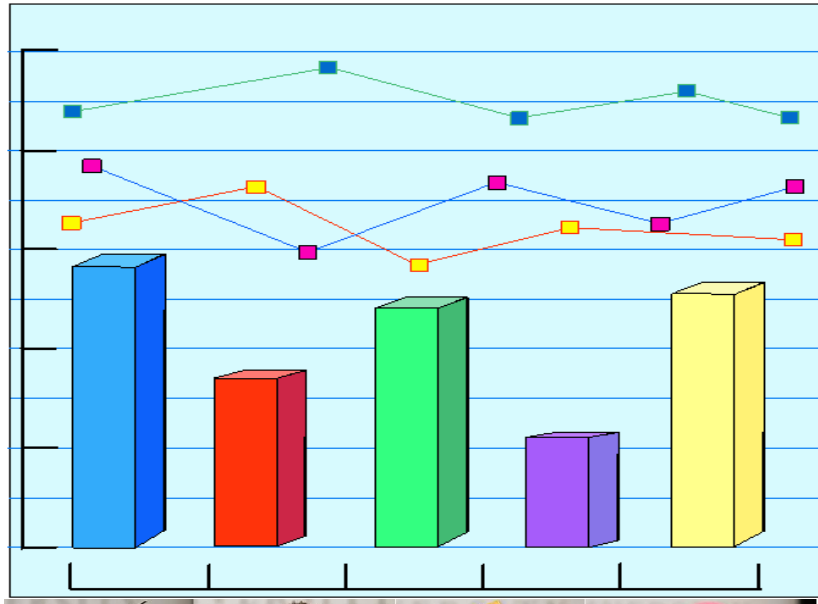
[simula . research laboratory]

How to Impact Software Engineering Practice Through Empirical Research



EASE 2009

Magne Jørgensen
Simula Research Laboratory

Empirical Software Engineering (ESE)

- Empirical software engineering is a field of research that emphasizes the use of empirical studies of all kinds to accumulate knowledge. (wikipedia)
- Basic assumption:
  - The use of empirical studies is an efficient way to evaluate knowledge or technology, to add knowledge and to guide technology innovations in software engineering.
- No assumption of ESE as “the only way” or “the perfect way”, only that it sometimes is worthwhile compared to its alternatives, such as guru-based, marked leader-based, fashion-based choice of practices, ...)

Why Do We Need Empirical Studies?
"I see it when I believe it"

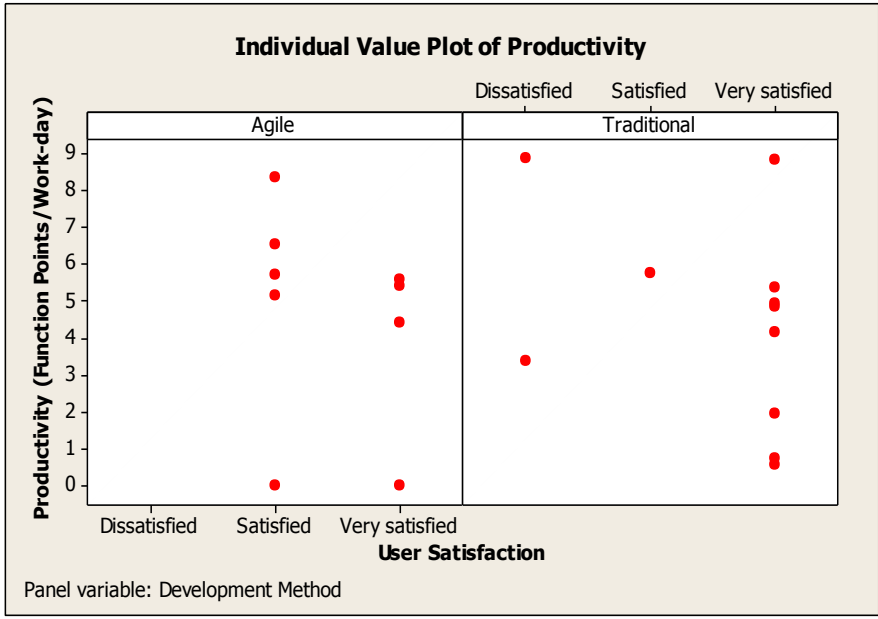
- **Research Question:** Do children get more hyperactive when given sugar?
- **Common belief**: Yes.
- **Answer**: Probably, no. At least 12 double blind, randomized controlled trials find no effect (see Vreeman & Carrol, Festive medical myths, British Medical Journal, 337:1288-1289, 2008) .
- **Why most believe it:** *"When parents think their children have been given a drink containing sugar (even if it is really sugar-free), they rate their children's behaviour as more hyperactive. The differences in the children's behaviour were all in the parents' minds*."

Study on “I see it when I believe it” in SE (Are Agile Methods Better?)

- **Participants**: 50 developers from a Polish company.
- **Strong belief in agile**: Before the study I collected their believes about agile methods.
  - 84% believed agile methods led to higher productivity (only 6% believed same or lower productivity), and 66% believed it led to more user satisfaction (only 8% same or lower).
- **Design of study**:
  - Generation of 10 project data sets (see example next page) with the triples: Development method (agile or traditional), Productivity (FP per work-day), and, User satisfaction (dissatisfied, satisfied, very satisfied).
  - All values were RANDOMLY generated.
  - A control gave that there were no (statistically) significant differences in the average values. The average values were slightly in favor of the traditional (non-agile) methods.
  - Each developer was randomly allocated to one of the data sets and asked to interpret it – **based on the measured data alone**.

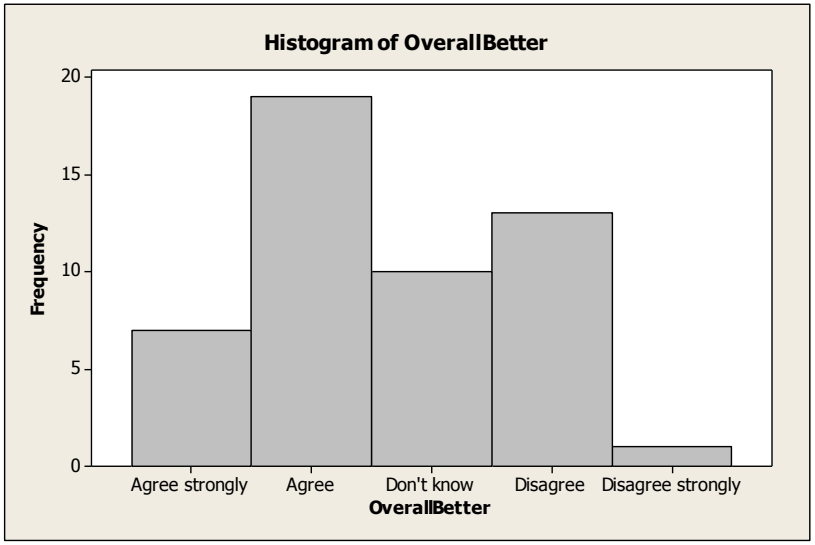
Study on “I see it when I believe it” in SE (Are Agile Methods Better?)

- **Instruction**:
  - *“Assume that this [the data set] is the only you know about the use of agile and traditional development methods in this company and that you are asked to interpret the data. The organization would like to know what the data shows related to whether they have benefited from use of agile methods or not.”*
- **Results:**
  - The interpretations of the data set related to productivity and user satisfaction as isolated variables were reasonable unbiased.
  - The interesting finding was related to the more complex interpretation of the **combined** (total) effect on productivity and user satisfaction.



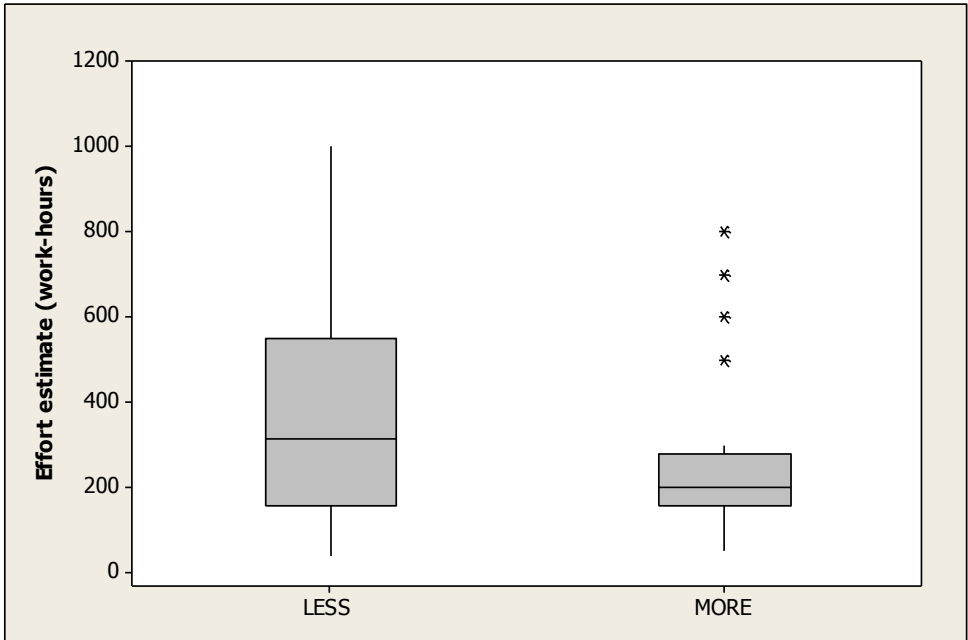
Study on “I see it when I believe it” in SE (Are Agile Methods Better?)

- **Question**: How much do you agree in: *“Use of agile methods has caused a better performance when looking at the combination of productivity and user satisfaction.”*
- **Result**: Strong bias in favor of agile methods (see figure).
  - The agreement in the claim depended on their previous belief in agile methods.
  - Previous belief: Agile methods are better (wrt productivity and user satisfaction) ➔ 20 of 32 agreed
  - Previous belief: Agile methods are not better (on at least one aspect) ➔ 1 of 7 agreed
  - Previous belief: Neutral ➔ neutral answers
- The real-life bias is probably much stronger:
  - Lack of objective measurement. More bias in favor of the preferred method.
  - More variables of importance, i.e., more complex interpretation and more space for wishful interpretation.



We Need ESE to Challenge "Obvious" Relationships: More Risk Analysis Make You More Realistic

- **Participants**: 50 developers from a Polish company (the same as in the previous study) randomly divided into two groups.
- **Group LESS**: Identify the most important risk, then estimate the effort.
- **Group MORE**: Think back on problems you have had in similar projects, identify the most important risk factors of the current project, analyze each risk factor with respect to probability and severity, then estimate effort.
- Actual effort: median of ca. 700 work-hours
- Those in Group MORE had:
  - Lower median effort estimates (200 vs 316 work-hours)
  - Higher mean confidence in low (<25%) estimation error (80% vs 70%).
- Results replicated in three other exp.



We Need ESE to ...

- Replace biased believes and opinions with evidence
- Replace non-representative experience with evidence representing a well-defined population
- Replace our tendency of seeing patterns where there are none (I see what I believe) with knowledge based on proper analysis methods
- Challenge existing practices
- Generate knowledge that cannot be derived from experience alone

... and many, many more good reasons.

The main problem of impact is, however, hardly to convince software practitioners about the above benefits of ESE.

Ok, ESE is needed, but ....

- Are we able to convince the software industry to use the **results** of ESE?
- What has been the role of ESE so far?
  - The IMPACT project (SE researchers) (www.sigsoft.org/impact/) claims: *"Software engineering research has significantly affected software engineering practice."*
  - It says, however, not much about the role of empirical studies.
- Although there are success stories of ESE, the software industry are, as far as I can see, currently not strongly impacted by it.
  - Why is this so?
  - What can we do to get more impact?

The Ivory Tower of ESE as Perceived by Software Professionals?

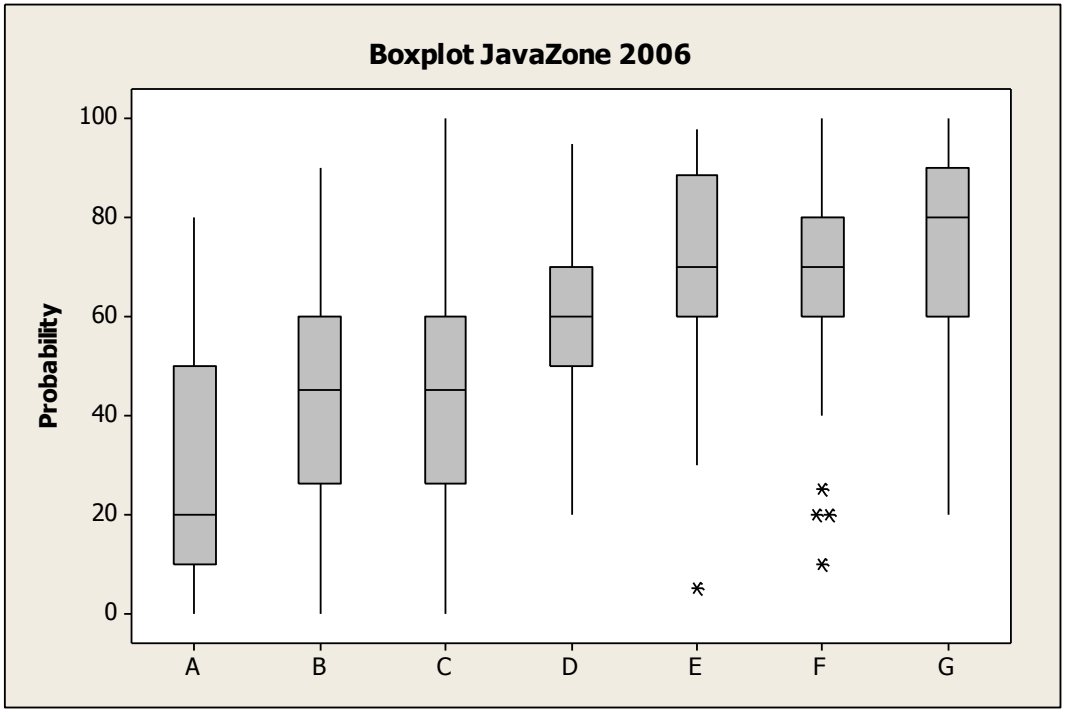
Is ESE Valid and Useful, but not Sufficiently Convincing? An Empirical Study at JavaZone 2006 (and 2007)

Context: Assume that a test course provider claims: ”*The course will lead to substantial increase in test efficiency and quality for most participants.*”

How likely do you think this claim is true, given *[reduced explanation]*:
A: No other information
B: Supporting claims from reference clients
C: Supporting study conducted by the course provider
D: Convincing explanation (but no empirical evidence)
E: Supporting experience from a colleague (It helped him)
F: Supporting scientific study completed at a renowned university
G: Own experience (It helped me)

Is ESE Valid and Useful, but not Sufficiently Convincing? An Empirical Study at JavaZone 2006

A: No other information
B: Support from reference clients
C: Supporting study conducted by the course provider
D: Convincing explanation (but no empirical evidence)
E: Supporting experience from a colleague (It helped him)
F: Supporting scientific study completed at a renowned university
G: Own experience (It helped me)



Is ESE Valid and Useful, but not Sufficiently Convincing? An Empirical Study at JavaZone 2006

- The results are probably “best case” results for ESE, i.e., the opinions are related to the **processes**, not their outcome.
- When the outcome of our own subjective judgment diverges from the outcome of analyses (scientific studies), we tend to trust the outcome of the subjective judgment more than that of the analyses; even when one tend to trust analytical **processes** more than those involved in subjective judgment (Hammond et al 1987).

An Example of Something that Has Had Impact: The Agile Manifesto

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

(agilemanifesto.org)

How Did Agile Methods Become a Leading Development Method?

It is easy to document several, from an academic point of view, limitations of elements of Agile methods:

- It is based on vague descriptions and poorly defined processes.
  - What is for example the meaning of the Manifesto's "*individuals and interactions over processes and tools*"?
- It contains nothing fundamentally new and most of its elements are common sense (and included in several existing methods).
  - Iterative and incremental development principles have been around since the 1950s. People describing iterative and incremental methods before 1990 include: Tom Gilb, Barry Boehm and Vic Basili.
- It attacks a straw man (naïve waterfall) that hardly exists and nobody would defend.
  - Who would claim that comprehensive documentation is more important than working software?

How Did Agile Become a Leading Development Method? (The Fashion Theory)

- **Rhetoric and Myth in Management Fashion** (Alfred Kieser, *Organization* 1997; 4; p 49-74) points out that the following factors are essential for success with a new method:

- Present one key principle that, according to the gurus, has been neglected in previous methods, e.g., the lack of frequent feedback in the naïve waterfall model.
- Describe how the old methods are bound to fail if not following the new method, e.g., how the old methods leads to systems that does not have the functionality that the clients need.
- Link the new method with highly treasured values, such as communication, individuals, flexibility and user value.

How Did Agile Become a Leading Development Method? (The Fashion Theory)

- Present stories about great successes when using the method. Go to practitioners' conferences and present these success stories.
- Avoid by all means the impression that the method has been created at a university or is based on academic research. Emphasize that the method is based on experienced professionals knowledge.
- Present the pioneers as exceptional professionals with long experience. Give them guru status.

How Could Agile Become a Leading Development Method?

- Base the messages on a mixture of simplicity and ambiguity. Use this to demonstrate the superiority of the new principles, e.g., “collaboration is better than contract negotiation”, and to demonstrate that the principles are strongly linked to common sense.
- Point out that the method may be hard to implement. Failures are thus explainable by poor implementation.
- Provide easy readable books with no academic jargon and direct speech.

How Could Agile Become a Leading Development Method?

- Time the introduction of the new method well.
  - Every new generation of software professionals need their “own” methods to separate themselves from the others and be the most knowledgeable.
  - The timing of and need for new development methods follows many of the same principles as those for cloth fashion.
  - This means that the success of a method (many followers) is also its path to destruction when it follows fashion-principles.
- Now and then, couple principles to science.
  - Low quality studies and strongly biased interpretations are no problem, since nobody will check the sources.

ELASTIC Development

- Together with some of the most experienced developers in the world we at Simula have developed a new best-practice method: ELASTIC.
- More and more experienced developers found that they SOMETIMES needed more design, more planning and more documentation. They could not anymore stand and look at failed projects due to religious beliefs in ONE method. We need to be flexible. We need to be more elastic!
- Manifesto of ELASTIC:
  - **Software projects vary, so should their development methods**

ELASTIC Development

- **Key principle**: ELASTIC has a phase where the developers together with their clients (based on project characteristics such as expected requirement stability, client maturity and technical complexity) agree on process elements. This tailoring method is unique and not part of any other development method.
- There is nothing fundamentally wrong with the traditional methods, such as agile, scrum, RUP and waterfall.
- They lack however:
  - The flexibility to deal with today's variation in clients and types of projects
  - Support on how to tailor a development method.

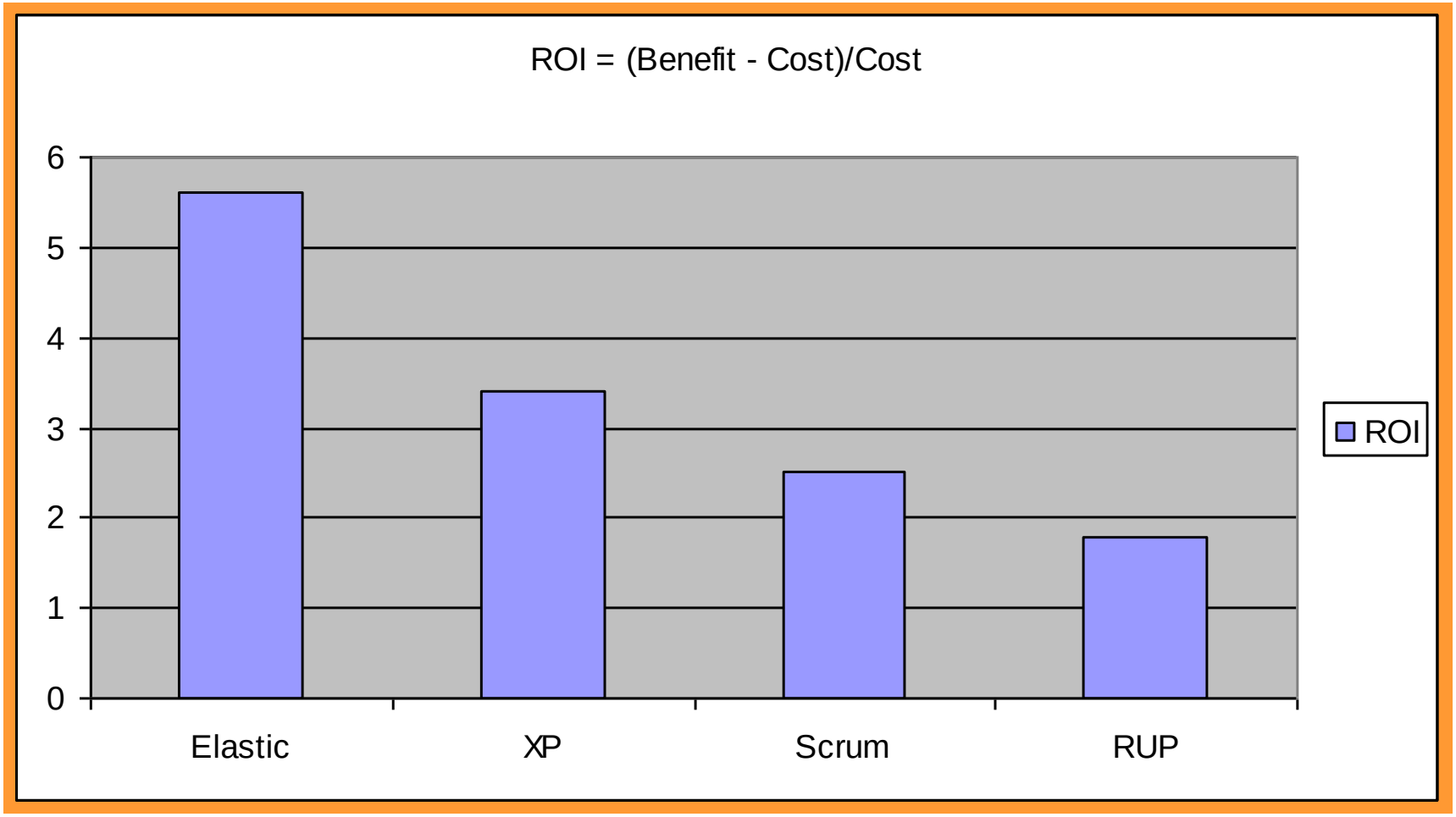
ELASTIC development

- The main values of ELASTIC are Communicate, Analyze, Reflect and Educate in close collaboration with the client (the CARE values).
- ELASTIC has so far been a great success!
- "*We have so many times been disappointed by the lack of professionalism and low ability of adapting the development method to our needs and maturity levels. Most software developers seem to be more concerned about their own religious belief in a method than creating value for the client. ELASTIC development takes us – the client – seriously. We will never again choose a software provider that does not follow ELASTIC development.*" Stein Mathisen, CEO Norwegian Hydropower.

ELASTIC development

- The client went from 50% to 0% failed projects and 200% in profitability by selecting a software providers following the ELASTIC method.
- Other studies also show great benefit from use of the ELASTIC method. The Johnson Group have summarized a study on more than 200 projects and found that ELASTIC gives the highest ROI (return on investment).

ELASTIC development



Can We Expect ESE to Have Strong Impact in Software Engineering?

- Probably not, unless the SE “culture” get more evidence-based!
- SE researchers don’t like and are not good at playing all the impact games (the rethoric) necessary to gain impact?
  - We have however our own games and rethoric, to please reviewers, research council, ... ;-)
- There may, however, be ways where we can impact software practice more than we currently do without ethical and research professionalism problems.

Warning:

- Now comes (as usual) the weakest part of the presentation, i.e., the part dealing with what I promised in the title (How to ....)
- Expect no brilliant ideas or recommendations based on strong evidence.

Suggested Ways to Impact Software Practice through ESE

- Improve the acceptance of Evidence-based SE (EBSE)
  - Teach software professionals EBSE (Training in formulating decidable questions, collection of valid evidence (including experience-based evidence), evaluation of strength of evidence and synthesis of evidence.)
  - Promote evidence-based principles at conferences
  - Train software professionals in completion of empirical studies (This is perhaps where ESE-elements has had most impact on practice, e.g. processes of measurement-based software improvement.)
  - Write SE books that are evidence-based
  - Demonstrate why we need ESE
- Like medicine, we should try to get to a stage where the professionals only accept evidence-based principles and methods.
  - It's a looooong way to go, and there may be inherent problems that stop us from reaching the stage where medicine currently is.

Suggested Ways to Impact Software Practice through ESE

- Select research topics where impact is more likely (relevance)
  - Increase the emphasis on relevance. Robert Glass in IEEE Software March/April, 2009 recommends that all studies should go through an "applicability check".
  - Do not conduct research where there are no opportunity to impact. Timing may be important.
- Include more research with high potential of impact. (Think bigger!)
- Emphasis money saving potential. A rough guideline by innovation advisors is that an idea should be able to save at least 10 times its implementation cost to be convincing for investors.

There is in my opinion far too much ESE research of low industry relevance! It's sometimes like doing research on typewriter improvement.

Suggested Ways to Impact Software Practice through ESE

- Improve the ESE methods
  - Higher quality studies.
    - I think I have reviewed more than 50 studies that show that their own estimation model is better than the other models. Most of these empirical evaluations have in my opinion been poorly designed.
  - More convincing studies.
    - Inclusion of real-life success stories, less use of students and small scale systems.
    - Forthcoming study (IEEE TSE, Jørgensen & Grimstad) compares estimation biases in laboratory settings and real-life settings. The main finding is that the biases are typically much larger in laboratory settings. We need real-life settings to evaluate effect sizes!
  - etc. (many papers on this)

Suggested Ways to Impact Software Practice through ESE

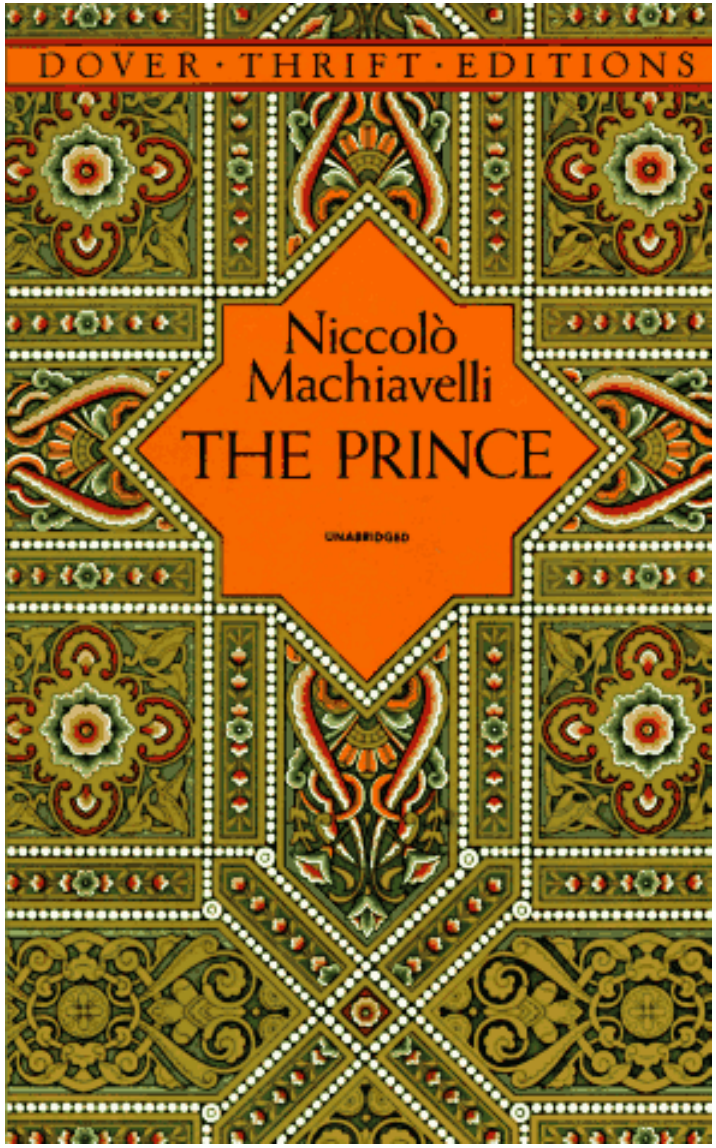
- Conduct ESE research in collaboration with the software industry.
  - Let them tell convincing success stories. Nothing beats success stories that can be linked to your ESE-results
  - Mean values and statistical significance may convince scientists, seldom software professionals.
  - Make win-win situations out of research results (see picture)

Suggested Ways to Impact Software Practice through ESE

- Better packaging/wrapping of ESE results
  - Tools
  - Processes/methods
  - Standards/certificates
  - Courses
- The type of package obviously depends on the content.
  - A innovative tool where the ESE contribution is the evaluation of it may be easier to package than ESE-based relationships. Even then, however, the packaging of the ESE results are of importance.

Suggested Ways to Impact Software Practice through ESE

- Better transfer of ESE results
  - Publish in practitioners' magazines
  - Write books without academic jargon
  - Be were practitioners meet
  - Package the ESE results as "experience" and "success stories"
  - Educate journalists to write about ESE (accept that good SE researchers are not necessarily good communicators)
  - Talk the "impact language" of successful gurus?
- Software practitioners are typically not even aware of our studies. If they find them, the studies are in a language they do not understand. This slows (or even inhibit) the impact.

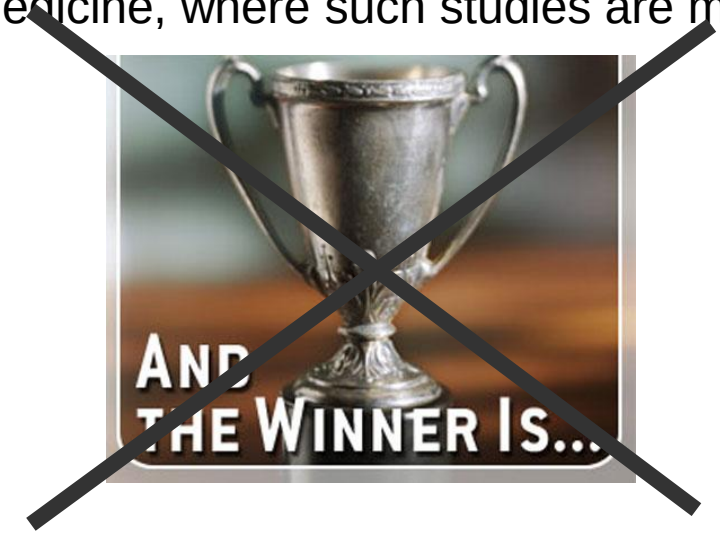


Suggested Ways to Impact Software Practice through ESE

- Better timing of ESE studies.
  - We are typically lagging behind.
  - When a method already is established, it is difficult to have an impact.
  - Being able to impact sometimes means that the ESE-based knowledge has to be there (and be known) when (or before) new technology emerges.
  - Agile will probably be replaced (as the leading method) with a new methods in 3-4 years. How will (and can) ESE impact the new method to be more evidence-based – and more efficient?
    - Providing input to the method gurus?
    - Examining emerging methods based on empirical knowledge?
    - Example: If we had collaborated with the Planning Poker guru (Mike Cohn) when he invented it, we could share with him relevant results on the Delphi-method and on group dynamics.

Suggested Ways to Impact Software Practice through ESE

- Focus on creation of evidence-based **principles**. Avoid “Is Method A better than Method B”-studies, where the methods consist of many (ill-defined) elements.
  - This “reductionism” may sound like a paradox, since the software industry wants exactly that kind of studies.
  - However, such studies do in my experience seldom produce results that are convincing (study the effect of own methods), seldom produce insight in cause-effects, seldom have the timing to enable impact (studies of already established practices).
  - We are different from medicine, where such studies are more meaningful.



Principles has Impacted Forecasting Practice

Examples of an evidence-based principle:

7.1 Keep forecasting methods simple.

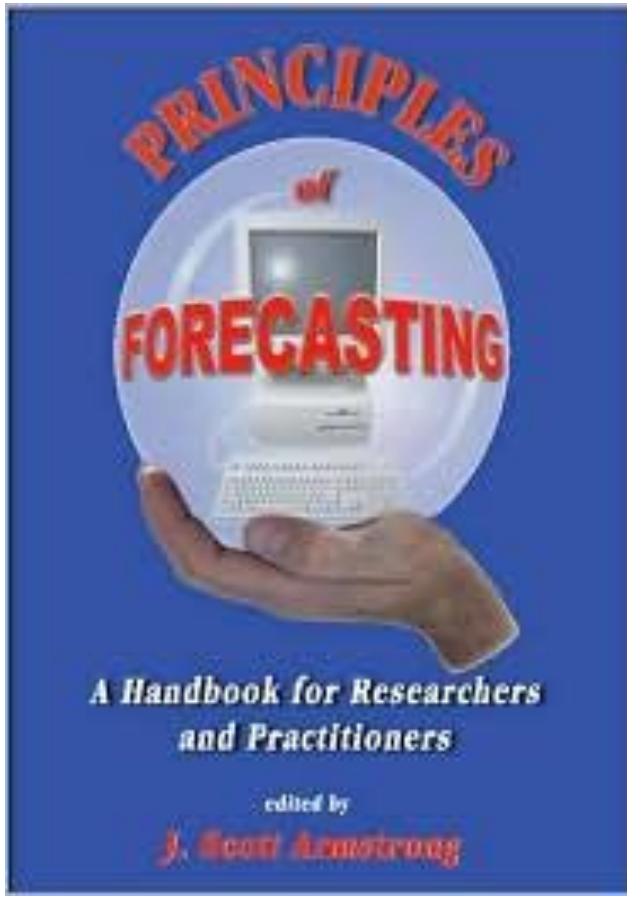
Description: Complex methods may include errors that propagate through the system or mistakes that are difficult to detect. Select simple methods initially (Principle 6.6). Then use Occam's Razor; that is, use simple procedures unless you can clearly demonstrate that you must add complexity.

Purpose: To improve the accuracy and use of forecasts.

Conditions: Simple methods are important when many people participate in the forecasting process and when the users want to know how the forecasts are made. They are also important when uncertainty is high and few data are available.

Strength of evidence: Strong empirical evidence. Many analysts find this principle to be counterintuitive.

Source of evidence: This principle is based on evidence reviewed by Allen and Fildes (2001), Armstrong (1985), Duncan, Gorr and Szczypula (2001), and Wittink and Bergestuen (2001).



There have been 1,069,597 visits to this website www.forecasting.com since February 14, 1998.

What I Wanted To Tell You

- Empirical software engineering is clearly needed, but does hardly play a major role in current software practice.
- The receipts of more impact on SE practice are (in principle) simple. The main problem is currently that ESE researchers are not good at playing the impact games.
- If we don’t want to play the impact games, we should aim at more acceptance for evidence-based principles, i.e., we should try to change the rules of the game. Demonstrations of the limitations of the principles currently in use is one element of this.
- Regardless of main strategy, we can (and should) influence software engineering practice through empirical research much more than we currently do.

It’s up to us!
Industry impact seldom happens
by publishing in academic journals.