Keynote – CESI 2015

ICSE 2015 Workshop, Florence, Italy, 18 May 2015

# The Maturation of Empirical Studies

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

TU Kaiserslautern & Fraunhofer IESE
Kaiserslautern, Germany

dieter.rombach@iese.fraunhofer.de



Technische Universität Kaiserslautern

Fraunhofer IESE

Florence
18 May 2015

# Dieter Rombach

- 1978: MS in Mathematics & Computer Science (Karlsruhe)
- 1984: PhD in Computer Science (Kaiserslautern)
- **1984-1991: Prof., CS Dept., University of Maryland, & Project manager, NASA GSFC (SEL)**
- **Since 1992: SE Chair, CS Department, University of Kaiserslautern**
- **1996-2014: Founding & Executive Director, Fraunhofer IESE**
- **Since 2015: Founding & Business Development Director, Fraunhofer IESE**
- Editor of many international journals (incl. IEEE TSE, ACM TOSEM, ESE)
- General & Program Chair of many intern. Conferences (incl. IEEE/ACM ICSE)
- NSF Presidential Investigator Award, ACM & IEEE Fellow, Federal Cross of Ribbon of Germany, Honorary PhD (Univ. of Oulu, Finland)
- Many advisory boards (industry, academia et al)

**Professional Life between Basic & Industrial Reserach**

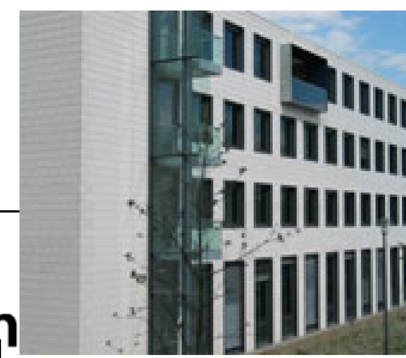
# IT/SoftwareCampus Kaiserslautern

- University Departments
  - Computer Science (3 chairs in SE)
  - 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 - 1000 Scientists in the area of Software, Software systems, Software Technology & Software Engineering**

## Fraunhofer IESE

- Applied Research & TT in Software & Systems Engineering
  - 230+ employees (growing)
  - € 14 M Budget
- High % of external income (~75%)
- International Presence
  - USA
  - Brazil
  - Japan, China, India
- Innovative Cooperation model
  - "Research & Innovation Labs"
  - Rapid Innovation (DevOps)
- Strategic cooperations with companies in all sectors of industry (e.g., automotive, aerospace, health, energy, ….)



IESE

**Top-ranked Applied Research Institute in Software & Systems Engineering**

# The Maturation of Empirical Studies

- **Motivation**
- **Basic Framework**
  - **Empirical Evidence**
  - **Empirical Software Engineering**
  - **Empirical Methods**
- **Maturation** (expanded version of VRB 2006)
  - **Phase 1: Isolated Studies**
  - **Phase 2: Multiple Studies (domain/environment specific)**
  - **Phase 3: Multiple Studies (across domains/environments)**
  - **Phase 4: Towards Creating Evidence**
- **Today & Future (Towards a Theory of Software Engineering Evidence)**
  - **Existing Body of Knowledge**
  - **Experimental Software Engineering in Kaiserslautern (Fraunhofer IESE) – Practical Examples**



- **Agenda for Research, Tech Transfer & Teaching**
- **Outlook**

# The Maturation of Empirical Studies

- **Motivation**
- Basic Framework
  - Empirical Evidence
  - Empirical Software Engineering
  - Empirical Methods
- Maturation (expanded version of VRB 2006)
  - Phase 1: Isolated Studies
  - Phase 2: Multiple Studies (domain/environment specific)
  - Phase 3: Multiple Studies (across domains/environments)
  - Phase 4: Towards Creating Evidence
- Today & Future (Towards a Theory of Software Engineering Evidence)
  - Existing Body of Knowledge
  - Experimental Software Engineering in Kaiserslautern (Fraunhofer IESE) – Practical Examples
- Agenda for Research, Tech Transfer & Teaching
- Outlook

c

## Motivation (1/2)

- **Engineering challenge**
  - find appropriate process/technique/method/tool **P**
  - to achieve the following goals **Q**
  - in context **C**
- **In order to answer to answer this challenge we require evidence**
  - regarding candidate processes/techniques/methods/tools **Pi**
  - about their effectiveness **F**
  - wrt. goals **Q**
  - in context **C**

**Q ==<var> F (Pi, C)**

–

**e.g., 95% Fault Detection Rate == F (PBR, Allianz AG)**



**Software Engineering must address engineering challenges!**

# Motivation (2/2)

- Physics offers laws for electrical eng.
  - precise
  - not circumventable
- Computer Science & …. offer "laws" for SE
  - empirically precise
  - circumventable (e.g., you may increase the complexity of any system and it still may work!)
    - is this really true?
      - not if one includes maintenance!
    - what defines bounds?
      - E.g., models that capture the negative consequences if you exceed complexity bounds

**Physical laws**

**Cognitive Laws**

**Cognitive Laws require „empirical evidence"!**

# The Maturation of Empirical Studies

- Motivation
- **Basic Framework**
  - **Empirical Evidence**
  - **Empirical Software Engineering**
  - **Empirical Methods**
- Maturation (expanded version of VRB 2006)
  - Phase 1: Isolated Studies
  - Phase 2: Multiple Studies (domain/environment specific)
  - Phase 3: Multiple Studies (across domains/environments)
  - Phase 4: Towards Creating Evidence
- Today & Future (Towards a Theory of Software Engineering Evidence)
  - Existing Body of Knowledge
  - Experimental Software Engineering in Kaiserslautern (Fraunhofer IESE) – Practical Examples



- Agenda for Research, Tech Transfer & Teaching
- Outlook

# Empirical Evidence (1/2)

- Empirical studies aim to capture quantitative evidence regarding (P)
  - product characteristics (definition, behavior)
    - "What is the complexity of a product?"
    - "What is the performance of a system?"
  - process characteristics (definition, behavior)
    - "What is the inherent degree of parallelism?"
    - "How much effort does it take?"
  - process-product relationships
    - "How does design complexity affect test effort?"
- Issues
  - How deterministic are studies?
  - How easy/hard is it to test/challenge results via replication?

Q == F (P,C)

Multiple evidence-based models qre required!

## Empirical Evidence (2/2): Observations – Laws - Theories

**Q == F ( P, C )**

- Observations
  - Mostly based on one or a **small number** of studies
  - There exists a **descriptive** relationship (F) between "goal" and "context"
  - The dependency is "instable"
- Laws
  - Based on a reasonably **large number** of similar experiments or studies
  - There exists a **correlational** relationship (F) between "goal" and "context"
  - The dependency is "**qualitatively**" stable (i.e., same pattern, but high variability)
- Theories
  - Based on a reasonably **large** & (for Context) **representative** number of similar experiments or studies
  - There exists a **causal** relationship (F) between "Goal" and "context"
  - The dependency is "**quantitatively**" stable (i.e., with acceptable variation)
  - The variation in "Goal" can be predicted based on specific values of the "Characteristics"; "characteristics" are the only cause of "goal" variation (cause-effect dependency)

## Observations

**Q == F ( Process, Context )**

- Mostly based on one or a small number of studies
- There exists a descriptive relationship (F) between "goal" and "context"
- No correlation established yet!

**- Repeatability (qualitatively) unclear?**
**- Predictability (quantitatively) unclear?**

- Example: We have found 60% of all requirements defects by means of perspective based requirements reading in project "X"

# The Maturation of Empirical Studies

## Laws

**Q == F ( Process, Characteristics )**

- Based on a reasonably large number of similar experiments or studies
- There exists a correlational relationship (F) between "goal" and "context"
- The dependency is "qualitatively" stable (i.e., same pattern, but high variability)
- No proven cause-effect relationship! The quantitative dependency may depend on other hidden context variables  (e.g., maturity)

**- Repeatability (qualitatively) assumed clear!**
**- Predictability (quantitatively) unclear?**



- Example: Systematic inspections always increase effectiveness/efficiency!

# The Maturation of Empirical Studies

**Theories**

**Goal == F ( Process, Characteristics )**

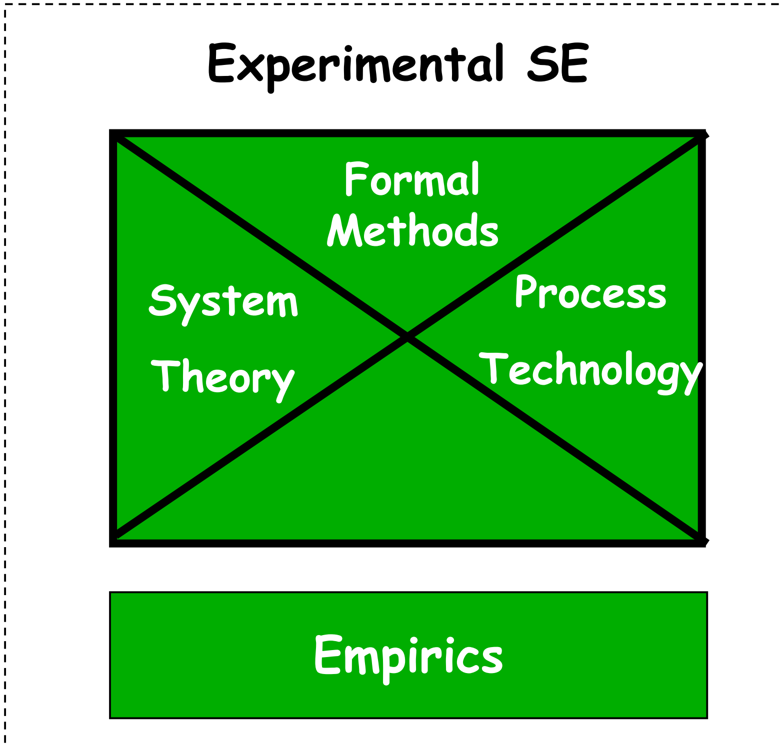
- Based on a reasonably large & (for Context) representative number of similar experiments or studie
- There exists a causal relationship (F) between “Goal” and “context”
- The dependency is “quantitatively” stable (i.e., with acceptable variation)
- The variation in “Goal” can be predicted based on specific values of the “Characteristics”; “characteristics” are the only cause of “goal” variation (cause-effect dependency)
- Realistic for certain contexts (e.g., company); hard to establish in general!

**- Repeatability (qualitatively) assumed clear!**
**- Predictability (quantitatively) assumed clear?**



- Example: Effort for reading preparation depends on human experience (Bosch)

# (Empirical) Software Engineering (1/2)




- Software Engineering comprises
  - (formal) methods (e.g., modeling techniques, description languages)
  - system technology (e.g., architecture, modularization, OO, product lines)
  - process technology (e.g., life-cyle models, processes, management, measurement, organization, planning QS)
  - empirics (e.g., experimentation, experience capture, experience reuse)

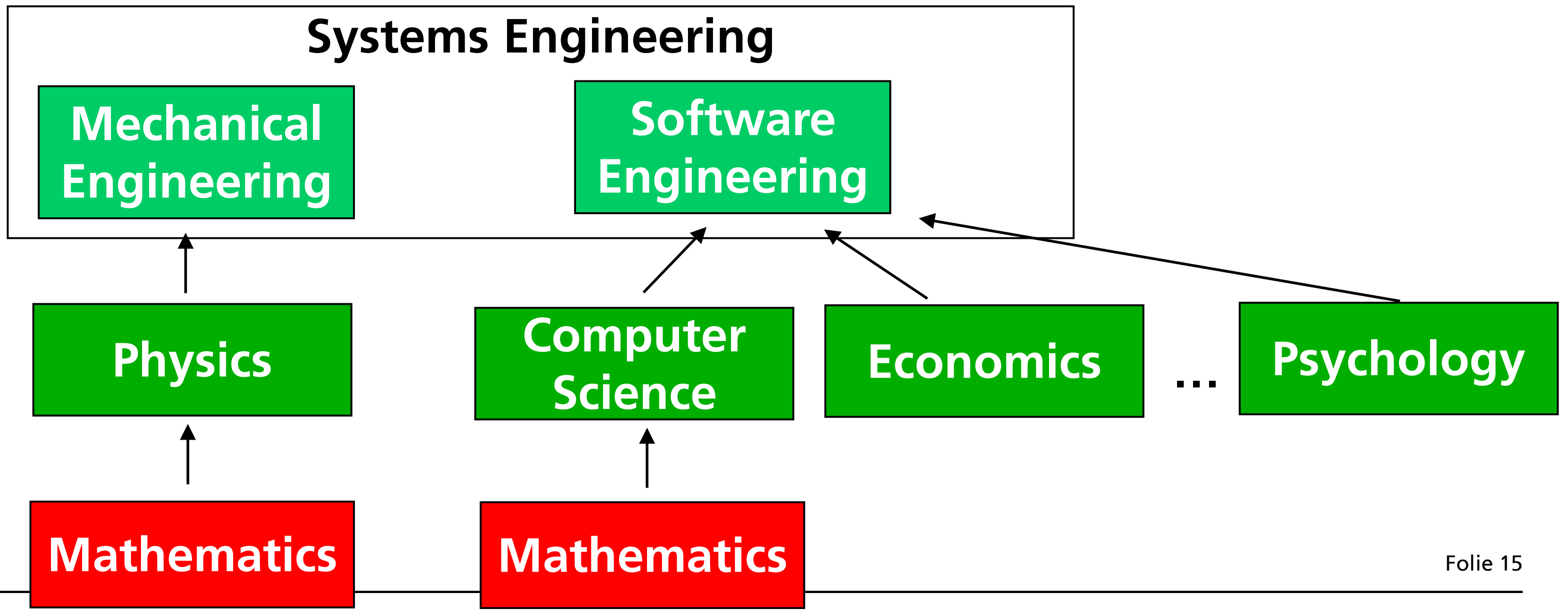




**Experimental Software Engineering recognizes the nature of our field**

# (Empirical) Software Engineering (2/2)

- Computer Science is one of the scientific base disciplines for the "engineering of large (software) systems"



## Empirical Methods (1/3)

- Traditional (quantitative) empirical evidence
  - controlled experiments

    (variation in *C* is controlled)
  - case studies

    (*C* is a constant – reflecting some environment)

- Questionnaires, Action Research, …. (mostly qualitative)
- Expert consensus (like in medicine)

$G == f(P, C)$

Practical acceptance increases

Statistical significance decreases



Scientists (aiming at testable cause-effect relations) prefer controlled expriments!

Practitioners (aiming at low-risk technology infusion) prefer case studies & expert consensus!

# Empirical Methods (2/3)

| | | # Projects | |
|---|---|---|---|
| | | 1 | m > 1 |
| # Teams per Project | 1 | **1 x 1** - Experiment<br>[single project] - [case study] | **1 x m** - Experiment<br>[multi-project variation] |
| | n > 1 | **n x 1** - Experiment<br>[replicated project] | **n x m** - Experiment<br>[blocked subject-project] |

**Sustained Technology Transfer requires combinations of studies!**

## Empirical Methods (3/3)

- Science in general involves
  - modeling of software product & process artifacts
  - empirical validation of hypotheses regarding their characteristics & behavior in testable/challengeable form
- Empirical foundation 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



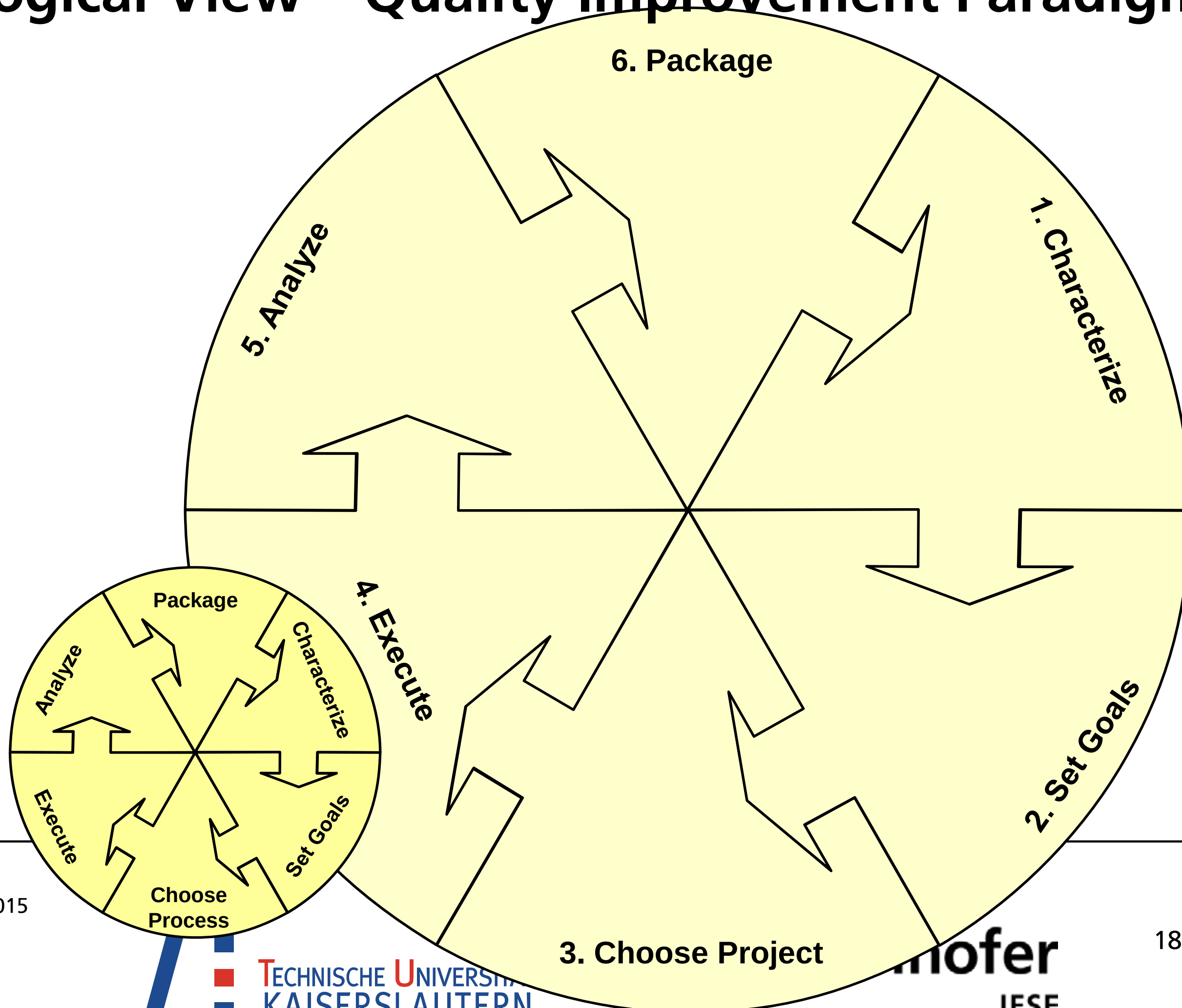
**There exists a comprehensive body of empirical methods!**
- Workshops (e.g., ISERN)
- Conferences (e.g., ESE Conference)
- Journals (e.g., ESE)

# GQM Abstraction Sheet

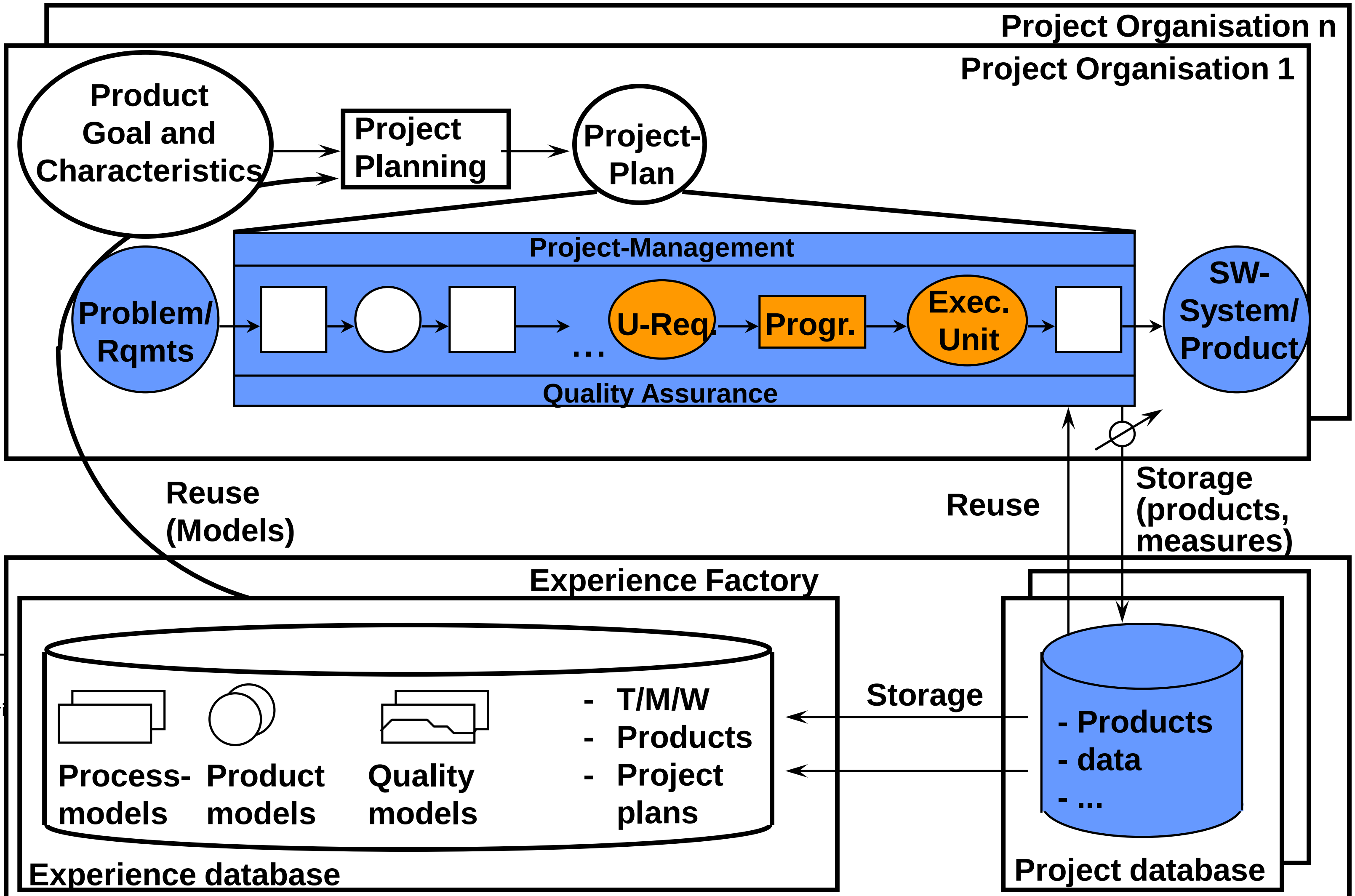
| Object | Purpose | Quality Aspect | Viewpoint | Context |
|---|---|---|---|---|
| Inspection | Understand | Effectiveness | Inspector | X |

| | |
|---|---|
| **Quality Focus**<br>• M1: # defects detected<br>• M2: # defects slipped<br>• M3: M1 / (M1 + M2) %<br>• M4: # hours per detection | **Variation Factors**<br>• M5: Experience of personnel ( - , **0** , + )<br>• M6: Size of program ( - , **0** , + )<br>• M7: Language ( **L1**, L2 , L3 ) |
| **Baseline Hypotheses**<br>• M3: 75%<br>• M4: 3 h | **Impact on Baseline Hypotheses**<br>• if (M5='+') then (M3='90%')&(M4='2.5 h')<br>• if (M7='L2')&(M6='+') then (M3='60%')&(M4='4 h') |

# Methodological View – Quality Improvement Paradigm (QIP)



# Organizational View – Experience Factory (EF)



# The Maturation of Empirical Studies

- Motivation
- Basic Framework
  - Empirical Evidence
  - Empirical Software Engineering
  - Empirical Methods
- **Maturation** (expanded version of VRB 2006)
  - **Phase 1: Isolated Studies**
  - **Phase 2: Multiple Studies (domain/environment specific)**
  - **Phase 3: Multiple Studies (across domains/environments)**
  - **Phase 4: Towards Creating Evidence**
- Today & Future (Towards a Theory of Software Engineering Evidence)
  - Existing Body of Knowledge
  - Experimental Software Engineering in Kaiserslautern (Fraunhofer IESE) – Practical Examples
- Agenda for Research, Tech Transfer & Teaching
- Outlook

# The Maturation of Empirical Studies

- **Example 1970's:**

  - **Question**: Can we quantitatively measure the effect of the application of a method on the product?
    - Method produced incremental versions of the product, each with more functionality

  - **Empirical Approach**: *Case study* measuring versions of the incrementally developed product to show what happened,

  - **Issues**: quantitative, observations over time, product metrics, comparing a product with itself (baseline issue), using feedback

  - V. Basili and **A. Turner**, "Iterative Enhancement: A Practical Technique for Software Development," IEEE Transactions on Software Engineering, vol. 1(4), December **1975**

# The Maturation of Empirical Studies

**Isolated Studies (1970's):**

**Q == F (P, C)**

- **Objectives:** Run isolated studies on a particular purpose
- **Methods:** Case Studies, Controlled Experiments
- **Results:** C fixed, observations (neither qualitatively, nor quantitatively repeatable),
- **Examples:** SEL (Basili/Turner 75, Basili/Zelkowitz 78)
- **Lessons Learned:** metrics, measurement process, performance of empirical studies, nonparametric statistics, context as given, local (often non-repeatable) evidence, …, SEL as empirical lab, GQM/QIP

Co

We (as a community) learned
- How to perform individual empirical studies!
- That they were not repeatbale (no context consideration)!

## Example 1980's: Inspections @ NASA GSFC

| | | # Projects | |
|---|---|---|---|
| | | One | More than one |
| # of Teams per Project | One | 3. Cleanroom (SEL Project 1) | 4. Cleanroom (SEL Projects, 2,3,4,...) |
| | More than one | 2. Cleanroom at Univ. of Maryland | 1. Reading vs. Testing<br>5. Scenario reading vs. ... |

# The Maturation of Empirical Studies

**Q == F (P, C)**

## Multiple Studies

## – environment/domain specific (1980's):

- **Objectives**: Tying studies together in one environment/domain
- **Methods**: Case Studies, Controlled Experiments, quasi experiments, qualitative studies

➔ **Results**: C variable within one environment/domain, mostly observations (neither qualitatively, nor quantitatively repeatable), some first laws (qualitatively repeatable), experimental framework, packages to repeat studies (Lott), evolved QIP (packaging) and GQM (templates and models) (Basili/Rombach, TSE 1988, "The TAME Project), formalized the Experience Factory Organization (Basili, "Software Development: A Paradigm for the Future," Compsac 89);

➔ **Examples**: Inspections based on solid reading (repeated studies ➔laws); Fraunhofer IESE

Co

**We (as a community) learned**

- **How to capture variations of effects for different context params!**
- **How to support effective tech transfer via combinations of studies!**

…

# The Maturation of Empirical Studies

## Examples 1990's: Fraunhofer IESE

| Method | Result | Publications |
|---|---|---|
| AcES | ■ 35% reduction of implementation and testing effort at same quality level | ■ ICSR 2008 |
| AcES/RATE, SAVE | ■ 60% less time needed for architectural analysis if architectures are visualized appropriately | ■ EMSE 2008 |
| SAVE-Life | ■ 60% fewer architecture violations if developers are getting live feedback on their architectural compliance | ■ PhD Knodel 2010 |
| AcES | ■ Architecture-compliant implementation reduces development effort by 50% | ■ PhD Knodel 2010 |

# The Maturation of Empirical Studies

## Multiple Studies across Domains (1990's):

**Q == F (P, C)**

- **<u>Objectives</u>**:, Expanding across environments/domains, trying to build evidence for a couple of techniques
- **<u>Methods</u>**: Build public repositories (e.g., VSEK, CeBASE) to establish evidences, Case Studies, Controlled Experiments, quasi experiments, qualitative studies
- **<u>Results</u>**: C variable across environments/domains, observations/laws, ISERN/EMSE/ESEM, Evolved empirical evidence about various techniques; more industry studies (e.g., **Fraunhofer IESE**)
- **<u>Examples</u>**: evolved empirical evidence about inspections, OO, and many other techniques (see IESE), Lessons learned (e.g., B. Boehm and V. Basili, "Software Defect Top 10 List," *IEEE Computer*, 2001; Basili/ Boehm, "COTS-Based Systems Top 10 List," *IEEE Computer* 2001

**We (as a community) learned**
**- How to share data/evidence across environments/domains?**
**➔ VERY HARD / VERY COMPLEX !!!**
**➔ Works only in trusted settings**
**- How to build initial communities of trust (e.g., ISERN, Fraunhofer IESE)!**

# The Maturation of Empirical Studies

**Towards Evidence (2000's):**

- Q == F (Pi, C)

- **Objectives**: Focusing on domain to build evidence and theories, understanding all relevant impact factors
- **Methods:** Case Studies, Controlled Experiments, quasi experiments, qualitative studies, GQM+Strategies
- **Results:** C variable within environments/domains, capture & understanding of all relevant context factors
- **Examples:** Bosch theory for inspection techniques to repeat results under varying contexts
- **Lessons Learned**: hard problem in development environment

We (as a community) learned
- How to involve industry (not empirical studies, but risk-averse technology transfer based on evidences?
- Foster trusting environments (ISERN, Fraunhofer IESE/CESE/FPG Bahia)!

# Towards a Theory of SE Evidence (1/2)

- Aggregation (P basic & constant)
  - to increase significance within same context C (i.e. reduce <var>)
  - to increase generality by varying context C (i.e. $C := C1 \times C2 \times C3 \times C4$)
- Significance increase
  - experiment replication (e.g., inspection area)
- Variation increase
  - experiment variation across contexts (e.g., applications, experiences, …)
- Challenges
  - Complexity: simple coverage for 5 variables with 4 values each requires "4 to the power of 5" = 1024 studies???
  - New hidden context variables appear: Combining contexts ➔ new hidden context variables *HC* appear (identified via meta analysis)!
    - E.g., (G1, P, C) & (G2, P, C) → (G1!G2, P, C x (HV1!HV2))

**Aggregation is hard**

- Even in a homogeneous case (e.g., just controlled experiments, PhD Ciolkowski)
  - Not to speak about heterogenous cases (i.e. different types of studies)

# Towards a Theory of SE Evidence (2/2)

- Aggregation (P complex &/ variable)
  - to scale up to "larger" processes P (e.g., Cleanroom software development process)
  - perform controlled experiments in "key elements" (e.g., unit inspections vs. testing)
  - perform integration case studies
  - acceptance of scaled-up evidence must be confirmed by expert consensus (organization or community)

**Scaleability wrt. Complexity of P requires**
- **Smart use of controlled experiments for key process components**
- **Scale-up case studies for complex process(es)**

# The Maturation of Empirical Studies

- Motivation
- Basic Framework
  - Empirical Evidence
  - Empirical Software Engineering
  - Empirical Methods
- Maturation (expanded version of VRB 2006)
  - Phase 1: Isolated Studies
  - Phase 2: Multiple Studies (domain/environment specific)
  - Phase 3: Multiple Studies (across domains/environments)
  - Phase 4: Towards Creating Evidence
- **Today & Future (Towards a Theory of Software Engineering Evidence)**
  - **Existing Body of Knowledge (Examples)**
  - **Experimental Software Engineering in Kaiserslautern (Fraunhofer IESE) – Practical Examples**

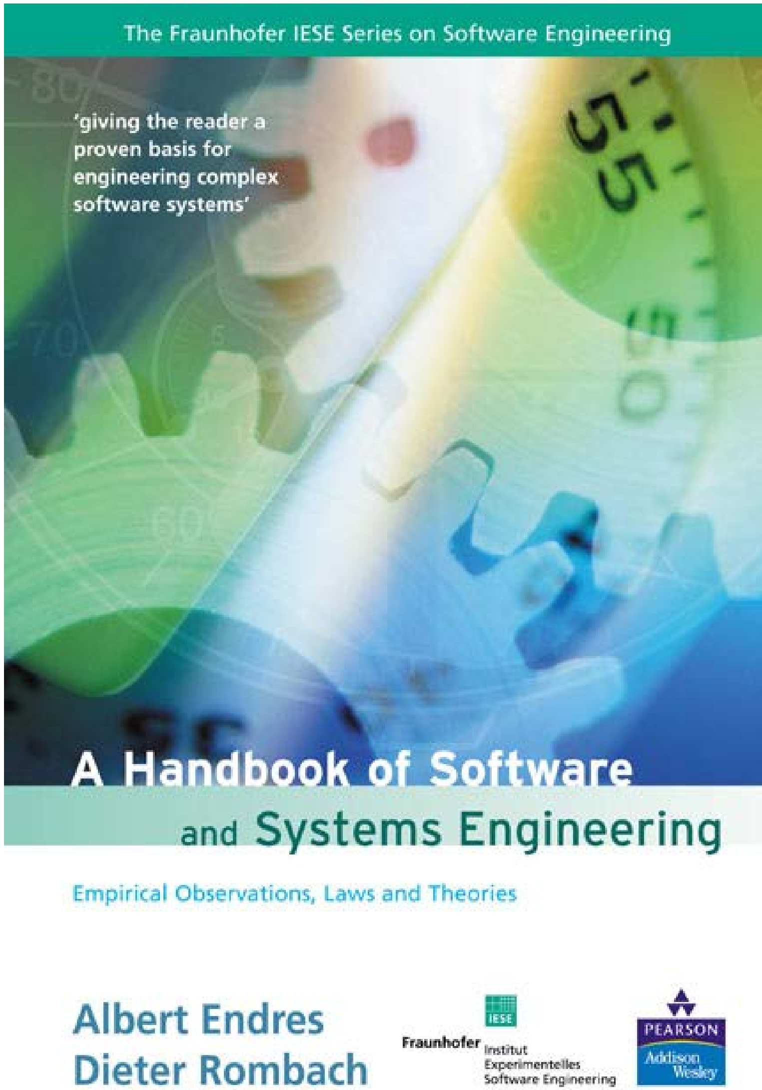
C

- Agenda for Research, Tech Transfer & Teaching
- Outlook

# Existing Body of Knowledge (1/3) – NASA SEL

- NASA SEL Experience (see Basili, JSS, 1997)
  - stepwise abstraction code reading vs. testing (Basili/Selby, TSE, 1987)
    - controlled experiment at UMD & NASA/CSC
    - effectiveness & cost (SAR > testing)
    - self-assessment (SAR > testing)
  - stepwise abstraction code reading in regular SEL project
    - case study at NASA/CSC
    - SAR did not show any benefits
    - diagnosis: People did stewise abstraction code reading not as well as they should have as they believed that testing would make up for their mistakes
  - Cleanroom vs. standard SEL software development
    - controlled experiment at UMD
    - more effective application of reading, less effort and more schedule adherence
  - stepwise abstraction code reading in SEL Cleanroom projects
    - case study(ies) at NASA/SEL
    - improved failure rates (- 25%) and productivity (+30%)

# Existing Body of Knowledge (2/3) – Community




**Handbook capturing existing body of knowledge**

**Students can learn about existing body of knowledge**

**Practitioners can avoid negligance of due dilligance**

**Additions are welcome for next edition of book (online?)**

## 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"
- There exist already more empirical "laws" than we typically recognize
  - book (Endres/Rombach, Addison, 2003)
  - inspections
  - design principles
- More studies need to be done
  - repeat (with variation)
  - generalize

# The Maturation of Empirical Studies

## 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

  **<u>Possible solutions</u>:**
  - 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)

## Design

- Good designs require deep application domain knowledge (L5)
  - Source: Bill Curtis et al [Curt88, Curt90]
  - “Goodness” is defined as stable and locally changeable (diagonalized requirements x component matrix)
  - Key principle: information hiding
  - Domain knowledge allows prediction of possible changes/variations
  - See: Y2K example
- Hierarchical (regular) structures reduce complexity (L6)
  - Source: Herb Simon [Simo62]
  - Examples: large mathematical functions, operating systems (layers), books (chapter structure), ….
- Incremental processes reduce complexity (L6a)
  - Source: Harlan Mills (Cleanroom) [MIL87]
  - Large tasks need to be refined in a number of comprehensible tasks
  - Examples: Arabic number division, iterative life-cycle model, incremental verification & inspection

## Design

- A structure is stable if cohesion is strong & coupling is low (L7)
  - Source: Stevens, Myers, and Constantine [Stev74]
  - High cohesion allows changes (to one issue) locally
  - Low Coupling avoids spill-over or so-called ripple effects

- Only what is hidden can be changed without risk (L8)
  - Source: David Parnas [Parn72]
  - Information hiding applied properly leads to strong cohesion/low coupling
  - See: Y2K-Problem

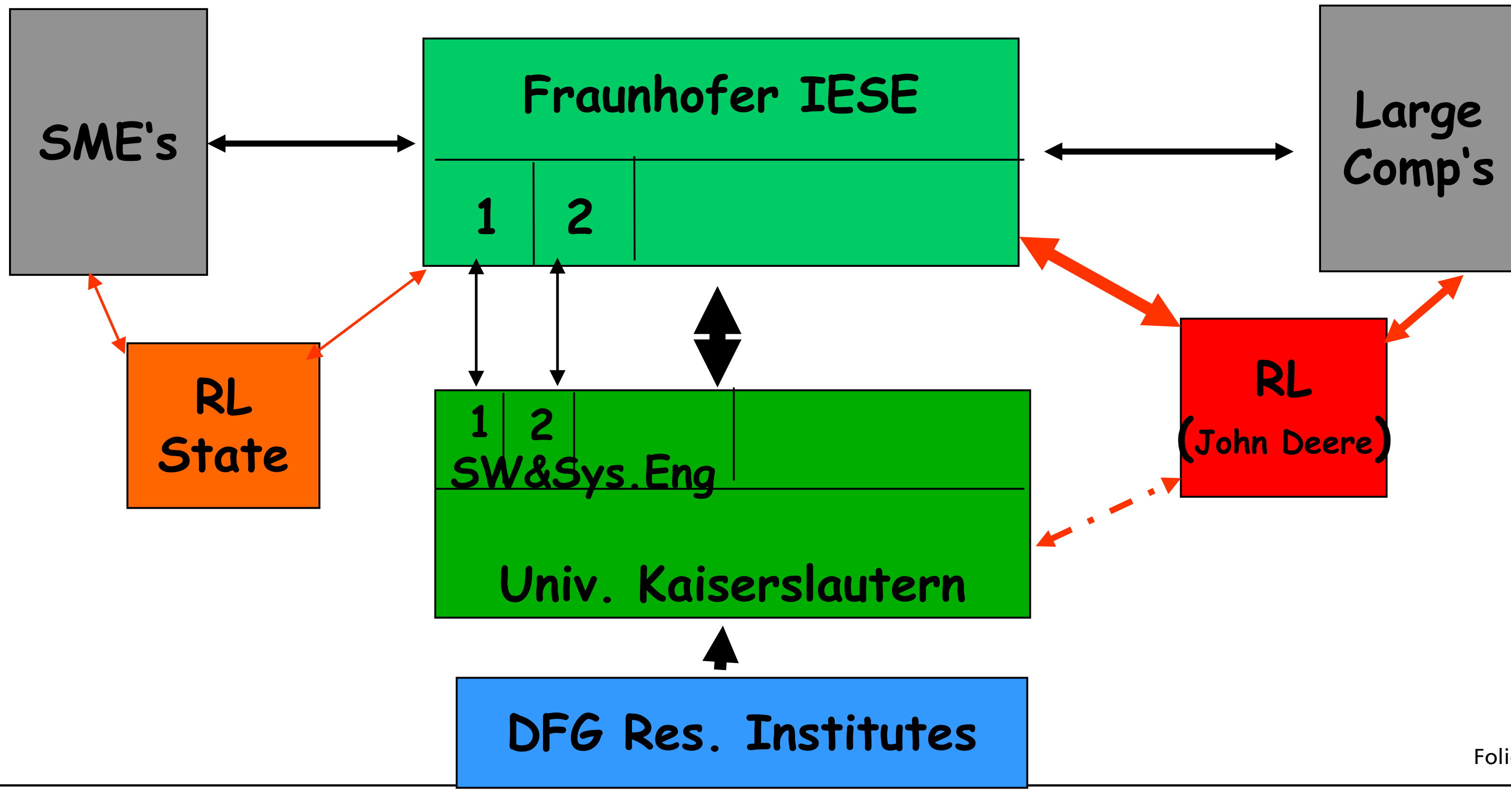
# Verification

- 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 planable due to fewer rework exceptions)
  - See: Inspections, Cleanroom
- Effectiveness of inspections is rather independent of its organizational form (process), but depends on the reading technique used (L18)
- Perspective-based inspections are highly effective and efficient (L19)
  - Source: Victor Basili [Bas96c, Shull00]]
  - Best suited for non-formal documents
  - See: PBR inspection

## Project Management

- Individual developer productivity varies considerably (variability is higher, if process guidelines are less detailed) (L31)
  - Source: Sackmann [Sack68]
- A multitude of factors influences developer productivity (L32)
- Development effort is a (non-linear) function of product size (L33)
  - Source: Barry Boehm [Boeh81, Boeh00c]
  - See: COCOMO-Model
- Most cost estimates tend to be too low (L34)
- Mature processes and personal discipline enhance planning, increase productivity and reduce errors (L35)
- Adding resources to a late project makes it later (L36)
  - Source: Fred Brooks [Broo75]

# Existing Body of Knowledge (3/3): Kaiserslautern








Technische Universität Kaiserslautern

Fraunhofer IESE

# The Maturation of Empirical Studies

- Further IESE work on inspections
  - investigation of effects in OO/UML environment (Laitenberger)
    - defined PBR for OO/UML (packaging of reading unit across views
    - controlled experiments
      - students at UKL (SE class)
      - PBR of requirements spec (UML) vs CBR
      - effectiveness & cost (PBR > CBR)
    - replication of existing (see NASA/SEL) studies in varying contexts (application domains, technology domains)
  - variation of existing studies to address new questions
    - optimal effort for preparation phase in inspection process (exists as demonstrated at Bosch; is used to manage inspection process)
- Industrial relevance
  - helped establish inspections with sustained success in several companies (e.g., Allianz, Bosch)
  - focus on inspections (with measurement-based feedback) matures development organizations (e.g., Bosch unit with inspections went from CMM1 to CMM 3 in one step!)

# The Maturation of Empirical Studies

- IESE Studies on OO/UML (Briand, Bunse, Daly)
  - operationalized good design principles such as coupling, information hinding & cohesion
  - hypotheses:
    - #1: "Good" OO designs are better understood
      - measured by the correctness of answers to a set of questions
    - #2: Impact analysis on "good" OO designs is performed beter and faster
      - measured by the time & correctness of all changes to perform a set of given change requests
  - controlled experiments at UKL
  - 2 systems ("good", "bad"); 2x2 factorial design
  - results
    - all results significantly in favor of "good" design
    - students made important self-experience regarding a set of engineering principles

# The Maturation of Empirical Studies

| Method | Result | Publications |
|---|---|---|
| PuLSE | ■ Strategic reuse program increases reuse level by 50%<br>■ Architectural divergences decreased from 17% to 1% | ■ ArQuE 02.09<br>■ CSMR 2008 |
| PuLSE | ■ With SPL approach, productivity has tripled<br>■ # of quality problems has been reduced to 20% | ■ Ricoh 2010 |
| PuLSE-EM | ■ 27% less effort on average for configuration management in a product line | ■ IWPSE-EVOL 2009 |





Technische Universität Kaiserslautern

Fraunhofer IESE

Florence
18 May 2015

# The Maturation of Empirical Studies

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





Technische Universität Kaiserslautern

Fraunhofer IESE

Florence
18 May 2015

# The Maturation of Empirical Studies

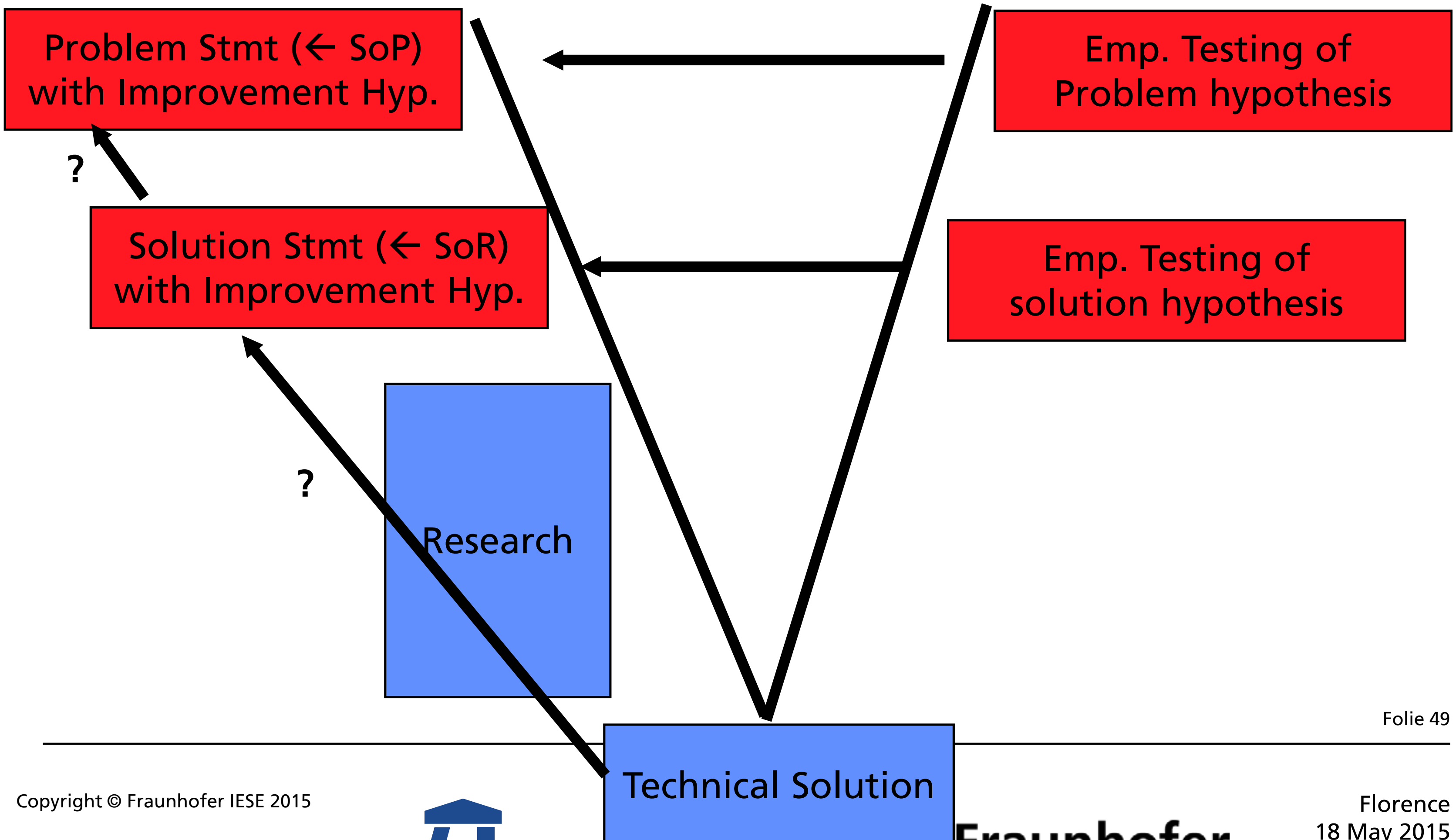
- Motivation
- Basic Framework
  - Empirical Evidence
  - Empirical Software Engineering
  - Empirical Methods
- Maturation (expanded version of VRB 2006)
  - Phase 1: Isolated Studies
  - Phase 2: Multiple Studies (domain/environment specific)
  - Phase 3: Multiple Studies (across domains/environments)
  - Phase 4: Towards Creating Evidence
- Today & Future (Towards a Theory of Software Engineering Evidence)
  - Existing Body of Knowledge
  - Experimental Software Engineering in Kaiserslautern (Fraunhofer IESE) – Practical Examples
- **Agenda for Research, Tech Transfer & Teaching**
- Outlook

## Agenda (for Research) (1/3)

- SE Research results require "some form of evidence"
  - notations, techniques, methods & tools w/o evidence are not accepted as software engineering results (e.g., PhD theses)
  - collaboration with SE practice & CS experts
- Future research focus on
  - empirical methods includes
    - Aggregation
    - Subjective & objective approaches
    - Better measures of significance (in case of complex processes)
  - empirical studies includes
    - Complex processes (e.g., agile)
    - Theory of evidence for (best practice) processes

**Without empirical evidence it is no software engineering contribution – as it**
- **does not allow scientific challenging!**
- **does not contribute to engineering challenge!**

# The Maturation of Empirical Studies



## Agenda (for Tech Transfer) (2/3)

- Apply "ESE" 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, German Telecom)
  - 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 in the past!**

**- Fraunhofer uses ESE as its business model engine!**

# Agenda (for Teaching & Training) (3/3)

- Learning in engineering is based on
    - reading
    - doing
    - experiencing
- Teaching must reflect by
    - first analyzing, then constructing (based on proven evidence)
    - performing "self-experience" studies
- At 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"

**Teaching engineering requires**
**- Learning of proven evidence (best practices)**
**- lecturing, doing & experiencing!**

# The Maturation of Empirical Studies

- Motivation
- Basic Framework
  - Empirical Evidence
  - Empirical Software Engineering
  - Empirical Methods
- Maturation (expanded version of VRB 2006)
  - Phase 1: Isolated Studies
  - Phase 2: Multiple Studies (domain/environment specific)
  - Phase 3: Multiple Studies (across domains/environments)
  - Phase 4: Towards Creating Evidence
- Today & Future (Towards a Theory of Software Engineering Evidence)
  - Existing Body of Knowledge
  - Experimental Software Engineering in Kaiserslautern (Fraunhofer IESE) – Practical Examples

C

- Agenda for Research, Tech Transfer & Teaching
- **Outlook**

# The Maturation of Empirical Studies

## Outlook

- SE is on its way to become a respected engineering discipline
  - automotive companies have more software than hardware engineers (since 2000)
  - mature software engineering includes empiricism (to create evidence)
  - system & service engineering (IoT&S) require mature software engineering (because we interact with real engineers)
- We need more community efforts
  - to provide trusted environments for industry collaboration
  - to create shared "handbooks of SE" (online)
- University of Kaiserslautern / Fraunhofer IESE
  - has  leading laboratory settings for empirically driven software engineering research
  - Maintains evidence-based innovation co-operations with industry for 20 years (successfully)
  - maintains international network (USA, Brazil, Europe)



The complexity of new (IoT&S based systems of systems requires evidence-based engineering!

# THANK YOU !