How to identify best practices?
– empiri and system development

SPIKE / Abelia Innovasjon theme conference
Klækken hotell, 26-27 Nov. 2003

Reidar Conradi
Dept. Computer and Information Science (IDI)
NTNU, NO-7491 Trondheim

conradi@idi.ntnu.no, Tel +47 73.593444, Fax +47 73.594466

Abelia/SPIKE: Good practice - empiri & syst.dev., Klækken, 26-27 Nov. 2003
Motivation for software improvement (1)

• **Large societal importance:** ubiquitous SW, vulnerable impact.
• **Want software faster, better, cheaper, …**
• **ICT sector:** second largest industry in Norway, 190 MNOK in annual revenues, 90 000 employees, 3.5 % of GNP value creation (SSB, 2001).
• 2/3 of SW developed **outside** traditional ICT industries (EU).
• 50-60 000 software developers (3%) in Norway– many with scant formal education in informatics.

• **Ex. Norwegian failures:** Skattedirektorates SCALA-system, NSBs billettsystem, NTNUs lønns/personalsystem, Rikstrygdeverkets TRESS, … But only the failures are heard of.

• **Ex. Problems in USA:** US Standish ”Chaos” report from 1995, cited in [PITAC99], on projects for tailored software:
  – 31% stopped before finish, 81 bill. $ loss/year (1% of GNP!)
  – 53% have serious overruns (189% average), 59 bill. $/year
Motivation for software improvement (2)

Some current challenges:
– Web-systems: time-to-market (TTM) vs. reliability?
– How do software systems evolve (”rot”) over time, cf. Y2K?
– How to use COTS components? [Basili01]
– How to estimate software development?
– ...
– What is empirically known about software technologies (techniques, methods, processes)?
– How to advice industry about software technologies, considering their context?
– How can SMEs carry out systematic improvement?
– How can we learn from each other – industry vs. research?
– How to perform valid sw.eng. research in a university -- by student projects and having industry serving as a lab?
Proposed “silver bullets” [Brooks87] (1)

What almost surely works:

• Software reuse/CBSE/COTS: yes!!
• Formal inspections: yes!!
• Systematic testing: yes!!
• Better documentation: yes!
• Versioning/SCM systems: yes!!
• OO/ADTs: yes?!, especially in domains like distributed systems and GUI.
• High-level languages: yes! - but Fortran, Lisp, Prolog etc. are domain-specific.
• Bright, experienced, motivated, hard-working, …developers: yes!!! – brain power.
• More powerful workstations: yes!! – computer power.
Proposed ”silver bullets” [Brooks87] (2)

What probably works:

• Better education: hmm?
• UML: often?, but need tailored RUP and more powerful tools.
• Powerful, computer-assisted tools: partly?
• Incremental development e.g. using XP: partly?
• More ”structured” process/project (model): probably?, if suited to purpose.
• Software process improvement: in certain cases?, assumes stability.
• Structured programming: conflicting evidence wrt. maintenance?
• Formal specification/verification: does not scale up? – only for safety-critical systems.

Need further studies (”eating”) of all these ”puddings”: what works with what results in what contexts – many challenges!
Empirical Software Engineering (ESE)

- Lack of formal validation in computer science / software engineering vs. other disciplines: [Tichy98] [Zelkowitz98].
- (New) technologies not properly validated: OO, UML, …
- Empirical / Evidence-based Software Engineering since 1992: writings by Basili, [Rombach93], [Wohlin00], Juristo.
- Sw.eng. group at NTNU since 1993, at UiO from 1997 – both with ESE emphasis.
- Sw.eng. group at Simula Research Laboratory from 2001: attn/ Dag Sjøberg, in coop. with NTNU, SINTEF et al.
SW Eng. characterization: need ESE

- SE learnt by “doing”, i.e. realistic projects in SE courses. Strong “soft” (human and organizational) factors.
- Problems in being more “scientific”:
  - Most industrial SE projects are unique (goals, technology, people, …), otherwise just copy software with marginal cost!
  - Fast change-rate between projects: goals, technology, people, process, company, … – i.e. no stability, meager baselines.
  - Also fast change-rate inside projects: much improvisation, with theory serving as back carpet.
  - So never enough time to be “scientific” – with hypotheses, metrics, collected data, analysis, generalization, and actions.
- How can we overcome these obstacles, i.e. to learn and improve systematically? – ESE as the answer?
- Tens of factors (“context”) in software projects – how to show effect and causality? Realism vs. rigour?
Possible “context” factors/variables

• To understand a discipline means to build models, that later can be validated and refined – but many context factors.

• People factors: number of people, level of expertise, group organization, problem experience, process experience, …

• Problem factors: application domain, newness to state of the art, susceptibility to change, problem constraints, …

• Process factors: life cycle model, methods, techniques, tools, programming language, other notations, …

• Product factors: deliverables, system size, required qualities such as time-to-market, reliability, portability, …

• Resource factors: target and development machines, calendar time, budget, existing software, …

• Example: 29 factors to predict software productivity [Walston77].

(from Basili’s CMSC 735 course at Univ. Maryland, fall 1999)
Ex. Estimation models, e.g. by Barry Boehm

- Effort = E1 * Size ** 0.91 + E2
- Duration = D1 * Effort ** 0.38 + D2
- And many other magic formulaes!

- Question: Can “E1” express 29 underlying factors?
- And how to calibrate for an organization, and use with sense?

- Formal vs. informal (expert) estimation [Jørgensen03]?
Ex. Model of fault rate vs. size

- [Basili84]: the fault rate of modules shrunk as module size and complexity grew in the NASA-SEL environment; other authors had inverse observation – who was right?:

- Explanation: smaller modules are normally better, but involve more interfaces and often chosen when “(re-)gaining” control.
- Above result confirmed by similar studies - but many more factors …
Four basic parameters in a study (GQM-method)

- **Object**: a process, a product, any form of model.
- **Purpose**: characterize, evaluate, predict, control, improve, …
- **Focus** (relevant object aspect): time-to-market, productivity, reliability, defect detection, accuracy of estimation model, …
- **Point of view** (stakeholder): researcher, manager, customer, …

- all this involves many factors/variables.
ESE: common kinds of empirical studies

- **Formal experiments**, “in vitro”, often among students: can control the artifacts, process and outer context.
- **Quasi experiments**, in “vivo”, in industry: costly and hard logistics. Use Simula’s SESE web-tool [Sjøberg02]?
- **Case studies**: try new technology in real project.
- **Post-mortems**: collect lessons-learned, e.g. by data mining or interviews [Birk02].
- **Surveys**: often by questionnaires.
- **Structured interviews**: more personal than surveys.
- **Observation**: being a “fly on the wall”.
- **General Theory**: Generalize from available data.
- **Action research**: researcher and developer overlap roles.
ESE: different data categories

- **Quantitative ("hard") data**: data (i.e. numbers) according to a defined metrics, both direct and indirect data. Need suitable analysis methods, depending on the metrics scale – nominal, ordinal, interval, and ratio. Often objective.

- **Qualitative ("soft") data**: prose text, pictures, … Often from observation and interviews. Need much human interpretation. Often subjective.

- **Specific data** for a given study (e.g. reuse rate) vs. **Common data** (cost, size, #faults, …) - “nice to have”?
ESE: validity problems

• **Construct validity**: the "right" (relevant, precise, minimal, …) metrics - use Goal-Question-Metrics?

• **Internal validity**: the "right" data values.

• **Conclusion validity**: the right (appropriate) data analysis.

• **External validity**: the "right" (representative) context.
ESE: combining different studies/data

- **Meta-studies**: aggregations over single studies. Cf. medicine with Cochran reporting standard. Need shared experience databases?
- A composite study may **combine** several study kinds and data:
  1. Prestudy, doing a survey or post-mortem
  2. Initial formal experiment, on students
  3. Sum-up, using interviews
  4. Final case study, in industry
  5. Sum-up, using interviews or post-mortem
Achieving validated knowledge: by ESE

- **Learn about ESE:** [Rombach93] [Conradi03].
- **Set goals,** e.g. use QIP [Basili95]?
- **Need operational methods** to perform studies: general [Kitchenham02], on GQM [Basili94]?
- **Cooperate** with others on repeatable studies / experiments (ISERN, ESERNET, …) [Vokác03].
- **Perform meta-analysis** across single studies. Need reporting procedures, databases etc.
- **Need more industrial** studies, not only with students.
- **Have patience,** allocate enough **resources.** Industrial studies will run into unexpected problems; SPI initiatives have 30-50% “abortion” rate [Conradi02] [Dybå03].
Ex. Some NTNU studies (all published)

CBSE/reuse:
- Modifiability of C++ programs and documentation, 1995.
- Ex3, INCO: COTS usage in Norway, Italy, and Germany 2002-04 (many).
- Assessment of COTS components, 2001-02.
- Ex2, INCO: CBSE at Ericsson-Grimstad, 2001-04 (many).

Inspections:
- Perspective-based reading, at U. Maryland and NTNU, 1995-96.
- Ex1, NTNU diploma theses: SDL inspections at Ericsson, 1993-97.
- UML inspections at U.Maryland, NTNU and at Ericsson, 2000-02.

SPI/quality:
- Role of formal quality systems in 5 companies, 1999.
- Comparing process model languages in 3 companies, 1999.
- Post-mortem analysis in two companies, 2002.
- SPI experiences in SMEs in Scandinavia and in Italy and Norway, 1997-2000.

And many more!

Abelia/SPIKE: Good practice - empiri & syst.dev., Klækken, 26-27 Nov. 2003
Ex1. SDL inspections at Ericsson-Oslo 1993-97, data mining study in 3 MSc theses (Marjara et al.)

General comments:

• AXE telecom switch systems, with functions around * and # buttons, teams of 50 people.
• SDL and PLEX as design and implementation languages.
• Data mining study of internal inspection database. No previous analysis of these data.
• Study 1: Project A, 20,000 person-hours. Look for general properties + relation to software complexity (by Marjara being a previous Ericsson employee).
• Study 2: Project A + Project-releases B-F, 100,000 person-hours. Also look for longitudinal relations across phases and releases, i.e. “fault-prone” modules - seems so, but not conclusive (by Skåtevik and Hantho)
• When results came: Ericsson had changed process, now using UML and Java, but with no inspections.
Ex1. General results of SDL inspections at Ericsson-Oslo 1993-97, by Marjara

<table>
<thead>
<tr>
<th>Activity</th>
<th>Yield = Number of Defects [#]</th>
<th>Total effort on defect detection [h]</th>
<th>Cost-efficiency [defect/h]</th>
<th>Total effort on defect correction [h]</th>
<th>Estimated saved effort by early defect removal (“formulae”) [h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspection preparation, design</td>
<td>928</td>
<td>786.8</td>
<td>1.17</td>
<td>311.2</td>
<td>8200</td>
</tr>
<tr>
<td>Inspection meeting, design</td>
<td>29</td>
<td>375.7</td>
<td>0.077</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Desk Check (Unit Test and Code Review)</td>
<td>404</td>
<td>1257.0</td>
<td>0.32</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Function Test</td>
<td>89</td>
<td>7000.0</td>
<td>0.013</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Total so far</td>
<td>1450</td>
<td>9419.5</td>
<td>0.15</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>System Test</td>
<td>17</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Field Use (first 6 months)</td>
<td>35</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1. Yield, effort, and cost-efficiency of inspection and testing, Study 1.

Study 1 overall results:
- About 1 person-hour per defect in inspections.
- About 3 person-hours per defect in unit test, 80 p-h/defects in function test.
- So inspections seem very profitable.
Ex1. SDL-defects vs. size/complexity (#states) at Ericsson-Oslo 1993-97, by Marjara

Study 1 results, almost “flat” curve -- why?:
- Putting the most competent people on the hardest tasks!
- Such contextual information is very hard to get/guess.
Ex1. SDL inspection rates/defects at Ericsson-Oslo 1993-97, by Marjara

Study 1: No internal data analysis, so no adjustment of insp. process:
- Too fast inspections: so missing many defects.
- By spending 200(?) analysis hours, and ca. 1250 more inspection hours:
  will save ca. 8000 test hours!

AbeliaSPIKE: Good practice - empiri & syst.dev., Klækken, 26-27 Nov. 2003
Ex2. INCO, studies and methods by PhD student
Parastoo Mohagheghi, NTNU/Ericsson-Grimstad

• Study *reusable middleware* at Ericsson, 600 KLOC, shared between GPRS and UMTS applications:
  – Characterization of quality of reusable comp. (pre-case study)
  – Estimation of use-case models for reuse – with Bente Anda, UiO (case study)
  – OO inspection techniques for UML - with HiA, NTNU, and Univ. Maryland (real experiment)
  – Attitudes to software reuse – with two other companies (survey)
  – Evolution of product families (post-mortem analysis)
  – Improved reuse processes (proposal for case study)
  – **Reliability and stability of reusable components**, based on 13,500 (!) change requests – with NTNU (case study/data mining), *next three slides*
Ex2. GPRS/UMTS system at Ericsson-Grimstad

Application A
Application B

Business Specific

Middleware
(& Component Framework)

System Platform

Application-specific components

Reused components in our study

Reused, but considered as COTS and OSS here
Ex2. Research design (data mining)
Ex.2 Hypotheses testing (as null-hyp.)

- **H01**: Reused components have same fault-density as non-reused components. *Rejected - reused more reliable.*
- **H02a**: There is no relation between #faults and component size for all components. Not rejected - *not incr. with size.*
- **H02b**: There is no relation between #faults and component size for reused components. Not rejected - *not incr. with size for reused.*
- **H02c**: There is no relation between #faults and component size for non-reused components. *Rejected - incr. with size for non-reused.*
- **H03a/b/c**: There is no relation between fault-density and component size for all/reused/non-reused components. Not rejected.
- **H04**: Reused and non-reused components are equally modified. *Rejected - reused more stable.*
Ex3. COTS usage contradicts “common wisdom”

In INCO, structured interviews of 7 Norwegian and Italian SMEs:

- **Thesis T1:** Open-source software is often used as closed source.
- **Thesis T2:** Integration problems result primarily from lack of compliance with standards; not architectural mismatches.
- **Thesis T3:** Custom code is mainly devoted to add functionalities.
- **Thesis T4:** Formal selection seldom used; rather familiarity with product or generic architecture.
- **Thesis T5:** Architecture more important than requirements to select components.
- **Thesis T6:** Tendency to increase level of control over vendor whenever possible.

See [Torchiano04].

To be extended with larger Norwegian survey by NTNU and Simula, later repeated in Germany and Italy.
From 50 software “laws” [Endres03]:

• L1, Glass: Requirement deficiencies are the prime cause of project failures.
• L5, Curtis: Good designs require deep application domain knowledge.
• L12, Corbató: Productivity and reliability depend on the length of a program’s text, independent of language level used.
• L16, Conway: A system reflects the organizational structure that built it.
• L23, Weinberg: A developer is unsuited to test his or her code.
• L27, Lehman-1: A system that is used will be changed.
More from 50 software “laws”:

• L30, Basili-Möller: Smaller changes have a higher error density than large ones.
• L36, Brooks: Adding manpower to a late project makes it later.
• L45, Moore: The price/performance of processors is halved every 18 month.
• L47, Cooper: Wireless bandwidth doubles every 2.5 years.
• L49, Metcalfe: The value of a network increases with the square of its users.
Some of the 25 hypotheses, also from [Endres03]:

- H2, Booch-2: Object-oriented designs reduce errors and encourage reuse.
- H5, Dahl-Goldberg: Object-oriented programming reduce errors and encourage reuse.
- H9, Mays: Error prevention is better than error removal.
- H16, Wilde: Object-oriented programs are difficult to maintain.
- H25, Basili-Rombach: Measurements require both goals and models.
Conclusion (1)

• Best practices: depend on context, so must know more about that relation!!

• Need feedbacks from and cooperation with industry to be helpful – our “laboratory”! Compensation to industry for participation.

• Seek data relevance to actual goal/hypothesis! But unused data worse than no data?

• ESE: promising, but hard.

• High ESE / SPI activity in Norway since 1997.

• Much international cooperation.
Conclusion (2)

• **Higher R&D spending in Norway?:** still 1.7% of GNP, in spite of parliamentary promises from April 2000 on reaching OECD-level (2.25%) in 4 years.

• **Large and growing ICT sector in Norway,** sparse funds for R&D. Too much at the bottom (“hw/tele”) and at the top (“applications”) – need more in the middle (“software engineering” and likewise).

• Ex. NFR is using 100 MNOK per year on basic software research – as much as the three best Norwegian football players earn per year!

• Ex. Kreftrregisteret for medicine, SSB for general data, Air traffic authority, Water research institute etc. – what public “bureau” is for (empirical) software engineering?

• **Chinese proverb:**
  – invest for one year - plant rice,
  – invest for ten years – plant a tree,
  – invest for 100 years – educate people.
Appendix 1: Some useful web addresses

- Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern: www.iese.fhg.de
- Fraunhofer Center for Experimental Software Engineering, Univ. Maryland (FC-MD): http://fc-md.umd.edu
- Software engineering group (SU) at IDI, NTNU: www.idi.ntnu.no/grupper/su/
- Industrial software engineering group (ISU) at UiO: www.ifi.uio.no/~isu/
- SINTEF Telecom and Informatics: www.sintef.no
- Simula Research Laboratory, at IT-Fornebu from 2001: www.simula.no (see under “research” and then “Software Engineering”)
- SPIKE project: www.abelia-innovasjon.no/spike/ (official web cite), www.idi.ntnu.no/grupper/su/spike.html (NTNU one).
Appendix 2: Literature list (1)


Literature list (2)


Appendix 3: SU group at NTNU

IDI’s software engineering (SU) group:

• Five faculty members: Reidar Conradi, Tor Stålhane, Letizia Jaccheri, Monica Divitini, Alf Inge Wang.
• One lecturer: MSc Per Holager.
• 15 active PhD-students, with 6 new in both 2002 and 2003: common core curriculum in empirical research methods.
• 35 MSc-cand. per year.
• Research-based education: students participate in projects, project results are used in courses.

• A dozen R&D projects, basic and industrial, in all our research fields – industry is our lab.
• Half of our papers are based on empirical research, and 25% are written with international co-authors.
Research fields of SU group (1)

- **Software Quality**: reliability and safety, software process improvement, process modelling
- **Software Architecture**: reuse and COTS, patterns, versioning
- **Co-operative Work**: learning, awareness, mobile technology, project work

In all this:
- **Empirical methods and studies** in industry and among students, experience bases.
- Software engineering **education**: partly project-based.
- **Tight cooperation** with Simula Research Laboratory/UiO and SINTEF, 15-20 active companies, Telenor R&D, Abelia/IKT-Norge etc.
Research fields of the SU group (2)

- Software quality
- Patterns, COTS, Evolution, SCM
- Reliability, safety
- Software architecture
- SPI, learning organisations
- Co-operative work
- Distributed Software Eng.
- Mobile technology
- Software Engineering Education
SU research projects, part 1

Supported by NFR:


3. **MOWAHS**, 2001-04: mobile technologies, Carl-Fredrik Sørensen (Conradi); with DB group.

4. **INCO**, 2001-04: incr. and comp.-based development, Parastoo Mohaghegi at Ericsson (Conradi); with Simula/UiO.


7. **SPIKE**, 2003-05: industrial sw process improvement, Finn Olav Bjørnson (Conradi); with Simula/UiO, SINTEF, Abelia, and 10 companies - successor of SPIQ and PROFIT. Also INTER-PROFIT in 2001-03.

8. **FAMILIER**, 2003-06: Product families, Magne Syrstad (Conradi), mainly with IKT-Norge but some IDI-support.

SU research projects, part 2

IDI/NTNU-supported:


4. **Component-based development**, 2002-05: Jingyue Li (Conradi).

5. **Creative methods in Education**, 2003-4 (NTNU): novel educational practices, no PhDs, Jaccheri at IDI w/ other dept.s.

Supported from other sources:

