Automation and Artificial Intelligence in Software Engineering: Experiences, Challenges, and Opportunities

Milan Latinović
Graz University of Technology, Graz, Austria
milan.latinovic@ieee.org

Viktoria Pammer-Schindler
Graz University of Technology, Graz, Austria
viktoria.pammer-schindler@tugraz.at

Abstract

Automation and Artificial Intelligence have a transformative influence on many sectors, and software engineers are the actors who engineer this transformation. On the other hand, there is little knowledge of how automation and Artificial Intelligence impact software engineering practice. To answer this question, we conducted semi-structured interviews with experienced software practitioners across frontend and backend development, DevOps, R&D, integration, and leadership positions. Our findings reveal 1) automation to appear as micro-automation in the sense of automation of tiny and specific tasks, 2) automation as a side product of work, and bottom-up driven in software engineering, and 3) automation as a possible cause for cognitive overhead due to automatically generated notifications. Furthermore, we notice that our interview participants do not expect automation and artificial intelligence tools to change software engineering's essence in the foreseeable future substantially.

1. Introduction

Automation and Artificial Intelligence have a transformative influence on many sectors. As a salient example, the transformation that manufacturing is still undergoing due to increased automation and deployment of data analytics and Artificial Intelligence has been coined Industry 4.0 or smart manufacturing [1, 2, 3]. This transformation is characterized by the convergence of sensor-enabled and networked things in production (Internet of Things), and thereby enabled large scale data collection, on the one hand; and capabilities for large scale, partially online, data analytics, often enabled by methods from Artificial Intelligence. This provides fine-granular process alignment and optimization, in parallel to increasing automation of production processes.

Beyond this, the automation and Artificial Intelligence are discussed as potentially transformative in other sectors, such as medicine or financial auditing, both knowledge-intensive sectors that are substantially based on expert work. For instance, in medicine, the performance of machine-learning computer-aided detection and automated diagnosis in radiology is comparable to that of experienced radiologists [4]. Similarly, auto-encoder neural networks can help financial auditors identify unusual journal entries in an audit context [5].

In this paper, we understand automation as a set of computational tools executing domain activities, based initially on conditional logic [6]. Thus, automation takes over activities done by human workers; or automation enables actions that humans cannot do. Through automation, tasks are carried out with different qualities, such as increased precision or reduced flexibility in the face of exceptions. By Artificial Intelligence, we understand a broad range of technologies and technological capabilities based on processing natural language, audio, images, and video; formal representations of knowledge and automated reasoning; and machine learning (cp. e.g., [7]). In this paper, we do not strictly differentiate between technologies that employ more traditional statistical methods of data analysis and machine learning. By now, automation tools in many sectors have taken up Artificial Intelligence technologies to a substantial degree, such that automation does not anymore mean that computational tools execute comparably static rules and equations. Nevertheless, instead, computational tools execute activities based on sophisticated and advanced computational methods from Artificial Intelligence.

Subsequently, humans' role in increasingly automated environments is mostly that of monitoring, handling exceptions, and overall managing automated services [8]. As "work becomes less predictable, less repetitive and more complex" ([9] in the context of smart manufacturing), work can be expected to become more knowledge-intensive, and learning at all levels to gain in importance. In parallel, this is likely to lead to

URI: https://hdl.handle.net/10125/70628
978-0-9981331-4-0

a polarization in the labor market. Those professions that demand high education and expertise, high manual dexterity, or high quality of social interaction (areas where modern computational technologies perform worse than humans) will remain or even rise in demand. In contrast, others need to expect to sink in the market as the automation costs decrease [10, 11].

Alongside automation and increasingly Artificial Intelligence enabled tools, it is now acknowledged that as computational tools grow in intelligence. Thinking broadly about machines as teammates is on the agenda of ongoing and future information systems research [12], which introduces socio-technical challenges, such as how machines affect decision-making, pace, and quality of social work, creativity, responsibility, and individuality (ibid).

Software engineers are - in principle - the actors who engineer such transformation. However, not every software engineer is an AI expert, or even working with AI. At the same time, the current role of automation and Artificial Intelligence enabled tools in software engineering practice is barely touched on in information systems or related research; at a time where technology-oriented research is, of course, already working on Artificial Intelligence enabled tools for software engineers (e.g., [13, 14, 15, 16, 17, 18, 19, 20, 21]).

Therefore, in this paper, we complement the existing discourse on the presence and future of work in the face of artificial intelligence an interview study that asks experienced software engineers about the role of and experiences with automation in their practice, and their expectations of the future role of Artificial Intelligence enabled tools for their work practice.

2. Background and related work

2.1. Challenges in software engineering practice

By software engineering, we here understand the professional practice of designing, developing, and maintaining software. Daily work practice of a software engineer typically falls into two major groups of work activities, namely coding and collaborative tasks, identified in Computer Support Collaborative Work (CSCW) literature as ”articulation work” - all that work that is needed to make cooperative work via division of labor happen [22].

Overall, software engineering professionals are independent workers who need an excellent capability to focus on productive work without a high amount of distractions [23], need to be able to solve problems with acceptable coding practices and without time pressure [24], as well as to communicate well in different contexts [25]. As software complexity increases and teams expand, communication and coordination get more involved, manifesting through the distribution of time over these two types of activities shifts from coding to articulation work [26, 27].

A variety of coordination tools and approaches such as retrospectives, rotation of team members, wiki pages for documentation, and digital boards [26] have already been investigated in order to solve typical challenges that software engineers face, such as increasing productivity, reducing time wasters, reusing code [28], and reducing technical debt [29].

However, while software engineering’s work practice and challenges are discussed within the field, the role of automation and artificial intelligence-enabled technologies in either addressing or adding to the software engineers’ challenges at work has not been discussed in the literature. This discussion is the gap that we explore in the present paper.

2.2. Opportunities for automation in software engineering

In literature, we find automation as playing a role in two phases of software engineering work: That of coding in the sense of problem-solving; and that of coding in the sense of integration of a specific piece of code into a larger whole.

The first phase of coding in the sense of problem-solving, means that a software engineer is focused on understanding and solving a specific problem without focusing on the broader software environment (except where necessary) like integration or software architecture. In this phase, tools like a programming assistant (PAPA) powered by IBM Watson’s cognitive computing technology [14] would be beneficial. PAPA can respond to theoretical questions based on NLP for a predefined knowledge base, but it does not support debugging code examples using program analysis techniques. However, a recent investigation on software engineer motivations and challenges to use Machine Learning frameworks highlighted the importance of having pre-made models that demonstrate best practices, useful examples, and support learning-by-doing [30]. This points to the need for artificially intelligent assisting tools to go further than programming assistants like PAPA. Further research is needed to acknowledge automated programming assistants’ possibility to provide pre-made models for specific cases and demonstrate best practices.

The second phase occurs when a software engineer has developed a working solution that now must be

integrated into the code base and validated. The common approach uses Continuous Integration [31, 32, 33] configured according to specific project needs, and accomplished by an automated build that aims to detect errors as soon as possible. *The build* is a process that executes predefined build *modules* which are defined by a *Continuous Integration pipeline*. Execution of pipeline results in either successful or failed build, which commonly includes a report on failure causes, i.e., bad code syntax or failed tests [34].

In a study of 20 open source projects, different Automated Static Code Analysis Tools have been identified as being used in Continuous Integration such as: check coding style; find bugs resulting in recognizable errors; identifying anti-patterns such as unused objects and unnecessary code blocks; reviewing licenses validity and missing license headers; dependency analysis; and compatibility issues recognition [35]. Although CI successfully detected issues in analyzed code, there were instances where CI tools' application and configuration caused false errors. This fact indicates the dual nature of automation tools, which can be useful and helpful, but like any other tool, they can also produce counter-effects if not used properly.

2.3. Artificial Intelligence and software engineering

The scarce research we have found on how Artificial Intelligence is changing software engineering was mostly within decision-support for product owners. For example, Dam et al. [15] proposed an Artificial Intelligence-powered agile project management tool, based on deep learning architecture, capable of assisting product owners and developers by identifying and refining backlog items, risk prediction, mitigation, and efforts estimation.

Another research from Dam et al. [17] proposes a concept where the Long Short Term Memory model (LSTM) is used to automatically learn semantic and syntactic features, which would lead to the possibility to recognize and predict code vulnerabilities.

Similarly, Choetkiertikul et al. [16] utilized LSTM to build a prediction system that supports developers and managers in estimating story points, where machine learning algorithm learns from previous estimates and supports teams in making consistent estimates by providing them with suggestions where teams make the final decision.

While [15, 17, 16] do discuss that Artificial Intelligence enabled tools that they proposed will act as support for software engineers rather than as a replacement of software engineers, they do not focus on how Artificial Intelligence-powered decision-support would affect software engineering teams, in the sense of studying who would be supported by such a tool (project managers or software engineering teams), or how such a tool would compare with other existing tools.

Another set of technologies from artificial intelligence with a potentially transformative effect on software engineering practice is formal verification, which automatically detects mismatches between program specification and implementation. For instance, tools such as the deductive software verification engine [21] can assist software engineers in debugging by recognizing and reporting which expressions might be responsible for errors that appeared in the source code.

Overall, however, in an overview study, Feldt et al. recognize the lack of a structured way to classify Artificial Intelligence applications on software engineering [20]. The authors propose to differentiate between different artificial intelligence roles in software engineering as follows: The point of Artificial Intelligence application, the type of Artificial Intelligence technology used, and the automation level allowed an interesting artifact. In parallel, the authors differentiate between using artificial intelligence as being applied at the process, product, or run-time to enable manual or autonomous automation. For example, independent change in run-time would be considered a higher risk, while automation based on the manual decision would be the lowest risk level. Feldt et al. conclude that most of the examined papers focus on supporting SE during the development phase but with uneven use of Artificial Intelligence approaches that exist. The authors conclude that artificial intelligence is not yet massively used in tools for programmers, but more at the level of embedding Artificial Intelligence into decision-supporting and decision-making processes.

Our study contributes and builds upon these findings by conducting interview-style research to get more information about Artificial Intelligence tools and use cases in software engineering. Interviewing seasoned software practitioners, recognized as the primary beneficiary and stakeholders affected by possible automation and AI, caused improvements.

2.4. Synthesis and research questions

In summary, we see that although automation and Artificial Intelligence-based tools are understood as transformative in many sectors and for many types of professions, there has not been an investigation of how

ID	Gender	Age	Position	YOE	NOC	Sector	Industry domain	Technology stack
P1	M	25-30	Frontend developer	3	2	Industry	SaaS	HTML, CSS, JS, Python, Git, Sql, Matlab
P2	F	30-35	Backend and DevOps	6	3	Industry	Logistic	Python, Git, Web services, Google Cloud
P3	M	40-45	Net-System Engineer	17	7	Industry	Networking	OS, NetOS, System Software
P4	M	35-40	Research & Backend	10	4	Academia	Security	Java, MySql, Git, SVN, Tomcat, GlassFish
P5	M	30-35	Frontend developer	15	5	Industry	SaaS	HTML, CSS, JS, UX/UI design
P6	M	25-30	Backend Team Lead	5	2	Industry	SaaS	PHP, MySql, Shell, Git, Pre-Commit
P7	F	30-35	Team Lead Sw. Integ.	3	2	Industry	Semiconductors	MongoDb, Python, Nginx, PostreSql, ELS
P8	F	25-30	Backend, Integration	3	2	Industry	SaaS	PHP, MySql, Git, Python, API, RestAPI
P9	F	35-40	Software developer	10	5	industry	SaaS	C#, MsSql, Git, Raact, Redux, WinSrv
P10	F	30-35	Researcher	8	4	Academia	Automotive	Semantic Web, NLP, Python
P11	M	30-35	Software Quality Ing.	9	3	Industry	Ind. Autom.	SW Test Automation, DevOps, Jenkins, C#
P12	M	40-45	Sr. Software Developer	20	8	Academia	SaaS	C++, PHP, JS, Arudino, Ruby, Go
P13	M	20-25	Software Engineer	3	2	Industry	Hardware	Jenkins, Python, XML, JSON, HDL, Linux

Table 1. List of software practitioners we interviewed: F=female, M=male; YOE=Years of Experience; NOC=Number of Companies that software practitioner worked for.

automation and Artificial Intelligence-based tools are used or impact software engineering practice. In this paper, we are taking up this question in two parts:

RQ1 What are approaches for automation and Artificial Intelligence enabled tools and practices adopted by interviewed software engineers? This question aims to understand the current practice.

RQ2 What are software engineers' outlook towards the future of automation and Artificial Intelligence enabled support for their work? This question aims to understand the attitudes, hopes, and concerns of software engineers.

3. Research method

We answer the above research questions with an interview study of 13 software practitioners with different backgrounds and working in various industries and academic institutions. An interview study is considered an appropriate method within an exploratory setting [36].

3.1. Interview structure and content

The interviews were semi-structured, with the following guiding questions to elicit the interviewee's professional background and answer the research questions: *Can you describe your job and what your company does? Which environment and tools are used in development? Which other tools are used for communication and management? Which part of work is automated, and what this automation "do for you"? Which part of work did you automate on your own? How does Artificial Intelligence fit in your activities, in your opinion?*. When introducing questions about automation or AI, examples as discussed above in the background section 2 was given. When asked about expectations for the future, again, these examples were used as prompts to stimulate discussion. Throughout the interviews, we encouraged participants to think about why they did something in a specific way or what could have been better by asking general reflective questions, such as *What was the benefit of this approach? What caused this problem? Could any of these be automated?* We designed the interviews for 45-60 minutes, but on some occasions, they took up to 90 minutes. Participation was not paid but voluntary. After the first interviews, P1-P4, we made a reflection of how interviewees perceive our questions and to recognize possible communication challenges and emerging topics. We aimed to identify which themes are triggering responses where participants knew something potentially novel, such as specific problems or difficulties, innovative ideas, solutions, or a combination of tools and processes to solve some of these challenges. Furthermore, we were interested in engaging our participants in creative discussion in terms of Artificial Intelligence to understand their level of knowledge of this topic and realize how they would utilize Artificial Intelligence in their daily work. Our analysis of the first round resulted in small refinement of our interview questions related to Artificial Intelligence and recognition of one emerging topic, "Who initiated automation in your company? Individual developer, team, or management?".

3.2. Data collection

The lead author conducted interviews via video conferencing and took quasi-verbatim notes simultaneously. These notes were shared with the interviewee directly via screen-sharing. This allowed immediate discussion of notes in case of disagreement by the interviewee. Furthermore, participants were asked to review the notes and provide

their consent afterward. Post-hoc corrections made by the participants were minimal and mostly related to GDPR[1] or NDA (Non-Disclosure-Agreement) compliance topics, i.e., related to personal sensitivity or organizational confidentiality of interview notes. Further, we interviewed professionals on equal standing with the researcher in terms of natural authority. Because of these characteristics, we can assume that the note-taking approach does not affect reliability, validity, or transparency (cp. [37]); on the other hand, this approach saved time and added clarity in understanding interviewees through the immediacy of using the interview notes to discuss any misunderstandings. Interviews were carried out in English.

3.3. Data analysis

Data analysis was done as inductive and reflexive thematic analysis [38, 39]. Our analysis constitutes an inductive thematic analysis, as we intended to identify a priori unknown patterns in the described experiences of our interviewees of how automation and AI support their current work practice; and how they envisage this to be in the future. Further, our analysis constitutes a reflexive thematic analysis, in the sense that themes were developed during the data analysis by discussion and reflection within the research team: We started by describing and summarizing the experiences of interviewees structured along with the types of guiding questions (professional background, current practice, future practice), and in further discussions identified and developed salient themes. The themes correspond to the subsections of Section 4, except for the first subsection 4.1, which is a descriptive account of ”commonly automated tools, activities, and challenges” and sets the background for the subsequently described themes. While reflexive and inductive thematic analysis means that we did not set out to structure data analysis by an a priori defined theoretical framework, the themes can, of course, still be related to and further interpreted with respect to existing literature; which we do in Section 5.

3.4. Interview participants

Our interview participants have experienced software practitioners across frontend, backend development, DevOps, R&D, integration, and leadership positions.
Our selection criteria were seniority (described as in the column ”YOE” abbreviating ”years of experience” in Table 1) in parallel to ongoing hands-on software engineering practice, i.e., we did not want to interview managers of software engineering teams. Further, we aimed for diversity in gender and age, sector (industry vs. academia), industry domain (Software as a Service - SaaS, logistics, networking, software security, semiconductors, automotive, industrial automation, or computer hardware), professional position, and technology stack (ongoing experience with various technologies).
Note that our interview partners had different levels of understanding of artificial intelligence and how it combines with software engineering. None of our participants had work experience in an AI software project. Experience with AI was *not* explicitly a selection or exclusion criterion; instead, our focus was to gather perspectives across the above-described range; and not only from people with direct experience in AI.

Participants were recruited online through the Linked-In network and in-person within corporate environments and conferences. Some participants recommended further potential interviewees.

3.5. Research limitations

Although we aimed to interview software practitioners with diverse demographics and professional backgrounds, our sample has some limitations: All of our participants are from Austria and neighboring countries in South-East Europe. Although territorial limits do not bound the nature of software engineering as a profession, our sample would benefit from a broader demographic. None of our participants worked for big corporations such as Google, Facebook, or Amazon, except P3, who pointed out that his experiences when working for big corporations are different from his experiences related to Small and Medium Enterprises (SMEs). Having this limitation in mind, the results of this research should be treated trough the scope of SMEs, taking into account that corporate experiences can differ from our findings.

4. Results

4.1. Commonly automated activities, tools, and challenges

Across all study participants, we see that automation occurs in many different types of software engineering activities (Table 2). The concrete tools utilized for automation and the precise manner of automation differ. There is no standard for automating similar activities, but there are popular tools that are combined differently. For instance, GitLabs and BitBucket are frameworks to automate server-side test and deployment, and

[1] https://gdpr-info.eu/

Automated activity group	Participants	Common tools
(Dev) Environment Configuration	P1, P3, P6, P7, P12	Docker, custom scripts
Coding, reviewing and compliance	P1, P2, P3, P4, P5, P8, P10, P11, P12	PyCharm, PHPStorm, Eclipse, Ansible
Versioning, Deployment, Releasing	P1, P2, P7, P11, P12	Jenkins, scripts, GitLabs
Tests and Continuous Integration	P1, P2, P5, P6, P7, P8, P10, P11, P12, P13	GitLabs, BitBucket, Pre-Commit, Jenkins, Ranorex, ReSharper
Orchestration and dependency management	P2	Kubernetes
Documentation generation	P1, P8, P10	DoxyGen, OpenApi
Knowledge share, notification and report	P1, P8, P9, P10, P13	Jira, Slack
Business performance tracking	P2, P4	Various BI tools

Table 2. Commonly automated activities, tools and participants that pointed them out

Pre-Commit hooks are used to automate client-side tests before sending to the server. However, there is no strict standard way to combine such tools; single software engineers or teams organize configuration, which constitutes both freedom and a workload.

Our results indicate substantial automation in environment configuration, static code checking, versioning, deployment, testing, continuous integration, and releasing.

Environment configuration is mostly related to either predefined Docker configurations or to custom scripts that automate steps needed for everyday tasks such as setup of development environment for new software engineer, extending infrastructure by adding new servers with predefined configuration or introducing additional dependencies to existing infrastructure.

Testing, versioning, deployment, and releasing activities tend to combine into Continuous Integration pipelines using Jenkins orchestration with custom scripting and CI environment provided by Git services such as GitLabs, BitBucket, or GitHub. Almost all participants execute tests within local environments before sending code to the server. Some participants reported a trend where tests are automated locally on development machines with the help of already mentioned pre-commit hooks framework. This local automation provides a better experience and faster response to developers and reduces the load of integration pipelines.

Developers also rely on automation provided by their Integrated Development Environments such as PyCharm, PHPStorm, Eclipse, or Visual Studio. Automation such as syntax checkers, linting, coding standard checkers, and necessary semantic checks are commonly used and expected as a default commodity.

However, in terms of knowledge creation and knowledge management activities, we have noticed a low level of automation efforts. We asked interviewees explicitly whether they use automated knowledge gathering or knowledge delivery mechanisms, smart search engines, or similar tools for assisting in knowledge management. Interviewees mentioned coordination and communication tools such as Jira and Slack, but only in task management and receiving direct notifications. P1, P8, and P10 confirmed activities related to documentation generation with DoxyGen and OpenApi standards and toolsets.

Most interview participants are not involved in implementing business performance tracking or analyzing data that stems from such tracking, with the exception of P3, who said on this: *[I do this for] "Black Friday" sales, where you need to predict what will be the load on services* and *Network telemetry is using algorithms for Artificial Intelligence and machine learning on Network environments*. By this, P3 means that he was analyzing data provided by an AI service that can predict network traffic to prepare for and support high-increase in traffic around events such as "Black Friday" sales.

Apparently, the business performance tracking and monitoring perspective are outside the sphere of responsibility of our interview partners; and the present interview study can, therefore, do not make statements about this. In particular, our study, therefore, does not indicate a lack of automation or Artificial Intelligence enabled tools for monitoring business performance.

4.2. Processes bridging with micro-automation

Software practitioners have similar groups of activities, such as requirements gathering, coding, testing, documenting, and communicating with other stakeholders. However, processes for enforcing these activities vary between companies and teams. Almost all interviewees mentioned some level of in-house built custom automation, but none of them reported any structured or standardized approach to automation. Established micro-automation solutions are usually built as custom scripts to automate multiple steps, automatic configurations, helpers for a development environment, quick optimizations for testing, and automated knowledge delivery systems (i.e., chatbots, smart notifications). Our participants reported various approaches to process bridging with micro-automation activities.

For example, P1 mentioned: *We have console commands which are build by our team [...] automation in the form of "Slack bots" that help developers. For*

example, if somebody creates a backup that I need, there is Slack channel with that backup hash.. In this example, the trick is that custom scripts report results of software engineering processes (backup is a typical one) directly within the chosen digital communication environment of the team, in this case, Slack. This is helpful, as results and reports from automated processes, therefore, are communicated and documented in more general-purpose environments.
P6 reports efforts invested in the automation of code quality check, stating: *Pre-commit hooks are automated, so we do not need to think if we will push something that is not according to code standards or that we will break parts that are covered with unit tests. When commit is pushed successfully, then our CI spins docker containers in [the] pipeline [...]*. This is helpful to automatically check for specific code standards even before publishing code to a remote repository (pre-commit).

The above are examples of what we call "micro-automation," i.e., automation of concrete steps in a process. An essential joint characteristic is that such micro-automation tends to bridge between different software engineering tools and environments that the developer uses to accomplish a specific activity.

When asked about micro-automation reasons, our interview partners commented that every company has processes that result in different needs to customize and automate various activities. For example, the Agile approach defines methodology and scope on what to do, but it does not prescribe exact steps on how to implement agile. A similar thing happens in code testing and validation. Many things in software engineering are scoped and turned into a methodology, but implementations change from company to company.

4.3. Automation of software engineering as a bottom-up driven side product of operative activity

We perceived our interview partners as being very enthusiastic about automation. Automation is, in general, perceived as valuable, helpful, and almost all of our participants reported automation in their daily work, mostly related to continuous integration and testing.
On the other hand, as companies are focused on the delivery of software or other operatives in the sense of billable project activities, the activity of automating tasks is, from our interview partners' perspective, a side-product of work. As an interesting example, P3 believes that automation is not recognized top level because there is no immediate result and understanding on how to prioritize automation: *There is no "immediate cache return" with automation. It is like a "defensive player" in football. The question is always, is there a need for huge automation or an approach where we automate only things that we need.*. On a similar note, P4 mentioned: "*members of these projects (management, team leads) were inert and did not have the initiative in introducing automation [...] some of these projects did not have stated Quality Assurance so this was not forcing any automation that would assist in Quality Assurance or development*. This suggests that in our interviewees' perception, automation is not initialized by top management because the strategic value of automation, nor an existing pain point that would be solved by it is clearly identifiable.

Most of our interviewees understand automation as a bottom-up driven initiative, in the sense that individual developers recognize an automation opportunity and then implement a quick solution tailored to a specific context and need. P3 says: "*automation is mostly individual effort and not organized company effort.*". P11 gives an example where he was the driver for bottom-up change in organization-wide practice: "*Since there was no company-wide tool for [continuous integration and continuous development], I suggested Jenkins. After some time, other teams started using it.*". We have indications that such initiatives are also understood as products of the nearer social context, namely the team, as when P12 states: "*The automation was introduced by either team or the individual developer [...]*". In all our interviews, however, there were no reports of more structured automation efforts organized at the level of the entire company.

The motivation for automating parts of the software engineering process is twofold. The first is to make work easier. The second is for software engineers to expand their knowledge and experiment with new tools and approaches. Making automated software is far more challenging and rewarding than manually repeating the same action over and over again. We see this motivation for automation as specific to the domain of software engineering.

4.4. Automation and effects on work efficiency

Despite enthusiasm for automation, and at least the partial motivation to automate in order to simplify work and make it more efficient, it is not that clear how automation really relates to work efficiency.

For instance, we have identified concerns that automation increases the cognitive overhead by making it more complicated to understand what is going on, and by overwhelming through notifications. Participant P1 describes how Jira's automated email notifications overwhelm him with a mix of information that he

already knows and does not need at the time of delivery: *Lots of these emails are not useful for me because it is spam because I was in the sprint refinement meeting and I was there when we updated these tickets. I would say that this is automation in my workflow [...] is not smart enough. [The] email notification system does not have information that I do not need something because I was already in a meeting.*. Other participants made similar comments, mostly related to overhead due to automatically generated emails or other forms of notifications, or irrelevant loads of error messages that should be helpful but have the opposite effect.
We, therefore, see that as automated routines generate automated messages (in the context of our interviews, this was mostly emails; but in other environments, this might be Slack messages or any other kind of notification), automated routines do take over work; but in parallel add a communication overhead. In computer-supported collaborative work, it has long been understood that in social, collaborative work, there is an overhead of articulation work (coordination, communication, scheduling, planning division of labor, and in general management - cp. [22]). Following this, we can reach farther based on the above observation of communication overhead through automation, and speculate, or at least ask, whether automated routines and in the future, increasingly autonomous and intelligent agents, will require a similar overhead.

4.5. Unclear impact of AI on the future of SW engineering practice

Overall, our interview partners are clearly enthusiastic with respect to the ongoing micro-automation within their work, as described above, but more ambivalent with respect to the future, and instead not seeing that automation or artificial intelligence-enabled tools would or could substantially change work practice in the near future.

When discussing technologies like programming assistants or AI-enabled pattern mining in continuous integration processes as discussed in the background section above, our interview participants were somewhat skeptical that such tools would be available in the near future with a performance that would substantially alter their work practice. On the other hand, our interview partners were looking with interest forward to more within the reach automation, such as for instance, smart testing tools: *there are companies with thousands of automated test cases, where executing all of them would last for days, therefore selecting only an appropriate subset of tests is very important. (P11)*, and more advanced versions of code quality checking support, or automatic documentation generation.

Some of the use cases our participants suggested are matching findings from our literature overview as experimental research and prototypes. However, none of our participants reported any productive implementations in their experience and working environments.

5. Discussion of Results

Our interview results point out that automation in software engineering tends to be on the side of quick-and-dirty micro-automation where no substantial strategic budget is set aside; hence all automation needs to make sense within operative projects. This type of automation aims to save time, meaning there is less incentive or possibility for reusable scripts/code for automation. Further, this means that automation is maybe even less perfect than in some other sectors; and this, in turn, may lead to adverse effects of automation, such as increased cognitive load for software engineers. Moreover, there are usually no benchmarks or quality criteria to evaluate the different socio-technical and cost-benefit effects of these automation efforts.

This micro-automation might be caused by the bottom-up nature of automation in software development. This is different from what we see in other sectors, where automation is a strategic activity that aims to optimize and respond to external pressures and opportunities. We acknowledge that the present study may be biased because we did not have strategic positions amongst the interviewees.
On the other hand, all interviewees were comparatively senior, and so it can be expected that they would at least have noticed strategic activities. One interpretation is that as software engineers are capable of recognizing automation bottlenecks and solving them, they just do automation themselves. In contrast, in other domains such as manufacturing, it has been repetitive and less creative tasks that have become automated [1, 2, 3]. In this case, those whose jobs have been most impacted by automation did not have the education (automation and software engineering), nor the tools within their work, nor the job profile (less flexibility in self-organizing work than software engineers) to ”just do automation themselves.” So, in software engineering as a domain, there is the possibility of bottom-up driven automation.

Secondly, when it comes to increasing artificial intelligence (AI) instead of rule-based and more simple automation, our interview partners were, on the one hand, optimistic and enthusiastic about having AI-enhanced tools for their work, but skeptical about AI ”being in a driver’s seat” and making decisions of

substantial importance. Combined with our findings from Section 2.3, we understand that AI-driven applications, such as for code error prediction [15], error localization using deductive verification [21] and assistance in effort estimations [16], would be useful and accepted by software practitioners, as long as these applications have advisory and not decision making purpose. Furthermore, scarce hands-on experiences with AI, which we recognized during our interviews, go along with the conclusion made by Feldt et al. [20] that AI applications are not yet massively used in decision-support and decision-making processes. We recognize and take into account that these results might be biased because software practitioners that we interviewed are not in the business of developing Artificial Intelligence and have no extensive professional knowledge of this topic.

On the other hand, it might be that software engineers as professionals do not seriously engage with the question of how their profession will be affected by Artificial Intelligence. In order to bring the discussion forward with experienced software engineering practitioners without in-depth knowledge or direct contact with Artificial Intelligence in their current practice, more in-depth discussions using different methods than interviews, such as co-design workshops to enable active learning and knowledge construction, and hence more informed discussion of the potential of Artificial Intelligence to transform software engineering practice, might be necessary [40, 41].

However, a third perspective might be that software engineering as a profession moves toward a skill diversity phase, where development might differ between two groups. Simple programming, which is heavily supported by automation and Artificial Intelligence tools where developers do not need actual awareness of Artificial Intelligence tools working in the background and actual creators of Artificial Intelligence enhanced tools that have an in-depth understanding of how Artificial Intelligence affects software engineering.

6. Conclusion

The interview study described in this paper has been concerned with the current usage of and practice around automation and Artificial Intelligence enabled tools in software engineering in Austrian- and Eastern-Europe centered SMEs, and with the expectations of interviewees on how automation and increasing deployment of methods from AI in automation will change software engineering practice.

A salient summary insight is that within software engineering, automation but not AI is substantially part of work practice, and is not necessarily seen by its practitioners as constituting a significant part in the future. We see the four themes of micro-automation, bottom-up automation, ambivalent automation benefits, and unclear future impact of AI on software engineering practice as a starting point for future work, in particular, to investigate in-depth characteristics and causalities around these themes and to map out, probably with co-design or future studies methods, potential roles of AI in future software engineering practice.

References

[1] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pp. 3928–3937, 2016.

[2] A. Kusiak, "Smart manufacturing," *International Journal of Production Research*, vol. 56, no. 1-2, pp. 508–517, 2018.

[3] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 6, pp. 1 – 10, 2017.

[4] S. Wang and R. M. Summers, "Machine learning and radiology," *Medical image analysis*, vol. 16, no. 5, pp. 933–951, 2012.

[5] M. Schultz and M. Tropmann-Frick, "Autoencoder neural networks versus external auditors: Detecting unusual journal entries in financial statement audits," in *53rd Hawaii International Conference on System Sciences, HICSS 2020, Maui, Hawaii, USA, January 7-10, 2020*, pp. 1–10, ScholarSpace, 2020.

[6] W. M. P. van der Aalst, M. Bichler, and A. Heinzl, "Robotic process automation," *Business & Information Systems Engineering*, vol. 60, pp. 269–272, Aug 2018.

[7] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2002.

[8] N. B. Sarter, D. D. Woods, C. E. Billings, *et al.*, "Automation surprises," *Handbook of human factors and ergonomics*, vol. 2, pp. 1926–1943, 1997.

[9] S. Thalmann, A. Fessl, and V. Pammer-Schindler, "How large manufacturing firms understand the impact of digitization: A learning perspective," in *53rd Hawaii International Conference on System Sciences, HICSS 2020*, pp. 1–10, 2020.

[10] C. B. Frey and M. A. Osborne, "The future of employment: How susceptible are jobs to computerisation?," *Technological Forecasting and Social Change*, vol. 114, pp. 254 – 280, 2017.

[11] S. Sampson, "Predicting automation of professional jobs in healthcare," in *53rd Hawaii International Conference on System Sciences, HICSS 2020*, pp. 1–9, 2020.

[12] I. Seeber, E. Bittner, R. O. Briggs, T. [de Vreede], G.-J. [de Vreede], A. Elkins, R. Maier, A. B. Merz, S. Oeste-Reiß, N. Randrup, G. Schwabe, and M. Söllner, "Machines as teammates: A research agenda on ai in team collaboration," *Information & Management*, vol. 57, no. 2, p. 103174, 2020.

[13] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: a comparative study," *Neural Computing and Applications*, vol. 27, pp. 2369–2381, Nov 2016.

[14] S. Memeti and S. Pllana, "Papa: A parallel programming assistant powered by ibm watson cognitive computing technology," *Journal of computational science*, vol. 26, pp. 275–284, 2018.

[15] H. K. Dam, T. Tran, J. Grundy, A. Ghose, and Y. Kamei, "Towards effective ai-powered agile project management," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pp. 41–44, May 2019.

[16] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 637–656, 2018.

[17] H. K. Dam, T. Tran, T. T. M. Pham, S. W. Ng, J. Grundy, and A. Ghose, "Automatic feature learning for predicting vulnerable software components," *IEEE Transactions on Software Engineering*, 2018.

[18] J. Xu, P. Yang, S. Xue, B. Sharma, M. Sanchez-Martin, F. Wang, K. A. Beaty, E. Dehan, and B. Parikh, "Translating cancer genomics into precision medicine with artificial intelligence: applications, challenges and future perspectives," *Human genetics*, vol. 138, no. 2, pp. 109–124, 2019.

[19] Y. Cheng, K. Chen, H. Sun, Y. Zhang, and F. Tao, "Data and knowledge mining with big data towards smart production," *Journal of Industrial Information Integration*, vol. 9, pp. 1–13, 2018.

[20] R. Feldt, F. G. de Oliveira Neto, and R. Torkar, "Ways of applying artificial intelligence in software engineering," in *2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, pp. 35–41, IEEE, 2018.

[21] R. Könighofer, R. Toegl, and R. Bloem, "Automatic error localization for software using deductive verification," in *Hardware and Software: Verification and Testing* (E. Yahav, ed.), (Cham), pp. 92–98, Springer International Publishing, 2014.

[22] K. Schmidt and L. J. Bannon, "Taking CSCW seriously - supporting articulation work," *Comput. Support. Cooperative Work.*, vol. 1, no. 1-2, pp. 7–40, 1992.

[23] V. Ivanov, A. Rogers, G. Succi, J. Yi, and V. Zorin, "What do software engineers care about? gaps between research and practice," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, (New York, NY, USA), p. 890–895, Association for Computing Machinery, 2017.

[24] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, "On the unhappiness of software developers," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, EASE'17, p. 324–333, 2017.

[25] B. Johnson, T. Zimmermann, and C. Bird, "The effect of work environments on productivity and satisfaction of software engineers," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[26] T. Dingsøyr, N. B. Moe, and E. A. Seim, "Coordinating Knowledge Work in Multiteam Programs: Findings From a Large-Scale Agile Development Program," *Project Management Journal*, vol. 49, no. 6, pp. 64–77, 2018.

[27] V. Stray, N. B. Moe, and A. Aasheim, "Dependency Management in Large-Scale Agile: A Case Study of DevOps Teams," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, pp. 7007–7016, 2019.

[28] D. Feitosa, A. Ampatzoglou, A. Gkortzis, S. Bibi, and A. Chatzigeorgiou, "Code reuse in practice: Benefiting or harming technical debt," *Journal of Systems and Software*, vol. 167, p. 110618, 2020.

[29] N. Rios, M. G. Mendonça, C. Seaman, and R. O. Spinola, "Causes and effects of the presence of technical debt in agile software projects," in *AMCIS 2019 Proceedings*, AMCIS Association for Information Systems, 2019.

[30] C. J. Cai and P. J. Guo, "Software developers learning machine learning: Motivations, hurdles, and desires," in *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 25–34, IEEE, 2019.

[31] D. Ståhl and J. Bosch, "Modeling continuous integration practice differences in industry software development," *Journal of Systems and Software*, vol. 87, pp. 48–59, 2014.

[32] M. Eyl, C. Reichmann, and K. Müller-Glaser, "Fast feedback from automated tests executed with the product build," in *Software Quality. The Future of Systems- and Software Development* (D. Winkler, S. Biffl, and J. Bergsmann, eds.), (Cham), pp. 199–210, Springer International Publishing, 2016.

[33] S. Ali, Y. Hafeez, S. Hussain, and S. Yang, "Enhanced regression testing technique for agile software development and continuous integration strategies," *Software Quality Journal*, pp. 1–27, 2019.

[34] P. M. Duvall, S. Matyas, and A. Glover, *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.

[35] F. Zampetti, S. Scalabrino, R. Oliveto, G. Canfora, and M. Di Penta, "How open source projects use static code analysis tools in continuous integration pipelines," in *2017 IEEE/ACM 14th Int. Conference on Mining Software Repositories (MSR)*, pp. 334–344, 2017.

[36] R. K. Yin, "Case study research: Design and methods. sage publications," *Thousand oaks*, 2009.

[37] A. S. Clausen, "The individually focused interview: Methodological quality without transcription of audio recordings.," *Qualitative Report*, vol. 17, p. 37, 2012.

[38] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[39] V. Braun and V. Clarke, "One size fits all? what counts as quality practice in (reflexive) thematic analysis?," *Qualitative Research in Psychology*, vol. 0, no. 0, pp. 1–25, 2020.

[40] A. Fessl, S. Feyertag, and V. Pammer, "Designing innovative digital technologies for knowledge management and data-driven business: A case study," in *15th Int. Conf. on Knowledge Technologies and Data-Driven Business (i-Know 2015)*, 2015.

[41] B. DiSalvo, "Participatory design through a learning science lens," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, (New York, NY, USA), p. 4459–4463, Association for Computing Machinery, 2016.