Software Engineering Research Strategies and the role of Empirical Studies

Mary Shaw
Carnegie Mellon University
www.cs.cmu.edu/~shaw/

Good Research in Software Engineering

Examine the kinds of research questions software engineers ask and the ways they study those questions

- Research questions are of different kinds
- Kinds of interesting questions change as ideas mature
- Research strategies also vary
- They should be selected to match the research questions
- Ideas mature over time
  - They grow from qualitative and empirical understanding to precise and quantitative models
- Good papers are steps toward good results
  - Each paper provides some evidence, but overall validation arises from accumulated evidence

Plan

- Engineering and software engineering
  - Quick history
- Life cycle of a technological innovation
  - Different issues, venues at different stages
- Focus on research papers
  - Various authors, conference advice
- Elements of a research presentation
  - Question, result, validation
- Research strategies that work
  - The logical structure of a project and paper
  - Examples from ICSE02, ICSE03, EASE04

Definition: What is "Engineer ing"?

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
The Coming-of-Age of Software Architecture Research

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

Virtuosos & talented amateurs

Intuition & brute force

Haphazard progress

Casual transmission

Extravagant use of available materials

Manufacture for use rather than sale

Skilled craftsmen

Established procedure

Pragmatic refinement

Training in mechanics

Economic concern for cost & supply of materials

Manufacture for sale

Educated professionals

Analysis & theory

Progress relies on science

Educated professional class

Enabling new applications through analysis

Market segmentation by product variety

Craft Commercial Professional

Production

Commercial

Science

Professional Engineering

"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
The Coming-of-Age of Software Architecture Research

Quick History of Software Engineering

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mnemonics, precise use of prose</td>
<td>Emphasis on small programs</td>
<td>Systems with complex specifications</td>
<td>Software integrated with hardware</td>
<td>Distributed, ad-hoc specs</td>
</tr>
<tr>
<td>Emphasis on small programs</td>
<td>Representing structure, symbolic information</td>
<td>Emphasis on process improvement, system structure</td>
<td>Emphasis on integrating resources, frameworks, integration structures</td>
<td></td>
</tr>
<tr>
<td>Elementary understanding of control flow</td>
<td>Programs execute once and terminate</td>
<td>Program assemblies execute continually</td>
<td>Embedded systems</td>
<td>Heavily interactive systems, multimedia</td>
</tr>
</tbody>
</table>

Timeline

- 1950 Programming-any-which-way
- 1960 Programming-in-the-small
- 1970 Programming-in-the-large
- 1980 Programming-in-the-world
- 1990 Programming-in-the-net
- 2000 Programming-in-the-net

Growth in Data Abstraction Granularity

- 1950 Programming-any-which-way
- 1960 Programming-in-the-small
- 1970 Programming-in-the-large
- 1980 Programming-in-the-world
- 1990 Programming-in-the-net
- 2000 Programming-in-the-net

Software Engineering Research

- 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 motivations (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”

Mary Shaw

1/14/2005
“No-No”s for Software Engineering Research

- Assume that a result demonstrated for a 10K-line system will scale to a 500K-line system
- Expect everyone to do things “my way”
- Believe functional correctness is sufficient
- Assume the existence of a complete, consistent specification
- Just build things without extracting enduring lessons
- Devise a solution in ignorance of how the world really works

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?

Plan

- Engineering and software engineering
  - Quick history
- Life cycle of a technological innovation
  - Different issues, venues at different stages
- Focus on research papers
  - Various authors, conference advice
- Elements of a research presentation
  - Question, result, validation
- Research strategies that work
  - The logical structure of a project and paper
  - Examples from ICSE02, ICSE03, EASE04

Redwine/Riddle Maturation Model

- 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

Key Idea

Seemal paper or system
Usable capability
Outsiders use
Production quality
Commercial support
The Coming-of-Age of Software Architecture Research

Another View: Progressive Codification

- Folklore
- Qualitative models
- Ad hoc solutions
- Improved practice
- Empirical models
- New problems
- Structural models
- Formal theories

Example: COCOMO

Warning: this is my interpretation

- COCOMO first appeared to be empirical curve fit
  > Basic COCOMO 81 was for quick estimates
    \[ MM = 2.4 \times (KDSI)^{1.05} \]
  and soon had three modes, with different constants
  > Intermediate COCOMO 81 had the same three development modes, plus 15 cost drivers
    - Organic \[ MM = 3.2 \times (KDSI)^{1.05} \prod (driver_i) \]
    - Organic \[ MM = 3.0 \times (KDSI)^{1.12} \prod (driver_i) \]
    - Organic \[ MM = 2.8 \times (KDSI)^{1.20} \prod (driver_i) \]
  > Detailed COCOMO81 also addressed phase distribution of costs

COCOMO Cost Drivers

- Drivers started with qualitative ratings (“very low”)
- These came with guidance (“slight inconvenience”)
- They also came with generic effort multipliers (.75)
- Better fits were available if the multipliers were calibrated specifically for an organization

COCOMO II

- Provides somewhat different, increasingly accurate models for applications composition, early design, post-architecture, and other development modes
- Enables organizations to tailor model to their needs
- Provides a constructive model, adding more cost drivers to help explain project descriptions
- Splits cost drivers into 16 effort multipliers and 6 scale factors that affect the exponent
  \[ PM = A \times (Size)^E \prod_i EM_i \]
  where \[ E = B + 0.01 \sum_j SF_j \]
  and \( A, B \) are calibrated to 161 projects in the database and determination of Size depends on sub-model
The Coming-of-Age of Software Architecture Research

Software Technology Maturation Points

Maturation Times

Basic Research Concept Form Dev + Ext Internal Exp External Expl Popularize

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

Phase Times and Publications
Success needs cumulative evidence

- A single paper has limited scope
  - Conference papers can hold one idea
  - Journal papers can wrap up individual results
- Results are more convincing if they are confirmed in different ways (triangulation)
- Each promising step justifies investment in next (often more expensive) step

Plan

- Engineering and software engineering
  - Quick history
- Life cycle of a technological innovation
  - Different issues, venues at different stages
- Focus on research papers
  - Various authors, conference advice
- Elements of a research presentation
  - Question, result, validation
- Research strategies that work
  - The logical structure of a project and paper
  - Examples from ICSE02, ICSE03, EASE04

Research Styles

- Physics and medicine have well-recognized research styles
  - Hypothesis, controlled experiment, analysis, refutation
  - Double-blind large-scale studies
- Acceptance of results relies on process as well as analysis
- Simplified versions help to explain the field to observers
- Fields can be characterized by identifying what they value:
  - What kinds of questions are “interesting”?
  - What kinds of results help to answer these questions?
    - What research methods can produce these results?
  - What kind of evidence demonstrates the validity of a result?

Critiques of Experimental CS/SE

“Computer scientists publish relatively few papers with experimentally validated results … The low ratio of validated results appears to be a serious weakness in CS research. This weakness should be rectified”

- Studies over past few years criticize computer science for failure to collect, report, analyze experimental data
- They start with the premise that data must be collected, then analyze papers and find data lacking
- I ask a different question:
  What are the characteristics of software engineering research that the field recognizes as quality research?


The Coming-of-Age of Software Architecture Research

Newman: Pro Forma Abstracts

- Asked, “To what extent is HCI an engineering discipline”?
- Characterized engineering research products
- Created three pro forma abstracts, templates describing research
- 90% of papers in engineering research fit these templates

Newman’s Pro Forma Templates for Engineering

**EM: Enhanced model**

Existing model-type models are deficient in dealing with properties of solution strategy. An enhanced model-type is described, capable of providing more accurate analyses / predictions of properties in solution strategy designs. The model has been tested by comparing analyses / predictions with empirically measured values of properties.

**ES: Enhanced solution**

Studies of existing solutions have shown deficiencies on property. An enhanced design for an existing solution is described, based on solution strategy. In comparison with existing solutions, it offers enhanced levels of property, according to analyses based on model-type. These improvements have been confirmed / demonstrated in tests of a working artifact-type based on the design.

**ET: Enhanced tool**

The effectiveness of model-type / solution strategy in supporting the design of artifact-type has been demonstrated. An enhanced tool / method is described for the design of artifact-type based on model-type / solution strategy. Examples are provided confirming the effectiveness of its support for model-type / solution strategy in design.

Newman: Pro Forma Abstracts

- Only 25-30% of HCI papers fit
- Created 2 more pro forma abstracts (arguably engineering)
- Now 95% of HCI papers fit
- Notes
  > Preliminary study, e.g., no check on inter-rater reliability
  > Found this a useful device for reading papers
  > Influenced refereeing in CHI

Additional Pro Forma Templates for HCI

**RS: Radical solution**

A radical solution to the problem of problem definition is described, based on solution strategy. In comparison with existing normal solutions it offers advantages, which have been demonstrated in preliminary tests, but it leaves a number of side effects to be addressed including list-of-side-effects. Strategies are suggested for addressing these side effects.

**XH: Experience and/or Heuristic**

Studies reported here of application supported by supporting technology generate a number of findings concerning issues, including list-of-findings. They indicate that requirement is / is not met by design-heuristic.
Brooks: Kinds of Research Results

Brooks proposed 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

Plan

- Engineering and software engineering
  - Quick history
- Life cycle of a technological innovation
  - Different issues, venues at different stages
- Focus on research papers
  - Various authors, conference advice
- Elements of a research presentation
  - Question, result, validation
- Research strategies that work
  - The logical structure of a project and paper
  - Examples from ICSE02, ICSE03, EASE04

What Do Good Software Engineers Do?

- I’ve been looking for the underlying scientific argument in software engineering papers for some time
  - Two “open-mike” sessions at FSEs
  - ICSE 2001 talk and paper
  - PhD course in critical reading of software engineering research
    - Spring 2000, Fall 2001
  - Observations of ICSE 2002, WICSA3 program committees
- I’ve summarized my observations in a model based on the questions, results, and validation methods of the research
  - v0.3.0, still alpha
  - Observations, in Brooks’ sense, perhaps becoming generalizations

Research Objectives

- Key objectives
  - Quality -- utility as well as functional correctness
  - Cost -- both of development and of use
  - Timeliness -- good-enough result, when it’s needed
- Address problems that affect practical software
## The Coming-of-Age of Software Architecture Research

### Types of Research Questions

<table>
<thead>
<tr>
<th>Research Question Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method/means of development</td>
<td>How can we do/create/automate X?</td>
</tr>
<tr>
<td>Method for analysis</td>
<td>What is a better way to do/create X?</td>
</tr>
<tr>
<td>Evaluation/analysis of an instance</td>
<td>How can I evaluate the quality of X?</td>
</tr>
<tr>
<td>Generalization/characterization</td>
<td>What is property X of artifact/method Y?</td>
</tr>
<tr>
<td>Feasibility</td>
<td>Is X always true of Y? Given X, what is Y?</td>
</tr>
<tr>
<td></td>
<td>What, exactly, do we mean by X?</td>
</tr>
<tr>
<td></td>
<td>Is Y a good formal/empirical model for X?</td>
</tr>
<tr>
<td></td>
<td>What are the types of X, how are they related?</td>
</tr>
<tr>
<td></td>
<td>Does X exist, and what is it?</td>
</tr>
<tr>
<td></td>
<td>Is it possible to do X at all?</td>
</tr>
</tbody>
</table>

### ICSE 2002 submissions

**Research Questions**

- **Method/means of development**: 142 (48%), 18 (42%), 18 (13%)
- **Method for analysis**: 95 (32%), 19 (44%), 20 (13%)
- **Evaluation/analysis of an instance**: 43 (14%), 5 (12%), 12 (12%)
- **Generalization/characterization**: 18 (6%), 1 (2%), 6% (6%)
- **Feasibility study or exploration**: 0 (0%), 0 (0%), 0 (0%)

**Total**: 298 (100.0%), 43 (100.0%), 14% (14%)

### Types of Research Results

<table>
<thead>
<tr>
<th>Research Result Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedure/technique</td>
<td>New/better ways to do development/analysis tasks (operational, not just guidelines)</td>
</tr>
<tr>
<td>Qualitative or descriptive model</td>
<td>Structure/taxonomy for problem areas, framework Informal guidance, informal domain analysis</td>
</tr>
<tr>
<td>Analytic model</td>
<td>Structural model that permits formal analysis, automation</td>
</tr>
<tr>
<td>Empirical model</td>
<td>Empirical predictive models based on real data</td>
</tr>
<tr>
<td>Tool/notation</td>
<td>Tool or notation that embodies model or technique</td>
</tr>
<tr>
<td>Specific solution</td>
<td>Solution to application problem applying SE principles, or result of specific analysis</td>
</tr>
<tr>
<td>Report</td>
<td>Interesting observations, rules of thumb</td>
</tr>
</tbody>
</table>

### ICSE 2002 submissions

**Research Results**

- **Procedure/technique**: 152 (44%), 28 (85%), 18% (18%)
- **Qualitative or descriptive model**: 94 (26%), 4 (7%), 6% (6%)
- **Analytic model**: 41 (12%), 1 (2%), 2% (2%)
- **Tool/notation**: 46 (14%), 7 (13%), 13% (13%)
- **Specific solution, prototype, axiom, or judgment**: 18 (14%), 1 (2%), 12% (12%)
- **Report**: 11 (3%), 0 (0%), 0% (0%)

**Total**: 248 (100.0%), 35 (100.0%), 10% (10%)
The Coming-of-Age of Software Architecture Research

Types of Research Validation

<table>
<thead>
<tr>
<th>Analysis</th>
<th>I have found my result satisfactory through ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal model</td>
<td>rigorous derivation and proof</td>
</tr>
<tr>
<td>Empirical model</td>
<td>data on use in controlled situation</td>
</tr>
<tr>
<td>Controlled experiment</td>
<td>carefully designed statistical experiment</td>
</tr>
<tr>
<td>Experience</td>
<td>My result has actually been used; the evidence is</td>
</tr>
<tr>
<td>Qualitative model</td>
<td>narrative</td>
</tr>
<tr>
<td>Empirical model, tool</td>
<td>data, usually statistical, on practice</td>
</tr>
<tr>
<td>Notation, technique</td>
<td>comparison of systems in actual use</td>
</tr>
<tr>
<td>Example</td>
<td>Here’s how my result works on a small example</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Given these criteria, my result ...</td>
</tr>
<tr>
<td>Descriptive model</td>
<td>adequately describes phenomena of interest</td>
</tr>
<tr>
<td>Empirical model</td>
<td>is able to predict ... because ...</td>
</tr>
<tr>
<td>Persuasion</td>
<td>I thought hard about this, and I believe...</td>
</tr>
<tr>
<td>Blatant assertion</td>
<td>No serious attempt to evaluate result</td>
</tr>
</tbody>
</table>

ICSE 2002 submissions

<table>
<thead>
<tr>
<th>Type of validation</th>
<th>Submitted</th>
<th>Accepted</th>
<th>Ratio Acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>48(16%)</td>
<td>11(26%)</td>
<td>23%</td>
</tr>
<tr>
<td>Evaluation</td>
<td>21(7%)</td>
<td>1(2%)</td>
<td>5%</td>
</tr>
<tr>
<td>Experience</td>
<td>34(11%)</td>
<td>8(19%)</td>
<td>24%</td>
</tr>
<tr>
<td>Example</td>
<td>82(27%)</td>
<td>16(37%)</td>
<td>20%</td>
</tr>
<tr>
<td>Persuasion</td>
<td>25(8%)</td>
<td>0(0.0%)</td>
<td>0%</td>
</tr>
<tr>
<td>No mention of validation in abstract</td>
<td>84(28%)</td>
<td>6(14%)</td>
<td>7%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>300(100%)</td>
<td>43(100%)</td>
<td>14%</td>
</tr>
</tbody>
</table>

Commonest Types of ICSE 2002 Papers

- Question
  - Most common: improved method or means of developing software
  - Also fairly common: papers about methods for analysis, principally analysis of correctness (testing and verification)
- Result
  - Most common: a new procedure or technique for some aspect of software development
  - Not unusual: a new analytic technique
- Validation
  - Most common: analysis and experience in practice
  - Also fairly common: example idealized from practice
  - Common in submissions but not acceptances: persuasion

Building Blocks for Research

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devlpmt method</td>
<td>Proc/technique</td>
<td>Analysis</td>
</tr>
<tr>
<td>Analysis method</td>
<td>Qual/des model</td>
<td>Experience</td>
</tr>
<tr>
<td>Evaluate instance</td>
<td>Analytic model</td>
<td>Example</td>
</tr>
<tr>
<td>Generalization</td>
<td>Tool/notation</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Feasibility</td>
<td>Specific solution</td>
<td>Persuasion</td>
</tr>
<tr>
<td>Report</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Observation about EASE04

- Typical software engineering papers take the form
  > I set out to do X
  > I produced result Y
  > I validated the result by Z
- However, a typical EASE paper takes the form
  > I wanted to understand P
  > I designed experiment Q
  > I learned R, with confidence S and caveats T

Another Observation about EASE04

- Paper structures much more standard than other conferences I’m familiar with

<table>
<thead>
<tr>
<th>Section title</th>
<th>Match</th>
<th>Year-match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Related work</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(Research) method or experimental design</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Results</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Analysis or Evaluation</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>At least 2 of conclusions/discussion/future work</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Conclusions</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Discussion</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Future work</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Plan

- Engineering and software engineering
  > Quick history
- Life cycle of a technological innovation
  > Different issues, venues at different stages
- Focus on research papers
  > Various authors, conference advice
- Elements of a research presentation
  > Question, result, validation
- Research strategies that work
  > The logical structure of a project and paper
  > Examples from ICSE02, ICSE03, EASE04

Complete Research Result

- Real World
  Practical problem
  Validation Task 2: Does the result help to solve the practical problem?
  Real World
  Solution to practical problem

- Research Setting
  Idealized problem
  Validation Task 1: Does the product solve the idealized problem?
  Research Setting
  Solution to idealized problem

Research product
  (technique, method, model, system, …)
The Coming-of-Age of Software Architecture Research

Two Common Plans

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can X be better?</td>
<td>New method</td>
<td>Analysis</td>
</tr>
<tr>
<td>Can X tell you Y?</td>
<td>Qual/desc model</td>
<td>Report</td>
</tr>
<tr>
<td>Evaluate instance</td>
<td>Analytic model</td>
<td></td>
</tr>
<tr>
<td>Generalization</td>
<td>Empirical model</td>
<td>Careful examples</td>
</tr>
<tr>
<td>Feasibility</td>
<td>Specific solution</td>
<td>Evaluation</td>
</tr>
<tr>
<td></td>
<td>Report</td>
<td>Persuasion</td>
</tr>
</tbody>
</table>

- Can X be better?
  - New method
  - Analysis

- Can X tell you Y?
  - Qual/desc model
  - Report

- Evaluate instance
  - Analytic model

- Generalization
  - Specific solution

- Feasibility
  - Report


*Question (Analysis method): How can we automatically verify that a finite state machine specification is a safe abstraction of a C procedure?*

*Result (Technique, supported by tool):*
  - Extract finite model from C source code (using predicate abstraction and theorem proving); show conformance via weak simulation.
  - Decompose verification to match software design so results compose.
  - Tool interfaces with public theorem provers

*Validation (Examples):*
  - Use examples whose correct outcome is known
  - Compare performance with various public provers incorporated
  - Verify OpenSSL handshake


*Question (Development method): How can we improve on the traditional approach to document authoring?*

*Result (Technique):*
  - Document authored by team in series of workshops
  - Workshops are highly structured around concrete issues

*Validation (Experience):*
  - In use in Nokia since 2000
  - Self-assessment by survey in 2001, good results
    - reduces calendar time for document
    - improves communication
    - reduces defects
## Empirical Validation

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devlpmt method</td>
<td>Can we predict cost?</td>
<td>Statistical comparison</td>
</tr>
<tr>
<td>Evaluate instance</td>
<td>Qual/desc model</td>
<td>Experience</td>
</tr>
<tr>
<td>Generalization</td>
<td>Empirical model</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Feasibility</td>
<td>Tool/notation</td>
<td>Persuasion</td>
</tr>
</tbody>
</table>

### Question (Analysis method): Can we estimate costs of developing web applications?

### Result (Technique): Tailor existing COBRA method for web applications
- Get data set from web development company

### Validation (Analysis, statistically valid):
- Establish evaluation criteria through interviews
- Apply tailored COBRA, least squares, and company’s informal model
- Compare results in several ways, including t-tests

---

## A Generalization Paper

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devlpmt method</td>
<td>Proc/technique</td>
<td>Analysis</td>
</tr>
<tr>
<td>Analysis method</td>
<td>Careful generalization</td>
<td>Report actual use</td>
</tr>
<tr>
<td>Evaluate instance</td>
<td>Analytic model</td>
<td>Example</td>
</tr>
<tr>
<td>Specific solution</td>
<td>Tool/notation</td>
<td>Evaluation</td>
</tr>
</tbody>
</table>

### Question (Generalization): What are benchmarks, in general, and how could using them improve software engineering research?

### Result (Qualitative model):
- Examine three successful benchmarks
- Formulate descriptive theory
- Describe how theory should inform practice

### Validation (Experience):
- Apply theory to interpret two reverse engineering benchmarks
- Identify three areas that are ripe for benchmarking
The Coming-of-Age of Software Architecture Research

Typical EASE paper (about half)

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devlpmt method</td>
<td>Proc/technique</td>
<td>Analysis</td>
</tr>
<tr>
<td>Analysis method</td>
<td>Qual/desc model</td>
<td>Experience</td>
</tr>
<tr>
<td>Evaluate instance</td>
<td>Empirical model</td>
<td>Example</td>
</tr>
<tr>
<td>Generalization</td>
<td>Tool/notation</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Feasibility</td>
<td>Specific solution</td>
<td>Persuasion</td>
</tr>
</tbody>
</table>

A Common, but Bad Plan

An Uncommon, but Good Plan

Can X be better? New method

Look, it works!

Sometimes a breakthrough (but sometimes nonsense)

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devlpmt method</td>
<td>Proc/technique</td>
<td>Analysis</td>
</tr>
<tr>
<td>Analysis method</td>
<td>Qual/desc model</td>
<td>Experience</td>
</tr>
<tr>
<td>Evaluate instance</td>
<td>Empirical model</td>
<td>Example</td>
</tr>
<tr>
<td>Generalization</td>
<td>Tool/notation</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Feasibility</td>
<td>Specific solution</td>
<td>Persuasion</td>
</tr>
</tbody>
</table>

ICSE02, ICSE03, EASE Paper Types

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devlpmt method</td>
<td>Proc/technique</td>
<td>Analysis</td>
</tr>
<tr>
<td>Qual Model</td>
<td>Emp Model</td>
<td>Analytic model</td>
</tr>
<tr>
<td>Anal Model</td>
<td>Notation</td>
<td>Example</td>
</tr>
<tr>
<td>Spec Soln</td>
<td>Report</td>
<td>Persuasion</td>
</tr>
</tbody>
</table>

Mary Shaw

15

1/14/2005
Newman’s “Enhanced Model”

**EM: Enhanced model**

Existing model-type models are deficient in dealing with properties of solution strategy. An enhanced model-type is described, capable of providing more accurate analyses/predictions of properties in software designs. The model has been tested by comparing analyses/predictions with empirically measured values of properties.

**Key:** EM provides new or better way of looking at problems

**Question**

Generalization / characterization: What, exactly do we mean by X? What is a good formal/empirical model of X?

**Result**

Models, preferably analytic or empirical, but precise descriptive or qualitative are acceptable

**Validation**

Empirical analysis, controlled experiment; perhaps experience

---

**Pro Forma Research Strategies**

Locating the *pro forma* abstracts in research strategy space

<table>
<thead>
<tr>
<th>Devlp Meth</th>
<th>Anal Meth</th>
<th>Eval Meth</th>
<th>Exp Meth</th>
<th>Qual Meth</th>
<th>Sim Meth</th>
<th>Purpose</th>
<th>Document Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qual Meth</td>
<td>Qual Meth</td>
<td>Qual Meth</td>
<td>Qual Meth</td>
<td>Qual Meth</td>
<td>Qual Meth</td>
<td>Qual Meth</td>
<td>Qual Meth</td>
</tr>
<tr>
<td>Emp Meth</td>
<td>Emp Meth</td>
<td>Emp Meth</td>
<td>Emp Meth</td>
<td>Emp Meth</td>
<td>Emp Meth</td>
<td>Emp Meth</td>
<td>Emp Meth</td>
</tr>
<tr>
<td>Anal Meth</td>
<td>Anal Meth</td>
<td>Anal Meth</td>
<td>Anal Meth</td>
<td>Anal Meth</td>
<td>Anal Meth</td>
<td>Anal Meth</td>
<td>Anal Meth</td>
</tr>
<tr>
<td>Notation</td>
<td>Notation</td>
<td>Notation</td>
<td>Notation</td>
<td>Notation</td>
<td>Notation</td>
<td>Notation</td>
<td>Notation</td>
</tr>
</tbody>
</table>

---

**Putting the Words on Paper**

- A research paper is a purposeful, designed artifact
  > Just like a software system
- Apply software design techniques to paper design
  > Start with the requirement: read the call for papers
  > Select an architecture: plan the sections, what they say
  > Plan a schedule: allow time for review, revision
  > Check consistency: type-check text like code
- See writing guidance at
  > [www.cs.cmu.edu/~shaw](http://www.cs.cmu.edu/~shaw) > Education > WordWright
The Coming-of-Age of Software Architecture Research

Examine the kinds of research questions software engineers ask and the ways they study those questions

• Research questions are of different kinds
  Kinds of interesting questions change as ideas mature
• Research strategies also vary
  They should be selected to match the research questions
• Ideas mature over time
  They grow from qualitative and empirical understanding to precise and quantitative models
• Good papers are steps toward good results
  Each paper provides some evidence, but overall validation arises from accumulated evidence

Final word – about this report

• In Brooks’ sense, a rule of thumb or generalization
• Not a technical result (a finding) …
  > No attempt to show anyone else can apply the model
  > No principled analysis
  > Limited data
    » one full set of abstracts and observation of PC
    » one set accepted papers as published
  > Use of abstracts as proxies for full papers is suspect
    » Though accepted 2003 papers suggest they’re not bad
  > Little discussion of related work