Smart Tools in Software Engineering: A Systematic Mapping Study

Dmitrii Savchenko, Jussi Kasurinen and Ossi Taipale
LUT University / School of Engineering Science (LENS), Lappeenranta, Finland
dmitrii.savchenko — jussi.kasurinen — ossi.taipale@lut.fi

***Abstract*—Software development processes such as waterfall development model have been around for over fifty years, but still, even modern software development approaches, such as DevOps or Test-driven development, fundamentally rely on the same principles and phases as everything before them. Yet, the modern world imposes new challenges for software businesses, and new ways of digital distribution require new ways of resource provisioning and ability to reduce the time-to-market to its absolute minimum. In this study, we identify and analyze the technologies which may be useful for software companies to ease the development and maintenance work by assisting the experts to collect relevant information and observe issues before they cause process disturbances. As a result, we describe a mapping study, which identifies different approaches to developing a smart software engineering tools applying potential technologies such as artificial intelligence, cloud-based service models, adaptive measurement, and other approaches, which could offer significant benefits to the software development process.**

I. Introduction

The software development environment is a socio-technical system [1], meaning that the process activities in the development of software are related to both the human aspects and technology. From the viewpoint of organizations, the software development principles have changed only to a small degree during the last fifty years; the waterfall approach was already known in the seventies [2], and even the more recent Agile methods are based on the project management principles first applied during the same era [3]. Even the innovative process models such as DevOps or Continuous Integration are more or less successful implementations of principles first presented in early agile models such as Extreme Programming (XP) [4].

However, the difference between the formal methods and methods-in-use has been acknowledged at least since from the nineties (for example [5]). In fact, there are survey studies from the software industry which indicate that the amount of formal methods and certified practices are in decline [6]. Similarly, the new business models such as digital distribution, self-publishing and free-2-play, with the technological innovations such as cloud computing and powerful tablet systems, change the way the software companies should operate, what types of tools they actually need [7] and the design challenges these companies face. In the more general level, there also is pressure on the managerial scalability aspects, to which concepts like Scrum of Scrums (for example [8]) try to find an answer. On the software engineering tools, even the established and traditional tools such as the UML notation face criticism for being unnecessarily complicated and hard to use efficiently (for example [9]). Overall, it seems that the technological innovations, new business aspects and organizational change needs will present a challenge to the software organizations.

To assess these challenges, we formulated a systematic mapping study based on the question how could the technological innovations affect the software development ecosystems and software processes?. To approach this problem, the following sub-problems were been identified:

- What technological innovations are currently studied to affect the software business in the near future or have recently affected the software industries?
- What are the recent modern software engineering tool-related innovations?
- How could the end-product quality and maintainability be promoted with the smart tools? What technological innovations would actually be useful for the software industry in practice?

The approach to answer to these research questions was in literature-based systematic mapping study (SMS), applying the widely accepted SMS process method to assess the recent innovations and scientific works regarding the different types of smart tools and smart development environments of the software engineering discipline.

Rest of this paper is structured as follows: Chapter 2 discusses the related works and concepts, and Chapter 3 presents the systematic mapping study approach. Chapter 4 presents the results and observations made from the literature, and the Chapter 5 discusses the meanings, risks and implications of the results. Finally, Chapter 6 closes the paper with summary and conclusions.

II. Related Work

The difference between the formal methods and methods-in-use has been acknowledged at least since from the nineties (for example [5]). In practice this means that the state-of-the-art software development and real-life-applied software process tends to supersede the existing codified standards and practices. Additionally, there are concerns that all process models have shortcomings [10], and that the process models tend to be really laborious to implement successfully even in the mature and professional organizations [11]. Similarly, even traditionally well-known and applied software design tools such as the UML2 notation face criticism for being

unnecessarily complicated and hard to use efficiently (for example [9]). For some areas of software industries, they even lack central components such as GUI design notations, or support for non-functional requirements [12]. In this sense, the search for more efficient software engineering tools and smart systems to aid the development work would be welcome.

In a survey study by Hynninen et al. [6] the different types of software development and quality assurance tools and processes were analyzed. Their main discovery was that while the amount of different types of development and testing tools was generally on the rise, while the amount of different formal process models and application level of certification training was on decline. Similar observations were made also by, for example, a survey by Garousi and Zhi [13]. In both studies, the rising trends in general were the agile approaches, with tools promoting automated activities and faster reaction times to the shifts in the customer demands or system requirements. Additionally, some areas of software industry, such as the entertainment software industry, do not necessarily even benefit from the application of the tools designed with the traditional software engineering mindset [7], and in any case, the major component of the software lifecycle costs is the maintenance, not the pre-launch development work [14].

III. Research approach

To assess the current state and recent trends of the smart tools for software engineering studies, one of the first steps in the systematic mapping study was to identify the appropriate venues and sources from which to collect the data material [15]. After a few trial searches with several databases and search engines such as ACM digital library or IEEExplore to assess the feasibility of our search terms, we decided to focus on the Google Scholar as the main data source because of its domain-neutral approach and large database. The utilization of the separate scientific publishers would have required searches from various different sources, leaving us with possibly biased results since these databases do not cover other works or other publication venues other than the ones the publisher endorses. The utilization of Google Scholar gave us less biased results while it cannot access absolutely all of the existing scientific data - it has a large pool of documents spanning probably most of the different domains from any research paper search portals. This approach also minimized the amount of duplicates and near copies, different versions of the same paper.

The objective of a systematic mapping study is to refine and summarize the collected data with a further analysis to present publishable results. In our mapping study, we applied the process model as defined by [16], since it is well-known and applied in software engineering domain. The Petersen model has six process steps; definition of the scope, search for paper pool, identification of primary documents, classification, data extraction and documentation. These process activities are summarized in the Table 1.

The research question definition was accomplished by assessing the amount of recent works discussing the smart tools and software engineering in the different publication venues. This research work was conducted by one of the authors, who also established the inclusion and exclusion criteria. After it was confirmed that there were significant document pool for the scope of the study, the decision to commit to a SMS study was made.

The search for primary studies was conducted on Google Scholar by formulating a list of rules and keywords. The initial search keywords were smart tools and 'software engineering' in the title or abstract of an article. This was a logical starting point as we were trying to gain knowledge on the recent studies regarding smart tools and technologies. Since the topic was very open-ended, we also added "smart IDE" to the second round of search to expand the document pool. Additionally, since we wanted to observe what were the recent trends and works in the smart tool research, we decided to add publication year limitation of "after 2015" to our research requirements. These search rounds are summarized in the Table 2.

These searches were conducted in January 2019 and were not restricted by any rules or filters except the limitations mentioned earlier, after 2015 and in English. That means in this stage we included all the articles regardless of, for example, the publication forum, type or the discussed technology or purpose.

We did not conduct any manual search beyond pilot studies with keyword feasibility, nor did we perform snowballing, since recent study by Petersen et al. [16] has found that database search is the most used and appropriate approach, with snowballing with far less used strategy. In our case, it was certain that we would have found more potential documents via snowballing, but since we wanted to focus on the most recent studies, we assessed that snowballing the reference lists would not have been very useful in our case since the recent documents probably do not reference other recent documents.

The screening and identification process was conducted manually by the research group. In this step the objective is to identify the primary documents from the pool of all documents generated by the search process. We followed the inclusion criteria principles similar to the Petersen et al. recommendations [15]: If the abstract explicitly mentions smart tool, smart IDE and software engineering or something similar, the paper was included into the primary documents. Only in the cases, where the appropriate terms appear in the keywords or index terms, but the abstract clearly indicates that the research work does not discuss software engineering, the paper was excluded. In this step, some sanity checks such as removal of the duplicate entries, and removal of partial, broken or non-peer-reviewed entries was also conducted.

Keywording by abstracting was conducted by one of the authors manually to minimize the amount of misclassifications. The keywording was done in two steps; in the first step all of the papers are classified by their content and context. After the initial set of keywords are given, they are grouped into larger abstract terms to establish categorizations. The categorization process is fairly similar to the open and axial coding steps of

TABLE I
SYSTEMATIC MAPPING STUDY STEPS, AS DEFINED BY PETERSEN ET AL. [15]

Step #	Process Activity	Output
1	Define research question, identify relevant forums	Defined the scope of the study
2	Conduct search for primary studies in the topic	The paper pool of the study (all papers)
3	Screening and identifying primary papers	The pool of primary papers
4	Keywording using abstracts	Classification of the papers
5	Data extraction and mapping process	Systematic map, statistical data
6	Documentation of the systematic map, analysis of results	Systematic mapping study

TABLE II
SEARCH ROUNDS AND CRITERIA FOR THE INCLUSION TO THE PAPER POOL. RESULTS FROM GOOGLE SCHOLAR

Round #	Search string	Other criteria	Documents
1	”software engineering” AND ”smart tools”	In English, since 2015, publication, citation data or patent	117
2	”software engineering” AND ”smart IDE”	In English, since 2015, publication, citation data or patent	7

the Straussian Grounded Theory [17]. In addition we applied a classification scheme and data extraction form to minimize the noise and ambiguity generated by having different types of documents and origin domains in the document pool.

Data extraction and mapping process consisted of several different measurements and metrics. The systematic mapping of the primary documents and their keywords were compiled for the visual presentation of our work by assessing the application domain besides software engineering, the type of research work done, the evidence provided by the work and the applied technology. These aspects were mapped and documented to a number of tables and diagrams which are presented in the next chapter.

IV. RESULTS AND OBSERVATIONS

Based on our analysis of the recent works on smart tools of software engineering, our research team identified a pool of 124 different documents consisting three scientific books which were compilations of scientific articles. By including these book chapters as separate documents, a total of 137 different documents describing the different types of documents discussion software engineering and smart tools was discovered. Out of these document, 48 documents (35.0 percent) were books or book chapters, 40 documents (29.2 %) journal articles, 22 (16.9 %) conference papers, 17 (12.4 %) thesis works and 10 (7.3 %) patents.

After classification and assessment, we accepted 54 documents as our pool of primary papers since they discussed software engineering and smart tools in some degree or form, bringing the inclusion rate to 39.4 percent. Since this study only included scientific sources, patents and thesis works were excluded from the further analysis regardless of the topic they discussed. Out of the primary documents, majority discussed one or more of the most five most common categories: Internet of Things or big data in general, metrics generation tools to assist design or development, different forms of artificial intelligence, recommendation tools to manage process or system analysis work either by analyzing the process or the architecture with smart tools. These are listed in Table 3. In addition, 11 papers were technology or solutions reviews in general, discussing the existing systems and existing smart solutions to solve different domain- or software engineering related development process issues.

TABLE III
THE MOST COMMON IDENTIFIED SMART SE-RELATED TECHNOLOGY CATEGORIES

IoT, Big data or distributed systems	12 (22.2 %) out of 54 papers
Metrics generator, programming or design assistance tool	11 (20.4%) out of 54 papers
Artificial intelligence or machine learning	11 (20.4 %) out of 54 papers
Recommendation tool (knowledge management, classification system or root cause analysis)	10 (18.5%) out of 54 papers
System analysis tool (process or architecture)	7 (13.0%) out of 54 papers

Observations from the different papers discussing the smart tools of software engineering are summarized in Table IV. The three most common attributes of the all analyzed papers were related to their content; the majority of the publications were theoretical papers discussing the smart technologies in general (64.8 percent) whereas implementations or proof-of-concepts were 29.6 percent of the works. In addition, 51.9 percent of the publications discussed the smart technology in other contexts besides software engineering. From these paper the impact to the manufacturing industry was most common theme, the second place being IoT and Healthcare-related systems.

In the assessment of the impact of smart tools, 9 publications discussed the performance differences between the ’traditional’ systems and the ’smart’ tools replacing them. Six publications included some form of assessment for the costs or needs based on real case study example, while 3 publications included no evidence or measurement data on the impact or efficiency of the applied smart technology.

TABLE IV
THE MOST COMMON OBSERVED ATTRIBUTES OF PUBLICATIONS RELATED TO SMART TOOLS OF SE

Theoretical paper concerning the technologies	64.8% (35)
Implementation, or proof of concept, for smart technology	29.6% (16)
Review of existing technologies and smart solutions	20.4% (11)
Application domain related to Manufacturing industry	18.5% (10)
Application domain related to Internet of Things	16.7% (9)
Application domain related to Healthcare	16.7% (9)
Performance evaluation between 'traditional' and 'smart' system	16.7% (9)
Case-study on costs and need to adopt a smart tool	11.1% (6)
Case study with no evidence or assessment presented on the impact of 'smart' technology	5.6% (3)
Survey or study on the users of smart tools	5.6% (3)

V. DISCUSSION AND IMPLICATIONS

Based on our observations, the results of this mapping study indicate that the term 'smart' is usually understood as some form of 'decision support system' in the publications. Smart tools are usually applied in two fields - Industrial IoT and software engineering. In software engineering, Smart tools are mostly applied in static analysis to generate metrics or further insights into the system structure. In the beginning of this study, our research team devised three main branches of 'smart tools'; 1) the tools which can adapt to solve different types of problems, 2) the tools which provide additional insightful information and 3) the tools, which guide the user and prevent common types of errors. Based on our observations, the recent studies seem to focus on the second category, providing more insight to help process management or to gain more useful information from the performance and architecture of the existing system. Based on our original research question *how could the technological innovations affect the software development ecosystems and software processes?* the results seem fairly straightforward: the current trends of smart tools seem to work towards providing more metrics and more information to the developers, with the IoT and big data-based solutions, and artificial intelligence-based metrics generators, being common topics. Based on these observations a tool which can create novel and accurate information from the current state of the system could be one of the outcomes combining the current trends, and enabling improvement of the overall quality of the software products and services it measures.

Obviously, this SMS study is not perfect and it has some shortcomings. For example, Grant and Booth [18] define systematic mapping study to have weaknesses in the lack of synthesis and in-depth analysis especially when compared against other research approaches, broad overall descriptive level and risk of oversimplifying the results of the studies. Overall, these limitations cannot be disputed, but they are recognized and there are ways to minimize their risk for the study validity. In our case, we selected a search tool which was not domain-specific, and avoided defining the "smart tool" to very restrictive degree. Basically, our only requirements were that the publication discussed software engineering-related topic, and was recent work, while self-identifying as discussing "smart tools". In general, the decision of using Google Scholar as the sole source of documents due its convenience, low cost, and broad coverage has been discussed by [19] and considers this approach of acceptable quality. For criticism, Giustini & Boulos [20] argue that the Google Scholar finds only approximately 95% of the articles in the controlled test group, and includes a large number of irrelevant objects. For our purposes of finding out what the "smart tools of software engineering" are, the accuracy of 95 percent was acceptable, since our aim was to gather articles from as wide range as possible instead of precision since we deliberately did not want to narrow the search criteria too much. From the 137 documents we ended up rejecting 83, mostly because they were not academic or research-based sources, or they did not discuss topics related to the software engineering discipline. Additionally, we did not evaluate the quality of the found documents in this study. As suggested by Petersen et al [16] and Kitchenham & Charters [21] the quality of articles should not be considered a major component in the systematic mapping studies, since the objective is to collect general information regarding the topic and research directions related to it, not to conduct an in-depth systematic literature review, which analyzes and discusses the publications and trends in detail.

VI. CONCLUSIONS

This systematic mapping study identifies the different types of smart tools and smart technologies, which are the current trends of software engineering research, while assessing how these technologies could assist the development and support of the software products and services. The need for this study origins from the observation that the applied tools, infrastructure and business models are evolving, and already now some areas of software industry are unable to successfully apply the organizational model intended to enable better organization practices and reach the intended end-product quality, or lack feasible tools to observe the overall state of their system. In general, the existing tools may not support the actual real-life needs, or at least could be improved.

Based on our observations, the current state of the smart tool research focuses on theoretical works, with a number of implementation projects or case studies discussed. The most "relevant" smart technologies are Internet of Things, big data or artificial intelligence-related tools, in many occasions providing in depth data or measurements to assist the development or design work. Additionally, recommendation systems or analysis tools exist, but the most common types of tools focus on providing novel information over existing domain, not -for example- in error prevention or adjusting to the different types of related problems. Based on these observations, for example smart analyzers and metrics generators as quality assurance assistance tools were proposed.

The application of these results could help in the development of new operational models or development tools to supplement the existing industry-adopted software development ecosystems. In any case, the results provide insight into the ways to further develop the software engineering tools and support systems towards the more technology-adapted approaches, and pinpoint areas of interest and innovations for the research of modern computer-assisted software engineering tools.

ACKNOWLEDGMENT

This work was partially funded by the Technology Development center of Finland (Business Finland), as part of the .Maintain project (project number 1204/31/2016).

REFERENCES

[1] F. W. Geels, "From sectoral systems of innovation to socio-technical systems," *Research Policy*, vol. 33, no. 6-7, pp. 897–920, Sep. 2004. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0048733304000496

[2] W. W. Royce, "Managing the development of large software systems: concepts and techniques," in *Proceedings of the 9th international conference on Software Engineering*. IEEE Computer Society Press, 1987, pp. 328–338.

[3] C. Larman and V. Basili, "Iterative and incremental developments. a brief history," *Computer*, vol. 36, no. 6, pp. 47–56, Jun. 2003. [Online]. Available: http://ieeexplore.ieee.org/document/1204375/

[4] S. Stolberg, "Enabling agile testing through continuous integration," in *Agile Conference, 2009. AGILE'09.* IEEE, 2009, pp. 369–374.

[5] H. v. Vliet, *Software engineering: principles and practice*, 3rd ed. Chichester, England ; Hoboken, NJ: John Wiley & Sons, 2008, oCLC: ocn191758457.

[6] T. Hynninen, J. Kasurinen, A. Knutas, and O. Taipale, "Software testing: Survey of the industry practices," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija: IEEE, May 2018, pp. 1449–1454. [Online]. Available: https://ieeexplore.ieee.org/document/8400261/

[7] J. Kasurinen, J.-P. Strandn, and K. Smolander, "What do game developers expect from development and design tools?" in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering - EASE '13*. Porto de Galinhas, Brazil: ACM Press, 2013, p. 36. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2460999.2461004

[8] M. Paasivaara, C. Lassenius, and V. T. Heikkil, "Inter-team coordination in large-scale globally distributed scrum: do scrum-of-scrums really work?" in *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '12*. Lund, Sweden: ACM Press, 2012, p. 235. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2372251.2372294

[9] M. Grossman, J. E. Aronson, and R. V. McCarthy, "Does UML make the grade? Insights from the software development community," *Information and Software Technology*, vol. 47, no. 6, pp. 383–397, Apr. 2005. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S095058490400134X

[10] K. N. Rao, G. K. Naidu, and P. Chakka, "A study of the agile software development methods, applicability and implications in industry," *International Journal of Software Engineering and its applications*, vol. 5, no. 2, pp. 35–45, 2011.

[11] M. Staron, "Adopting Model Driven Software Development in Industry A Case Study at Two Companies," in *Model Driven Engineering Languages and Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 4199, pp. 57–72.

[12] J. Kasurinen and K. Smolander, "What do game developers test in their products?" in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '14. New York, NY, USA: ACM, 2014, pp. 1:1–1:10. [Online]. Available: http://doi.acm.org/10.1145/2652524.2652525

[13] V. Garousi and J. Zhi, "A survey of software testing practices in canada," *Journal of Systems and Software*, vol. 86, no. 5, pp. 1354–1376, May 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121212003561

[14] B. W. Boehm, *Software engineering economics*, ser. Prentice-Hall advances in computing science and technology series. Englewood Cliffs, N.J: Prentice-Hall, 1981.

[15] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering." in *EASE*, vol. 8, 2008, pp. 68–77.

[16] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[17] J. M. Corbin and A. L. Strauss, *Basics of qualitative research: techniques and procedures for developing grounded theory*, fourth edition ed. Los Angeles: SAGE, 2015.

[18] M. J. Grant and A. Booth, "A typology of reviews: an analysis of 14 review types and associated methodologies: A typology of reviews, *Maria J. Grant & Andrew Booth*," *Health Information & Libraries Journal*, vol. 26, no. 2, pp. 91–108, Jun. 2009. [Online]. Available: http://doi.wiley.com/10.1111/j.1471-1842.2009.00848.x

[19] D. Rosenstreich and B. Wooliscroft, "Measuring the impact of accounting journals using Google Scholar and the g-index," *The British Accounting Review*, vol. 41, no. 4, pp. 227–239, Dec. 2009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0890838909000614

[20] D. Giustini and M. N. Kamel Boulos, "Google Scholar is not enough to be used alone for systematic reviews," *Online Journal of Public Health Informatics*, vol. 5, no. 2, Jun. 2013. [Online]. Available: http://journals.uic.edu/ojs/index.php/ojphi/article/view/4623

[21] B. A. Kitchenham, "Systematic review in software engineering: where we are and where we should be going," in *Proceedings of the 2nd international workshop on Evidential assessment of software technologies*. ACM, 2012, pp. 1–2.