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
Civil Engineering

Example:

Bridges and Arches
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

Renaissance & Industrial Revolution

Scientific Engineering

Empirical progress via failure and repair

No deliberate application of mathematics to determine size or shape

Little theory, but construction methods lasted until 19th century

Vitruvius: De Architectura [about 25 BC]
The Evolution of the Stone-arch Bridge

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

*Ring and Western Civilization, James Kip Finch, Hill Book Company, Inc., New York, NY, 1951, p33*
Fig. 28. Arch bridge, according to Leon Battista Alberti.


15th century
Ironbridge at Coalbrookdale, 1779
Dee Bridge disaster, 1847
Business of bridges

Romans

Renaissance & Industrial Revolution

Scientific Engineering

Increasingly long spans, lighter structures

Rules of thumb about proportions

Explanation of structures:
  - Brunelleschi on arches and domes 15th century
  - Galileo on beams 17th century

Introduction of cast iron, wrought iron, steel, and reinforced concrete
Hardest problem was identifying the proper basic concepts, e.g. force.

New mathematics was needed (calculus).
### PROPERTIES OF VARIOUS SECTIONS

<table>
<thead>
<tr>
<th>Section</th>
<th>Area of Section $A$</th>
<th>Distance from Axis $y$ in Expression of Inertia $I_{yy}$</th>
<th>Moment of Inertia $I_{yy}$</th>
<th>Section Modulus $S = \frac{I_{yy}}{y}$</th>
<th>Radius of Gyration $r = \sqrt{\frac{I_{yy}}{A}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d + a$</td>
<td>$\frac{a}{2}$</td>
<td>$\frac{a}{2}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{2}$</td>
<td>$\frac{d}{\sqrt{2}}$</td>
</tr>
<tr>
<td>$d + 2a$</td>
<td>$\frac{3a}{2}$</td>
<td>$\frac{3a}{2}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{3}$</td>
<td>$\frac{d}{\sqrt{3}}$</td>
</tr>
<tr>
<td>$d + 3a$</td>
<td>$\frac{4a}{3}$</td>
<td>$\frac{4a}{3}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d}{\sqrt{4}}$</td>
</tr>
<tr>
<td>$d + 4a$</td>
<td>$\frac{5a}{4}$</td>
<td>$\frac{5a}{4}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{5}$</td>
<td>$\frac{d}{\sqrt{5}}$</td>
</tr>
<tr>
<td>$d + 5a$</td>
<td>$\frac{6a}{5}$</td>
<td>$\frac{6a}{5}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{6}$</td>
<td>$\frac{d}{\sqrt{6}}$</td>
</tr>
</tbody>
</table>

### PROPERTIES OF VARIOUS SECTIONS

<table>
<thead>
<tr>
<th>Section</th>
<th>Area of Section $A$</th>
<th>Distance from Axis $y$ in Expression of Inertia $I_{yy}$</th>
<th>Moment of Inertia $I_{yy}$</th>
<th>Section Modulus $S = \frac{I_{yy}}{y}$</th>
<th>Radius of Gyration $r = \sqrt{\frac{I_{yy}}{A}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d + c$</td>
<td>$\frac{c}{2}$</td>
<td>$\frac{c}{2}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{2}$</td>
<td>$\frac{d}{\sqrt{2}}$</td>
</tr>
<tr>
<td>$d + 2c$</td>
<td>$\frac{3c}{2}$</td>
<td>$\frac{3c}{2}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{3}$</td>
<td>$\frac{d}{\sqrt{3}}$</td>
</tr>
<tr>
<td>$d + 3c$</td>
<td>$\frac{4c}{3}$</td>
<td>$\frac{4c}{3}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d}{\sqrt{4}}$</td>
</tr>
<tr>
<td>$d + 4c$</td>
<td>$\frac{5c}{4}$</td>
<td>$\frac{5c}{4}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{5}$</td>
<td>$\frac{d}{\sqrt{5}}$</td>
</tr>
<tr>
<td>$d + 5c$</td>
<td>$\frac{6c}{5}$</td>
<td>$\frac{6c}{5}$</td>
<td>$\frac{d^2}{4}$</td>
<td>$\frac{d^2}{6}$</td>
<td>$\frac{d}{\sqrt{6}}$</td>
</tr>
<tr>
<td>Sections</td>
<td>Area of Section $A$</td>
<td>Distance from Axis to Extremities of Section $y$ and $y_1$</td>
<td>Moment of Inertia $I$</td>
<td>Section Modulus $S = \frac{I}{y}$</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>----------------------------------------------------------</td>
<td>----------------------</td>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$bd - ah$</td>
<td>$y = \frac{d}{2}$</td>
<td>$\frac{1}{12} (bd^3 - ah^3)$</td>
<td>$\frac{bd^3 - ah^3}{6d}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$bd - ah$</td>
<td>$y = b - y_1$</td>
<td>$\frac{1}{3} (2mb^3 + ha^3) - Ay_1^2$</td>
<td>$\frac{I}{y}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$bd - 2ah$</td>
<td>$y = \frac{d}{2}$</td>
<td>$\frac{1}{12} (bd^3 - 2ah^3)$</td>
<td>$\frac{bd^3 - 2ah^3}{6d}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$bd - 2ah$</td>
<td>$y = \frac{b}{2}$</td>
<td>$\frac{1}{12} (2mb^3 + ha^3)$</td>
<td>$\frac{2mb^3 + ha^3}{6b}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$bm + At$</td>
<td>$y = d - y_1$</td>
<td>$\frac{1}{3} (ry^2 + by_1^3 - 2a(y_1 - m)^3)$</td>
<td>$\frac{I}{y}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$bm + At$</td>
<td>$y = \frac{b}{2}$</td>
<td>$\frac{1}{12} (mb^3 + Ah^3)$</td>
<td>$\frac{mb^3 + Ah^3}{6b}$</td>
<td></td>
</tr>
</tbody>
</table>
Figure 10.2 Types of arch bridge.

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

Engineering of bridges

Romans

Renaissance & Industrial Revolution

Scientific Engineering

1700: good theories (statics, strength of materials)

1750: tabulations of properties of materials

1850: formal analysis of a bridge structure

1900: structural analysis worked out

1950: systematic theory

2000: design automation
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.
Table 2.3-2 Matrix of Abutment Types versus Superstructure Types

<table>
<thead>
<tr>
<th>Superstructure Type</th>
<th>Abutment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Traditional High</td>
</tr>
<tr>
<td>Prestressed Concrete Adjacent Box Beam</td>
<td>❌</td>
</tr>
<tr>
<td>Prestressed Concrete Spread Box Beam</td>
<td>❌</td>
</tr>
<tr>
<td>Prestressed Concrete I-Beam</td>
<td>❌</td>
</tr>
<tr>
<td>Steel Rolled Beam</td>
<td>❌</td>
</tr>
<tr>
<td>Steel Plate Girder</td>
<td>❌</td>
</tr>
</tbody>
</table>

Automates production of scaled bridge contract drawings

BRADD

Engineering Input

Design Drawings

www.dot.state.pa.us
Evolution of civil engineering

1700: statics
1700: strength of materials
1775: hydraulics

1750: properties of materials
1850: full analysis of a bridge
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 introduced 1968 to draw attention to “the software crisis”

Aspiration, not description

By some reports, “software engineering” was coined by Margaret Hamilton a few years earlier; the 1968 and 1969 NATO conferences brought the phrase into widespread use.
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
Commerce drives science

Science is often stimulated by problems in commercial practice

- safety-critical tasks ➞ safety analysis
- large systems ➞ architectural patterns
- concurrency ➞ parallel logics & languages
- large state spaces ➞ model checking
- many versions ➞ program families, inheritance
- huge data sets ➞ MapReduce scalability
- adaptive systems ➞ MAPE model
Codified knowledge

Data structures, algorithms
Programming languages and semantics
Verification and model checking
Objects and abstract data types
Static and dynamic analysis
Software architectures
Model-based engineering
Pattern languages
Computability

...
Research feeds practice

Research and development stimulates creation of innovative ideas and industries.
Research and development stimulates creation of innovative ideas and industries.
Increasing Abstraction Scale

Software architecture

1990

1980

Abstract architectures

1970

Programming-in-the-large

1960

Programming-in-the-small

1950

Programming-any-which-way

1940

Software development environments

1930

Inheritance

1920

Abstract data types

1910

Programming-in-the-large

1900

Information hiding

NATO SE conference

1890

Separate compilation

Subroutines

1880

Macros

1870

Beyond Programming

2010

Domain platforms

Sociotechnical systems

Cloud computing arch

Service-oriented arch

End user software engineering

Web development tools

Model-driven development

Component-based Systems

Integrated product lines

Vanishing system boundaries

Democratization of Internet
Design guidance

Choosing among algorithms based on the problem setting

Fig. 21. Suggestions on the use of the various sorting techniques.
Design guidance

Choose among algorithms based on the problem setting.

- **Are there 11 items or are you unwilling to write 20 lines instead of 10 lines of program to get an n log n sort time instead of n^2?**
  - **Yes:** Use a bubble sort.
  - **No:**
    - **Are items to be added or deleted from the sorted order?**
      - **Yes:** Use a distributive sort.
      - **No:**
        - **Is the CPU space-time product more important than I/O time?**
          - **Yes:** Use a tournament replacement sort.
          - **No:**
            - **Is the sort to be done on two disks?**
              - **Yes:** Interleave the strings to be merged.
              - **No:**
                - **Are six or fewer tape drives and disks available?**
                  - **Yes:** Use a polyphase merge.
                  - **No:**
                    - **Can 7 or more drives be read in both directions and quickly replaced?**
                      - **Yes:** Use an oscillating merge.
                      - **No:**

- **No:**
  - **Are the items less than an average of log_2 n positions out of place?**
    - **Yes:** Use Treesort 3 or a Shell sort.
    - **No:**
      - **Would you make the sort dependent on the distribution of key values to get a sort linear with n in time and storage?**
        - **Yes:** Try to use an address calculation sort.
        - **No:** Are items to be added or removed from the sorted order?
          - **Yes:** Use an item at a time.
          - **No:**
            - **Can 7 or more drives be read in both directions and quickly replaced?**
              - **Yes:** Use an oscillating merge.
              - **No:** Use a polyphase merge.
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
with a program transformation
OPERATOR
USER PROGRAM
I/O MANAGEMENT
OPERATOR/PROCESS COMMUNICATION
MEMORY (MAIN/SECONDARY) MANAGEMENT
PROCESSOR ALLOCATION
MULTIPROGRAMMING
HARDWARE

LAYERED SYSTEM (THE System, Dijkstra)
A layered system!!

http://www.multicians.org/architecture.html
Craft practice

Software has always had structure
  o Informal vocabulary
    – Objects, pipes/filters, interpreters, repositories …
  o Intuitions and folklore about fitness to task

Ancient examples (since NATO69):
  o Software bundled with hardware
  o Compilers, layered operating systems
  o Databases for accounting
FIGURE 7. Flight Computer Operating System (The FCOS dispatcher coordinates and controls all work performed by the on-board computers.)

<table>
<thead>
<tr>
<th>Client Layer*</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Access domain management</td>
<td></td>
</tr>
<tr>
<td>Buffering and record-level I/O</td>
<td></td>
</tr>
<tr>
<td>Transaction coordination</td>
<td></td>
</tr>
<tr>
<td>Agent Layer</td>
<td></td>
</tr>
<tr>
<td>Implementation of standard server interface</td>
<td></td>
</tr>
<tr>
<td>Logger, agent, and instance tasks</td>
<td></td>
</tr>
<tr>
<td>Helix Directories</td>
<td></td>
</tr>
<tr>
<td>Path name to FID mapping</td>
<td></td>
</tr>
<tr>
<td>Single-file (database) update by one task</td>
<td></td>
</tr>
<tr>
<td>Procedural interface for queries</td>
<td></td>
</tr>
<tr>
<td>Object (FID directory)</td>
<td></td>
</tr>
<tr>
<td>Identification and capability access (via FIDs)</td>
<td></td>
</tr>
<tr>
<td>FID to tree-root mapping; table of (FID, root, ref_count)</td>
<td></td>
</tr>
<tr>
<td>Existence and deletion (reference counts)</td>
<td></td>
</tr>
<tr>
<td>Concurrency control (file interlocking)</td>
<td></td>
</tr>
<tr>
<td>Secure Tree</td>
<td></td>
</tr>
<tr>
<td>Basic crash-resistant file structure</td>
<td></td>
</tr>
<tr>
<td>Conditional commit</td>
<td></td>
</tr>
<tr>
<td>Provision of secure array of blocks</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
</tr>
<tr>
<td>Commit and restart authority</td>
<td></td>
</tr>
<tr>
<td>Disk space allocation</td>
<td></td>
</tr>
<tr>
<td>Commit domains</td>
<td></td>
</tr>
<tr>
<td>Cache</td>
<td></td>
</tr>
<tr>
<td>Caching and performance optimization</td>
<td></td>
</tr>
<tr>
<td>Commit support (flush)</td>
<td></td>
</tr>
<tr>
<td>Frame allocation (to domains)</td>
<td></td>
</tr>
<tr>
<td>Optional disk shadowing</td>
<td></td>
</tr>
<tr>
<td>Canonical Disk</td>
<td></td>
</tr>
<tr>
<td>Physical disk access</td>
<td></td>
</tr>
</tbody>
</table>

*Also called client Helix.

**Figure 2. Abstraction layering.**

Figure 2. Display PostScript interpreter components.

A7E avionics architecture, as shown in Bachman et al. Software Documentation in Practice, SEI 2000.
Commerce stimulates science

uncertainty about quality, performance, changeability, etc

models to support software metrics

ad hoc structure, multiple versions, interoperability issues, design drift

styles/patterns for software architecture
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

Virtual machine

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

http://www.pcmag.com/encyclopedia/term/53927/virtual-machine
Commercial practice

1970s: batch processing
  o modules and procedure calls, Cobol

1980s: informal “architecture” in papers
  o colloquial use of architectural terms

1990s: early structure
  o software product lines

2000s: architecture research enters practice
  o company-specific overall architectures
  o frameworks, UML
  o objects everywhere
Maturation of scientific ideas

Key Idea

Basic Research
Recognize problem, Invent ideas

Concept Formation
Refine ideas, publish solutions

Development & Extension
Try it out, clarify, refine

Internal Exploration
Stabilize, port, use for real problems

External Exploration
Broaden user group, extend

Popularization
Propagate through community

Seminal paper or system
Usable capability
Outsiders use it
Production quality, commercial support

15-20 years

Sam Redwine, Jr. and William Riddle: Software Technology Maturation, Proc ICSE-8, May 1985
Maturation of software architecture

Foundations

Basic Research

Concepts

Development

Internal Exp/Ext

External Exp/Ext

Popularization

Garlan and Shaw. Software architecture: reflections on an evolving discipline. ESEC/FSE keynote 2011
Supporting concepts from software engineering evolved on their own 15- to 20-year cycles, related concepts continue to evolve.
Basic research, 1985-1993

Basic descriptive models:
Product lines for specific domains
Catalogs of common idioms
Connectors as well as components

Concept formation 1992-1996

Elaboration of basic models
Languages and formalizations
Taxonomies of architectural patterns
Workshops and books

Foundations

Basic Research

Concepts

Unification and refinement

Second generation concepts

Institutions, conferences

Development

Internal Exp/Ext

External Exp/Ext

Popularization
Internal exploration: 1996-2003

Explicit attention to architecture in design
Architecture’s role in quality attributes
Analysis and evaluation techniques
Books on practice
External exploration: 1998-present

Technologies useful beyond development group

Tools and frameworks

Company-specific architectures

External Exp/Ext

Popularization
Popularization: 2000-present

Production-quality, supported, commercialized technology, standards
Education, professional organizations
Architect as senior technical leader
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 THE FIREFOX AND ITS GEOcko 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.

I AM A GOD

http://xkcd.com/676/
Systematically Organized Knowledge

SEI Series organizes knowledge about architecture and its analysis
## Architectural styles and reasoning

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

Rules of thumb on \textit{data flow}

If your problem is decomposed into sequential stages, consider \textit{batch sequential} or \textit{pipeline} architectures.

If each stage is incremental, so that later stages can begin before earlier stages finish, consider a \textit{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 \textit{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 \textit{closed loop} architectures.

Generality-power trades
Styles, Platforms, and Product Lines

- Bosch Engine Control
- Siemens Healthcare for 3D
- AUTOSAR
- HLA
- IOS
- Domain-Specific Component Integration Platforms

- Generic Component Integration Platforms
- Generic Style Specializations
- Pipes & Filters
- Process Control
- Data Flow
- Call-Return
- Events

But is it “Engineering” yet?
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
- Many data breaches (retail, government, ...)
- Samsung Galaxy S5, S6 keyboard exploit
- 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 $\rightarrow$ 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

# Identity Theft Resource Center

## 2014 Data Breach Category Summary

<table>
<thead>
<tr>
<th>Totals for Category</th>
<th># of Breaches</th>
<th># of Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banking/Credit/Financial</td>
<td>43</td>
<td>1,198,492</td>
</tr>
<tr>
<td>Business</td>
<td>258</td>
<td>68,237,914</td>
</tr>
<tr>
<td>Educational</td>
<td>57</td>
<td>1,247,812</td>
</tr>
<tr>
<td>Government/Military</td>
<td>92</td>
<td>6,649,319</td>
</tr>
<tr>
<td>Medical/Healthcare</td>
<td>333</td>
<td>8,277,991</td>
</tr>
</tbody>
</table>

**Totals for All Categories:**

<table>
<thead>
<tr>
<th># of Breaches</th>
<th># of Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>783</td>
<td>85,611,528</td>
</tr>
</tbody>
</table>

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

**Total Breaches:** 783

**Records Exposed:** 85,611,528
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
software development methods

production

commerce

science

professional engineering

~ 1990, adoption of development methods

emerging, but spotty
Want to be part of this?
http://isri.cmu.edu/education/

Making Progress
Catastrophe

Consequences of Failure

- Therac-25
- Drug interactions
- Ambulance scheduling
- Patient monitoring
- Missile guidance
- Medical implants
- Nuclear safety devices

Greatest Need for Engineering Discipline

- Car cruise control
- IRS 1040 on your own
- Web search health info
- Finding restaurant
- Appointment scheduling

Inconvenience

Degree of Oversight

- Full oversight, manual operation
- None: full automation, unattended operation

- Advance cruise control
- IRS 1040 TurboTax
- Stock market alerts
- Stocks program trading
- Near real time weather
- Automatic sports stats
- Self driving car
Adapting to evolving technology

- Technology outruns traditional manuals
  - Understand how search supplants indexing
  - Analog of MapReduce for documentation?

- Agility, “perpetual beta”, and evolution
  - Exploit power end of generality tradeoff, embedding knowledge in task-specific tools

- Scaling cost to consequence, predictably
  - 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
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

### 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%

Estimated counts in American workplace

"Professional and enthusiast programmers" (international)


Demographics of US Internet users

<table>
<thead>
<tr>
<th>Overall</th>
<th>Total adults</th>
<th>87%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Women</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>Men</td>
<td>86</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Age</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>18-29</td>
<td>97%</td>
</tr>
<tr>
<td>30-49</td>
<td>93</td>
</tr>
<tr>
<td>50-64</td>
<td>88</td>
</tr>
<tr>
<td>65+</td>
<td>57</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Geography</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>urban</td>
<td>88%</td>
</tr>
<tr>
<td>suburban</td>
<td>87</td>
</tr>
<tr>
<td>rural</td>
<td>83</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Education</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= high school</td>
<td>76%</td>
</tr>
<tr>
<td>some college</td>
<td>91</td>
</tr>
<tr>
<td>college +</td>
<td>97</td>
</tr>
</tbody>
</table>

### Generations Online 2010

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

#### Key:
- **% of internet users in each generation who engage in this online activity**
- **90-100%**: Most users
- **80-89%**: Users in the minority
- **70-79%**: Users in the majority
- **60-69%**: Users frequenting the activity
- **50-59%**: Users occasionally using the activity
- **0-9%**: Users of this activity

#### Source:
Pew Internet surveys.

---

Here is the table with the activities and their popularity across different generations:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Millennials</th>
<th>Gen X</th>
<th>Younger Boomers</th>
<th>Older Boomers</th>
<th>Silent Generation</th>
<th>G.I. Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>Email</td>
<td>Email</td>
<td>Email</td>
<td>Email</td>
<td>Email</td>
<td>Email</td>
</tr>
<tr>
<td>Search</td>
<td>Search</td>
<td>Search</td>
<td>Search</td>
<td>Search</td>
<td>Search</td>
<td>Search</td>
</tr>
<tr>
<td>Health info</td>
<td>Health info</td>
<td>Health info</td>
<td>Health info</td>
<td>Health info</td>
<td>Health info</td>
<td>Health info</td>
</tr>
<tr>
<td>Social network sites</td>
<td>Get news</td>
<td>Govt website</td>
<td>Travel reservations</td>
<td>Buy a product</td>
<td>Govt website</td>
<td>Govt website</td>
</tr>
<tr>
<td>Watch video</td>
<td>Watch video</td>
<td>Buy a product</td>
<td>Travel reservations</td>
<td>Travel reservations</td>
<td>Govt website</td>
<td></td>
</tr>
<tr>
<td>IM</td>
<td>IM</td>
<td>Bank online</td>
<td>Social network sites</td>
<td>Social network sites</td>
<td>Social network sites</td>
<td></td>
</tr>
<tr>
<td>Listen to music</td>
<td>Listen to music</td>
<td>Financial info</td>
<td>Financial info</td>
<td>Financial info</td>
<td>Financial info</td>
<td>Religious info</td>
</tr>
<tr>
<td>Travel reservations</td>
<td>Travel reservations</td>
<td>Bank online</td>
<td>Bank online</td>
<td>Bank online</td>
<td>Bank online</td>
<td>Religious info</td>
</tr>
<tr>
<td>Online classifieds</td>
<td>Online classifieds</td>
<td>Rate things</td>
<td>Social network sites</td>
<td>Social network sites</td>
<td>Social network sites</td>
<td></td>
</tr>
<tr>
<td>Bank online</td>
<td>Bank online</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
</tr>
<tr>
<td>Govt website</td>
<td>Govt website</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
</tr>
<tr>
<td>Play games</td>
<td>Play games</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
</tr>
<tr>
<td>Read blogs</td>
<td>Read blogs</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
<td>IM</td>
</tr>
<tr>
<td>Financial info</td>
<td>Financial info</td>
<td>Religious info</td>
<td>Religious info</td>
<td>Religious info</td>
<td>Religious info</td>
<td>Religious info</td>
</tr>
<tr>
<td>Rate things</td>
<td>Rate things</td>
<td>Play games</td>
<td>Play games</td>
<td>Play games</td>
<td>Play games</td>
<td>Play games</td>
</tr>
<tr>
<td>Religious info</td>
<td>Religious info</td>
<td>Read blogs</td>
<td>Read blogs</td>
<td>Read blogs</td>
<td>Read blogs</td>
<td>Read blogs</td>
</tr>
<tr>
<td>Online auction</td>
<td>Online auction</td>
<td>Online auction</td>
<td>Online auction</td>
<td>Online auction</td>
<td>Online auction</td>
<td>Online auction</td>
</tr>
<tr>
<td>Podcasts</td>
<td>Podcasts</td>
<td>Podcasts</td>
<td>Podcasts</td>
<td>Podcasts</td>
<td>Podcasts</td>
<td>Podcasts</td>
</tr>
<tr>
<td>Donate to charity</td>
<td>Donate to charity</td>
<td>Donate to charity</td>
<td>Donate to charity</td>
<td>Donate to charity</td>
<td>Donate to charity</td>
<td>Donate to charity</td>
</tr>
<tr>
<td>Virtual worlds</td>
<td>Virtual worlds</td>
<td>Virtual worlds</td>
<td>Virtual worlds</td>
<td>Virtual worlds</td>
<td>Virtual worlds</td>
<td>Virtual worlds</td>
</tr>
</tbody>
</table>

---

http://www.pewinternet.org/2010/12/16/generations-2010/
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

Ideas evolve over time from pure research to practical production

The greatest need for engineering is in the most critical applications