Empirical Models: Towards a Science of Software Engineering

Dieter Rombach

TU Kaiserslautern & Fraunhofer IESE
IT Campus Kaiserslautern

- University Departments
  - Computer Science
  - Mathematics
  - Electrical Engineering
  - Mechanical Engineering

- Affiliated Research Institutes
  - MPI for Software systems
  - FhI for Experimental SW Engineering (IESE)
  - FhI for Industrial Mathematics (ITWM)
  - German Research Center for AI (DFKI)

app. 800 Scientists in the area of Software, Software systems, Software Technology & Software Engineering
Fraunhofer IESE

• Applied Research & TT
  • 200+ employees (growing)
  • € 13 M Budget
  • High % of external income (~75%)

• International Presence
  • USA
  • Brazil
  • Australia

• Innovative Cooperation model
  • “Research Labs”

• High international reputation in
  • “Systems & Software Engineering”
    • No. 1 in Europe (JSS, 2006)
• Empirical Software Engineering
  • Top 3 worldwide (ACM)
AGENDA

• Motivation

• Traditional Engineering (Physical Laws)
• Software Engineering (?Cognitive Laws?)

• Existing Observations, Laws & Theories

• TODOs (incl. a joint Community Effort)
AGENDA

• Motivation

• Traditional Engineering (Physical Laws)
• Software Engineering (?Cognitive Laws?)

• Existing Observations, Laws & Theories

• TODOs (Community Effort)
Motivation

- Engineering requires (among others)
  - Understanding of technologies & prediction of their effects
  - Scaling based on regularities
- Traditional engineering disciplines do this based on physical laws
- Software engineering (partially) uses empirical models
  - Could they eventually define the (cognitive) laws in our field?
- Need to expand empirical studies to all relevant process areas
  - Replication
  - Aggregation
  until trends become qualitatively stable (for a company, sector of industry, entire field) \(\rightarrow\) Laws?

We need more than notations, methods & tools!
We need models regarding their impact under specific conditions!
AGENDA

• Motivation

• Traditional Engineering (Physical Laws)
• Software Engineering (?Cognitive Laws?)

• Existing Observations, Laws & Theories

• TODOs (incl. a joint Community Effort)
Traditional Engineering (How do they scale engineering?)

- Physical laws limit geometrical options!
- Repeated usage of components results in regular structures & allows scaling of engineering

Application of Divide & Conquer is visible! The consequences of violations can be experienced instantly!
Traditional Engineering (How do they structure disciplines?)

- Engineering practices
- Physical laws
- (Continuous) mathematics
AGENDA

- Motivation
- Traditional Engineering (Physical Laws)
- Software Engineering (?Cognitive Laws?)
- Existing Observations, Laws & Theories
- TODOs (incl. a joint Community Effort)
Application of Divide & Conquer is not visible!
The consequences of violations can be experienced with a time delay only!
Software Engineering (How should we structure our discipline?)

- Systems Engineering
  - Mechanical Engineering
  - Physics
  - Mathematics
  - Software Engineering
  - Computer Science
  - Mathematics
  - Economics
  - Psychology

Empirical Studies and Modeling is part of science & engineering!
Empirical Methods (Do they exist?)

- A software science must involve
  - modeling of software product & process artifacts
  - validating empirically hypotheses regarding their characteristics & behavior in testable/challengeable form

- Empirical foundation in software engineering includes methods for
  - relating goals to measurements (GQM)
  - piggy-bagging empirical studies on real projects (QIP)
  - organizing empirical observations for reuse (EF)
  - specific activities such as experimental design, data analysis
    - importance of combining quantitative & qualitative analysis

No excuses accepted!

ISERN Community defines Standards!
Empirical Models

• Testing technique T identifies
  - 85% of all defects (+/-5%) iff (Code complexity <= C0 & Exp = high)
  - 65% of all defects (+/-10%) iff (Code complexity > C0 & Exp = high)
  - 50% of all defects (+/-10%) iff (Code complexity > C0 & Exp < high)
  - Etc.

• 4 types of models exist
  - Product – product (e.g., design complexity → final product Q)
  - Product – process (e.g., design complexity → test effort)
  - Process – process (e.g., effort distribution across phases)
  - Process – product (e.g., “process – product models”)
Empirical Process-Product Models?

- For non-deterministic human processes
  - Function f can only be determined empirically
  - Variance of function f depends on
    - sufficient # data points
    - inclusion of relevant C
    - stability of non-included C

- important use of qualitative analysis: assure that there are no non-identified context factors

- Variance may increase when
  - Repeated in new environment
  - Reason: hidden context factor (i.e., factor which was constant in one or both environments) becomes influential
  - Consequence: New empirical cycle OR keep two different domains (i.e. identical recommendation exists for product lines!)

\[ Q/P/T (A1) = f (A2, C, HC) \]

- Q/P/T: any quality/productivity/time aspect of interest
- A1/A2: Software artifact / Process
- C: context factors such as experience, lifecycle model
- HC: hidden context factors that only show up at certain levels of generalization
- f: empirical relationship (with variation)

Dieter Rombach

Technische Universität Kaiserslautern

Fraunhofer IESE
AGENDA

• Motivation

• Traditional Engineering (Physical Laws)
• Software Engineering (?Cognitive Laws?)

• Existing Observations, Laws & Theories

• TODOs (incl. a joint Community Effort)
Observations, Laws & Theories

• Observations
  - One study in one context
  - No qualitative/quantitative stability guaranteed
  - E.g., Systematic requirements inspection reduces rework in project p

• Laws
  - A representative set of studies for some context
  - Qualitative stability, quantitative variability
  - E.g., Systematic requirements inspections always reduce rework in projects of some type (different amounts for different contexts)

• Theories
  - A representative set of studies for some context (all relevant independent variable captured)
  - Qualitative stability, quantitative stability (small variability)
  - E.g., Use of a systematic requirements inspection in a specific context reduces rework by X% (with +/- Y % variation)

What exists are mostly observations! We need more „laws“ & theories!
Existing Body of Knowledge

• There exists more knowledge than we typically recognize
  - mostly in terms of context-specific empirical observations
  - rarely in terms of generalized “laws”

• Fraunhofer Institute for Experimental Software Engineering (IESE) maintains an Experience Base
Executive Director
Prof. Dr. D. Rombach

Scientific Director
Prof. Dr.-Ing. P. Liggesmeyer

Deputy Director
Prof. Dr. F. Bomarius

Business Areas

Product Sectors, e.g.
R. Kalmar

- Automotive & Transportation Systems
- Automation & Plant Engineering
- Medical Systems

Service Sectors, e.g.
M. Ochs

- Information Systems
- eGovernment
- Health Care

Divisions

Embedded Systems (ES) Dr. Mario Trapp

- Embedded Systems Development (ESD)
  Dr. M. Becker
- Embedded Systems Quality Assurance (ESQ)
  Dr. R. Eschbach

Process Management (PM) Dr. J. Heidrich

- Measurement, Prediction & Empiricism (MPE)
  Dr. J. Heidrich
- Process Compliance & Improvement (PCI)
  R. van Lengen

Information Systems (IS) Dr. J. Dörr

- Information Systems Development (ISD)
  Dr. Marcus Trapp
- Information Systems Quality Assurance (ISQ)
  M. Eisenbarth
## ES: Overview of Empirical Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcES</td>
<td>■ 35% reduction of implementation and testing effort at same quality level</td>
<td>■ ICSR 2008</td>
</tr>
<tr>
<td>AcES/RATE, SAVE</td>
<td>■ 60% less time needed for architectural analysis if architectures are visualized appropriately</td>
<td>■ EMSE 2008</td>
</tr>
<tr>
<td>SAVE-Life</td>
<td>■ 60% fewer architecture violations if developers are getting live feedback on their architectural compliance</td>
<td>■ PhD Knodel 2010</td>
</tr>
<tr>
<td>AcES</td>
<td>■ Architecture-compliant implementation reduces development effort by 50%</td>
<td>■ PhD Knodel 2010</td>
</tr>
</tbody>
</table>
### ES: Overview of Empirical Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>PuLSE</td>
<td>- Strategic reuse program increases reuse level by 50%</td>
<td>ArQuE 02.09</td>
</tr>
<tr>
<td></td>
<td>- Architectural divergences decreased from 17% to 1%</td>
<td>CSMR 2008</td>
</tr>
<tr>
<td>PuLSE</td>
<td>- With SPL approach, productivity has tripled</td>
<td>Ricoh 2010</td>
</tr>
<tr>
<td></td>
<td>- # of quality problems has been reduced to 1/5</td>
<td></td>
</tr>
<tr>
<td>PuLSE-EM</td>
<td>- 27% less effort on average for configuration management in a product line</td>
<td>IWPSE-EVOL 2009</td>
</tr>
</tbody>
</table>
## MPI: Overview of Empirical Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CoBRA Cost Estimation</strong></td>
<td>■ Best-in-class estimation error between 5 and 20% initially</td>
<td>■ ICSE 1998 &amp; 2006</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ METRICS 2002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ PhD Trendowicz 2008</td>
</tr>
<tr>
<td><strong>HyDEEP Defect Prediction</strong></td>
<td>■ Defect prediction accuracy ~30% (normally 40-150%)</td>
<td>■ ICSE 2010</td>
</tr>
<tr>
<td></td>
<td>■ Effort for estimation model &lt; 1.5 PD / expert</td>
<td>■ Journal ESE 2010</td>
</tr>
<tr>
<td><strong>Specula PM Dashboards</strong></td>
<td>■ Earlier detection of plan deviations: 25-83%</td>
<td>■ ESEM 2007 &amp; 2008</td>
</tr>
<tr>
<td></td>
<td>■ More complete detection of plan deviations: 17-63% more</td>
<td>■ PROFES 2008</td>
</tr>
<tr>
<td></td>
<td>■ Overall effort (including training and data collection): 10%</td>
<td>■ PhD Heidrich 2008</td>
</tr>
</tbody>
</table>
## MPI: Overview of Empirical Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
<th>Publications</th>
</tr>
</thead>
</table>
| Defect Flow Models            | ■ More reliable defect classification: Kappa 0.65-079 (substantial)  
■ Detect the defects more locally, e.g. 72% to 100% of analysis defects are detected in the origination phase, etc.  
■ Substantial rework reductions up to 90%                                                                                           | ■ METRICS 2005  
■ METRIKON 2007  
■ EuroMICRO 2009                                                              |
| Aggregation of Empirical Studies | ■ Current (unsystematic) summaries often lead to wrong conclusions  
■ PBR: 50% of assumptions have proven to be wrong; 50% could be phrased more accurately  
■ Complexity models: 25% of assumptions have proven to be wrong                                                                    | ■ ESEM 2009  
■ METRIKON 2010                                                            |
# IS: Overview of Empirical Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfy (IESE NFR Method part)</td>
<td>- Method leads to almost 100% measurable NFRs</td>
<td>PhD Dörr, 2010</td>
</tr>
<tr>
<td></td>
<td>- Method resulted in ROI values between 2 and 17!</td>
<td>RE’ 05</td>
</tr>
<tr>
<td></td>
<td>- Method resulted in more complete NFRs (up to 622% more relevant NFRs identified)</td>
<td>…</td>
</tr>
<tr>
<td>GoMobile Product (new)</td>
<td>- Half-day usability walkthroughs of iPhone business apps (performed by two usability experts) can bring up more than 40 usability issues, with more than 25% of them being critical for the acceptance of the app.</td>
<td>Not yet published</td>
</tr>
</tbody>
</table>
## IS: Overview of Empirical Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
<th>Publications</th>
</tr>
</thead>
</table>
| UXelerate (FUN/FUN-NI) | - Subjects who used UX Patterns worked on 23% more tasks without making mistakes than subjects who did not use UX Patterns  
- With UX Patterns, 77% of the employees collected their time in a timely manner compared to 41% without UX Patterns (improvement of 36%).  
- Call Center customers who were treated by agents with UX Patterns in the software were rating customer satisfaction 8% higher than customers who were served by agents without UX Patterns. | - Usability Professionals 2008 & 2009  
- CHI 2010 |
| ACES/Save/PuLSE    | - See ESD slides                                                      |                                   |
Existing Body of Knowledge

• There exists more knowledge than we typically recognize
  - mostly in terms of context-specific empirical observations
  - rarely in terms of generalized “laws”

• Fraunhofer Institute for Experimental Software Engineering (IESE) maintains an Experience Base

• There exists a large body of empirical facts in our community (e.g., Endres/Rombach, Addison, 2003)
  - inspections
  - design principles
Handbook capturing existing body of knowledge

Students can learn about existing body of knowledge

Practitioners can avoid negligence of due diligence
Requirements

- Requirements deficiencies are the prime source of project failures (L1)
  - Source: Robert Glass [Glas98] et al
  - Most defects (> 50%) stem from requirements
  - Requirements defects (if not removed quickly) trigger follow-up defects in later activities

Possible solutions:
- early inspections
- formal specs & validation early on
- other forms of prototyping & validation early on
- reuse of requirements docs from similar projects
- etc.

- Defects are most frequent during requirements and design activities and are more expensive the later they are removed (L2)
  - Source: Barry Boehm [Boeh 75] et al
  - >80% of defects are caused up-stream (req, design)
  - Removal delay is expensive (e.g., factor 10 per phase delay)

It would be nice to have effect measures for these possible solutions!
Inspection

- Inspections significantly increase productivity, quality and project stability (L17)
  
  - Source: Mike Fagan [Faga76, Faga86]
  
  - Early defect detection increases quality (no follow-up defects, testing of clean code at the end → quality certification)
  
  - Early defect detection increases productivity (less rework, lower cost per defect)
  
  - Early defect detection increases project stability (better plannable due to fewer rework exceptions)

- Effectiveness of inspections is rather independent of its organizational form (i.e. process), but depends on reading technique (L18)
  
  - Source: Ross Jeffery [Jeff90]

- Perspective-based inspections are highly effective and efficient (L19)
  
  - Source: Victor Basili [Bas96c, Shull00]
  
  - Best suited for non-formal documents
Existing Body of Knowledge

• There exists more knowledge than we typically recognize
  - mostly in terms of context-specific empirical observations
  - rarely in terms of generalized “laws”

• Fraunhofer Institute for Experimental Software Engineering (IESE) maintains an Experience Base

• There exists a large body of empirical facts in our community (e.g., Endres/Rombach, Addison, 2003)
  - inspections
  - design principles

• More studies need to be done (see ISERN)
  - repeat (with variation)
  - generalize

and results need to be organized and maintained (Handbook) !!!
AGENDA

• Motivation

• Traditional Engineering (Physical Laws)

• Software Engineering (?Cognitive Laws?)

• Existing Observations, Laws & Theories

• TODOs (incl. a joint Community Effort)
TODOs (Teaching) (1/5)

- Learning is based on
  - reading
  - doing
  - experiencing

- Human-based engineering activities depend heavily on the latter approaches to learning

- Teaching must reflect this
  - first analysis, then construction
  - perform “self-experience” experiments

- At Technical University of KL/CS department
  - 1st semester: NO programming (just reading & changing)
  - SE experiments (GSE: final UG class)
    - #1: Unit inspection more efficient than testing
    - #2: Traceable design documentation reduces effort & risk of change
    - #3. Informal (req) documents can be inspected efficiently (> 90%)
  - practical semester-long team projects with “data collection & process improvements”
TODOs (Research) (2/5)

• SE Research results require “some form of evidence”
  • notations, techniques, methods & tools w/o evidence are not accepted (e.g., PhD theses)
  • collaboration with SE/CS experts

• Current research focus (SE techniques)
  • notations like Java, UML, …
  • architecture (specifically “product line architectures”)
  • processes such as inspection & testing, design, requirements

FROM
  • quality (reliability, security, safety) perspective
  • efficiency (cost, time) & effectiveness (“dependable SE”)

• Current research focus (ESE methods)
  • Theory for “Evidence”
TODOs (PhD Research) (3/5)

- Problem Stmt (↔ SoP) with Improvement Hyp.
- Solution Stmt (↔ SoR) with Improvement Hyp.
- Research
- Technical Solution

- Emp. Testing of Problem hypothesis
- Emp. Testing of solution hypothesis

? -> ?
TODOs (Tech Transfer) (4/5)

- Apply “Evidence” as transfer vehicle to create sustained improvements
- Use empirical studies to
  - evaluate major process-product relations prior to offering to industry (e.g., in vitro controlled experiments)
  - method prototyping: Evaluate new methods together with industry experts in order to provide ROI potential insight for decision makers (e.g., Ricoh, Bosch, John Deere)
  - motivate candidate pilot project (developers & managers) with semi-controlled training experiment
  - evaluate pilot project (in vivo case studies) in order to adapt & motivate
  - continuously evaluate wide-spread use in order to motivate & optimize

Without empirical evidence, no human-based process is lived!
This has contributed to the growing gap between research & practice!
Todos (Community Effort) (5/5)

- Fraunhofer IESE (Rombach/Heidrich) wants to support the establishment of Community Experience Bases
  - **Research** Organizations (like IESE’s Experience Base)
    - Focus on experiences wrt. new notations, methods & processes
  - **Development** Organizations (with companies, e.g., SPES 2020)
    - Focus on key notations, methods & processes for development
  - **Community at large, e.g.**
    - Springer may host Experience Base
    - IESE is responsible for content management
    - Individual areas will be chaired by community reps

Is the time right to shift into the next gear?
Who wants to participate actively?
What are incentives for contribution?

Lets discuss this at ISERN‘2012 in Sweden !!!
Conclusion

• Why do we build systems that
  • have critical defects?
    ➔ Understanding the true characteristics of software
      (e.g., does not tolerate mistakes, has „design“
      defects only)
    ➔ Engineering-style techniques for spec & verification?
      (e.g., divide & conquer, laws & patterns ➔
      scaleability)
  • are hard to maintain?
    ➔ Empirical Laws / Models that explain / predict
      (e.g., maintainability) early
    ➔ Development for maintainability

This is Fraunhofer IESE‘s Mission based on strategic cooperations with industry !!!
Contact Data

Prof. Dr. Dr. h.c. Dieter Rombach

Software Engineering Chair, CS Dept, TU KL, Germany
Executive Director, Fraunhofer IESE, KL, Germany

Fraunhofer-Institut für Experimentelles Software Engineering IESE
Fraunhofer-Platz 1
67663 Kaiserslautern
Germany

Tel/Fax: +49 631 6800 1000 / 1099
Email: Dieter.Rombach@iese.fraunhofer.de
Website: www.iese.fraunhofer.de