Research Methods in Software and Systems Engineering

Aldeida Aleti and Adel N. Toosi
Software Engineering Research process

- Research question
- Research results
- Research validation
# Types of research questions

<table>
<thead>
<tr>
<th>Types</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FEASIBILITY</strong></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>
<tr>
<td><strong>CHARACTERIZATION</strong></td>
<td>What are the characteristics of X?</td>
</tr>
<tr>
<td></td>
<td>What exactly do we mean by X?</td>
</tr>
<tr>
<td></td>
<td>What are the varieties of X, and how are they related?</td>
</tr>
<tr>
<td><strong>METHOD/MEANS</strong></td>
<td>How can we do X?</td>
</tr>
<tr>
<td></td>
<td>What is a better way to do X?</td>
</tr>
<tr>
<td></td>
<td>How can we automate doing X?</td>
</tr>
<tr>
<td><strong>GENERALIZATION</strong></td>
<td>Is X always true of Y?</td>
</tr>
<tr>
<td></td>
<td>Given X, what will Y be?</td>
</tr>
<tr>
<td><strong>DISCRIMINATION</strong></td>
<td>How do I decide whether X or Y?</td>
</tr>
<tr>
<td>Category</td>
<td>Research Question</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FEASIBILITY</td>
<td>Is it possible to automatically generate code from an architectural specification?</td>
</tr>
<tr>
<td>CHARACTERIZATION</td>
<td>What are the important concepts for modeling software architectures?</td>
</tr>
<tr>
<td>METHOD/MEANS</td>
<td>How can we exploit domain knowledge to improve software development?</td>
</tr>
<tr>
<td>GENERALIZATION</td>
<td>What patterns capture and explain a significant set of architectural constructs?</td>
</tr>
<tr>
<td>DISCRIMINATION</td>
<td>How can a designer make tradeoff choices among architectural alternatives?</td>
</tr>
</tbody>
</table>
## Research Results

### Qualitative & Descriptive Models
- Report interesting observations
- Generalize from (real-life) examples
- Structure a problem area; ask good questions

### Techniques
- Invent new ways to do some tasks, including implementation techniques
- Develop ways to select from alternatives

### System
- Embody result in a system, using the system both for insight and as carrier of results

### Empirical Models
- Develop empirical predictive models from observed data

### Analytical Models
- Develop structural models that permit formal analysis
Example: SA research results

QUALITATIVE & DESCRIPTIVE MODELS
- Early architectural models
- Architectural patterns
- Domain-specific software architectures

TECHNIQUES SYSTEM
- UML to support object-oriented design
- Architectural languages

EMPIRICAL MODELS
- Communication metrics as indicator of impact on project complexity

ANALYTICAL MODELS
- Formal specification of higher-level architecture for simulation
Research Validation

PERSUASION
I thought hard about this, and I believe…

IMPLEMENTATION
Here is a prototype of a system that…

EVALUATION
Given these criteria, the object rates as…

ANALYSIS
Given the facts, here are consequences…
   - Rigorous derivation and proof
   - Data on use in controlled situation

EXPERIENCE
Report on use in practice
   - Narrative
   - Comparison of systems in actual use
   - Data, usually statistical, on practice

Formal model
Empirical model

Qualitative model
Decision criteria
Empirical model
Example: Automated testing (AT) research validation

<table>
<thead>
<tr>
<th>PERSUASION</th>
<th>Early automated testing, random testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPLEMENTATION</td>
<td>Implementation of AT on an industrial system</td>
</tr>
<tr>
<td>EVALUATION</td>
<td>Comparison of search-based software testing with random testing</td>
</tr>
<tr>
<td>ANALYSIS</td>
<td>Algorithm selection for Automated Software Testing</td>
</tr>
<tr>
<td>EXPERIENCE</td>
<td>User studies with industry experts on the usefulness of automated software testing</td>
</tr>
</tbody>
</table>

- Formal model
- Empirical model
- Qualitative model
- Decision criteria
- Empirical model
Building blocks for SE research

Question
- Feasibility
- Characterisation
- Methods/means
- Generalisation
- Selection

Results
- Qualitative model
- Technique
- System
- Empirical model
- Analytical model

Validation
- Persuasion
- Implementation
- Evaluation
- Analysis
- Experience
### A common good plan

<table>
<thead>
<tr>
<th>Question</th>
<th>Results</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility</td>
<td>Qualitative model</td>
<td>Persuasion</td>
</tr>
<tr>
<td>Characterisation</td>
<td>Technique</td>
<td>Implementation</td>
</tr>
<tr>
<td>Can X be done better?</td>
<td>Build Y</td>
<td>Measure Y, compare to X</td>
</tr>
<tr>
<td>Generalisation</td>
<td>Empirical model</td>
<td>Analysis</td>
</tr>
<tr>
<td>Selection</td>
<td>Analytical model</td>
<td>Experience</td>
</tr>
</tbody>
</table>
A common, but **bad** plan

<table>
<thead>
<tr>
<th>Question</th>
<th>Results</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility</td>
<td>Qualitative model</td>
<td>Persuasion</td>
</tr>
<tr>
<td>Characterisation</td>
<td>Technique</td>
<td>Implementation</td>
</tr>
<tr>
<td>Methods/means</td>
<td>System</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Generalisation</td>
<td>Empirical model</td>
<td>Analysis</td>
</tr>
<tr>
<td>Selection</td>
<td>Analytical model</td>
<td>Experience</td>
</tr>
</tbody>
</table>
Two other good plans

Question

Can X be done at all?
Characterisation
Methods/means
Is X always true of Y?
Selection

Results

Qualitative model
Technique
Build Y that does X
Empirical model
Formally model Y, prove X

Validation

Look it works
Implementation
Evaluation
Check proof
Experience
What do program committees look for?

Interesting, novel, exciting results that significantly enhance our ability to develop and maintain software
to know the quality of the software we develop
to recognize general principles about software
or to analyze properties of software

You should explain your result in such a way that someone else could use your ideas
What is new here?

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awful</td>
<td>I completely and generally solved ... (unless you actually did)</td>
</tr>
<tr>
<td>Bad</td>
<td>I worked on (studied, investigated, sought, explored) skedaddling</td>
</tr>
<tr>
<td>Poor</td>
<td>I worked on improving (contributed to, participated in, helped with) skedaddling</td>
</tr>
<tr>
<td>Good</td>
<td>I showed the feasibility of predicting software defects with machine learning. I significantly improved the accuracy of detecting software defects (or proved, demonstrated, created, established, found, developed)</td>
</tr>
<tr>
<td>Better</td>
<td>I automated the generation of software tests. With a novel application of search techniques, I achieved a 10% increase in coverage and a 15% improvement in detecting bugs over the standard method.</td>
</tr>
</tbody>
</table>
What has been done before? How is your work different or better?

<table>
<thead>
<tr>
<th>Awful</th>
<th>The skedaddling problem has attracted much attention [2, 3, 4, 5, 7].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>Trer’s jumping approach to skedaddling [4] achieved 60% coverage [8]. Amil [6] achieved 80% by skipping, but only for light-free cases [34].</td>
</tr>
<tr>
<td>Better</td>
<td>Trer’s jumping approach to Skedaddling [4] achieved 60% coverage [8]. Amil [6] achieved 80% by skipping, but only for light-free cases [34]. We modified the jumping approach to use the agility representation of skipping and achieved 90% coverage while relaxing restrictions so that only running is prohibited.</td>
</tr>
</tbody>
</table>
Distributed System definitions - many and varying:

- A system in which hardware or software components located at networked computers communicate and coordinate their actions only by passing message [Coulouris]
- A collection of independent computers that appears to its users as a single coherent system [Tanenbaum]

Computer Networks vs Distributed Systems:

A Computer Network: Is a collection of spatially separated, interconnected computers that exchange messages based on specific protocols. Computers are addressed by IP addresses.

- A Distributed System: Multiple computers on the network working together as a system. The spatial separation of computers and communication aspects are hidden from users.
Distributed system challenges

- Heterogeneity
- Openness
- Security
- Scalability
- Failure Handling
- Concurrency
- Transparency
- Quality of Service
Programming Models

- Thread
- Task
- Map-Reduce
- Message Passing
- Data Flow
- Workflow
- Parameter Sweep
- Bag of Tasks
Platforms

Cluster
Grid
Cloud
Peer-to-Peer Systems
Supercomputers
Mobile Computing
Sensor Networks
Internet of things
Edge and Fog Computing
Content delivery networks (CDN)
Software Defined Networks (SDN)
...
Objectives

Cost-related
- Energy-related
- Monetary cost
- Utility
- Welfare

Time-related
- Response time
- Throughput
- Availability
- Delay

Others
- Accuracy
- Ease of use
- Utilization
- Security
- Privacy
- Reliability
- Robustness
- Interoperability
Constraints

- Budget
- Deadline
- Accuracy
- Capacity
- Regulation
Scheduling: One of the challenging areas of research in DS

- Planning scheme
  - Dynamic (online)
  - Static (deterministic, offline)

- Decision Making
  - Local
  - Global

- Optimality
  - Optimal
  - Sub-optimal

- Goal
  - Load Balancing
  - Admission Control
  - Mapping
  - Resource Provisioning
    - Selection
    - Centralized
    - Decentralized
    - Hierarchical
    - Peer-to-Peer
    - hybrid

- Architecture
  - Approximate
  - Heuristics
  - Meta-Heuristics
Evaluations Methods

- Analytical Modeling
- Measurement
- Empirical Analysis
- Emulation
- Simulation
Common Mistakes in Performance Evaluation

No Goals/Biased Goal
Unsystematic Approach
Analysis without understanding the problem
Incorrect Performance Metrics
Unrepresentative workload
Wrong Evaluation Technique
Overlooking Important Parameters
Ignoring significant factors
  - Sensitivity analysis
Inappropriate Experimental Design
  - Full factorial design
Inappropriate level of detail
No analysis/Erroneous Analysis
Ignoring Errors in Input
Improper Treatment of Outliers
Too complex Analysis
Assuming No change in the Future
Ignoring Social Aspects
  – Weak presentation leads to rejection of the high-quality analyses
Ignoring Variability
  – If the variability is high the mean alone is misleading.
Improper Presentation of Results
Ignoring or Omitting Assumptions and limitations

Please find more complete slides here: [http://adelnadjarantoosi.info/ppt/common.pptx](http://adelnadjarantoosi.info/ppt/common.pptx)
How to write a research paper in DS domain

**Problem**
- Short Background (If necessary)
- Scope
  - Application model, e.g., Map-reduce
  - Platform, e.g., Cluster
- Objective, e.g., Cost and Energy Consumption
- Constraints, e.g., Capacity and Available Renewable Energy

**Methodology**
- E.g., Online scheduling using meta-heuristics

**Evaluation Method**
- Analytical proofs, Simulation, Emulation, Real Implementation

**Results/Findings**

**Conclusion/Implications**
Problem:

- In this paper, we present BlinkDB, a massively parallel, approximate query engine for running interactive SQL queries on large volumes of data. BlinkDB allows users to trade-off query accuracy for response time, enabling interactive queries over massive data by running queries on data samples and presenting results annotated with meaningful error bars.

Methodology:

- To achieve this, BlinkDB uses two key ideas: 1) an adaptive optimization framework that builds and maintains a set of multi-dimensional stratified samples from original data over time, and 2) a dynamic sample selection strategy that selects an appropriately sized sample based on a query’s accuracy or response time requirements.

Evaluation:

- We evaluate BlinkDB against the well-known TPC-H benchmarks and a real-world analytic workload derived from Conviva Inc., a company that manages video distribution over the Internet. Our experiments on a node cluster show

Results and Conclusions:

- that BlinkDB can answer queries on upto 17TBs of data in less than seconds (over 200x faster than Hive), with in an error of 2-10%.

Summary

Software Engineering Research process
  – Research Question, Research Results, and Research Validation

Three Evaluation Technique:
  – Measurement
  – Simulation
  – Analytical Modeling

Common mistakes in performance evaluation
  – Sensitivity analysis
  – Factorial design