What Makes Good Research in Software Engineering?

17-939A
Course Overview

Mary Shaw
September MMI

http://spoke.compose.cs.cmu.edu/ser01/
Course Description

Other science and engineering disciplines have well-refined explanations of their research paradigms. Software engineering research lacks these.

> Within the field, this impedes the design of research projects.
> Outside the field, it leads to characterizations such as "software engineering research is awfully soft and mushy."

This course addresses the problem by studying the body of software engineering research not only for its specific content but also to determine the research strategies that lead us to believe the results.

We will attempt to create a report that explains the paradigms by analyzing several subfields.
Objectives

Develop an understanding of the issues in doing research that on improving practical large-scale software construction together with some skill in addressing these issues.

A student should gain proficiency in three areas:

> General research skills
  • Read research papers critically, identifying and evaluating the reasoning.
  • Understand how research results evolve over time.
  • Apply software engineering results to your own work.

> Software engineering research methods
  • Exercise good judgement in recognizing good research problems.
  • Recognize types of research problems and select the appropriate paradigm.
  • Understand several validation techniques and when to apply them.

> Specific software engineering results
  • Understand results that span the field of software engineering.
Definition: What is “Engineering”?

Definitions abound. They have in common:

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

Engineering enables ordinary people
to do things that formerly required virtuosos

Engineering entails making decisions under the constraints of
limited time, knowledge, and resources
Characteristics of Engineering Practice

Iteration between analysis and design
Heavy use of earlier designs
Reliance on science when possible
Tradeoffs between alternatives
Handbooks, manuals
Pragmatic approach to cost-effectiveness
<table>
<thead>
<tr>
<th>Craft</th>
<th>Commercial</th>
<th>Professional Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtuosos &amp; talented amateurs</td>
<td>Skilled craftsmen</td>
<td>Educated professionals</td>
</tr>
<tr>
<td>Intuition &amp; brute force</td>
<td>Established procedure</td>
<td>Analysis &amp; theory</td>
</tr>
<tr>
<td>Haphazard progress</td>
<td>Pragmatic refinement</td>
<td>Progress relies on science</td>
</tr>
<tr>
<td>Casual transmission</td>
<td>Training in mechanics</td>
<td>Educated professional class</td>
</tr>
<tr>
<td>Extravagant use of available materials</td>
<td>Economic concern for cost &amp; supply of materials</td>
<td>Enabling new applications through analysis</td>
</tr>
<tr>
<td>Manufacture for use rather than sale</td>
<td>Manufacture for sale</td>
<td>Market segmentation by product variety</td>
</tr>
</tbody>
</table>
“Software Engineering”

Rallying Cry:
Phrase coined in 1968 to draw attention to software problems
Aspiration, not description

Requirement:
Cost-effective solutions ... ... to practical problems ... ... of increasing scale

Technical basis
Computer science
# Quick History of Software Engineering

<table>
<thead>
<tr>
<th>Period</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960-1965</td>
<td>Programming-anymwhich-way</td>
</tr>
<tr>
<td>1980-1985</td>
<td>Programming-in-the-large</td>
</tr>
<tr>
<td>1990-1995</td>
<td>Programming-in-the-world</td>
</tr>
</tbody>
</table>

- **1960-1965**: Mnemonics, precise prose, emphasis on small programs, representing structure, symbolic information, elementary understanding of control flow.
- **1970-1975**: Simple input-output specifications, emphasis on algorithms, data structures and types, programs execute once and terminate.
- **1980-1985**: Systems with complex specifications, emphasis on system integration, management, long-lived databases, program assemblies execute continually.
- **1990-1995**: Software integrated with hardware, emphasis on process improvement, system structure, abstractions for system design (client-server, object, ...), heavily interactive systems, multimedia.
Timeline

Programming-any-which-way

1950

Programming-in-the-small

1960

Programming-in-the-large

1970

Programming-in-the-world

1980

2000
Growth in Data Abstraction Granularity

- 1950: nonics, macros
- 1960: types in languages
- 1970: data structures
- 1980: abstract data types
- 1990: objects
- 2000: architectural chunks

- Programming-in-the-small
- Programming-in-the-large
- Programming-any-which-way
Key objectives

> Quality -- utility as well as functional correctness
> Cost -- both of development and of use
> Timeliness -- good-enough result, when it’s needed

Key strategies (e.g., for validating software)

> Do current things better
  » improve test coverage analysis, test case generation from specifications
> Make structural changes to make current things less necessary
  » formalize specifications to find more flaws earlier
> Develop new technology to exploit new opportunities
  » scale up model checking tools
> Refine understanding of real requirements
  » change objective from “correct” to “good enough for intended purpose”
Maturity: Progressive Codification Cycle

- Ad hoc solutions
- Folklore
- Improved practice
- Models & theories
- New problems
- Codification
Maturity: Kinds of Research Results

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
Progressive Codification

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

Sample progression:

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

To understand a discipline, ask what it produces, how it behaves, and what it values

Phenomena

What does this science investigate, design, create, and produce?

Questions

What sorts of questions does this science ask?
What implicit structure does it put on the world?

Methods

What research paradigms does this science accept as legitimate?

Criteria

What standards are used to evaluate the quality of research results?
Maturation model

> Six phases of software technology maturation
> Defined by five transition points

Interpretation

> Technology takes 17 years from key idea to widespread use
> Details depend on type of technology

Claim/conjecture

> Critical factors for moving technology into widespread use
  » Conceptual integrity
  » Clear recognition of need
  » Tuneability
  » Prior positive experience
  » Management committee
  » Training
Inhibitors and Facilitators of Transition

Inhibitors (slow down transition)
- Internal transfer
- High cost
- Contracting disincentives
- Psychological hurdles
- Easily modified technology

Facilitators (speed up transition)
- Prior success
- Incentives
- Technically astute managers
- Readily available help
- Latent demand
- Simplicity
- Incremental extension to current technology
Redwine/Riddle Maturation Model

Key Idea

Basic search
Identify problems
Identify 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

Usable capability

Outsiders use it

Production quality, commercial support

Final paper or system

Seminal paper or system
# Software Technology Maturation Points

<table>
<thead>
<tr>
<th>Technology Areas</th>
<th>58</th>
<th>59</th>
<th>60</th>
<th>61</th>
<th>62</th>
<th>63</th>
<th>64</th>
<th>65</th>
<th>66</th>
<th>67</th>
<th>68</th>
<th>69</th>
<th>70</th>
<th>71</th>
<th>72</th>
<th>73</th>
<th>74</th>
<th>75</th>
<th>76</th>
<th>77</th>
<th>78</th>
<th>79</th>
<th>80</th>
<th>81</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Engineering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compiler Construction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knowledge-Based Systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metrics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology Concepts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structured Programming</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abstract Data Types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Methodology Technology</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCR Methodology</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOD-STD-SDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFR 800-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Related Technology</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smalltalk 80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost Models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SREM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic Resch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concept Form</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev+Ext</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Exp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Exp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Popularize</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Before and After Clear Solution Appears

[Diagram showing the timeline and associated technologies or concepts with timeframes and milestones.]

Basic Resch  Concept Form  Dev+Ext  Internal Exp  External Exp  Popularize
Before and After Usable Capabilities
Phase Times and Publications

<table>
<thead>
<tr>
<th>Basic Resch</th>
<th>Concept Form</th>
<th>Dev+Ext</th>
<th>Internal Exp</th>
<th>External Exp</th>
<th>Popular</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Basic Research: 4 years
- Concept Form: 6 years
- Development + Extension: 8 years
- Internal Expansion: 8 years
- External Expansion: 10 years
- Popularization: 12 years
- Total: 30 years

Publication Venues
- Research Workshops
- Conferences
- Experiments
- Archival Journals
- Reviews
- Development Workshops
- Popular Journals
- Trade Publications
Joint Objective

Publishable report that explains software engineering research to software engineers and other computer scientists

Report organization (preliminary)

> Overview of research paradigms
  » based on pro forma abstracts

> Overview of idea evolution
  » based on Redwine/Riddle

> A set of reports on specific research areas
  » organize consistently
  » survey major results
  » explain research methods
  » show how field matures
## Current interests

<table>
<thead>
<tr>
<th>Who</th>
<th>Topic 1</th>
<th>Topic 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owen Cheng</td>
<td>SE for security</td>
<td>SW economics</td>
</tr>
<tr>
<td>George Fairbanks</td>
<td>Tools &amp; environments</td>
<td>SW analysis</td>
</tr>
<tr>
<td>Tim Halloran</td>
<td>Maintenance &amp; evolution</td>
<td>Tools &amp; environments</td>
</tr>
<tr>
<td>Beth Latronico</td>
<td>Reliability &amp; dependability</td>
<td>SW safety</td>
</tr>
<tr>
<td>Paul Li</td>
<td>Education</td>
<td>Economics driven</td>
</tr>
<tr>
<td>Annie Luo</td>
<td>Testing</td>
<td>Performance</td>
</tr>
<tr>
<td>Elissa Newman</td>
<td>Reverse engineering</td>
<td>Maintenance &amp; evolution</td>
</tr>
<tr>
<td>Vahe Poladian</td>
<td>SW analysis</td>
<td>SE for security</td>
</tr>
<tr>
<td>Charles Shelton</td>
<td>SW architecture</td>
<td>Middleware</td>
</tr>
<tr>
<td>Bridget Spitznagle</td>
<td>Reliability &amp; dependability</td>
<td>Reverse engineering</td>
</tr>
<tr>
<td>Dean Sutherland</td>
<td>Rev engr, maint/evol, anal</td>
<td>Safety, reliability, tools/env</td>
</tr>
</tbody>
</table>
# Current interests

<table>
<thead>
<tr>
<th>Topic</th>
<th>Choice 1</th>
<th>Choice 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW analysis</td>
<td>Vahe Poladian, (Dean Sutherland)</td>
<td>George Fairbanks</td>
</tr>
<tr>
<td>Maintenance &amp; evolution</td>
<td>Tim Halloran, (Dean Sutherland)</td>
<td>Elissa Newman, (Dean Sutherland)</td>
</tr>
<tr>
<td>Reliability &amp; dependability</td>
<td>Beth Latronico, Bridget Spitznagle</td>
<td>Bridget Spitznagle</td>
</tr>
<tr>
<td>Reverse engineering</td>
<td>Elissa Newman, (Dean Sutherland)</td>
<td>George Fairbanks</td>
</tr>
<tr>
<td>SE for security</td>
<td>Owen Cheng</td>
<td>Vahe Poladian</td>
</tr>
<tr>
<td>Tools &amp; environments</td>
<td>George Fairbanks</td>
<td>Tim Halloran, (Dean Sutherland)</td>
</tr>
<tr>
<td>SW architecture</td>
<td>Charles Shelton</td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td>Paul Li</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>Annie Luo</td>
<td></td>
</tr>
<tr>
<td>SW economics</td>
<td></td>
<td>Owen Cheng, Paul Li</td>
</tr>
<tr>
<td>Middleware</td>
<td></td>
<td>Charles Shelton</td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td>Annie Luo</td>
</tr>
<tr>
<td>SW safety</td>
<td></td>
<td>Beth Latronico, (Dean Sutherland)</td>
</tr>
</tbody>
</table>

-939A