Software Engineering Research: Challenges, Successes, and Opportunities

Lori A. Clarke
Leon J. Osterweil
Dewayne Perry
Richard N. Taylor

26 July 1999
What is at Stake Here?

• Risks in doing software engineering badly:
  – Economic loss:
    • Loss of value, productivity. Demise of companies
    • Loss of dominance to European Comm, India, Singapore, Israel
      – This is not just alarmist; these countries are doing impressive things
  – Societal vulnerability:
    • Software is the infrastructure in all other infrastructure
    • Potential for death, destruction, calamity
      – There have been plenty of incidents, there will be more (more serious?)

• Rewards in doing it well:
  – Economic: greater efficiencies, profits, balance of trade
  – Societal: Unprecedented services, convenience
  – Scientific: Other grand challenges hinge on software
What Does “Do Software Well” Mean?

- Keep costs down
  - Currently hundreds or thousands of dollars/line
- Know what the system really does
  - Current testing is a very unreliable sampling technique
- Be sure it can be trusted with lives, property
- Effectively harness teams of thousands around the world to build millions of lines of code?
- Deal with legacy software (tens of billions of lines!)
  - Clean it up (Y2K is just the tip of the iceberg)
  - Evolve it
  - Rehost it
  - Keep from continuing to write more of it!

Typical Engineering Challenges
Our Aspirations Go Beyond Simply Meeting These Challenges

• Software is the “Grand Enabler” for all the Grand Challenges

• But there are also Grand Challenges for Software Engineering
  – Software that evaluates, changes, improves itself
  – Keeping one billion lines of code under control
  – Making N people N times as productive as 1
A Notable Case in Point: Web Protocol Research

- Major goal: Global Software Engineering
  – Using the internet and web protocols as environment
- Engineering Artifacts Developed
  – HTTP/1.1 and Apache
- Enormous Impact
  – Huge community gains efficiencies and new power
- Basis in Science
  – Programming languages, software system structure
- Contributions to Science
  – Interplay among structure, security, efficiency
Meeting Challenge and Engineering Problems Requires A Scientific Basis

- Much current practice is ad hoc, poorly integrated
- Scientific structure and principles are needed to provide order, coherence
  - Lead to systematic practice, orderly improvement
- Current situation has many historical antecedents
  - Ad hoc practice precedes science
  - Disciplined engineering springs from basis in science
- Need to accelerate the establishment of scientific underpinnings to hasten the maturity of software engineering practice
Some Key Scientific Issues

• What IS software?
  – What are its components, constraints, structure?
  – How to engineer artifacts whose nature is unclear?

• How to measure and evaluate something that is
  – Intangible, insensible, yet very REAL
  – Would lead to useful and effective measurement

• What do we mean by “retain control over software”
  – We need an analog of heat sinks, grounds, containment

• How to extract and integrate applicable aspects of other scientific disciplines?
  – Mathematics, Psychology, Sociology, … ?
SW Engineering Research Aims To

- Establish scientific/mathematical principles to explain software properties and behaviors
  - Develop new Science
  - Grapple with impediments to traditional science
    - Measurement obstacles, difficulty in repeating experiments
- Establish empirical methods and approaches that help in building, evaluating, evolving software
- Use mathematical and empirical scientific bases to go from craft/guild mentality to sound engineering practices that enable predictable development
- Deliver unprecedented value-adding capabilities for creativity, imagination, reasoning, communication
Past Community Successes

• Process:
  – Engineering: The waterfall model, the spiral model (Boehm), prototyping processes.
  – Science: How are processes like software? How can studying one help the other?
• Configuration Management/Version Control:
  – Engineering: Early work by Tichy, Rochkind. Now all software companies do this
  – Science: Syntactic/Semantic consistency. The essential nature as a set of constraints and relations. Absence of physical objects fosters a clearer view
• Internet Protocols:
  – Engineering: HTTP. (author lists confirm it springs from software engineering).
  – Science: Web shatters old preconceptions of everything, invites new understandings
• Integrated Development Environments:
  – Engineering: Gandalf, Arcadia, etc. anticipated products from Rational, Symantec
  – Science: Many types of “integration”. What is “seamless”? What do developers do?
• Testing Tools and Technology:
  – Engineering: Early research (coverage metrics, regression test) now industry standards
  – Science: What are the dimensions of quality? How to measure them? How to achieve them? How to know they have been achieved?
More Community Successes

• Inspections:
  – Engineering: Fagan, Mills pioneered this. Now a key part of most company practice
  – Science: What psychology, sociology, cognition apply here? A Theory of inspections

• Metrics:
  – Engineering: The COCOMO model, cyclomatic complexity, McCabe. The only thing non-technical managers understand.
  – Science: What can be measured? what should be measured? how to capture a wraith?

• High Reliability:
  – Engineering: Program verification (Hoare, Djikstra) now in use on real projects. Probably the European examples are the best.
  – Science: How to reason about software (Floyd)? What logics, rules of inference

• Requirements:
  – Engineering: Teichroew, Alford/Estrin, systems led to successful commercial products
  – Science: What are dimensions? How to define them? Relation to evaluation?

• Design:
  – Engineering: Theory, technology for modularity, encapsulation, separation of concerns.
  – Science: What is design? How does it relate to requirements, implementation? How to evaluate design. Lack of tangible product forces focus on essential issues here
Need More of This--at a Faster Pace

- Current practice shortcomings:
  - Lack of scientific frameworks leads to small tools, and narrow approaches
  - Lack of quality has led to acquiescence, but:
    - Superior quality chases out poor quality (at internet speed)
    - Will auto industry experience ca. 1960’s be repeated here?

- The Future must be characterized by:
  - Efficiencies from coherence, integration, reuse
  - Quality as a key competitive factor

- Will spring from empirical and theoretical science

- Where will essential underlying science come from?
  - Who will fund it? Demonstrate its tie to practice?
Current Frustrations

• Too much focus on software as a vehicle, not as the focus itself
• Mission Agency impatience with immaturity of the discipline
• Bravura engineering feats mistaken for science
• Important basic research too often sacrificed to fad-driven research
• Lack of commitment to careful concept evaluation
• Too little empirical validation/evaluation
• Ineffective technology insertion into real world
Strong NSF programs critical now

• Empirical research for problem understanding
• Develop scientific underpinnings
• Build and strengthen technology
• Lock these three into a synergistic feedback loop with each other
• Heighten awareness of quality, reliability issues
• Show power of scientific frameworks to provide coherence, speed, evolvability, portability
• Train skilled personnel to execute
• Promote application of basic science to other application areas
Community Action Plan

- Boehm Committee:
  - Panel being formed
  - Active discussion
  - Workshop in LA end of August

- Draft report
  - Strategic situation assessment
  - Community achievements summary
  - Software Grand Challenges
  - Strategic directions proposals

- Meeting with NSF Management on 27 September
  - A key opportunity for all of us to come together on this