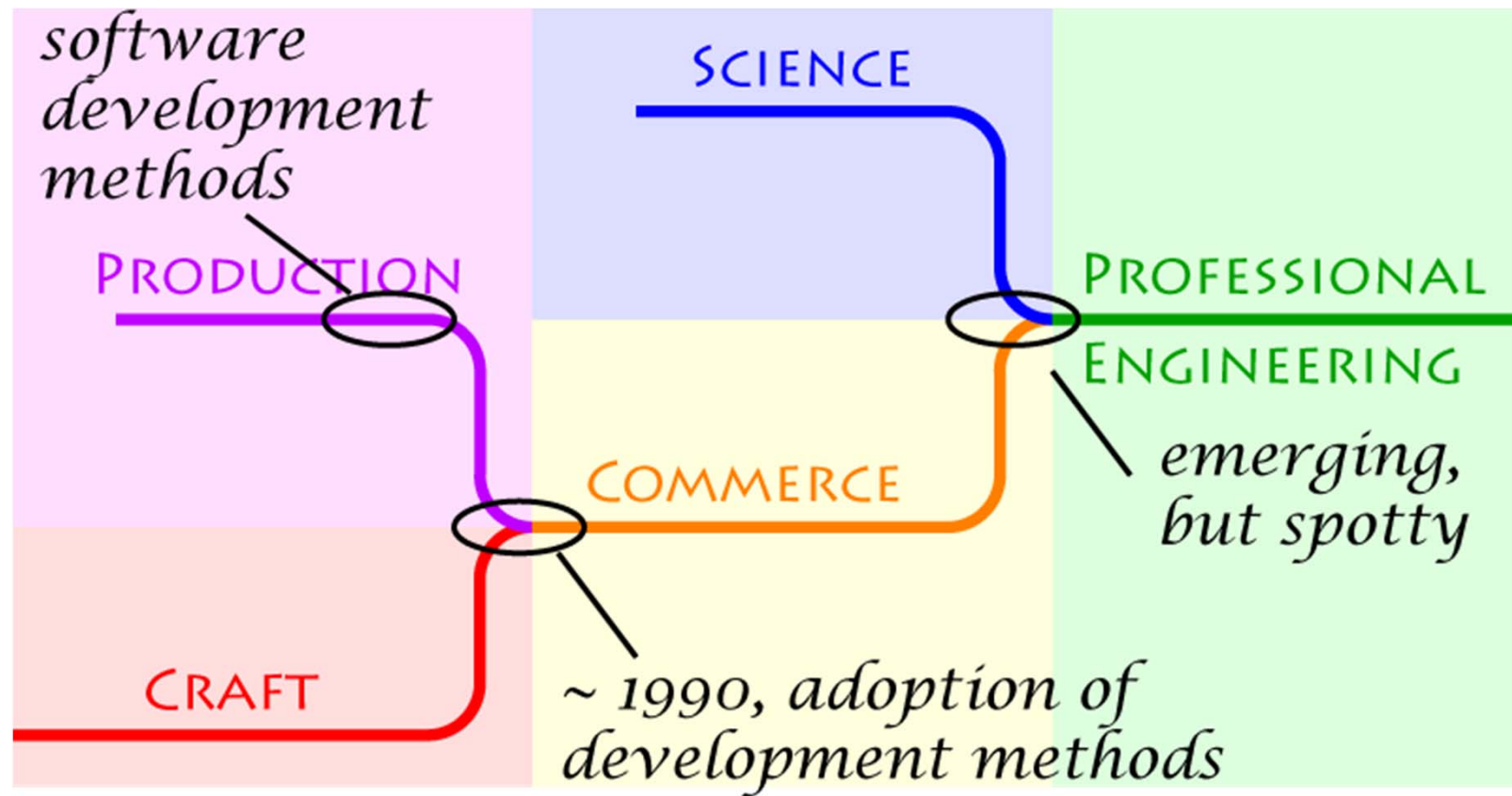
GAO-14-871T: Published: Sep 18, 2014.

CMS Has Taken Steps to Address Problems, but Needs to
Further Implement Systems Development Best Practices

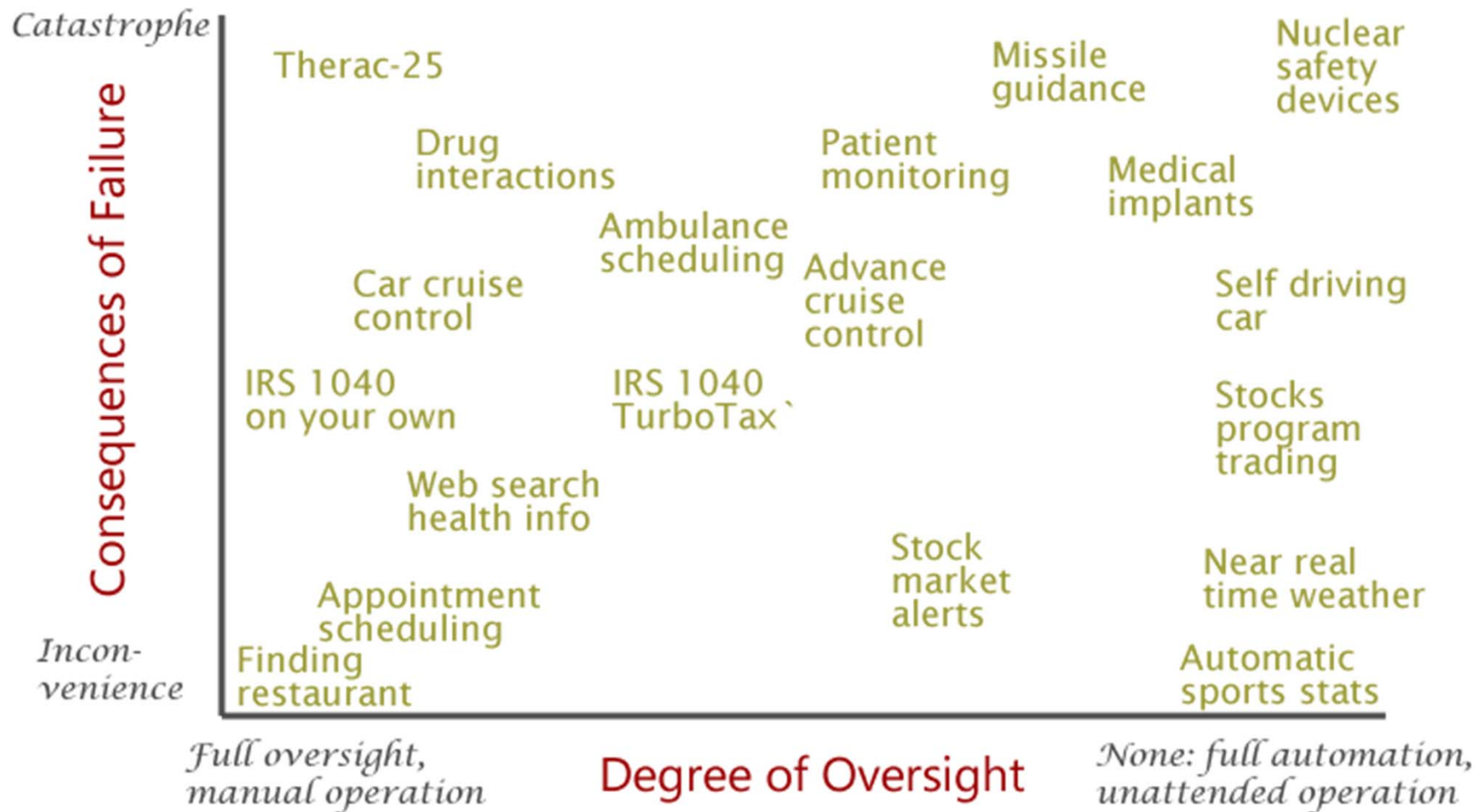
GAO-15-238: Published: Mar 4, 2015.

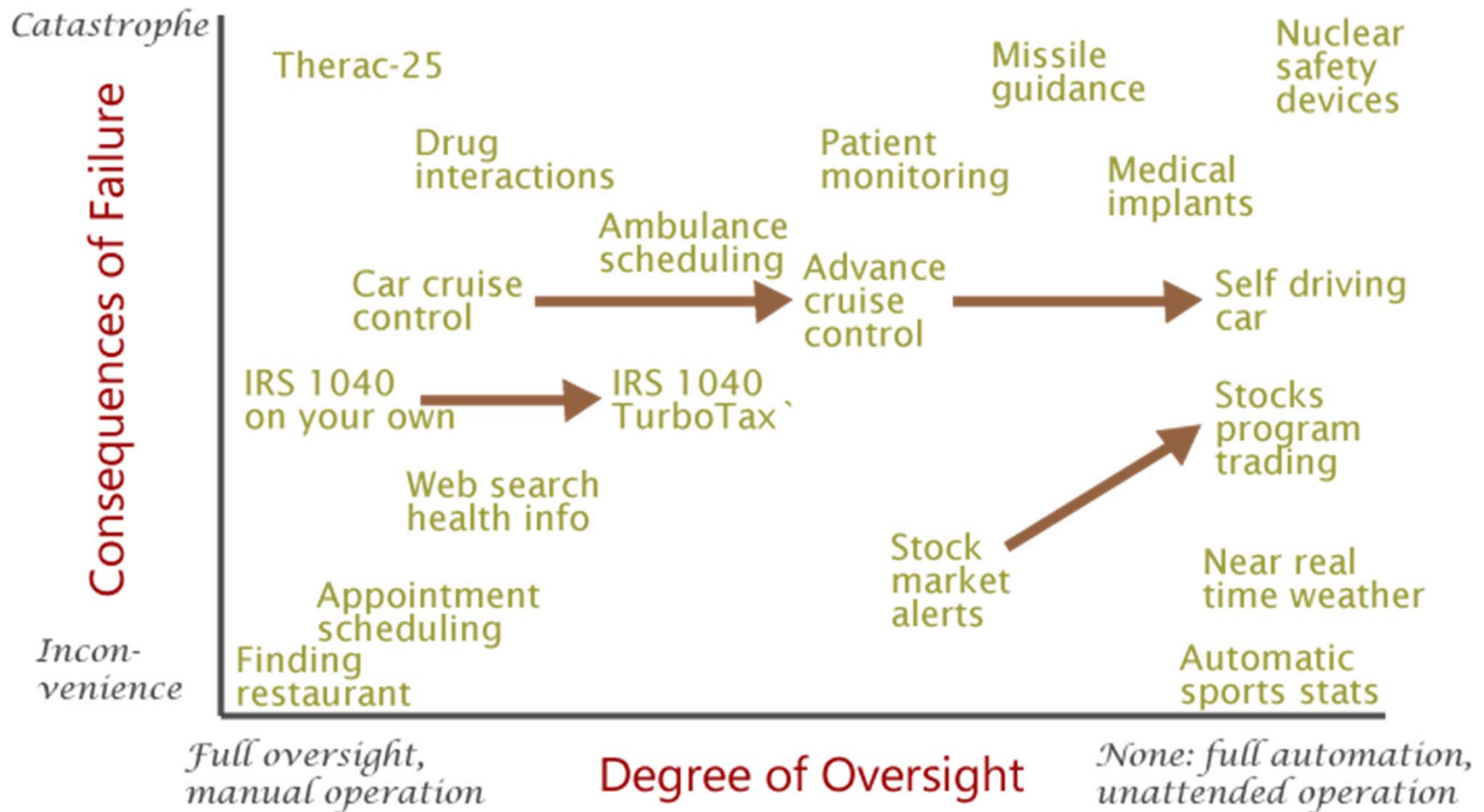
Characteristics of engineering

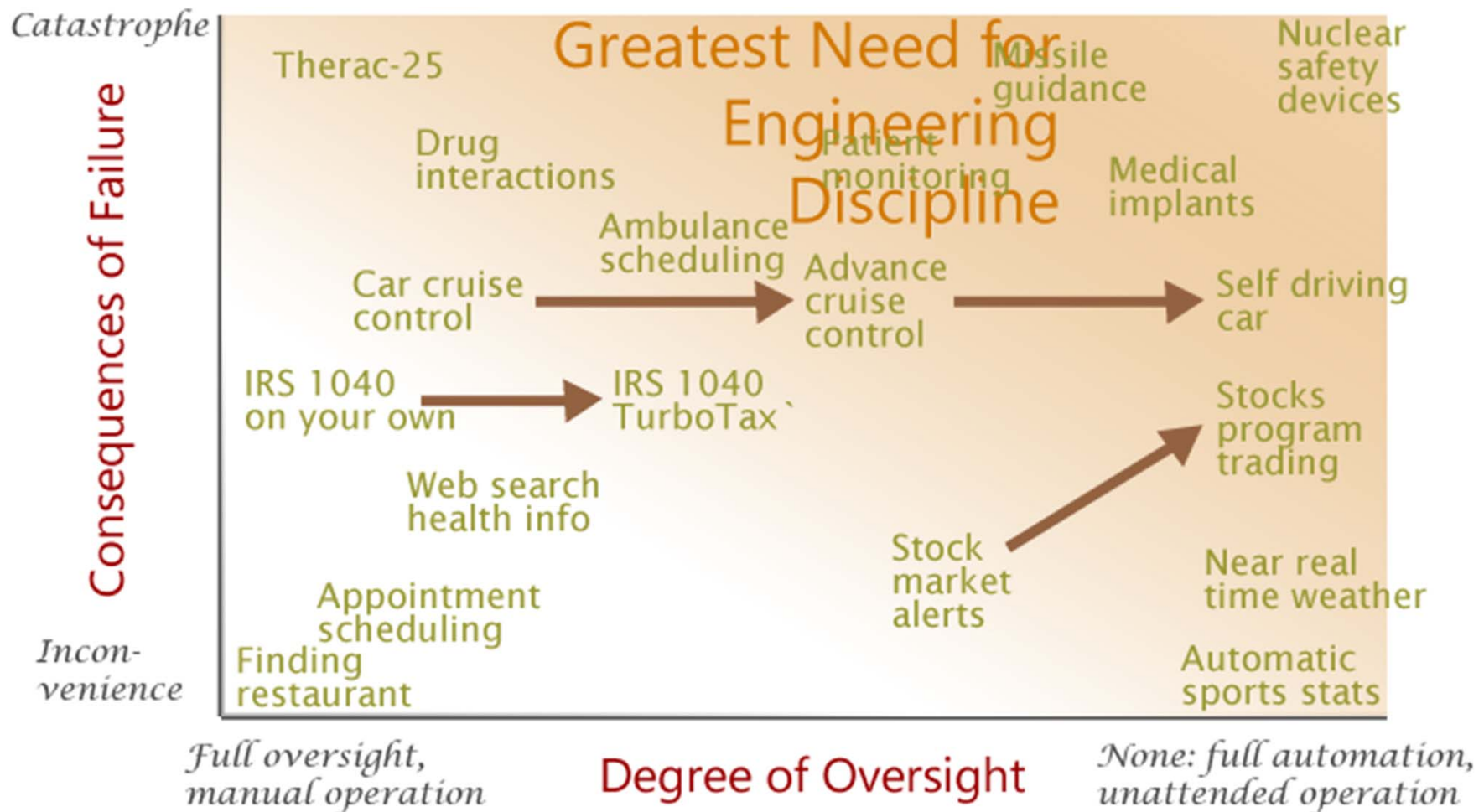
-  limited time, knowledge, and resources force decisions on tradeoffs
-  best-codified knowledge, preferentially science, shapes design decisions
-  reference materials make knowledge and experience available
-  analysis of design predicts properties of implementation



Making Progress







Adapting to evolving technology

- Technology outruns traditional manuals
 - Understand how search supplants indexing
- Agility, “perpetual beta” vs overall design
 - Exploit power end of generality tradeoff, embedding knowledge in task-specific tools
- Scaling cost to consequence
 - High stakes applications have rigorous engineering, mashups are fine for throwaways – but where is middle ground?

*How do we bring codified knowledge to design?
Exhortation won't work*

Architectures at scale

- Highly distributed, dynamically-formed task-specific coalitions of distributed autonomous resources (fix “mashups”)
- Balance among privacy, data quality, data mining (address “credibility” of data)
- Pervasive cyber-physical systems: control, security, adaptation (“Internet of Things”)
- Socio-technical ecosystems: platforms, extensions, and people as part of system (“wicked” problems, end user development)

Civilize the electronic frontier

Infrastructure and amenities

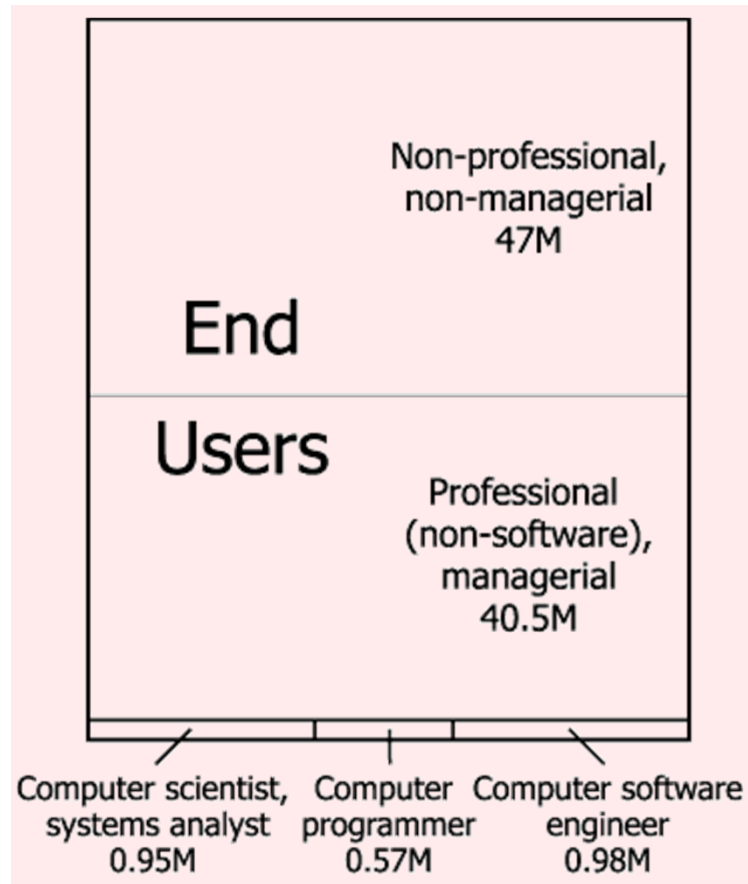
Civil order, good manners, rule of law

Empowerment of citizens to manage their own affairs

Clarity on personal security/responsibility

This requires widespread understanding of the technology and shared expectations about its use

There are *lots* of casual developers



Estimated counts in American workplace

Education

Self-taught	41.8%
BS in CS (or related)	37.7%
On-the-job training	36.7%
MS in CS(or related)	18.4%
Online class	17.8%
Some univ, no degree	16.7%
Industry certification	6.1%
Other	4.3%
Boot-camp	3.5%
PhD in CS(or related)	2.2%
Mentorship program	1.0%

“Professional and enthusiast programmers”
(international)

Demographics of US Internet users

Overall	Total adults	87%
	Women	87
	Men	86
Age	18-29	97%
	30-49	93
	50-64	88
	65+	57
Geography	urban	88%
	suburban	87
	rural	83
Education	<= high school	76%
	some college	91
	college +	97

Millennials Ages 18-33	Gen X Ages 34-45	Younger Boomers Ages 46-55	Older Boomers Ages 56-64	Silent Generation Ages 65-73	G.I. Generation Age 74+
Email	Email	Email	Email	Email	Email
Search	Search	Search	Search	Search	Search
Health info	Health info	Health info	Health info	Health info	Health info
Social network sites	Get news	Get news	Get news	Get news	Buy a product
Watch video	Govt website	Govt website	Govt website	Travel reservations	Get news
Get news	Travel reservations	Travel reservations	Buy a product	Buy a product	Travel reservations
Buy a product	Watch video	Buy a product	Travel reservations	Govt website	Govt website
IM	Buy a product	Watch video	Bank online	Watch video	Bank online
Listen to music	Social network sites	Bank online	Watch video	Financial info	Financial info
Travel reservations	Bank online	Social network sites	Social network sites	Bank online	Religious info
Online classifieds	Online classifieds	Online classifieds	Online classifieds	Rate things	Watch video
Bank online	Listen to music	Listen to music	Financial info	Social network sites	Play games
Govt website	IM	Financial info	Rate things	Online classifieds	Online classifieds
Play games	Play games	IM	Listen to music	IM	Social network sites
Read blogs	Financial info	Religious info	Religious info	Religious info	Rate things
Financial info	Religious info	Rate things	IM	Play games	Read blogs
Rate things	Read blogs	Read blogs	Play games	Listen to music	Donate to charity
Religious info	Rate things	Play games	Read blogs	Read blogs	Listen to music
Online auction	Online auction	Online auction	Online auction	Donate to charity	Podcasts
Podcasts	Donate to charity	Donate to charity	Donate to charity	Online auction	Online auction
Donate to charity	Podcasts	Podcasts	Podcasts	Podcasts	Blog
Blog	Blog	Blog	Blog	Blog	IM
Virtual worlds	Virtual worlds	Virtual worlds	Virtual worlds	Virtual worlds	Virtual worlds

Generations Online 2010

This chart shows the popularity of internet activities among internet users in each generation

90-100%	40-49%
80-89%	30-39%
70-79%	20-29%
60-69%	10-19%
50-59%	0-9%

Key: % of internet users in each generation who engage in this online activity

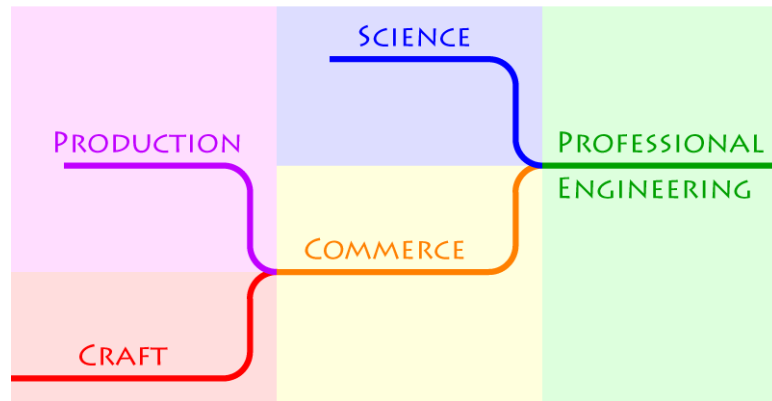
Email	Email	Email	Email	Email	Email
Search	Search	Search	Search	Search	Search
Health info	Health info	Health info	Health info	Health info	Health info
social network sites	Get news	Get news	Get news	Get news	Buy a product
Watch video	Govt website	Govt website	Govt website	Travel reservations	Get news
Get news	Travel reservations	Travel reservations	Buy a product	Buy a product	Travel reservations
Buy a product	Watch video	Buy a product	Travel reservations	Govt website	Govt website
IM	Buy a product	Watch video	Bank online	Watch video	Bank online
Listen to music	social network sites	Bank online	Watch video	Financial info	Financial info
Travel reservations	Bank online	Social network sites	Social network sites	Bank online	Religious info
Online classifieds	Online classifieds	Online classifieds	Online classifieds	Rate things	Watch video
Bank online	Listen to music	Listen to music	Financial info	Social network sites	Play games
Govt website	IM	Financial info	Rate things	Online classifieds	Online classifieds
Play games	Play games	IM	Listen to music	IM	social network sites
Read blogs	Financial info	Religious info	Religious info	Religious info	Rate things
Financial info	Religious info	Rate things	IM	Play games	Read blogs
					Donate to

Civilizing the electronic frontier

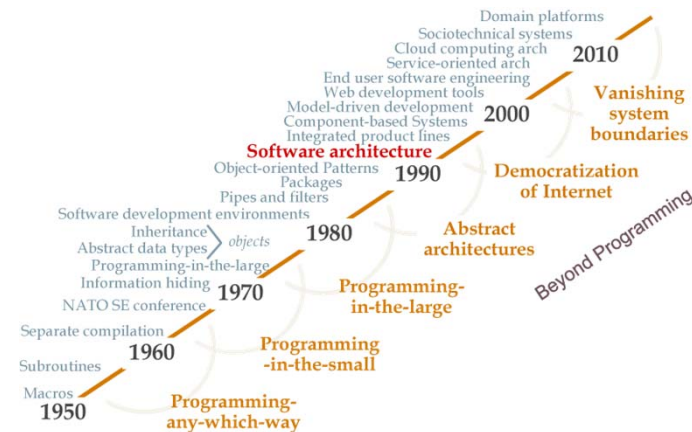
- **Policy, requiring technology**
 - Balance anonymity and accountability
 - Balance security and privacy
 - Balance individual and corporate objectives
 - Address product liability
- **Technology**
 - Apply known best practices and designs
 - Address new forms of information access (search) and software creation (independent parts)
- **User models**
 - Improve the explanations and intuitions we provide the public at large

Recapitulation

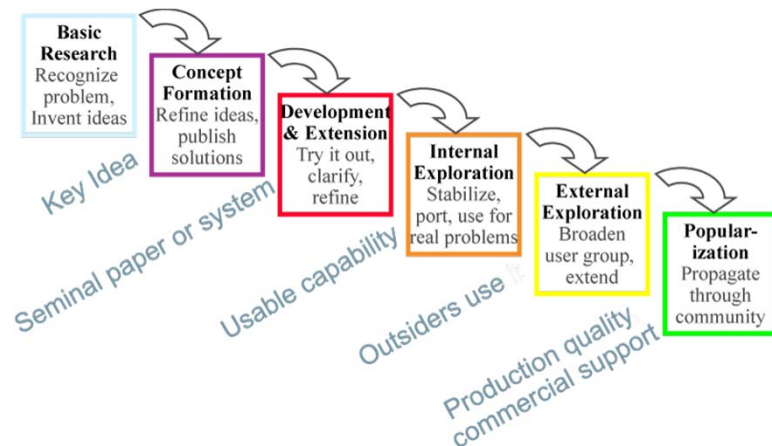
Engineering evolves from craft and commercial practice via science



Engineering basis evolves via increasingly powerful abstractions



Ideas evolve over time from pure research to practical production



The greatest need for engineering is in the most critical applications

