What makes software development projects successful, and what makes them fail? (and how to find out)

Summer School, August 2019

Magne Jørgensen
SimulaMet & OsloMet

What do we get in return from our huge investments in digitalization?

You can see the computer age everywhere but in the productivity statistics.
Robert Solow (1987)

Is this (still) true?

ELSEVIER
Information Economics and Policy
Volume 25, Issue 3, September 2013, Pages 109-125

ICT and productivity: conclusions from the empirical literature

M. Cardona [a], T. Kretschmer [a, b], T. Strobel [b]

⊞ Show more

https://doi.org/10.1016/j.infoecopol.2012.12.002

Get rights and content

"… 10% increase in ICT investment leads to a 0.6% (points) increase in growth"
(➔ around half of the current (very low) increase in productivity is due to ICT-investments!)
"… the growth impact of ICT has grown over time."

Many studies show positive effect

Research method quiz: Is there anything strange with Fig 6?

Information Economics and Policy
Volume 38, March 2017, Pages 38-54

The productivity paradox: A meta-analysis

Petr Polák

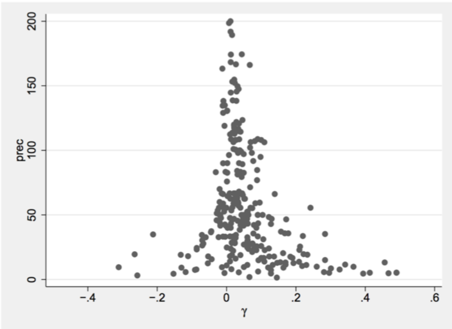


Figure 5: Funnel graph – before 2002

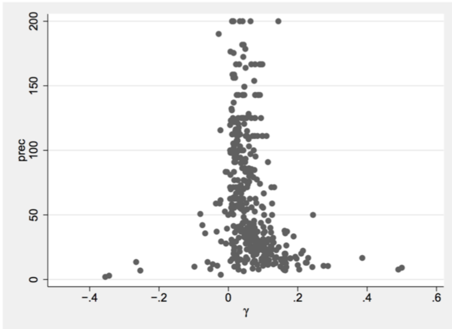
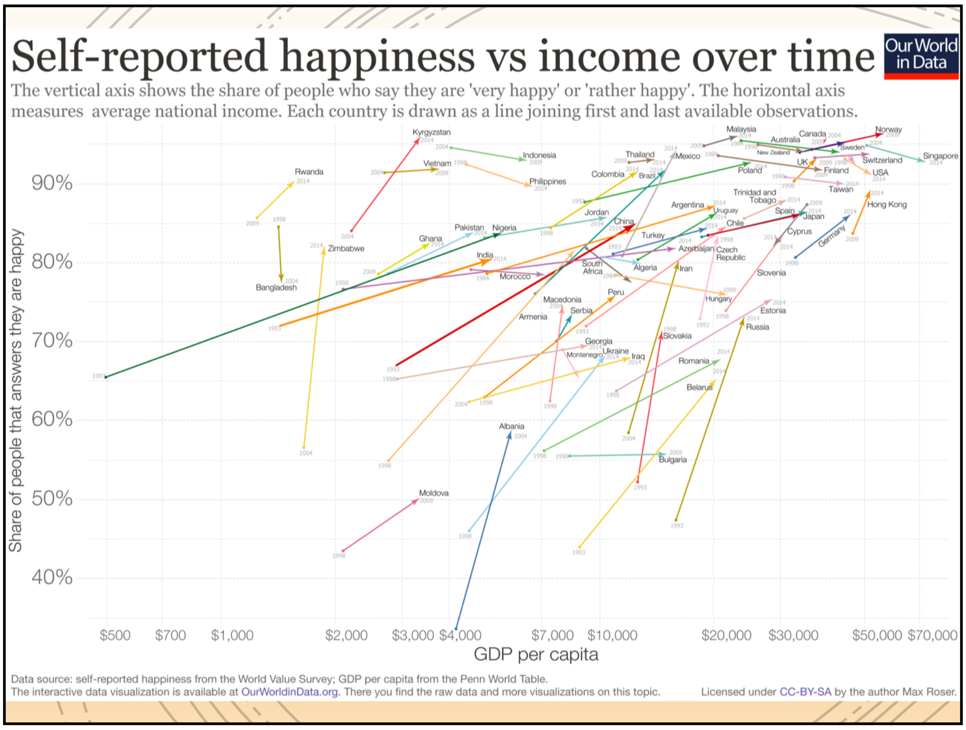


Figure 6: Funnel graph – after 2002

There is more to productivity increase than productivity increase



So, productivity through digitalization is important – How well do we succeed with it?

Are we this bad?
(or are just the study bad?)

THE STANDISH GROUP

The most frequently reported results on software projects (from 1994, repeated bi-yearly) found that:

- 31% of all projects are cancelled before they complete
- Average cost overrun of 189%

(page 13 of their 1994-report): "*We then called and mailed a number of confidential surveys to a random sample of top IT executives, asking them to share failure stories.*"

Another frequently referred study:
The consequences of software failures (2017)

LO$$E$ FROM SOFTWARE FAILURES (USD)

1,715,430,778,504

ONETRILLIONSEVENHUNDREDFIFTEENBILLIONFOURHUNDREDTHIRTYMILLIONSEVENHUNDREDSEVENTY-EIGHTTHOUSANDFIVEHUNDREDFOUR

TRICENTIS

What would you ask/look for to find out how reliable this number is?

What they actually calculated (and still calculate) has nothing to do with "losses" and makes no sense

The Software Fail Watch is an analysis of software bugs found in a year's worth of English language news articles. To find the stories, we set up a Google account with alerts for phrases such as "software glitch" and "software bug."

Then we manually sort through each of the alerts, picking out promising headlines, reading the articles for relevance, and noting down any specific details of interest. While reading the articles we ask ourselves questions like: What industry does this story fall into? Does the article say how much the affected software cost to implement? Does it mention how many products were recalled? How long was the system down? Is the associated company public, private, or a government contractor?

You get the idea.

In short, they:

1) Find news articles about bugs.
2) Find a number related to cost present in the article (e.g., «how much the affected software cost to implement»)
3) Add these numbers

Article below: "Losses" are the total development cost of F-35! (counted twice, since two reported faults)

Controversial $400bn F-35 fighter jet now has computer 'brain' problem which could see entire fleet grounded

- Lack of testing on software may mean it's not ready for its deployment
- The problem is with the Autonomic Logistics Information System (ALIS)
- Major issue is data produced by ALIS goes through a on operating unit
- The lack of back up could mean that the entire feet has to be grounded

By ELLIE ZOLFAGHARIFARD and MARK PRIGG FOR DAILYMAIL.COM
PUBLISHED: 18:14 BST, 21 April 2016 | UPDATED: 19:33 BST, 21 April 2016

Clearly not all investments are successful

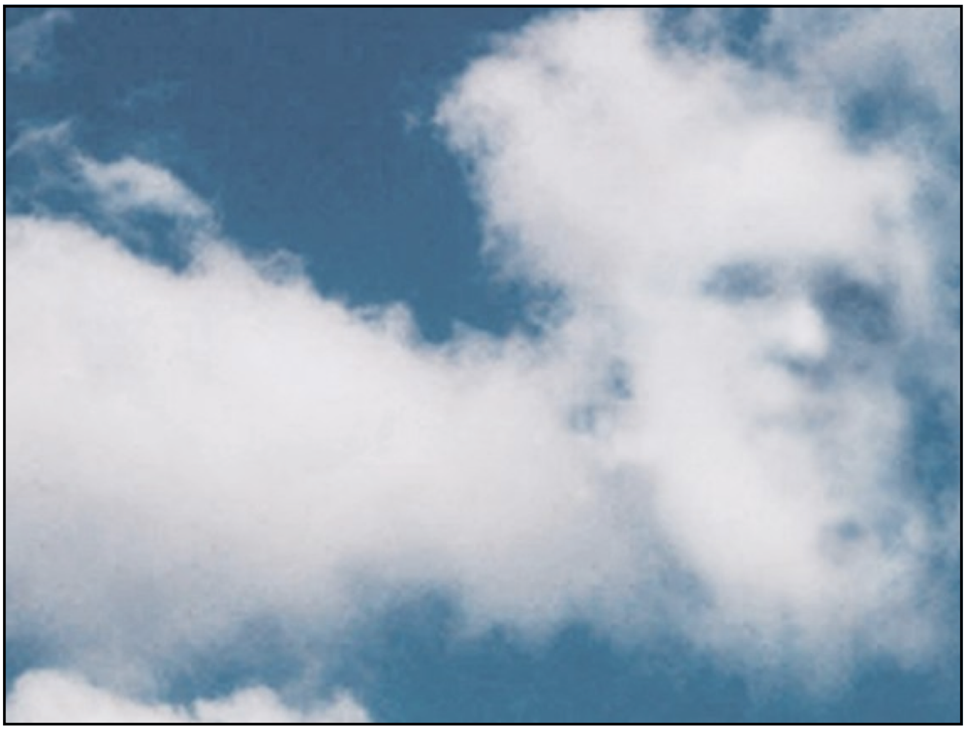
Around 10% of all digitalization projects are cancelled or completed with little or no client benefits.

About 50% get into substantial problems with either client benefits, technical quality, cost control, time control or development productivity.
(below: a selection of Norwegian IT failures)

MODERNISERINGSPROGRAMMET

Nav stanser IT-prosjekt til 3,3 milliarder

A selection of results on patterns underlying digitalization success and failures

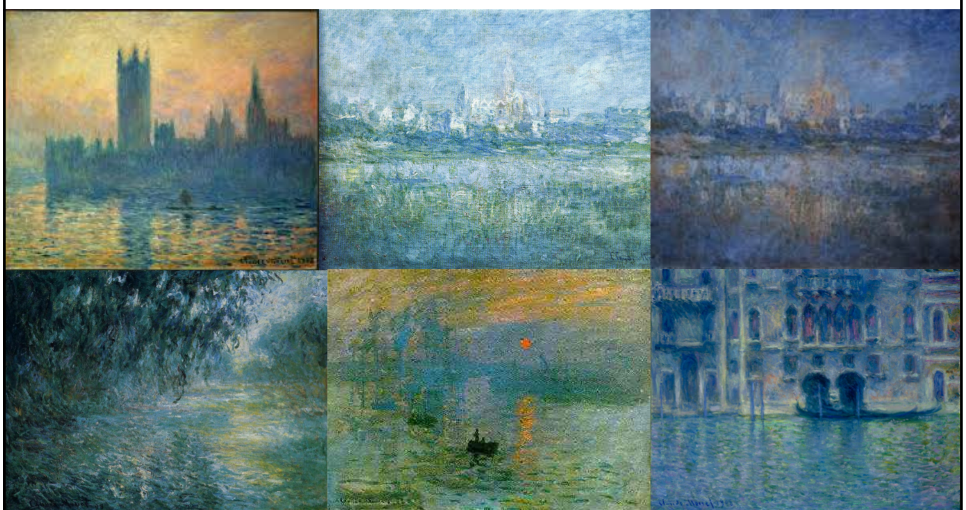


Leonard Koppett, Sporting News 1978

Year	Super Bowl Winner	Original League	Stock Market
1967	Green Bay	National	Up
1968	Green Bay	National	Up
1969	New York Jets	American	Down
1970	Kansas City	American	Down
1971	Baltimore	National	Up
1972	Dallas	National	Up
1973	Miami	American	Down
1974	Miami	American	Down
1975	Pittsburgh	National	Up
1976	Pittsburgh	National	Up
1977	Oakland	American	Down

What is the probability that this connection is by random?

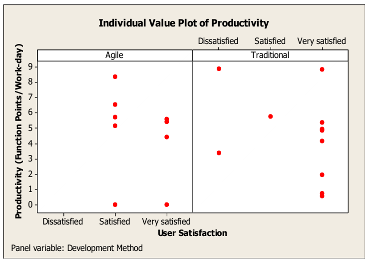
When making a decision or choice,
the world is no more the same (Dan Gilbert)



ted.com/talks/lang/eng/dan_gilbert_asks_why_are_we_happy.html

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

- **Design:**
  - Data sets with randomly set performance data comparing "traditional" and "agile" methods.
  - Survey of each developer's belief in agile methods
- **Question**: How much do you, based on the data set, agree in: *"Use of agile methods has caused a better performance when looking at the combination of productivity and user satisfaction."*
- **Result**:
  - Previous belief in agile determined what they saw in the random data



The ease of creating beliefs: Are risk-willing or risk-averse developers better?

Group

Initially
Average 3.3
Debriefing
Average 2: 3.5
2 weeks later
Average 3: 3.5

Group
—
Initially
Average 5.4
Debriefing
Average 2: 5.0
2 weeks later
Average 3: 4.9

Study design: ~~Research evidence~~ + Self-generated argument.

Question: Based on your experience, do you think that risk-willing programmers are better than risk-averse programmers?
1 (totally agree) – 5 (No difference) - 10 (totally disagree)
Neutral group: Average 5.0

Why don’t we know how to avoid failures and be successful with software development?

There are thousands of reports, research papers and presentations on how to succeed with software development projects

Successful Software Project Delivery in 10 Steps | Appnovation
https://www.appnovation.com/.../successful-software-project-deli... ▾ Oversett denne siden
15. des. 2014 - So spend some time here to not only identify and document what the change is, but also identify **how to** refocus the team and their efforts to ...

How to Manage a Successful Software Project: Methodologies ...
https://www.amazon.com/...Successful-Software-Project.../04710... ▾ Oversett denne siden
Buy **How to** Manage a **Successful Software Project**: Methodologies, Techniques, Tools on Amazon.com ✓ FREE SHIPPING on qualified orders.

Empathy: The key to a successful software project - O'Reilly Media
https://www.oreilly.com/.../empathy-the-key-to-a-successful-soft... ▾ Oversett denne siden
22. jun. 2016 - Empathy: The key to a **successful software project** ... But it is not enough to safeguard the **success** of even relatively simple software .... Learn **how to** pass data to a command without violating the command pattern in C#. Video.

Communities Over Code: How to Build a Successful Software Project ...
https://www.linux.com/.../communities-over-code-how-build-suc... ▾ Oversett denne siden
9. jan. 2017 - Healthy productive FOSS projects don't just happen, but are built, and the secret ingredient is Community over code. Purpose and details are ...

How to Ensure your Software Project is a Success
https://www.castle-cs.com/.../how-to-ensure-your-software-projec... ▾ Oversett denne siden
How to Ensure your **Software Project** is a **Success**. Thursday 9th April 2015. In the first of a series of posts on **software project** management, George Strathie, ...

Software Project Success - InfoWorld
www.infoworld.com/blog/software-project-success ▾ Oversett denne siden
23. apr. 2015 - When outsourcing a **software project**, companies will often negotiate a fixed-bid contract with ... Here's **how to** get started in the right direction.

How to Be a Successful Software Project Manager
https://books.google.no/books?isbn=1482848392 - Oversett denne siden
Dr. Tuhin Chattopadhyay - 2015 - Business & Economics
The **success** of the project largely depends upon the satisfaction of the users and ... Lot of effort goes into making a **software project successful**; right from ...

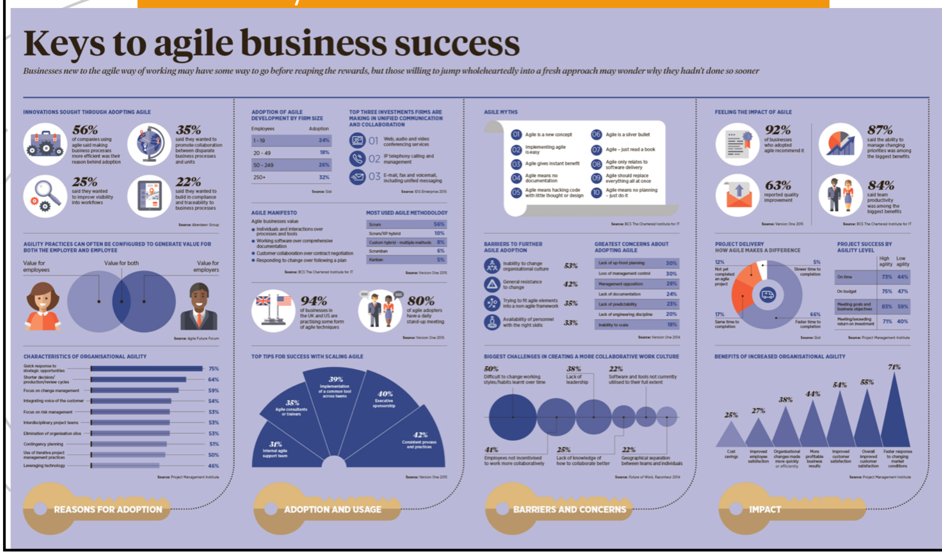
review - How to measure the success of a small software project ...
softwareengineering.stackexchange.com/.../how-to-measure-the-s... ▾ Oversett denne siden
6. mar. 2014 - Our team is putting together a quick review process for measuring the **success** of a **software project**. Projects are mostly internal, which means ...

Reliable Software Project Success [GreyLoud Guide to Software ...

SUCCESS CRITERIA	IMPORTANCE (POINTS)
1. User Involvement	19
2. Executive Management Support	16
3. Clear Requirements	15
4. Proper Planning	11
5. Realistic Expectations	10
6. Short Project Milestones	9
7. Competent Staff	8
8. Ownership	6
9. Clear Vision & Objectives	3
10. Hard-Working, Focused Staff	3
TOTAL	100

The list of success factors has not changed much since the 1960s!
More or less the same list is for example presented in:
Gotterer, M.H. *Management of computer programmers*. Proceedings of the spring joint computer conference. 1969. ACM

Many methods claim success



Cobb's paradox?

We know why projects fail, we know how to prevent their failure – so why do they still fail?

What is a proper response to Cobb's paradox? Do software professionals ignore the knowledge?

Cobb's paradox is no paradox. We don't know that much about why something fails and how to succeed.

The simple truth is that …

… we'll probably never fully understand and control what it takes to succeed

- The high complexity and innovativeness of product, process and people organization means that we can hardly expect to succeed all the time
- Much of what happens is outside of the control of the project
- Connections are context dependent and hard to identify and understand
- There is a network of connections and we're inherently poor at identifying and understanding indirect relationships
- The relationships are ***probabilistic*** and we're inherently poor at understand non-deterministic relationships

A diversion into probabilistic relationships

Just to illustrate how poor we are at identifying them
(and a bit just for the fun of it)

Representativeness bias
(seeing patterns that are not there)

Question: Assume five throws with a fair coin. Which of the following sequences is more likely to occur?
Alt. 1: Head-Head-Head-Head
Alt. 2: Head-Tail-Head-Tail

Answer: Same probability

Relevance: We tend to use to the representative heuristic (Alt 2. is more "representative" of sequence of coin flipping) and think that non-representative sequence (such as Alt. 1) are surprising patterns.

Failure of seeing true patterns

Question: Assume a sequence of throws with a fair coin. Which of the following two sequences is more likely to occur **FIRST**?
Alt. 1: Head-Head
Alt. 2: Tail-Head
Example: Head-Tail-Tail-Head-Head….
→ Tail-Head occurs before Head-Head

Answer: It is 75% likely to first observe Tail-Head and only 25% likely to first observe Head-Head

Relevance: Some probabilistic connections are connected, hidden and non-intuitive. Difficult to see them …

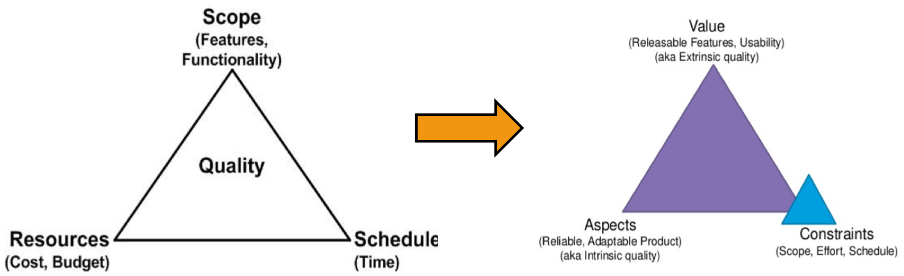
One more…
(mainly for fun, but also to show how poor our probabilistic intuition is)

- A country has regulated that no family is allowed to have more than one son, but as many daughters as they want.
- This means that allowed sequences of child-births are:
  - Boy (stop, not allowed to have more children)
  - Girl-Boy (stop)
  - Girl-Girl-Boy (stop)
  - etc.
- **Question**: How does this law affect the proportion of men and women in the country?
- **Answer**: Not effect at all. There will still be about 50-50 men and women

Back to software projects

What does it mean to succeed and to fail with software development?

Software project success: What is it?



Success dimensions:

- Client benefits delivered
- Cost control
- Time control
- Development efficiency
- Software properties (technical quality)

We need to be evidence-based to improve success: Evidence-based software engineering (EBSE)

- Tore Dybå, Barbara Kitchenham and Magne Jørgensen, Evidence-based Software Engineering for Practitioners, IEEE Software, Vol. 22, No. 1, Jan-Feb 2005.
- **The main steps of EBSE are as follows:**
  - Convert a relevant problem or need for information into an answerable question.
  - Search the literature and practice-based experience for the best available evidence to answer the question.
  - Critically appraise the evidence for its validity, impact, and applicability.
  - Integrate the appraised evidence with practical experience and the client's values and circumstances to make decisions about practice.
  - Evaluate performance in comparison with previous performance and seek ways to improve it.

What is valid evidence? A real-life example (1)

- A software development department wanted to replace their "old-fashioned" development tool with a more modern and hopefully more efficient one.
- They visited many possible vendors, participated at numerous demonstrations, and contacted several "reference customers". Finally, they chose a development tool. The change cost about 10-20 million NOK + training and other indirect costs.
- A couple of years after the change, the department measured the change in development efficiency (not common – most software organizations never study the effect of their choices).
- Unfortunately, the development efficiency had not improved and the new development tool was far from as good as expected.
- This illustrated that even when applying much resources and time to collect evidence, software professionals may fail in making good decisions. What went wrong in this case?

What went wrong? A real-life example (2)

- The collection and evaluation of evidence had focused on "tool functionality", following the principle "the more functionality, the better".
- The demonstrations focused on strengths of the tools, not on weaknesses. Although, the software professionals were aware of this, they probably failed to compensate for what the demonstrations did not demonstrate. (We are not good at identifying lacking information!)
- The reference customers had themselves invested much money in the new tool. As long as they do not plan to replace the tool, then they would however not be reference customers anymore, they will tend to defend their decisions. (Avoidance of cognitive dissonance.)
- Although the amount of information (evidence) was high, they organization lacked the most essential information (independent evaluations of the tools in context similar to their own) and processes for critical evaluation of the information.
- In addition, they lacked the awareness of how they were impacted by the tool vendors persuasion techniques.
- Guidance in the principles of evidence-based software engineering would, we think, improved the decision.

What could have been done better?

- Formulate the problems and goals more precisely
- Collect evidence (research, experience from neutral sources, …).
  - At that time, there were no research studies, but possibly studies on related tools and neutral experience, available.
  - They could, for example, try to find tool customers similar to one's own organization and use more structured and critical experience elicitation processes.
  - They should avoid that the tool vendor chose the reference customers.
- Complete of own empirical studies.
  - Invite the tool vendors to solve problems specified by the department itself at the department's own premises.
  - Many vendors seem to accept this type of "competition", given an important client. If not, pay them to to some work on a representative project.
- Avoid decision biases, such as those from vendor demonstrations, dinners with the tool vendors and other situations known to include more persuasion than valid information (or, at least, they should not let those who were exposed to this type of impact participate in the decision.)

Exercises
How would you test the following claims in an evidence-based manner?

1) "**Most (93%) of our communication is non-verbal"** (common claim in presentation courses and books)
2) **"45% of features of "traditional projects" are never used** (Standish Group, again …)
3) **"There is an increase in cost of removing errors in later phases**" (common claims in testing)
4) **"Agile is better than traditional methods"** (common claim by agile people)

tinyurl.com/origami-berlin

Does the software development method matter?
(Does it help to work agile?)

Common belief (amongst agile people): Yes

Try to explain what these agile claims (values) mean

Our studies: Yes, agile helps, but …

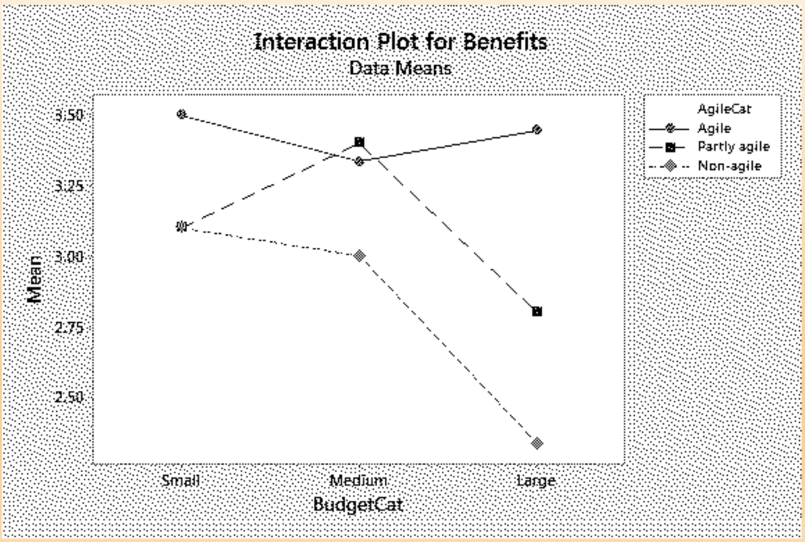
The numbers show the increase (in percent points) in proportion of successful projects

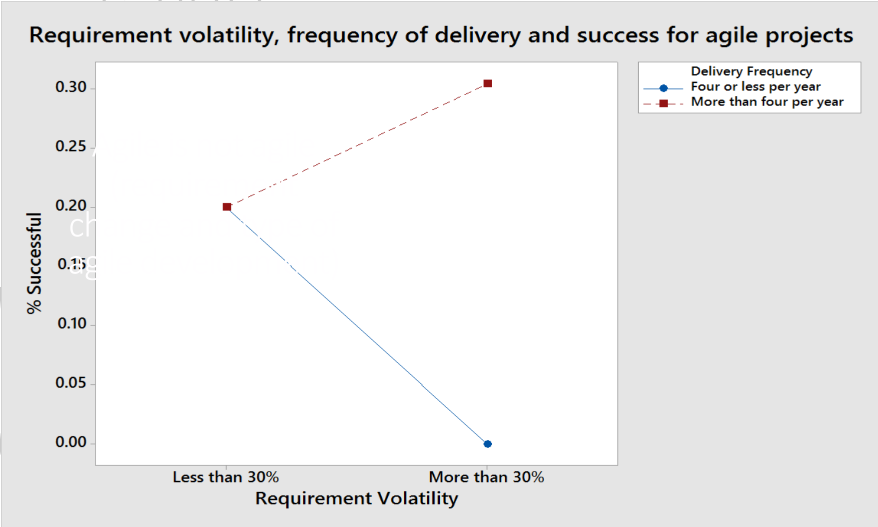
	Agile	Frequent delivery to production	Flexible scope
Client benefits	16%	22%	29%
Technical quality	21%	6%	32%
Budget control	2%	22%	29%
Time control	8%	11%	24%
Efficiency	11%	5%	24%

… only when including frequent delivery to production and flexible scope.
Agile projects not including these practices were LESS successful than non-agile projects! We need to emphasize individual practices to understand connections with success.

Similar results in our follow-up surveys and studies. NB: Correlation is not (necessarily) causation.

«True» agile is particularly good at delivering client benefits in larger projects (mean success wrt delivered benefits 1 (failure) ..5 (very successful)





Are larger
(and presumably more complex) projects
less successful?

Common belief: Yes

Our (initial) result: No
Large projects not less successful
than smaller ones (similar finding in all studies)

Criterion	< 1 mill Euro	1-10 mill Euro	> 10 mill Euro
Client benefits	31%	47%	35%
Tech. quality	24%	28%	25%
Budget control	24%	47%	47%
Time control	29%	35%	35%
Efficiency	24%	12%	24%

The numbers (percentages) represent the proportion of projects
assessed to be successful or very successful with respect to a success criterion.

But, the first results hid that we only had studied <u>completed</u> projects

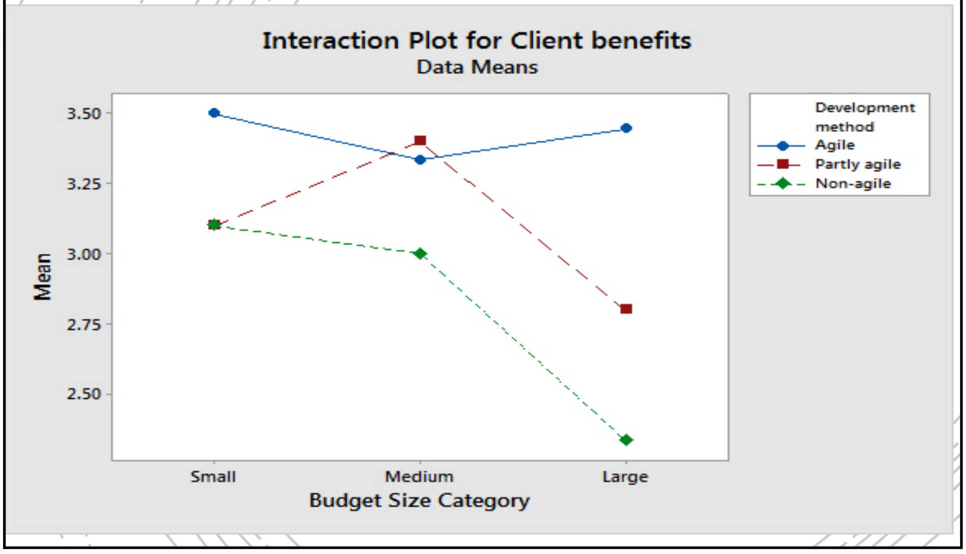
Adding non-completed projects in follow-up studies gave that the largest projects (> 10 mill Euro) were strongly over-represented in the group of failed projects (2-3 times more frequent).

A rule of thumb (based on offshoring projects) is that ten times larger project size leads to twice the risk of failure.

Also of interest:

- Different reasons for problems for small and large projects.
- Higher risk of failure with larger projects should not be used to divide ”logical connected deliveries” into separate projects.

Agile software projects seem to be less affected by large project size



Does contract type matter?

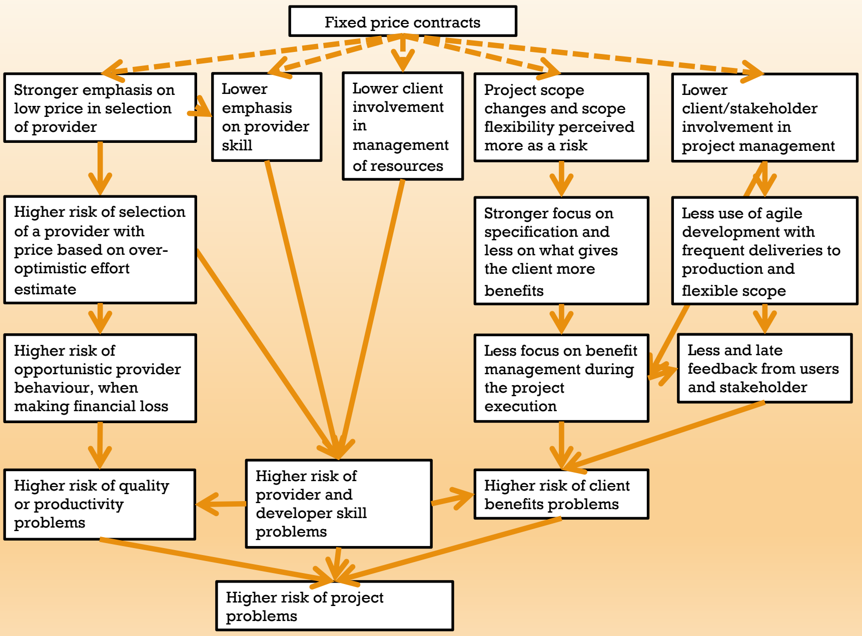
Common belief:
Clients: Fixed price contracts is better
Providers: Time & materials payment is better

Our finding: Time & materials type of contracts much better for both the client and the provider (several studies)

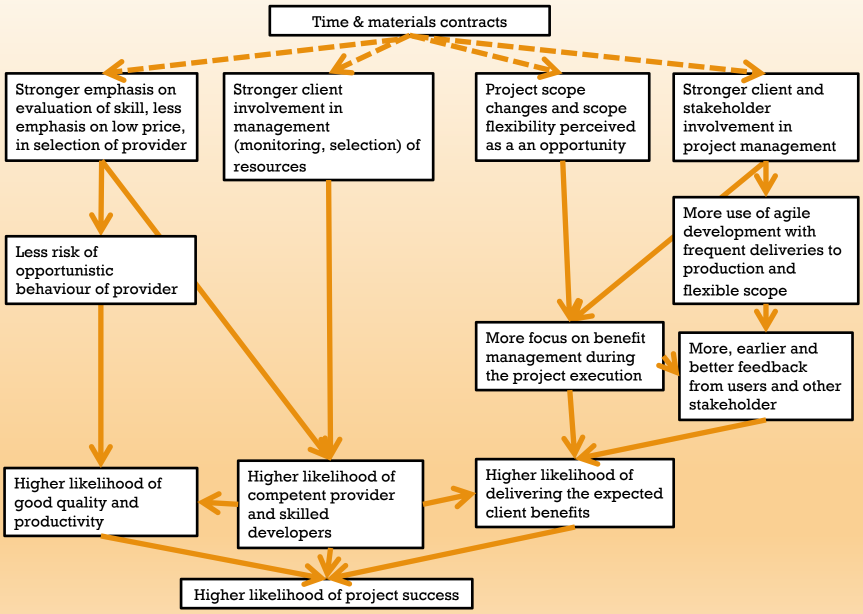
First study: Extremely negative results for Fixed price contracts.

	Fixed price	Time & Material
Client benefits	**0% (success rate)**	**59%**
Technical quality	22%	24%
Budget control	33%	31%
Time control	11%	29%
Efficiency	0%	19%

Failure pattern (Fixed price behaviour)



Success pattern (Time and materials behavior)



Does it help with “benefits management”?

Common belief: Yes (but few do it)

Our finding: Not all benefit management practices led to much improvements

Survey 1: Survey

Benefit management practices	Proportion	Increase in success rate (wrt benefits)
Cost-benefit analysis (up front)	47%	6%
Benefit responsible appointed	57%	22%
Plan for benefit management	33%	31%
Benefit management during proj. execution	53%	34%
Evaluation of benefit during/after proj. exec.	31%	19%

Survey 2 (in-depth study):

Benefit management practices	Present	Not present/don't know
Cost-benefit analysis (up front)	31% with problems	22% with problems
Benefit responsible appointed	28% with problems	29% with problems
Plan for benefit management	29% with problems	28% with problems
Benefit management during proj. execution	20% with problems	35% with problems

Characteristics of the successful project

Characteristics of the successful project

- Good control of ambition level. Avoiding ”too much” at the same time and good at saying ”no” to adding complexity.
- Use of contracts that avoid ”fixed price”-behavior.
- Client with competence to select and manage competent providers and individual resources (not so much focus on low price)
  - Selection of resources from more than one provider
- Flexibility in scope (not only ”must have”-functionality)
- Client is (as a minimum) strongly involved in the planning and execution of benefits management.
- Use of agile development with frequent deliveries to production (or at least with proper testing/feedback from real users)
- Early start of involvement of stakeholders (especially the users) and planning and preparing for deployment.

54

Exercise: Evidence-based practice (group work - if we have enough time)

1) Formulate a question (or problem) about the how you can positively influence software development success.

<This could be anything from the effect of a particular programming tool/language to contracts, development methods and team organization.>
NB: Remember to formulate this in a way that makes it possible and meaningful to collect evidence about it and answer the question.
➔ Short discussion

2) Collect empirical evidence (here: use google scholar to find at least one relevant paper – if available, a systematic literature review)

3) Evaluate the paper critically, both related to relevance and validity of the evidence.

[4) Aggregate the evidence … Another time …]

Give a 5 minutes presentation of what you found out …

Failure factors from a study of 400.000 small projects

Predictor variable	Coefficient	p-value	Odds ratio	95% confidence interval	
				Lower	Upper
Constant	-2.90	0.00			
SatisfactionScoreProviderCat=Low	0.35	0.00	1.42	1.39	1.45
SatisfactionScoreProviderCat=No Scores	0.91	0.00	2.49	2.33	2.67
FailureRateProviderCat=Low	-0.66	0.00	0.52	0.51	0.53
FailRateProviderCat=No Projects	-0.34	0.00	0.71	0.67	0.76
SkillTestPassRateProviderCat=Low	0.07	0.00	1.07	1.02	1.12
SkillTestPassRateProviderCat=No Tests	0.58	0.00	1.79	1.74	1.85
SatisfactionScoreClientCat=Low	0.18	0.00	1.20	1.17	1.23
SatisfactionScoreClientCat=No Scores	0.25	0.00	1.28	1.23	1.33
FailureRateClientCat=Low	-0.64	0.00	0.53	0.52	0.54
FailureRateClientCat=No Projects	-0.63	0.00	0.53	0.51	0.56
PreviousCollaboration=Yes	-1.74	0.00	0.17	0.17	0.18
FocusLowPriceCat=Low	-0.19	0.00	0.83	0.81	0.85
FocusLowPriceCat=Medium	-0.08	0.00	0.92	0.89	0.95
FailureRateProviderRegionCat=High	0.27	0.00	1.31	1.28	1.33
FailureRateClientRegionCat=High	0.42	0.00	1.53	1.48	1.58
GeographicalDistance=Neighbor	-0.07	0.02	0.93	0.90	0.97
GeographicalDistance=Offshore	0.02	0.10	1.02	1.00	1.05
logProjectSize	0.71	0.00	2.03	1.99	2.06

Jørgensen, Magne. "Failure factors of small software projects at a global outsourcing marketplace." Journal of systems and software 92 (2014): 157-169.

Regional differences in failure rate

Table: Client = columns, Provider = rows

Client / Provider	AF	EA	EE	LA	ME	NA	OC	SA	WE	Total
AF (Africa)	14% (92)	22% (289)	26% (137)	19% (105)	23% (195)	16% (3944)	12% (692)	26% (306)	15% (183)	17% (7633)
EA (East Asia)	20% (332)	16% (1660)	19% (856)	15% (662)	18% (970)	12% (27447)	12% (3953)	25% (1416)	15% (10576)	14% (48023)
EE (East Europe)	11% (1285)	14% (5010)	13% (5278)	11% (2618)	14% (4325)	9% (114728)	10% (11473)	18% (4355)	10% (51088)	10% (201565)
LA (Latin America)	12% (127)	16% (523)	14% (540)	11% (985)	15% (493)	10% (17245)	9% (1888)	20% (499)	12% (6369)	11% (28868)
ME (Middle East)	16% (231)	25% (622)	16% (635)	17% (320)	17% (824)	13% (15881)	13% (1973)	26% (792)	15% (6494)	14% (27883)
NA (North America)	19% (2713)	20% (2713)	16% (2143)	20% (1352)	19% (2112)	13% (86346)	15% (8161)	25% (2049)	15% (23947)	14% (130919)
OC (Oceania)	14% (58)	18% (260)	26% (149)	26% (82)	19% (182)	12% (6656)	9% (1474)	24% (205)	15% (2303)	13% (11484)
SA (South Asia)	17% (2614)	23% (7729)	22% (4861)	19% (3599)	20% (5632)	16% (143699)	15% (18958)	24% (10934)	18% (54710)	17% (254075)
WE (Western Europe)	13% (470)	17% (2070)	14% (1779)	14% (960)	15% (1927)	13% (38544)	14% (4250)	23% (1529)	13% (20111)	13% (72297)
Total	16% (5734)	19% (20935)	17% (16393)	16% (10702)	18% (16714)	13% (456106)	13% (52894)	23% (22113)	14% (177852)	

Journal of Systems and Software
Volume 116, June 2016, Pages 133-145

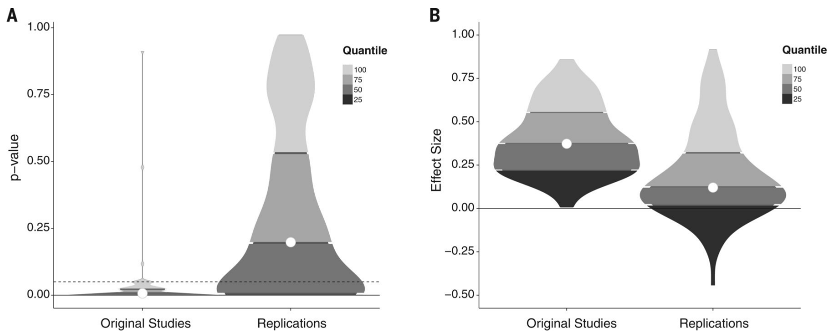
Incorrect results in software engineering experiments: How to improve research practices

Magne Jørgensen [a, b], Tore Dybå [b, c], Knut Liestøl [b], Dag I.K. Sjøberg [b]

Assume	Incorrect results	Incorrect significant results
50% true relationships	Ca. 40%	Ca. 35%
30% true relationships	Ca. 60% (most results are false)	Ca. 45% (nearly half of the significant results are false)

The study also – perhaps more importantly – shows that there must be a large amount of researcher and publication bias in our studies

Replication of 100 experiments reported in papers published in 2008 in three top psychology journals (Replication sample size 3-4 times the original size)



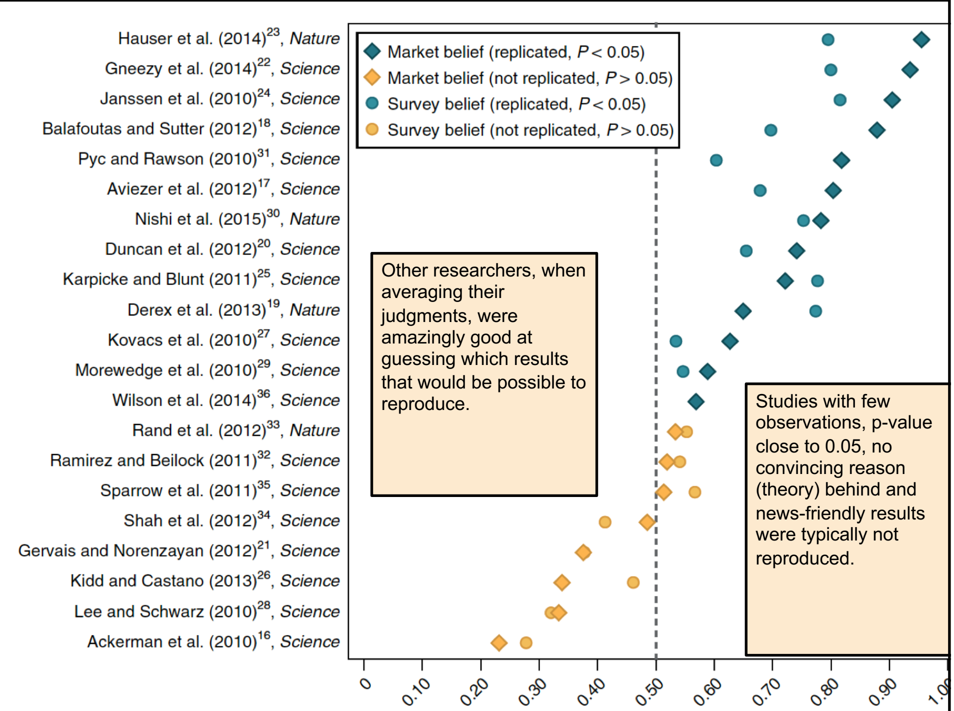
Open Science Collaboration. *Estimating the reproducibility of psychological science.* Science 349.6251 (2015).

Reproduced effect size was on average about one third of the originally reported effect size.

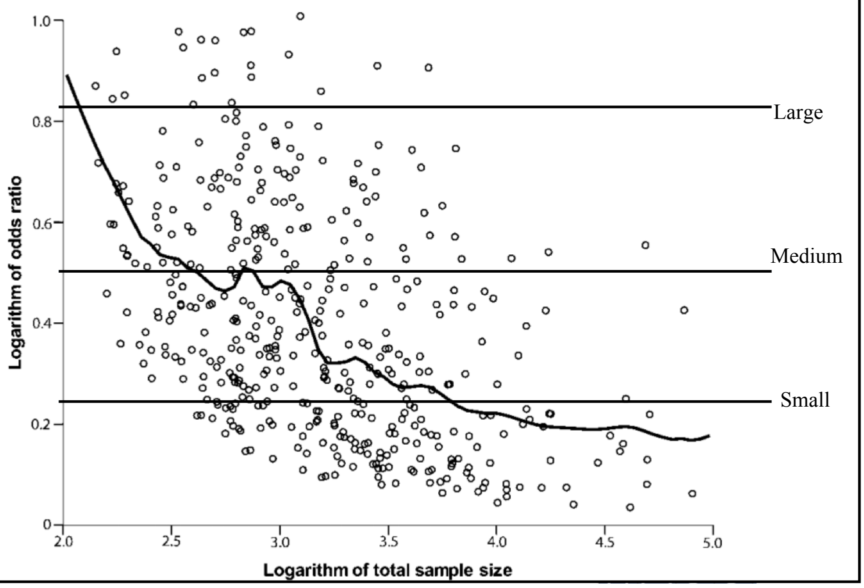
nature
human behaviour
LETTERS
https://doi.org/10.1038/s41562-018-0399-z

Evaluating the replicability of social science experiments in *Nature* and *Science* between 2010 and 2015

- Sample sizes on average about five times higher than in the original studies.
- Statistically significant effect in the same direction as the original study for 13 (62%) studies, and the effect size of the replications was on average about 50% of the original effect size.

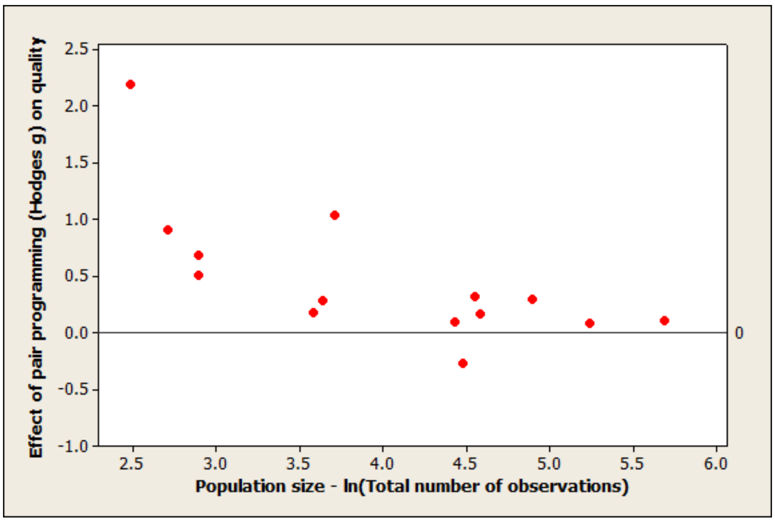


This is further illustrated in: “Why most discovered true associations are inflated”, Ioannidis, Epidemiology, Vol 19, No 5, Sept 2008



Example from empirical software engineering

Data from: Hannay, Jo E., et al. "The effectiveness of pair programming: A meta-analysis." Information and Software Technology 51.7 (2009): 1110-1122.



TITLE: What makes software development projects successful, and what makes them fail?

Abstract: Numerous research studies and consultancy reports make claims about how often, or rather how seldom, software projects are successful, why so many of them fail, and how to succeed more often. These studies and reports have reported very much the same success and failure factors and the same advices since the 1960s. If we already know how to make a successful software project, why is the proportion of failed software projects about the same as earlier? Are software professionals ignorant of the published knowledge or are there other reasons? Important reasons for the little use of the knowledge may be that previous studies have had very little focus on the most important success dimension, i.e., delivering value, contain very little practical advice on how to succeed, and have not managed to include the context-dependency and complexity of the connections between process choices and outcome. In this course I present evidence-based, practical advices based on a set of own and other researchers' empirical studies on software projects. It starts with an attempt to better define and operationalize what we should mean with project success and how to analyze and describe the context-dependent and probabilistic network of connections between essential choices and behavior, and the outcome of software projects. Then evidence connecting software development success to sourcing models, contract types, competence evaluation, cost-benefit analyses, benefits management, software development processes and project management is presented. Finally, the evidence is summarized and presented as context-dependent patterns of software project success and failure.

Biography: Magne Jørgensen is a chief research scientist at Simula Metropolitan Center for Digital Engineering, a professor at Oslo Metropolitan University, a consultant at Scienta and a guest professor at Kathmandu University. His research includes work on management of software projects, evidence-based software engineering and human judgment. He has published on these and other topics in software engineering, forecasting, management and psychology journals. He has been ranked the top scholar in systems and software engineering four times and was in 2014 given the ACM Sigsoft award for most influential paper the last ten years for his work on evidence-based software engineering. He is member of the Norwegian Digitalization Advisory Board.

Time: August 28, 13-17.
Place: Architecture (A) Building, «Strasse des 17. Juni, 152». Lecture room A053 (ground floor).