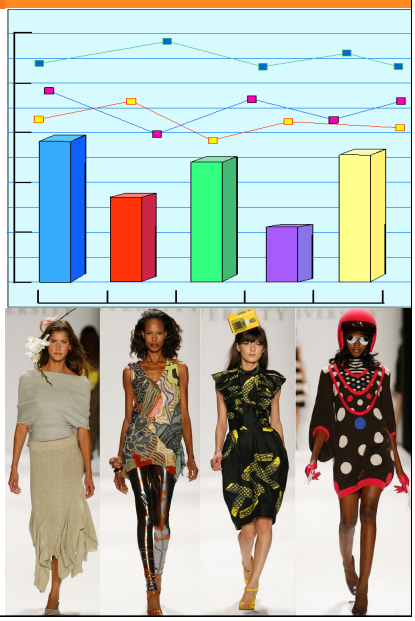
[ simula . research laboratory ]

From fashion and opinion-based to:

# Evidence-Based Software and Knowledge Engineering

ICKE-2011

Magne Jørgensen
Simula Research Laboratory



## Empirical evidence

## Evidence-Based Software Engineering

1. Convert a relevant problem or need for information into an answerable question.
2. Search the literature or relevant experience for the best available evidence to answer the question.
3. Critically appraise the evidence for its validity, impact, and applicability.
4. Integrate the appraised evidence with practical experience and the client's values and circumstances to make decisions about practice.
5. Evaluate performance in comparison with previous performance and seek ways to improve it.



## We Need EBSE to ...

- Replace biased believes and opinions with evidence in the software industry
- Replace non-representative experience with evidence representing a well-defined population
- Replace our tendency of seeing patterns where there are none 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.

## Why evidence-based practice?

- **Research Question:** Do children get more hyperactive when given sugar?
- **Common belief**: Yes.
- **Evidence-based answer**: 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*."

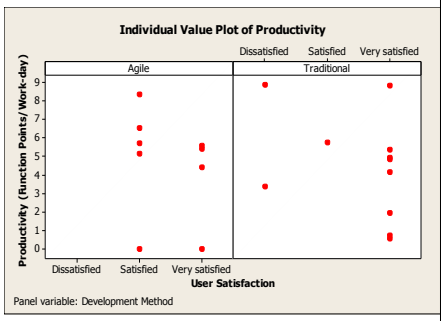


## Agile Methods in Software Development

- **Participants**: Professional software developers.
- **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.
  - Each developer was randomly allocated to one of the data sets and asked to interpret it – **based on the measured data alone**.

## Agile Methods in Software Development

*"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."*



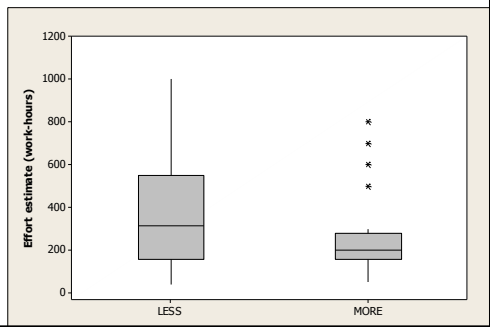



## "I see it when I believe it" vs "I believe it when I see it"

- **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.
  - 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.

## Challenge "Obvious" Relationships: More Risk Analysis Make You More Realistic

- **Participants**: Professional software developers 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.






## Is EBSE Valid and Useful, but not Sufficiently Convincing?

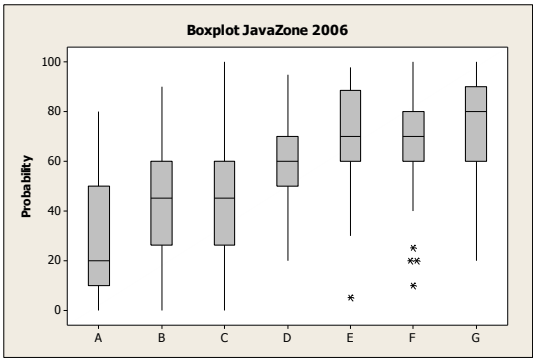
### An Empirical Study at JavaZone 2006 (and 2007)

**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 EBSE 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)






## Most of these methods have been "fashion"

Waterfall model, sashimi model, agile development, rapid application development (RAD), unified process (UP), lean development, modified waterfall model, spiral model development, iterative and incremental development, evolutionary development (EVO), feature driven development (FDD), design to cost, 4 cycle of control (4CC) framework, design to tools, re-used based development, rapid prototyping, timebox development, joint application development (JAD), adaptive software development, dynamic systems development method (DSDM), extreme programming (XP), pragmatic programming, scrum, test driven development (TDD), model-driven development, agile unified process, behavior driven development, code and fix, design driven development, V-model-based development, solution delivery, cleanroom development , ….
**+ 1000s of company specific methods.**

## Current fashion: Agile development
## 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 the most Fashionable 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 to create a new fashion

- 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 to create a new fashion

- 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 to create a new fashion

- 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 to create a new fashion

- 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.

[ simula . research laboratory ]

# How to transfer from fashion and opinion to evidence-based software and knowledge engineering?

## Acceptance

- Improve the acceptance of the importance of evidence
  - Teach software professionals and university students
  - Promote evidence-based principles at conferences
  - Train software professionals in completion of empirical studies (This is perhaps where EBSE-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 EBSE
- 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.


[ simula . research laboratory ]

## Relevance

- 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 far too much research of low industry relevance! The research is sometimes similar to doing research on typewriter improvement.**



## Quality

- Improve the quality of the evidence
  - 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!

## Success stories

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




## Transfer

- Better transfer of research (evidence-based) results
  - Publish in practitioners' magazines
  - Write books without academic jargon
  - Be were practitioners meet
  - Package the EBSE results as "experience" and "success stories"
  - Educate journalists to write about EBSE (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.

## Timing

- Better timing of research studies (presentation of evidence)
  - 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 empirically 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) research 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 I had collaborated with the Planning Poker guru (Mike Cohn) when he invented this estimation method, we could share with him relevant results on the Delphi-method and on group dynamics.




## Development of principles

- 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.

## Example:
## Principles has Impacted Forecasting Practice

**Examples of an evidence-based principle:**

**7.1 Keep forecasting methods simple.**

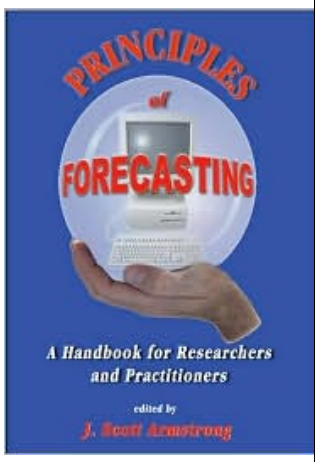
**Description:** Complex methods may include errors that propagate through the system or mistakes that are difficult to detect. Select simple methods initially (Principle 6.6). Then use Occam's Razor; that is, use simple procedures unless you can clearly demonstrate that you must add complexity.

**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.

[ simula . research laboratory ]

[ simula . research laboratory ]

# Teaching Evidence-Based Software Enginering to University Students and Software Engineers

## Learning goal:

- *"... to learn to practice evidence-based software engineering. This means the ability to identify, evaluate and apply valid and relevant research results and practice-related experience as the basis for judgments and decisions in software development."*



## Repitition: Main Steps of Evidence-Based Software Engineering (EBSE)

1. Convert a relevant problem or need for information into an answerable question.
2. Search the literature (and practice) for the best available evidence to answer the question.
3. Critically appraise the evidence for its validity, impact, and applicability.
4. Integrate the appraised evidence with practical experience and the client's values and circumstances to make decisions about practice.
5. Evaluate performance in comparison with previous performance and seek ways to improve it.

## Students

- University students:
  - MSc students at University of Oslo (Norway), Hedmark University College (Norway), Kathmandu University (Nepal)
- Software engineers:
  - Software professionals in Norway, Australia, Costa Rica and Vietnam



## Exam (project delivery)

In order to pass, a university student must produce a project report, which includes:

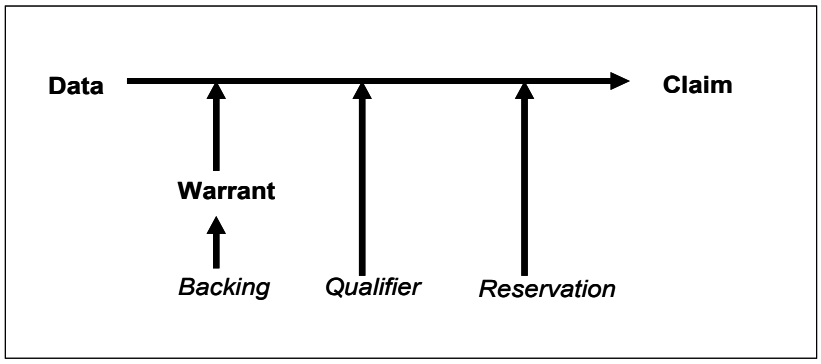
- An answerable software engineering question/problem.
- An extensive search for relevant research results and practice-related experience.
- Evaluation of the relevance and validity of the results, opinions, and argumentation found in the information sources.
- A synthesis of the available evidence to support a conclusion related to the initial question.
  - The conclusion may be that it is not possible to form a definitive opinion, or that the evidence in favor of a particular decision or answer is weak.
- Design of an empirical study that would extent the evidence on the chosen topic

## Lectures are exercise-based

- Exercises in problem formulation, systematic collection of evidence, critical evaluation of evidence, synthesis of evidence, design of empirical studies.
- Typical exercise in critical evaluation
  - Example: Evaluation of ”Aim, fire”-paper by Kent Beck




## Toulmin’s Model of Argumentation



## Lessons learned (1)

- Most students and software engineering like using the EBSE and find it useful.
- Most students produce good work.
- The students had never before had any teaching in how to examine scientific studies critically or how to systematically evaluate the arguments presented in course textbooks and computer magazines.
  - There were used to reproduce, not to critizise.
- The lack of previous skill meant that they easily
  - Reproduced instead of evaluated knowledge
  - Evaluated how much they agreed with the conclusion instead of the validity of the argumentation.



## Lessons learned (2)

- The students were much more critical towards what they read (and heard from gurus) towards the end of the course.
- The students now had the means to decompose and evaluate argumentations!
- Students must be urged to use sources other than the web.
  - Initially, all studets relied too much on an unsystematic search of the internet.
- Less scientific evidence in EBSE, compared to Evidence-Based Medicine (EBM).
  - This leads to more emphasis on collection and evaluation of practice-based experience, less on evaluation of scientific studies compared to EBM.

## Lessons learned (3)

- The ability to evaluate studies improves a lot with teaching about:
  - The importance of random treatment
  - The importance of representative samples
  - Valid and invalid generalization
- How to collect experience based evidence is not part of evidence-based medicine, but the principles are very much the same.
- Toulmin's model of argumentation is a useful means to teach the students to evaluate argumentations.



## Teaching EBSE: Lessons learned summarized

- Universities should have a stronger focus on how to acquire new knowledge and evaluate argumentations.
- Teaching EBSE to university students (and software professionals) may be an important means for software engineering to become a more mature discipline with more resistance towards "hype".
- There are, however, not much relevant evidence available and the teaching should emphasize how to collect practice-based evidence, as well.

# What I Wanted To Tell You

- Evidence-based software engineering is clearly needed in the software industry. We should stop relying so much on fashion and opinions in software and knowledge engineering.
- We, as researchers, have a role to play in making the software industry more evidence-based, e.g., in producing and spreading useful evidence. Currently we are not taking this responsibility very seriously.
- There should be university and industry courses on evidence-based software engineering, including proper collection and evaluation of practice-based evidence.
  - Teaching people how to formulate problems, collect evidence, evaluate evidence and summarize evidence is a very robust and useful knowledge and skill.
- Much of the evidence may benefit from being presented as context-dependent principles. Hopefully, the first evidence-based text-book on software engineering (and knowledge engineering?) will be written.

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

[ simula . research laboratory ]