Software Engineering: Building on Past Successes

Leon J. Osterweil, University of Massachusetts, Amherst
Lori A. Clarke, University of Massachusetts, Amherst
Richard N. Taylor, University of California, Irvine

Software Engineering Is Important and Challenging

- **Software is the Grand Enabler**
  - The principal "building material" for information systems
  - Most flexible and least constrained of all materials
- **The most complex engineered systems are software systems**
- **Software Engineering has a broad scope**
  - Effective use of this building material to create and evolve systems and embedded components spans:
    - Design, Analysis, Configuration Management, High-assurance, Team coordination, Software architecture and components, Software process, Metrics and empirical studies
  - Intellectual depth: deep scientific and engineering issues, interdisciplinary
  - Thrives best as a research area with practical application and evaluation

Software Engineering Has Been Successful

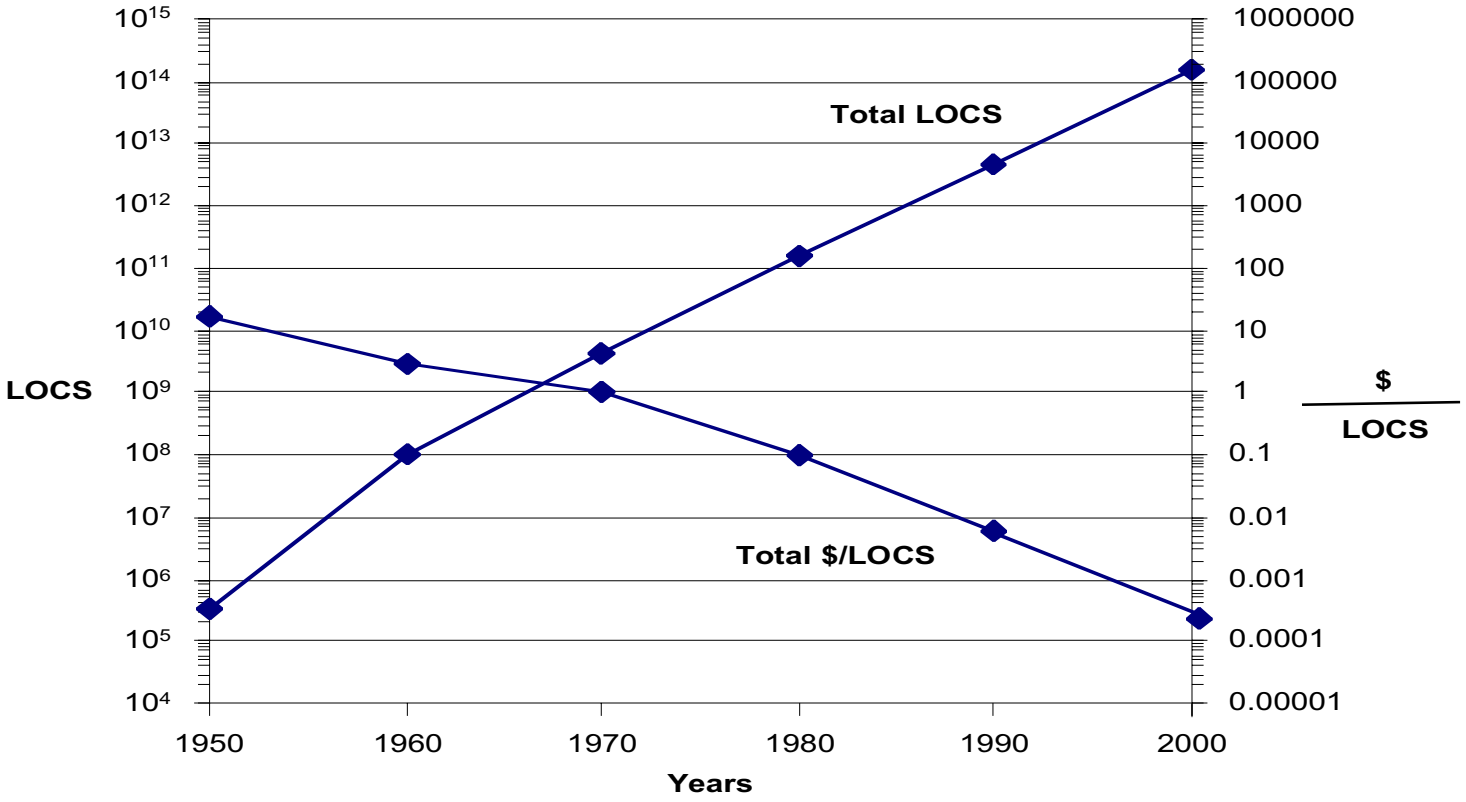
Capability has grown

- In total lines of code produced
- In total lines of code deployed
- In trustworthiness of systems
- In interconnectivity
- In magnitude of capability
- In predictability of processes
- In applications of increasing
  - Size
  - Complexity
  - Application domain diversity
- In meeting key societal and military challenges

Demands/Expectations have grown

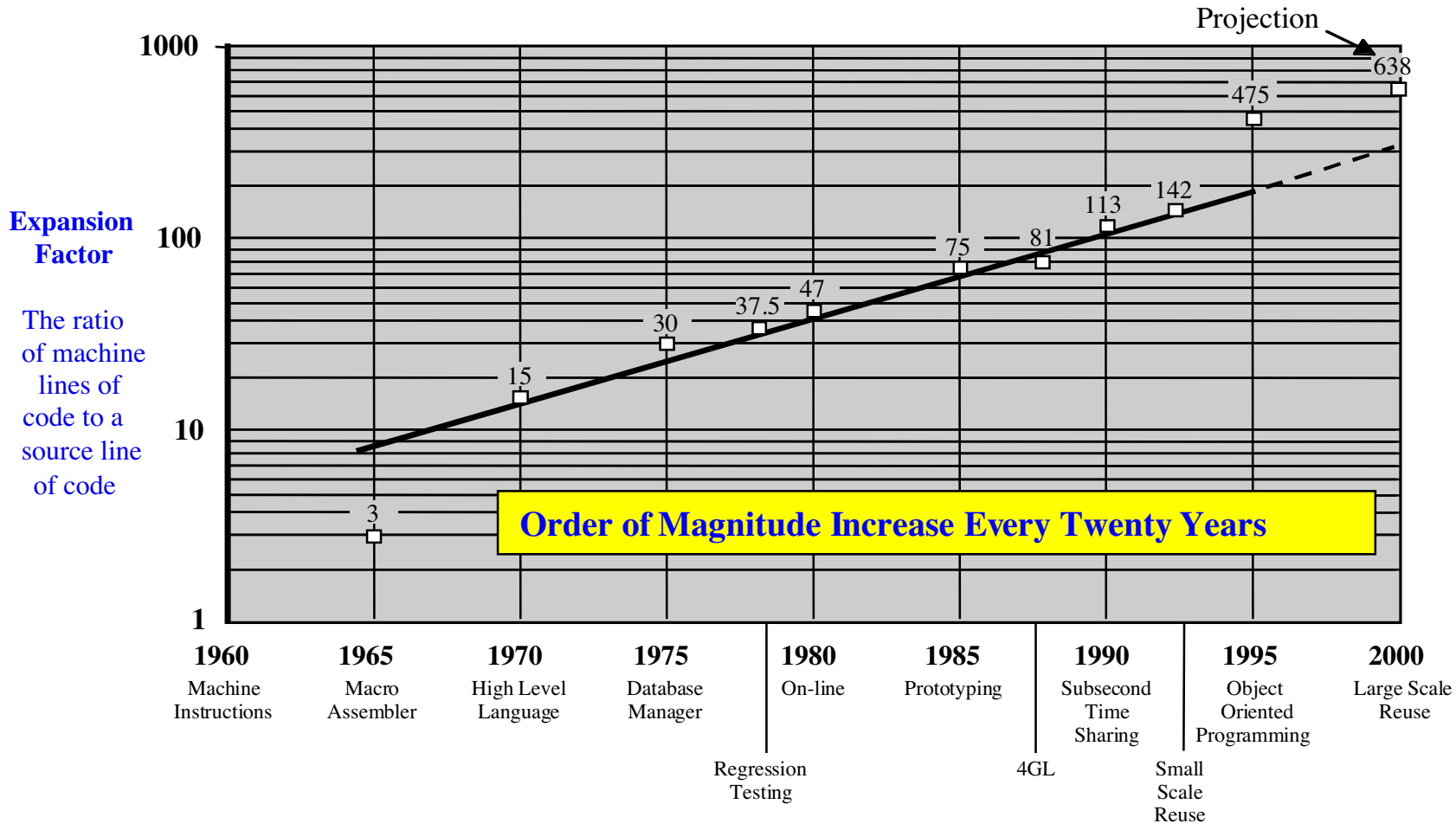
- Broader deployment
- Need for greater reliability
- More diversity of application
- Greater interconnectedness
- Faster development/cycles
- Greater adaptability/evolvability
- More robustness
- More protection of privacy
- Greater security

Lines of Code in Service: U.S. DoD



Growth in Impact of Lines of Code

(Bernstein, 1997)



Research Contributions to this Success

- **Requirements Engineering:**
  - Early research: Teichroew's PSL/PSA ('72), Alford's SREM ('77), Estrin's SARA('86), Parnas's tables ('88), Heitmeyer('96)
  - Tech Transfer: Numerous requirements specification tools
  - Industrial impact: Nearly universal use of requirements in DoD and industry
- **Process: Systematizing how software is built and evolved**
  - Early research: Royce's waterfall model ('62), Boehm's spiral model ('87), Process programming (Osterweil '87)
  - Industry impact: Nearly universal use of lifecycle development
- **Configuration Management/Version Control:**
  - Early research: Rochkind RMS ('76), Tichy's RCS ('79), Feldman's Make('79)
  - Tech transfer: Variety of CASE products (e.g., Atria/Rational's Clear Guide)
  - Industry impact: Use of CM nearly universal (it is also easier to do it than to not do it)
- **Integrated Development Environments:**
  - Early research: Interlisp('74), MESA('79), Gandalf ('79), Arcadia ('88)
  - Tech Transfer: Products from Rational, Symantec, etc.
  - Industry Impact: Java development environments, the "middleware" industry

More Contributions

- **Testing Technology:**
  - Early Research: Miller's coverage metrics ('74), Howden ('76), Stucki's assertions ('75)
  - Tech Transfer: Software Research Assoc., Poston &Co., Sun's SoftTest
  - Industrial Impact: Widespread graph based coverage metrics and tools, assertion-checking compilers/debuggers
- **Inspections:**
  - Early Research: Fagan inspections ('77), Mills Cleanroom ('79)
  - Tech Transfer: Several inspection technology companies
  - Industrial Impact: Now an accepted practice in many companies; proven effectiveness
- **Metrics:**
  - Early Research: Boehm's COCOMO model('81), McCabe's metrics ('75)
  - Tech Transfer: Numerous companies suggest, define, take metrics, sell metric data
  - Industrial Impact: The only thing non-technical managers understand, influenced CMM

More Contributions

- **Software Design Methodology:**
  - Early research: Parnas's modules (1972), Data Abstractions (Guttag, Wulf, Liskov, Goguen--mid 1970's), Objects and OO Design (mid-1980's)
  - Tech Transfer: Myriad software design companies (eg. IDE, Rational)
  - Industrial impact: Widespread use of design notations and methods; OO Design
- **Internet Protocols:**
  - Research: HTTP/1.1, URI spec, WebDAV protocol (author lists confirm their roots in software engineering)
  - Tech Transfer: Numerous startups, standards, MS Office 2000, etc.
  - Industry Impact: Web shatters old preconceptions of everything, revolutionizes the world
- **Component technology**
  - Early research: Reiss' Field(92), Purtilo's software bus('85), Heimbigner's Q('86)
  - Tech Transfer: Numerous s/w products (CORBA, DCE)
  - Industrial impact ; "Middleware", Application Generators

Still More Contributions

- **High Assurance:**
  - Early Research: Formal verification (Hoare-1972, Djikstra-1970), finite state verification (Dave('76), Cesar('86), Model checking('86), SPIN ('82), FLAVERS('94))
  - Tech transfer: Numerous European examples, gaining acceptance for hardware
  - Industrial Impact: Concepts incorporated into strongly typed programming languages (Ada, Java), supported in tools (e.g., Lint); increasing adoption in life critical projects
- **Persistence and OO DB**
  - Early Research: PS AL□GOL('82), Pleiades ('89), Napier ('89)
  - Tech transfer: workshops and "shootouts" between the DBMS and SW Engineering communities
  - Industrial Impact: persistence is now an integral part of Java, commercial OO data base systems

A Detailed Recent Success Story: Current WWW Protocols and Software

- **What:** the HTTP/1.1 protocol, the URI (URL) standard, the Apache Web server, and the WebDAV protocol (1994-1999)
- **Impact:** HTTP/1.1 is the dominant protocol on the WWW today. Apache has over 60% market share. URI is the basis for naming on the WWW. WebDAV is being supported by the major vendors. These protocols and the underlying software architecture brought the original (Berners-Lee) web to its modern form.
- **Who:** Roy Fielding & Jim Whitehead at UCI led these efforts and coordinated contributions from people around the world. (Evidence: author lists on the standards and articles; COB of The Apache Foundation)
- **Funding:** DARPA Software Environments program then DARPA EDCS program

Challenges and Opportunities Remain

- **Quantity and diversity of world needs**
- **Software problems continue to grow**
  - Titan IVB, USS Hue City, cost, schedule, suitability problems
  - Ariane launch disaster, telephone, e-commerce outages
- **SE community is continuing to rise to these challenges**
  - Often in response to challenges
  - Often from seeing beyond immediate challenges
- **Each community grows best in reaction to the other; a complementary relationship**
- **Suggests the importance of a diversified portfolio**
  - Basic research
  - Synergistic, multi-discipline investigations
  - Industrial scale demonstrations

Pragmatic Concerns

- **Technology Transition**
  - Marketplace often makes “fixing it later” more attractive than “doing it right the first time”
  - Strong coupling among technologies, processes, acquisition practices, cultures
  - Rapidly evolving commercial technology
  - Slowly evolving Government acquisition practices
  - Risk-averse program managers
  - Leaky intellectual property
  - Payoffs take a long time to demonstrate
  - And are hard to trace back to particular technology insertions
- **Empirical evaluations**
  - Many dimensions , most of which are hard to measure
    - (e.g., ease of use)
  - Realism versus repeatability
  - Limited resources to develop prototypes and

Community Visions/ Potential Program Areas

- **Anytime, anywhere knowledge acquisition & interaction**
- **Highly adaptive software**
- **Bullet-proof software**
- **Domain Specific Languages**
- **Component approaches to embedded systems**
- **Teamwork support for software engineering**
- **Safe end-user programming of applications**
- **Software understanding**

Example 1: Anytime, Anywhere Knowledge Acquisition & Interaction

- **Vision:** Anytime, anywhere knowledge of what you need to know to accomplish your purposes, and ability to interact in that context. The current WWW hints at what can be accomplished with convenient access to globally distributed information. The vision requires application and information integration involving 2 orders of magnitude more information agents.
- **Technologies:**
  - Event-based notification
  - Software architectures
  - Protocols
  - Information meta-models
  - Task models
  - Wireless technologies
  - Names and namespaces
  - Security
- **Impact and Applications:**
  - Battlefield information systems
  - Just-in-time everything
  - Process control
  - Logistics
- **Rationale:** DoD needs are more demanding technically and are needed in a shorter timeframe. Current commercial inertia will restrain innovation.

Example 2: Self Improving Systems

- Vision: Systems that sense faults/shortcomings, diagnose problems, synthesize solution approaches, suggest and evaluate modifications. Human involvement dramatically reduced. Systems stabilize, settle in, with time
- Technologies:
  - Process
  - Automated testing/analysis
  - Architecture
  - Distributed Systems
  - Component composition
  - Mixed initiative systems
- Impact and Applications:
  - Faster deployment
  - Faster evolution
  - Increased quality
  - More stable applications
  - Increased robustness
  - Greater reuse/lower cost
- Rationale: Making software "perfect" is an unrealizable goal. It is more realistic to assure rapid, steady, measurable improvement. Aspects of this approach are appearing in some programs, but it needs a central focus.

Example 3: Bullet-Proof Software

- **Vision:** Light-weight verification techniques that are accessible by well-trained software developers and provide *guarantees* that systems will adhere to important safety properties. Requires hiding the "formal" in formal specifications and improving finite state verification techniques by an order of magnitude. Integrate testing and analysis approaches and incorporate incremental validation processes throughout lifecycle.
- **Technologies:**
  - Finite state verification
  - Integrated testing/analysis
  - S/W architecture
  - Specification patterns
  - Distributed systems
  - Process
- **Impact and Applications:**
  - Early fault detection
  - Reduced development/maintenance costs
  - High assurance systems
  - Faster, more reliable evolution
- **Rationale:** Industry is more focussed on time to market then software quality and, thus, not addressing society's high assurance needs. National infrastructure is fragile, as evidenced by the Y2K problem and numerous failed projects (e.g., IRS). Emerging approaches show incredible promise but need to be nurtured.

Why Now: Technical Opportunities

- **Component-oriented software engineering**
  - *Wide adoption*: Beans, COM
  - Reality vs. theory (EJB and COM studies)
  - Consensus on technical agenda
- **Scaling challenges**
  - Assurance, mobility, reliability, self-adaptation
  - Embedded software
  - Very large distributed systems
  - Interoperation, heterogeneity
  - Event-based integration
- **Assurance**
  - Model checking
  - Hybrid and model-based
- **Human computer interaction**
  - Engineering the human interface
  - Collaboration
  - Design and evaluation
- **Domain-specific systems**
  - Analysis and architecture
  - Generation and specialization
- **Legacy**
  - Understanding and analysis
  - Adaptation
  - Informing future design decisions
- **Team-based engineering**
  - Work coordination
  - Organizational memory (cf. DoD reverse engineering costs)
  - Coordination and performance

Conclusions

- **Software engineering is a critical area for the nation**
- **Software engineering is a difficult area to engage**
  - A quickly changing field
  - Suffers from light-under-the-lamppost measures
  - Embraces a very broad range of issues
- **Now is a critical time**