Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria

Empirical Software Engineering: the Good, the Bad, and the Useful (and don't forget the ugly)

Luigi Lavazza
Università degli Studi dell'Insubria, Varese, Italy
luigi.lavazza@uninsubria.it

The 17th International Conference on Software Engineering Advances
ICSEA 2022 - October 16-20, 2022 - Lisbon, Portugal

Luigi Lavazza

- Luigi Lavazza is associate professor of Computer Science at the University of Insubria at Varese, Italy. Formerly he was assistant professor at Politecnico di Milano, Italy. Since 1990 he cooperates CEFRIEL, where he acts as a scientific consultant in digital innovation projects.
- His research interests include: Empirical software engineering, software metrics and software quality evaluation; Software project management and effort estimation; Software process modeling, measurement and improvement; Open Source Software.
- He was involved in several international research projects, and he also served as reviewer of EU funded projects.
- He is co-author of over 170 scientific articles, published in international journals, or in the proceedings of international conferences or in books.
- He has served on the PC of a number of international Software Engineering conferences; from 2013 to 2018 he was the editor in chief of the IARIA International Journal On Advances in Software.
- He is a IARIA fellow since 2011

Many low quality papers

- Often observed in the recent literature:
  - Not interesting papers
    - Addressing hardly relevant issues
  - Not scientifically sound
  - Not practically applicable
  - Any combination of the problems mentioned above
- The observed quantity of low-quality papers is growing

A taxonomy of “bad” papers

- Most often, papers that are likely to turn out bad are the result of an “original sin” concerning the goal of the paper and/or the context in which it is conceived.
  - Paper industry
    - Nor specific of software engineering: see for instance the "Bartleby industry" in literary studies
  - Original & irrelevant papers
  - Superficial systematic literary reviews
  - Ultra-sophisticated studies
  - It is new: let’s use it!
  - Combinatorial research
  - …

Paper industry

- Every now and then, research topics emerge that become fashionable.
- In some cases it is because the topic concerns emerging problems that actually need a lot of research
  - This is likely to give rise to good papers
- Quite often, a topic attracts attention just because a lot research can be done. The usefulness of that research is not considered.
  - A paper industry is created. New conferences and workshops are created dedicated to the topic.
  - Examples:
    - Code smells
    - Technical debt
    - ML for AI (defect detection, effort estimation, …)
    - …

The case of code smell

- The book that introduced code smells was published in 1999
  - Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: improving the design of existing code. Addison-Wesley (1999)
- The possibility to detect code smell automatically was suggested in 2004
  - Marinescu, R.: Detection strategies: Metrics-based rules for detecting design flaws. 20th IEEE Int. Conf. on Software Maintenance. (2004)
  - This is the paper that originated the industry
- According to scholar.google.com, the papers published on software "code smell" have been
  - 65 before 2005 (15 per year, on average)
  - 4390 since 2005 (244 per year, on average)
  - Note that the papers mentioning "bad smell" and software are over 11000. Of these, the majority deals with software engineering issues

The case of code smell: what happened

- Code smells were introduced as items of checklists for manual software inspections
  - Programmers should be able to recognize code smells as reasons to refactor code
- In 2004, the idea of automatically detecting code smells was introduced
- This was the start of the industry
  - Alternative methods to identify, visualize and detect code smells
  - Study of relationships among code smells
  - Study of the impact of code smells
  - Evolution of code smells
  - (semi)automated refactoring
  - New types of code smells
    - So, all the previous research can be remade, taking into account also the new smells.

The smell industry growth

- Why limiting the interest to *code* smell?
- New types of software-related “smells” were introduced
  - Architectural smells and architecture smells (169+117 papers)
  - Test smells and testing smells (364+61 papers)
  - Design smell (431 papers)
  - Community smells (230 papers)
  - …

The code smell industry: results

- What did we get from all this research?

- Some academic prototype tools for code smell detection.
  - Hardly any was integrated into actual development tools.

- In practice, developers who want to improve their code use anti-pattern detection tools, whose development started before and proceeded independently of code smell research.
  - PMD
  - Spotbugs (formerly Findbugs)
  - CppCheck
  - …

The original & irrelevant papers

- These papers are original and definitely correlated with software development. They are also technically sound.
- Unfortunately, these papers are generally useless
  - They do not create new knowledge, rather they provide evidence for common sense
  - They are not actionable. Or, better, they suggest the same kind of actions as common sense.
- They get accepted because it is difficult to criticize them, and often they look as sophisticated and innovative research,

The original & irrelevant papers

- Examples:
  - “Need for Sleep: The Impact of a Night of Sleep Deprivation on Novice Developers’ Performance”
    - Appeared on IEEE TSE
    - concludes that sleep deprivation has possibly disruptive effects on software development activities
  - “Morning or Evening? An Examination of Circadian Rhythms of CS1 Students”
    - IEEE/ACM Int. Conference on Software Engineering: Software Engineering Education and Training
  - A number of studies on the effects of Covid-19 on software development
  - …

Superficial systematic literary reviews

- In principle, performing Systematic literary reviews is a good idea.
- In practice, most SLR are hardly useful.
- Problems:
  - There are too many
    - SLRs are easy to make and are easily accepted (there have been conferences where half of the program was made of SLR papers)
  - The emphasis has moved from summarizing the knowledge in a given field to
    - Showing that the review is actually systematic
    - Providing a lot of “descriptive data” (papers per year, per venue, etc.; frequency of methods, various types of classifications, …)
  - At best, the reader gets a (usually incomplete) list of pointers
- They have also perverse affects: in some cases you are requested to perform a SLR before you can state that there is a problem to tackle.

Ultra-sophisticated studies about something

- These are esoteric studies
  - The problem is extremely specific, difficult to understand
  - The techniques employed are many and complex, and often under-specified.
  - The paper is long, difficult to follow, full of data and graphs

- My feeling is the reviewers accept these papers without fully understanding them. Maybe they are afraid of admitting they did not understood something…

It is new: let’s use it!

- Techniques, methods, measures, etc. are often proposed without proper validation.

- The research that follows uses the proposed technique, measure or method as though it was reliable.
- The rational is that the newer the employed technique the more easily is the paper accepted, and the most often it will be cited.

Combinatorial research

- Many methods, many datasets, many measures, etc.
- The number of combinations is extremely large, so there are a number of subsets that can be selected to write a paper.
  - Addressing a subset of the methods, applied to one or more of the available datasets, evaluating results using a subset of the available datasets
- Often combined with cherry picking?
  - Sometimes selection appear strange: there is the suspect that the combinations that provide most remarkable results are selected.

The diseconomy of scale of paper quality

- The pressure to write many papers has a perverse effect on authors.
- Example:
  - PhD students need to publish
  - They produce papers
  - The supervisor is busy with a number of tasks and duties, so he/she does not review the paper written by the inexperienced PhD student
  - Result:
    - low quality paper are submitted
    - PhD students do not learn how to write good papers
      - Although in some cases they learn from reviews
      - But very often reviewers do not work well either, as we shall see…

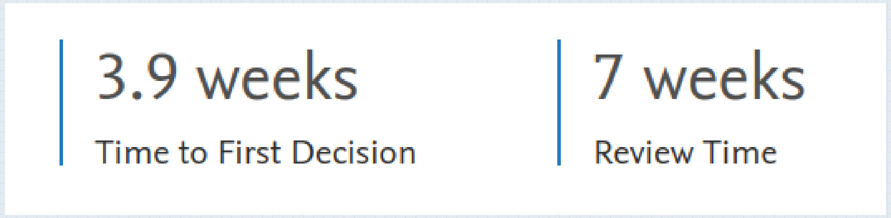
The diseconomy of scale of paper quality

- Having more and more paper to review implies that
  - There is a shortage of experienced reviewers
  - There is a shortage of time for performing reviews

- Consequences (at review level):
  - More and more inadequate reviewers are involved
  - The need to provide reviews in short time makes it difficult to write good reviews
  - Many reviewers delegate reviews to their inexperienced PhD students
    - How many papers written by inexperienced students are reviewed by equally inexperienced students?
- Consequences (at publishing level):
  - Many low quality papers get published

Faster, faster, faster

- Authors look for journals that offer quick reviewing cycles.
- Publishers are aware that quick reviews attract authors
  - i.e., quicker reviews = greater business
- Example:
  - Elsevier Journal of Informetrics



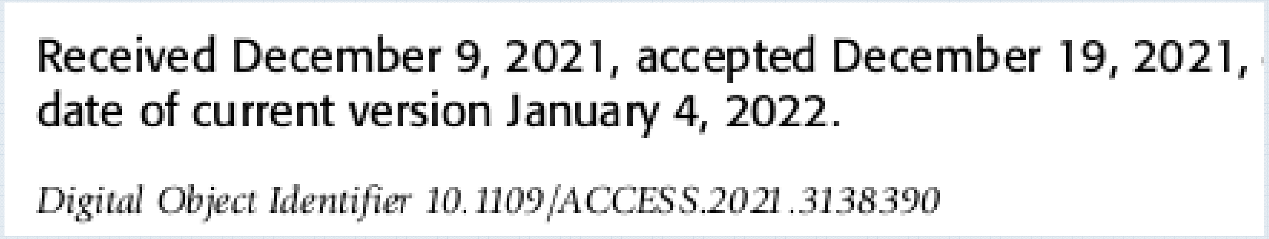
3.9 weeks	7 weeks
Time to First Decision	Review Time

  - IEEE Access

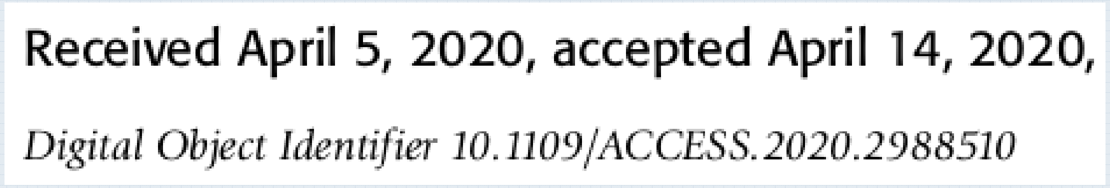
Rapid peer review
and publication
4-to-6
weeks
IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Faster than faster

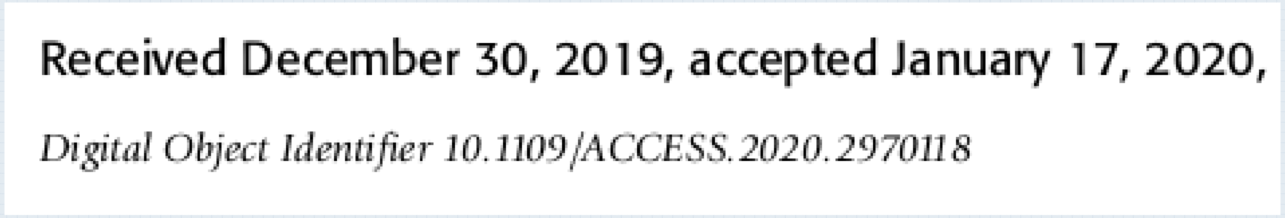
- From IEEE Access papers:


Received December 9, 2021, accepted December 19, 2021,
date of current version January 4, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3138390


Received April 5, 2020, accepted April 14, 2020,

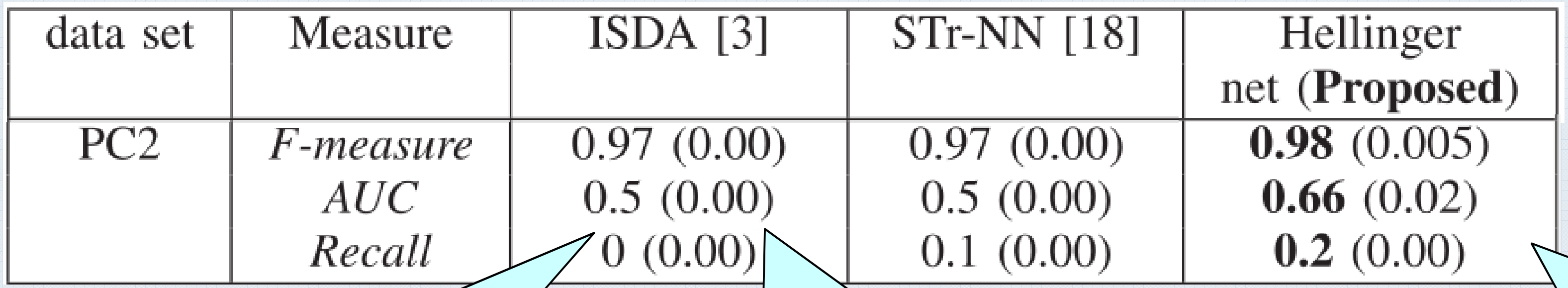
Digital Object Identifier 10.1109/ACCESS.2020.2988510


Received December 30, 2019, accepted January 17, 2020,

Digital Object Identifier 10.1109/ACCESS.2020.2970118

Inadequate reviews: top-level journal example

- Published on IEEE Transactions on reliability
  - 16th on 175 journals on Safety, Risk, Reliability and Quality, according to Scimago
- The paper bases its conclusions on the following results:



data set	Measure	ISDA [3]	STr-NN [18]	Hellinger net (**Proposed**)
PC2	*F-measure* *AUC* *Recall*	0.97 (0.00) 0.5 (0.00) 0 (0.00)	0.97 (0.00) 0.5 (0.00) 0.1 (0.00)	**0.98** (0.005) **0.66** (0.02) **0.2** (0.00)

It is not possible that F-measure is close to 1 when Recall is zero

AUC-0.5 indicates random classification. You cannot get F-measure 0.97 with random classification with the PC2 dataset

It is not possible that F-measure is 0.98 when Recall is 0.2

- The problems are obvious: how is it ever possible that no reviewer noticed them?

Inadequate reviews: top-level journal example

- Published on Empirical Software Engineering (Springer)
  - 45th of 404 papers in Software, according to Scimago

$$R^2 = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

- Probably just an oversight, but no reviewer noticed it.

Inadequate reviews: another example

- Published on Mathematical Problems in Engineering (Hindawi, i.e., Wiley)
  - Ranked Q2 by Scimago
- No reviewer noticed the following quite obvious mistakes:
  - The definition of recall and precision are mixed up

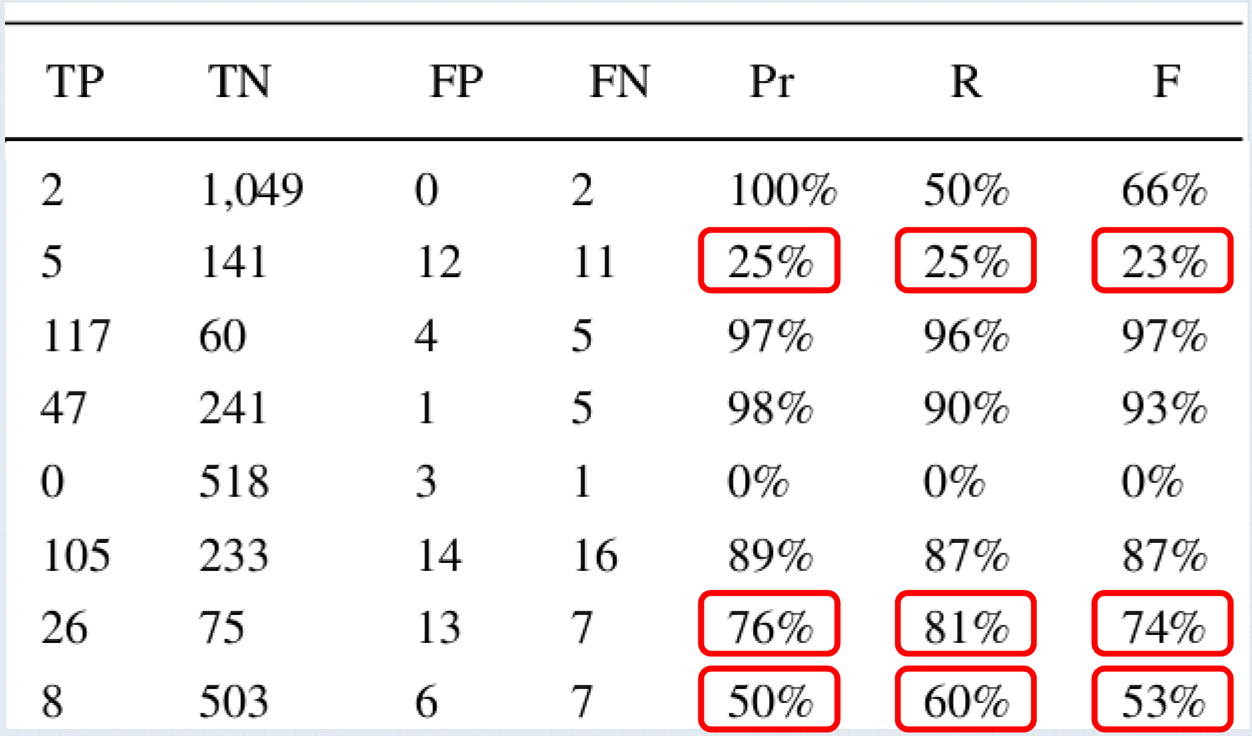
$$\text{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad \text{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

  - F1-score is systematically less than both precision and recall

Recall	Precision	F1-score
0.7358	0.6814	0.6767
0.7096	0.6722	0.6685
0.7504	0.7264	0.7142
0.7826	0.7581	0.7451

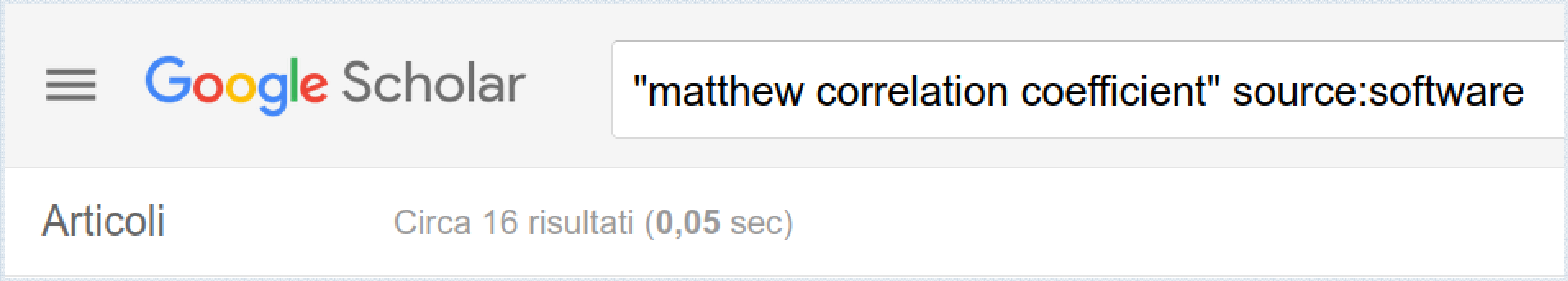
Inadequate reviews: another example

- Published on Empirical software engineering (Springer)
  - 45th of 404 papers in Software, according to Scimago
- No reviewer noticed the following quite obvious mistakes:



TP	TN	FP	FN	Pr	R	F
2	1,049	0	2	100%	50%	66%
5	141	12	11	$\boxed{25\%}$	$\boxed{25\%}$	$\boxed{23\%}$
117	60	4	5	97%	96%	97%
47	241	1	5	98%	90%	93%
0	518	3	1	0%	0%	0%
105	233	14	16	89%	87%	87%
26	75	13	7	$\boxed{76\%}$	$\boxed{81\%}$	$\boxed{74\%}$
8	503	6	7	$\boxed{50\%}$	$\boxed{60\%}$	$\boxed{53\%}$

Inadequate reviews: yet another example



"matthew correlation coefficient" source:software

Articoli

Circa 16 risultati (**0,05** sec)

- Matthews Correlation Coefficient became Matthew Correlation Coefficient in several papers
  - Published by Wiley, IEEE, Springer, Elsevier, …

- Not a big problem in itself, as it is just a typo that does not affect the value of papers …
- but how is it ever possible that no reviewer noticed this mistake? To anybody who is expert in performance metrics the error is immediately evident.

The diseconomy of scale of paper quality

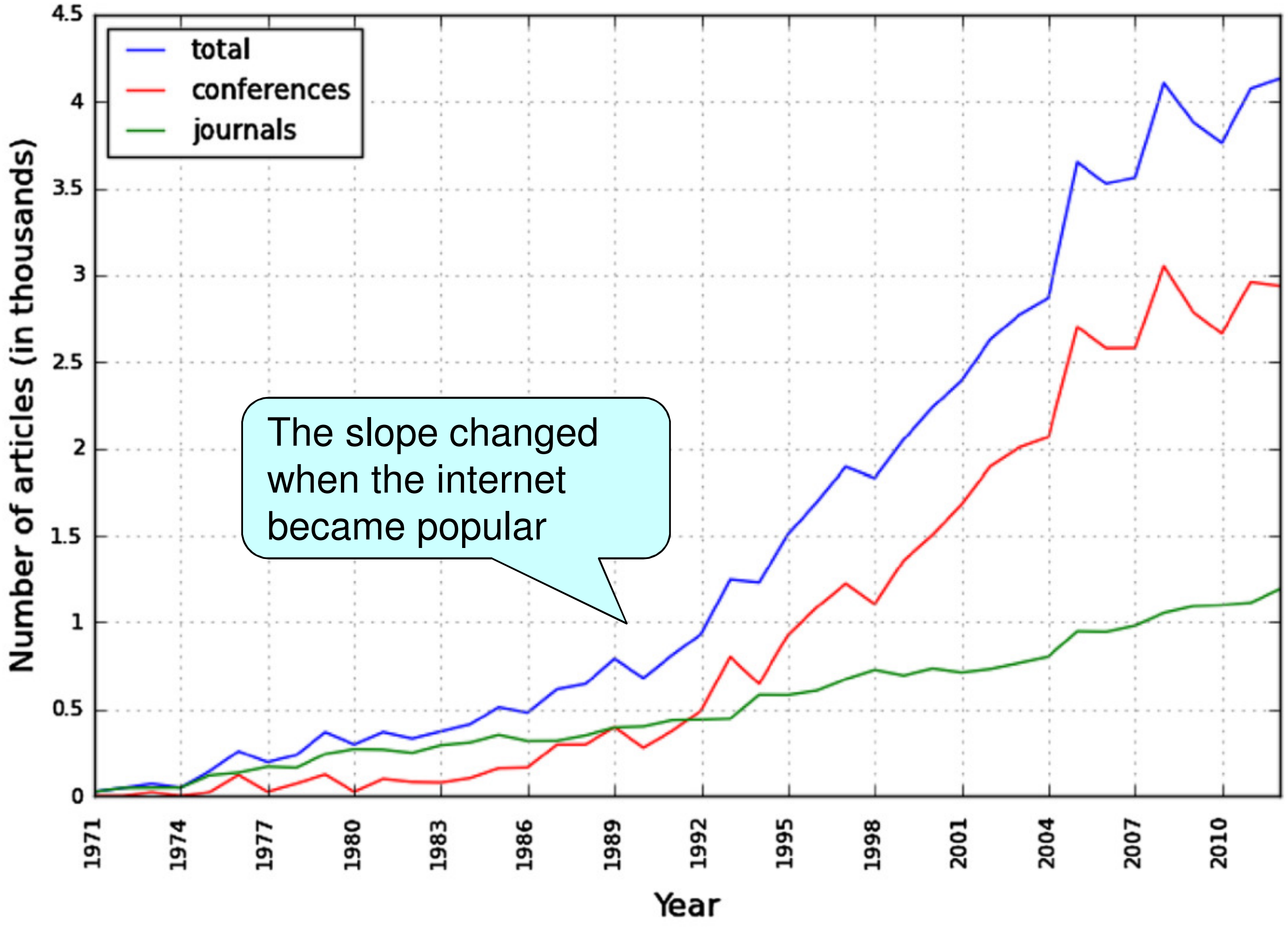
- The need to “put together” a program for a conference may have perverse effects
- Example:
  - Few papers have been submitted
  - To accept enough papers to compose a program of reasonable extent, the acceptability threshold is lowered.
  - Result: more low quality papers

Reasons why so many bad papers get published

Many papers

- A reason why the amount of “bad” papers is growingly is that there is a continuously growing number of published papers.
  - The more papers, the more bad papers
  - Besides, the relative value of the paper decreases, thus there is a pressure to publish more papers, even those that one would probably not publish, in “normal” conditions.

Trend of published articles in SE conferences and journals (up to 2010)



The slope changed when the internet became popular

João M. Fernandes Authorship trends in software engineering
Scientometrics · October 2014
DOI: 10.1007/s11192-014-1331-6

Many conferences and journals

- Many researchers are willing to publish
- Publishing journals and organizing conferences is a lucrative business
  - Especially with open-access mechanisms (authors pay to get the paper published as open access)

➔ hence the number of conferences and journals is continuously growing

High pressure to publish

- Motivations
  - “publish or perish”
- Opportunity
  - There are so many conferences and journals where one can publish
  - Publishing is easy
    - Produce a pdf and submit is via a web site

Publication indexes are easily available

- Publishing papers on the internet makes indexing easy
- Publishing papers on the internet makes publishing easy

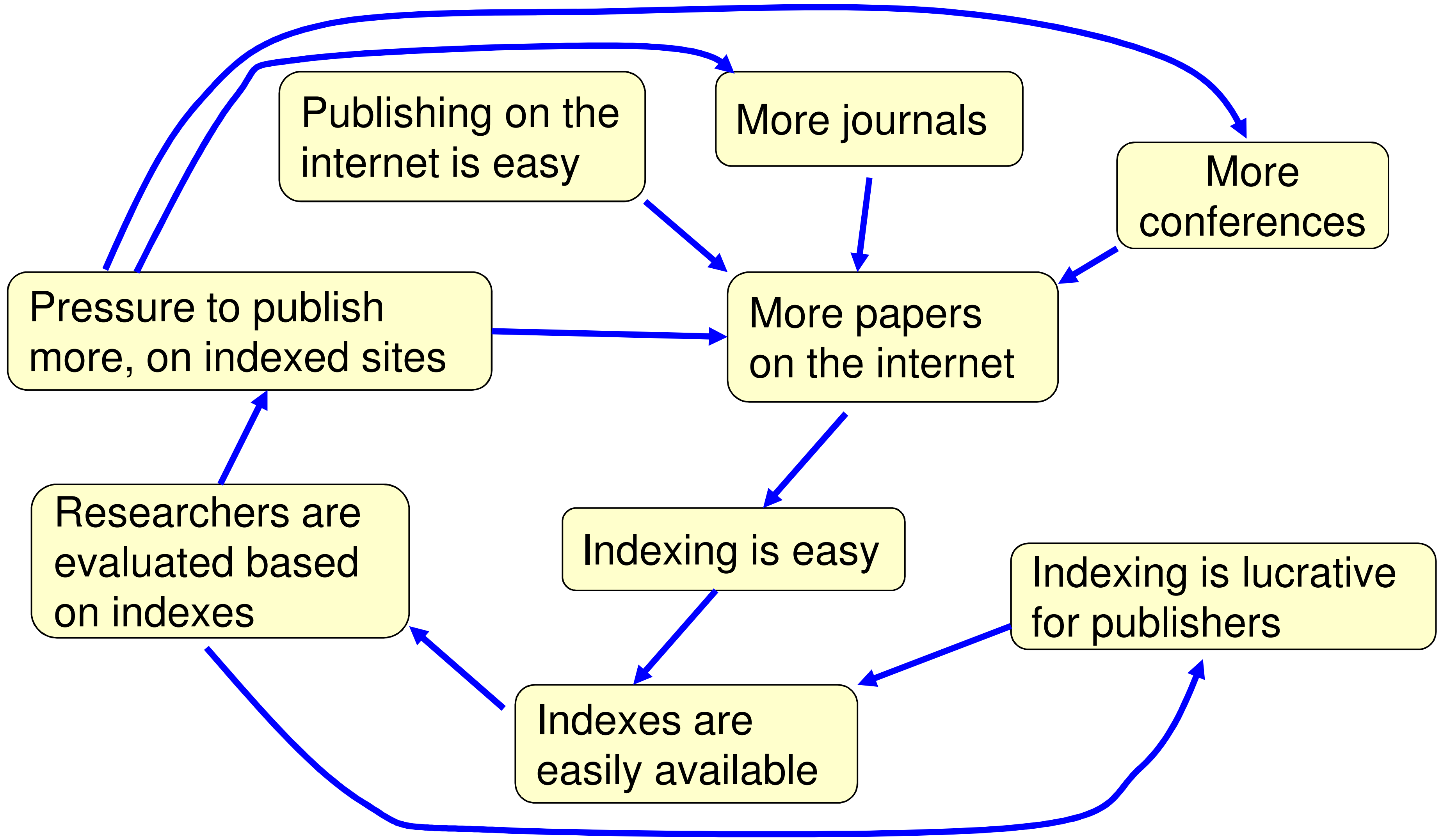
Index-based evaluations

- Evaluating the contribution of a researcher based on the real value of the produced research or the real impact of the produced research is difficult.
- Evaluating the contribution of a researcher based on number that can be easily retrieved from the internet is easy

- Most institutions evaluate researchers based on indexes, just because many indexes are easily available
  - E.g., the number of publications, the number of citations, the H-index, …

- Funding and careers depend on these evaluations

The vicious cycle


Publishing on the internet is easy
More journals
More conferences
Pressure to publish more, on indexed sites
More papers on the internet
Researchers are evaluated based on indexes
Indexing is easy
Indexing is lucrative for publishers
Indexes are easily available

The result

- Quantity ↑
- Quality ↓
- Practical impact on real software development: for most papers, NIL

What can we do to improve the situation?

- We should break the vicious circle.
- How?
- Improving the evaluation of research
  - Making evaluations less dependent on paper indexes
  - Focusing on the real value of research
    - Recognizing the impact on the real software development practice
- Reducing the amount of bad papers
  - Removing the need for authors to publish a lot of papers
  - Accepting less irrelevant or flawed papers
    - Involving actual developers in the publishing process?
    - Establishing evaluation mechanisms for conferences and journals NOT based on current indexes?

Anything good and useful?

- Is there anything good in the many published papers dealing with Empirical Software Engineering?

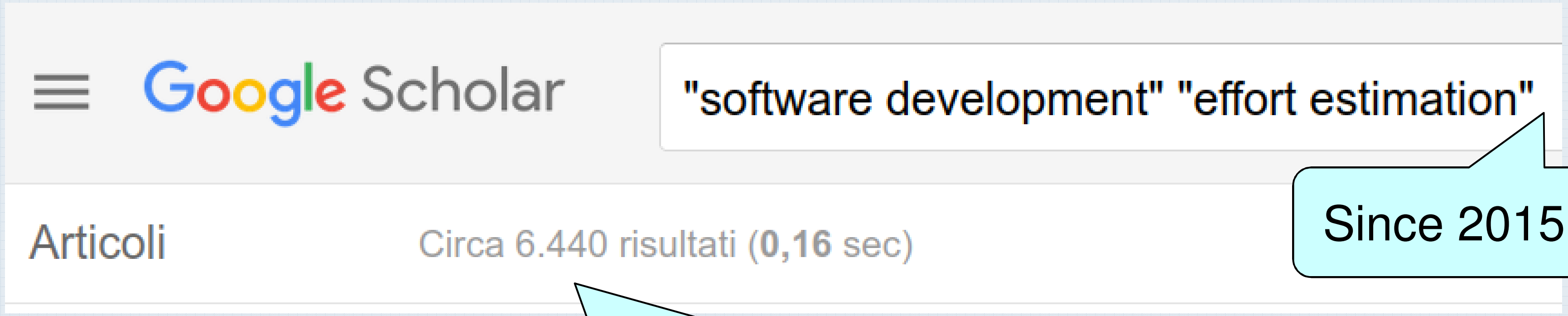
Several papers provide good results, but …

- Very often, results are
  - Very specific, i.e., they concern a very small subset of the potentially interesting situations
  - Very fine-grained, i.e., they concern details rather than a broad scope

- In general, all this research does not provide clear answers to the questions that practitioners may ask.

Example: the case of effort estimation

- Estimate the effort required to develop the requested software is among the most important objectives of software managers.
- Accordingly, a great deal of research has been and is still being dedicated to this topic.


Google Scholar

"software development" "effort estimation"

Articoli

Circa 6.440 risultati (**0,16** sec)

Since 2015

Over 850 papers per year!

Example: the case of effort estimation

- A practitioner would typically ask:
  "considering that I have to develop a program in domain X, according to lifecycle Y, using the development environment Z, with a development team characterized as W, how should I estimate the effort and time required?"
- The available answers make still reference to well consolidated models (e.g., COCOMO II)
- The impressive amount of research produced in the recent years has not been converted into knowledge that can be used in practice.

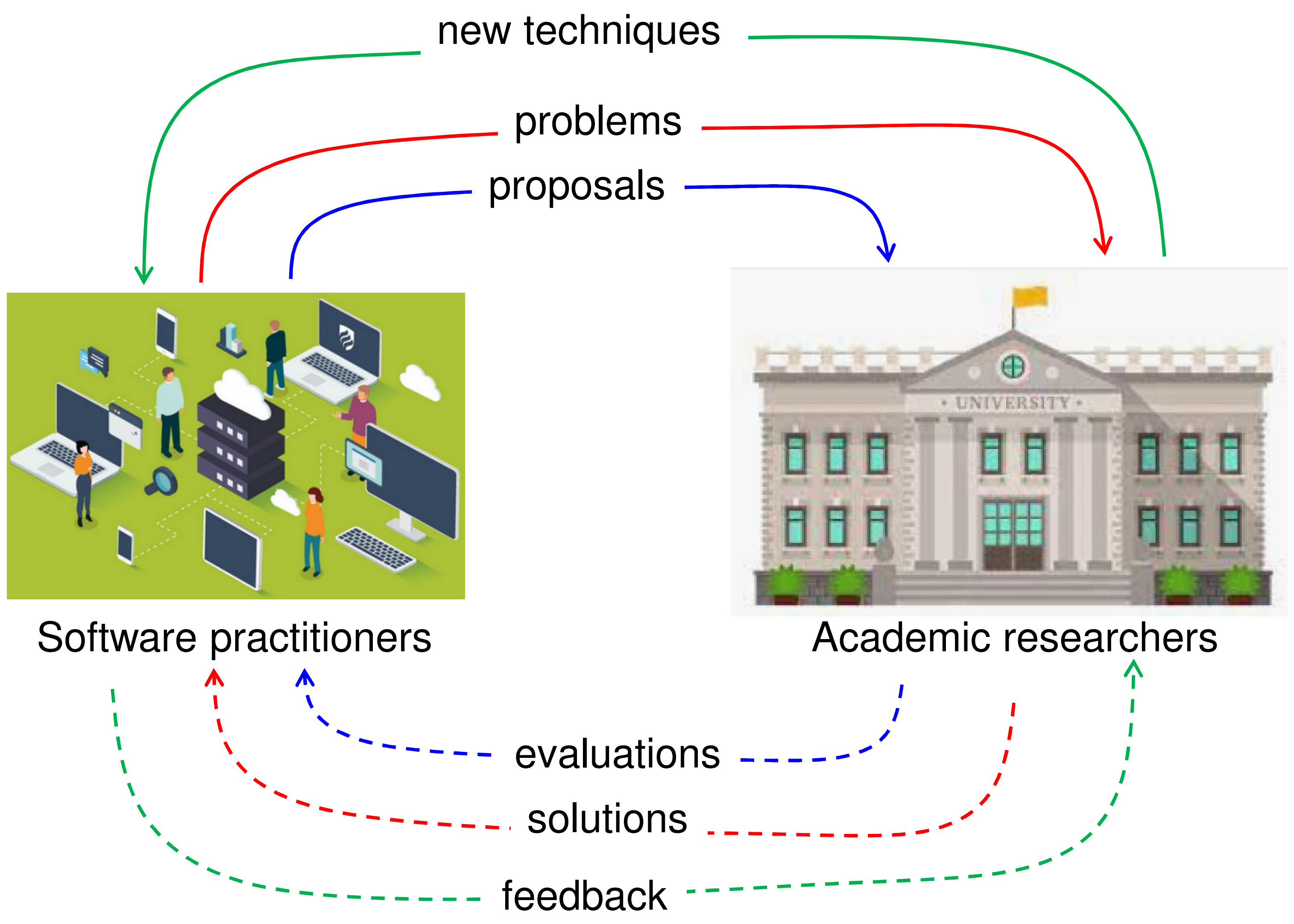
What is lacking?

- We need to “put together” and “distil” the results of research, so that usable knowledge is derived.

- Problems:
  - It is difficult
  - It involves dealing with even thousands of papers

- From the point of view of researchers
  - The result is uncertain
  - If something useful for practitioners is produced, it is not certain that it will be cited, thus contributing to the index moloch.
  - Writing yet another specific & fine-grained paper is so much easier!

The virtuous cycle



The importance of industry & practitioners

- Industry & practitioners should be more involved
  - In proposing topics, needs, etc.
  - In evaluating research
    - Mainly practically
- How?

The importance of industry & practitioners

- How to involve software professionals in research evaluation?

- At a very abstract level the answer is simple:
  - provide them with something useful that they are interested in using and providing feedback about
  - For instance:
    - “executive summaries” that provide a quick-to-read and clear description of research results
    - Code they can readily use to apply the proposed techniques
    - …

Suggestions are welcome!

An example of schematic summary

- Let us see how we could provide a schematic description of the results of the papers
  - Luigi Lavazza, Angela Locoro, Roberto Meli, “Using Locally Weighted Regression to Estimate the Functional Size of Software: a Preliminary Study”, IARIA congress 2022.
  - Luigi Lavazza , Angela Locoro, Geng Liu, Roberto Meli, “Using Locally Weighted Regression to Estimate the Functional Size of Software: an Empirical Study” submitted to J. on Advances in Software

An example of schematic summary

- Topic
  - Functional size estimation
- Addressed problem
  - How to get functional measures (namely IFPUG Function Point) without applying the standard method
    - To save time and money
    - To get measures when software specifications are not yet fully detailed
- Precondition
  - Historic data are available
    - At least 40 project data
    - For each project: size in IFPUG FP, number of EI, EO, EQ, ILF and EIF

An example of schematic summary

- Methods used
  - IFPUG HLFPA
  - Linear regression
  - LOESS (Locally Weighted Regression)
  - IFPUG SFP
- Research description
  - Two datasets (ISBSG from a single organization, and “Chinese” from multiple organizations) were used to compare the estimation accuracy provided by the mentioned methods
  - Evaluation procedures:
    - Within-dataset: subsets of a dataset were used to estimate the remaining projects of the same dataset
    - A dataset was used to build an estimation model that was then applied to the other dataset

References should be given

An example of schematic summary

- Research results

Estimation model	Within-dataset	Cross-dataset
HLFPA	8.8%–10.6%	8.8%–10.6%
Lin. Regr. 4 param.	6%–9.5%	8.4%–10.6%
Loess 4 param.	6.4%–9.4%	7.3%–19.2%
SFP	8.9%–11.4%	8.9%–11.4%
Lin. Regr. 2 param.	9.6%–9.7%	NA–11.2%
Loess 2 param.	8.9%–9.4%	11.4%–11.7%

- Provided support
  - The R code to build estimation model is available
  - URL: …
- For further information:
  - Luigi.Lavazza@uninsubria.it

A legend
should be given

How can we deal with fragments of knowledge?

- I wrote papers on Functional Size estimation using
  - Traditional “fixed weight” estimation
  - Machine learning
  - LOESS
  - …
- We cannot expect practitioners to read all these (and other authors’) papers.
- A synthesis is needed
- How?
  - SLR not suited
  - What kind of paper that provide clear and practical suggestions could be accepted?
    - Anecdote: negative results considered “not actionable”

But then you have many schematic summaries

- What is missing:
  - Collection, merging, connection, classification, …

- Maybe we should go back to past proposals. E.g., the experience factory by Vic Basili.
- Today we need a knowledge factory.

Thanks for your attention!

Questions?