The Coming-of-Age of Software Architecture Research

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

Executive Summary

• Technical ideas in software engineering evolve in predictable ways
• This evolution pattern sets expectations for software architecture research

• Lessons for developers: what to expect of software architecture research, now and in the future
• Lessons for researchers: useful research paradigms and validation techniques

Definitions abound. They have in common:
Creating cost-effective solutions ...
... to practical problems ...
... by (preferring to) apply scientific knowledge ...
... building things ...
... in the service of mankind

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

Software Architecture Research

Software architecture research is the principled study of the overall structure of software systems, especially the relations among subsystems and components.
The Coming-of-Age of Software Architecture Research

Redwine/Riddle Maturation Model

Key Ideas
- Basic Research: Recognize problem, form ideas
- Concept Formation: Refine ideas, publish solutions
- Development & Extension: Try it out, clarify, refine
- Internal Exploration: Stabilize, port, use for real problems
- External Exploration: Broadcast, user group, external
- Popularization: Propagate through community

Seminal paper or system
- Usable capability
- Production quality, commercial support

Software Technology Maturation Points

Maturation Times

Transition Points for Abstract Data Types

- Basic research ==> concept formation
  > 1968: formulation of information hiding
- Concept formation ==> development & extension
  > 1973: abstract data type models
- Development & extension ==> internal exploration
  > 1977: incorporation in programming languages
- Internal exploration ==> external exploration
  > 1980: incorporation in other technologies
- External exploration ==> Popularization
  > late 80’s: object models, C++, Java
The Coming-of-Age of Software Architecture Research

**Basic Research in Software Architecture**

- Architectures for particular product lines or application domains (e.g., oscilloscopes, missile control)
- Catalogs that captured informal use of box and line diagrams
- Key idea: models for architectural style

*Late 1980s to 1992/3*

**Concept Formulation in Software Architecture**

- Architecture description languages
  > capture design abstractions in system definition
  > exploit properties of particular styles
  > support style-specific compilation or analysis
- Prototype implementations support further experiments
- Formalization, both within language and standalone
- Catalogs expanded to taxonomies

*Mid 1990s and ongoing*

**“Software Architecture” citation counts**

- NEC Bibliography citation counts
  - “software architecture” in entry (2000 of 3595 citations)
  - “software architecture” in title (1831 citations)

Falloff in 1997-99 may reflect delay in updating bibliography and/or normal publication lags

**Software Architecture 24 Most Cited**

- Published 1990-1998
- 20 papers
  > 8 architectures for particular domains (1990-1995)
  > 7 surveys or models for field (1992-1995)
  > 3 formalizations (1993-1996)
  > 1 architecture description language (1995)
  > 1 reverse engineering (1997)
The Coming-of-Age of Software Architecture Research

**Development and Extension in Software Architecture**
- Unification and refinement
  - ACME interchange language
  - Refinement of taxonomies
- Maturing institutions
  - Workshops, special issues, conference tracks
  - Books
  - Independent conference

*Late 1990s and ongoing*

**Internal Enhancement and Exploration in Software Architecture**
- Architectural styles used informally as design guides
- Some formal analyses of real systems
- Books/courses on software architecture

*From 1998 onward*

**External Enhancement and Exploration in Software Architecture**
- Recognition of the value of architects and architectural design considerations
  - Processes require architectural design reviews and explicit architectural documentation
  - Emerging use of product line architectures, commercial architectural standards, component integration frameworks
- Tool-supported methods
  - Architectural views in UML
  - Architectural Tradeoff Analysis Method

*Very late 1990s and ongoing*

**Popularization of Software Architecture**
- Emerging standards
  - IEEE-Std-1471-2000, recommended Practice for Architectural Description of Software-Intensive Systems
  - Standards for particular component families
    - But these reflect component reuse interests as much as architecture

*Just beginning*
The Coming-of-Age of Software Architecture Research

Redwine/Riddle Maturation Model

Focus on Research

• Recall …
  
  SE research is a prime source of the systematic knowledge
  SE research problems should reflect the constraints

• Turn now to research strategies

Research Strategy

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

• Ideas mature over time
  
  They grow from qualitative and empirical understanding to precise and quantitative models

• 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

Research Objectives

Real World
Practical problem

Real World
Solution to practical problem

• 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>Types</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility</td>
<td>Does X exist, and what is it?</td>
</tr>
<tr>
<td>Characterization</td>
<td>What are the characteristics of X?</td>
</tr>
<tr>
<td>Method/Means</td>
<td>How can we do X?</td>
</tr>
<tr>
<td>Generalization</td>
<td>Is X always true of Y?</td>
</tr>
<tr>
<td>Evaluation/Discrimination</td>
<td>How can I evaluate the quality of X?</td>
</tr>
<tr>
<td>Product</td>
<td>What is a (better) design for application X?</td>
</tr>
</tbody>
</table>

### Examples of Research Questions about Software Structure

| Feasibility      | Is it possible to describe relations among components of a software system? |
| Characterization | What is the important information to share and hide about a component?    |
| Method/Means     | How can we incorporate abstract data types in a programming language?   |
| Generalization   | What is a formal relation between the specification of an abstract data type and its implementation? |
| Evaluation/Discrimination | What organization should I choose for a user interface component? |
| Product          | What’s a good structure for avionics?                                    |

### Research Strategy

- **Real World**
  - Practical problem
  - Research Setting
    - Idealized problem
  - Research product
    - (technique, method, model, system, …)
  - Real World
    - Solution to practical problem

---

Mary Shaw  6  10/29/01
The Coming-of-Age of Software Architecture Research

Types of Research Results

- **Qualitative & descr. model**: Report interesting observations, generalizations
  - Structure a problem area; ask good questions
- **Empirical model**: Develop empirical predictive models from observation
- **Analytic model**: Develop structural models that permit formal analysis
- **Notation**: Embodi model in notation with a calculus, semantics
- **Technique**: Invent new ways to do some tasks, including
  - evaluation, implementation, selection from alternatives
- **Tool**: Embodie technique (development or eval) in useful tool
- **Specific sol’n**: Report solution to application problem that shows
  - SE principles – may be design, not implementation
- **System (incl. prototype)**: Embodi result in a system, using the system
  - both for insight and as carrier of results

Maturity: 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

Examples of Software Architecture Research Results

- **Qualitative & descr. model**: Early architectural models
  - Architectural patterns
- **Empirical model**: Communication metrics as indicator of project complexity
- **Analytic model**: Formal specification of HLA for simulation
- **Notation**: Architecture Description Languages
- **Technique**: Domain-specific software architectures
  - Selection of UI system structure from alternatives
- **Tool**: UML to support object-oriented design
- **Specific sol’n**: Higher Level Architecture for integrating simulations
- **System (incl. prototype)**: Various architecture description languages

Research Objectives

Research Setting

- **Research Setting**: Validation Task 1: Does the product solve the idealized problem?

Research Setting

- **Research Setting**: Solution to idealized problem

Research product

- **Research product**: (technique, method, model, system, …)

Real World

- **Practical problem**: Solution to practical problem

Real World

- **Practical problem**: Solution to practical problem
The Coming-of-Age of Software Architecture Research

### Types of Research Validation

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persuasion</td>
<td>I thought hard about this, and I believe...</td>
</tr>
<tr>
<td>Implementation</td>
<td>Here is a prototype of a system that...</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Given these criteria, the object rates as...</td>
</tr>
<tr>
<td>Analysis</td>
<td>Given the facts, here are consequences...</td>
</tr>
<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>Report on use in practice</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>
</tbody>
</table>

### Examples of Software Architecture Research Validation

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persuasion</td>
<td>Early architectural models</td>
</tr>
<tr>
<td>Implementation</td>
<td>Early architecture description languages</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Taxonomies; performance improvement</td>
</tr>
<tr>
<td>Analysis</td>
<td>(Many predictive models)</td>
</tr>
<tr>
<td>Formal model</td>
<td>HLA for distributed simulation</td>
</tr>
<tr>
<td>Empirical model</td>
<td>User interface structure</td>
</tr>
<tr>
<td>Controlled experiment</td>
<td>Emp stud of O-O vs ad hoc inspection</td>
</tr>
<tr>
<td>Experience</td>
<td>(Many case studies)</td>
</tr>
<tr>
<td>Qualitative model</td>
<td>Architectural patterns</td>
</tr>
<tr>
<td>Empirical model, tool</td>
<td>Domain-specific architectures</td>
</tr>
<tr>
<td>Notation, technique</td>
<td>Communication and project complexity</td>
</tr>
</tbody>
</table>

### Complete Research Result

- **Real World**
  - Practical problem
  - Research Setting: Idealized problem
  - Research product (technique, method, model, system, ...)
  - Validation Task 1: Does the product solve the idealized problem?
  - Validation Task 2: Does the result help solve the practical problem?

- **Real World**
  - Solution to practical problem

### “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
Newman: Pro Forma Abstracts

- Asked, “To what extent is HCI an engineering discipline”?
- Characterized engineering research products
- Created 3 pro forma abstracts, templates describing research
- 90% of papers in engineering research fit these templates
- 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

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 artifact-type have shown deficiencies on property. An enhanced design for an artifact-type 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.

Newman’s Pro Forma Templates for Engineering

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.

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.
The Coming-of-Age of Software Architecture Research

Building Blocks for Research

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility</td>
<td>Qualitative model</td>
<td>Persuasion</td>
</tr>
<tr>
<td>Characterization</td>
<td>Empirical model</td>
<td>Implementation</td>
</tr>
<tr>
<td>Method/Means</td>
<td>Analytic model</td>
<td></td>
</tr>
<tr>
<td>Generalization</td>
<td>Notation</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Eval/discriminate</td>
<td>Tool</td>
<td>Analysis</td>
</tr>
<tr>
<td>Product</td>
<td>Specific solution</td>
<td>Experience</td>
</tr>
<tr>
<td></td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

A Common Plan

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility</td>
<td>Qualitative model</td>
<td>Persuasion</td>
</tr>
<tr>
<td>Characterization</td>
<td>Empirical model</td>
<td>Implementation</td>
</tr>
<tr>
<td>Can X be better?</td>
<td>Analytic model</td>
<td>Compare Y to X</td>
</tr>
<tr>
<td>Generalization</td>
<td>Notation</td>
<td></td>
</tr>
<tr>
<td>Eval/discriminate</td>
<td>Tool</td>
<td>Analysis</td>
</tr>
<tr>
<td>Product</td>
<td>Specific solution</td>
<td>Experience</td>
</tr>
<tr>
<td></td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

A Common, but Bad, Plan

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility</td>
<td>Qualitative model</td>
<td>“Look, it works!”</td>
</tr>
<tr>
<td>Characterization</td>
<td>Empirical model</td>
<td>Implementation</td>
</tr>
<tr>
<td>Can X be better?</td>
<td>Analytic model</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Generalization</td>
<td>Notation</td>
<td>Analysis</td>
</tr>
<tr>
<td>Eval/discriminate</td>
<td>Tool</td>
<td>Experience</td>
</tr>
<tr>
<td>Product</td>
<td>Specific solution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

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 solution strategy designs. The model has been tested by comparing analyses / predictions with empirically measured values of properties.

Key: EM provides a new or better way to look at problems

Question
- Characterization: What, exactly do we mean by X?
- Generalization: Is X always true of Y?

Result
- Models, preferably analytic or empirical, but at least precise

Validation
- Empirical analysis, controlled experiment
The Coming-of-Age of Software Architecture Research

Building Blocks for Research

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility</td>
<td>Qualitative model</td>
<td>Persuasion</td>
</tr>
<tr>
<td>Characterization</td>
<td>Empirical model</td>
<td>Implementation</td>
</tr>
<tr>
<td>Method/Means</td>
<td>Notation</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Generalization</td>
<td>Technique</td>
<td>Analysis</td>
</tr>
<tr>
<td>Eval/discriminate</td>
<td>Specific solution</td>
<td>Experience</td>
</tr>
<tr>
<td>Product</td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

Two Other Good Plans

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can X be done?</td>
<td>Qualitative model</td>
<td>&quot;Look, it works!&quot;</td>
</tr>
<tr>
<td>Characterization</td>
<td>Empirical model</td>
<td>Implementation</td>
</tr>
<tr>
<td>Method/Means</td>
<td>Notation</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Is X always true?</td>
<td>Technique</td>
<td>Check proof</td>
</tr>
<tr>
<td>Eval/discriminate</td>
<td>Build Y that X'es</td>
<td>Experience</td>
</tr>
<tr>
<td>Product</td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

Sometimes a breakthrough (but sometimes nonsense)

<table>
<thead>
<tr>
<th>Question</th>
<th>Strategy/Result</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility</td>
<td>Propose new approach</td>
<td>Persuasion</td>
</tr>
<tr>
<td>Change assumptions</td>
<td>Empirical model</td>
<td>Implementation</td>
</tr>
<tr>
<td>Method/Means</td>
<td>Notation</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Generalization</td>
<td>Technique</td>
<td>Analysis</td>
</tr>
<tr>
<td>Eval/discriminate</td>
<td>Specific solution</td>
<td>Experience</td>
</tr>
<tr>
<td>Product</td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

Pro Forma Abstracts and Research Strategies

Locating the pro forma abstracts in research strategy space

Mary Shaw 11 10/29/01
The Coming-of-Age of Software Architecture Research

Executive Summary

- Technical ideas in software engineering evolve in predictable ways
- This evolution pattern sets expectations for software architecture research

Lessons for developers: what to expect of software architecture research, now and in the future

Lessons for researchers: useful research paradigms and validation techniques

I offer these as observations, in Brooks’ sense

What Next?

- Software architecture
  > Don’t just continue in the style of the 90’s
  > Make existing results more robust, better evaluated
  > Address new problems: highly distributed systems, dependability, humane systems
  > Help transition to practice

- Software engineering
  > Be more explicit about research paradigms, when and why they work
  > Be more careful about validation, especially real-world significance