The Journal of Systems & Software 185 (2022) 111144

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

A user survey on the adoption of crowd-based software engineering instructional screencasts by the new generation of software developers☆

Parisa Moslehi [a], Juergen Rilling [a,*], Bram Adams [b]

[a] Concordia University, Montreal, Canada
[b] Queens University, Kingston, Canada

ARTICLE INFO

Article history:
Received 16 September 2020
Received in revised form 1 November 2021
Accepted 3 November 2021
Available online 25 November 2021

Keywords:
Crowd-based documentation
User survey
Tutorial screencast
Software engineering

ABSTRACT

Context: In recent years, crowd-based content in the form of instructional screencast videos has gained popularity among software engineers. For organizations to remain competitive in attracting and retaining their workforce, they must accommodate the use of such crowd-based documentation content.
Objective: We conduct a user survey to gain insights on how a user's background and work tasks influence the use of different documentation media types. In our analysis we focus on how-to tutorial screencasts and how the new generation of software engineers is using such videos as an information resource.
Methods: For this research, we report results from our user survey, including benefits and challenges software engineers face in using screencasts as project documentation. We discuss potential avenues on how to improve the usefulness of how-to tutorial screencasts.
Results: The level of professional experience, job position or reason for resorting to a documentation, affect the type of resource being used. As most (78%) of our survey participants were junior software engineers, our survey results are in particular applicable to this user group.
Conclusion: We conclude our paper with lessons learned and provide some recommendations for screencast creators, such as: building a dedicated platform, making screencasts searchable and link their content to other artifacts.

1. Introduction

The last three decades have been characterized by new technologies (e.g., collaborative development, social media) that had not only a major effect on peoples' daily lives but also organizations and their workplaces (Storey et al., 2010). These technology and social changes were not instant rather they evolved slowly, with various age groups (a.k.a. generations) of the population being exposed and immersed differently within these new environments. Organizations have to adapt to these changes to be able to provide a work environment and *media ecology* (Storey et al., 2014) that not only supports complementary tools of communication but also allows organizations to take full advantage of the changes in the skill sets of their workforce (McLure Wasko and Faraj, 2000).

For example, depending on their level of familiarity with digital technologies of the 21st century (Eisner, 2011), software engineers will rely on resources they are familiar with, to support or complete their daily tasks or document their work. Educational systems have already embraced these technologies and learning style changes in our society by adapting existing and introducing new learning and teaching approaches to accommodate the learning behavior and needs of the younger generations (Seemiller and Grace, 2017). An example of such new learning styles is *blended-learning* (a.k.a., *hybrid learning* and *mixed-mode learning*) (Garrison and Kanuka, 2004), which has emerged as an integral part in teaching learners who are more familiar with the new technologies (i.e., digital natives) (Wells et al., 2012).

At the same time, traditional well-formed and structured software documentation and processes have been replaced with new, more agile processes reducing the amount of documentation that is explicitly generated for projects (Highsmith and Highsmith, 2002). This transition has been further accelerated through the wide-spread use of social media and crowd-based documentation and by having a generation of younger developers in the

☆ Editor: Kelly Blincoe.

* Corresponding author.
E-mail addresses: p_mosleh@encs.concordia.ca (P. Moslehi), juergen.rilling@concordia.ca (J. Rilling), bram.adams@queensu.ca (B. Adams).

https://doi.org/10.1016/j.jss.2021.111144
0164-1212/

workforce, who are immersed in these new technologies while growing up (Seemiller and Grace, 2017).

Many software engineers have adopted social media as part of their daily workflows to communicate and collaborate with each other. These employees are considered being part of the emergence of a participatory culture (Storey, 2015), who are using these social media channels to share knowledge and interactively learn from each other (e.g., how to perform certain tasks or to improve their work (Storey et al., 2014; MacLeod et al., 2015; Ponzanelli et al., 2016; Barzilay et al., 2013; Black et al., 2010)). Key to this wide-spread social and multimedia environments adoptions is that these environments are readily available and provide cheap and fast many-to-many communication and information distribution (Storey et al., 2014). In addition, they provide new communication channels and media resources such as images (e.g., captured from whiteboards), Wikis, Q&A repositories (e.g., Stack Overflow) and video repositories (e.g., YouTube or Instagram) (MacLeod et al., 2015; Parnin et al., 2012). These resources are often created by many and used by many (i.e., crowd-based) (Parnin et al., 2012), with contributors being motivated by the fact that they can gain an online reputation and also learn more through the process of teaching to others. Such crowd-based documentation or resources are considered a subset of what is known as *public good* (McLure Wasko and Faraj, 2000), which refers to content that is socially generated by the community or within organizations (e.g., documentation, code, activities, etc.)

One type of multimedia documents that has gained popularity, are screencasts. Screencasts deliver content in the form of audio (through a narrator), video (images), and textual (meta) data (e.g., subtitles, title, description, publish date). An instructional screencast includes a digital video recording of a computer screen and are often used to transfer knowledge, to demonstrate a process, show how to do something or explain a concept. In the context of this research, we also refer to this type of screencasts as how-to tutorial videos. While other text-based documentation such as blog posts, question and answers, and traditional software artifacts are used to share explicit knowledge or facts about an application (MacLeod et al., 2017), screencasts are used for sharing tacit knowledge (MacLeod et al., 2017) that is in peoples' heads (McLure Wasko and Faraj, 2000). Screencasts can be created by educational platforms (MacLeod et al., 2017) (e.g., edx,[1] coursera[2]) that provide online courses for certain topics and often include granting certificates to people who enroll and successfully complete these online courses and their assessments. More often, these screencasts are created by the crowd for other purposes, such as: explaining how to use a software feature, providing step-by-step instructions for workarounds to a given problem or to demonstrate security issues that may result from known security vulnerabilities (Eghan et al., 2020). Platforms (e.g., YouTube[3], Vimeo[4]) are commonly used to host such crowd-based tutorials and documentation screencasts that are available for free.

Crowd-based screencasts are mostly created and consumed by users which are part of the generation Y (born between 1977–1994) and generation Z (born after 1994) users who are more likely to be visual learners and prefer digital content over textual documentation (Eisner, 2011; Turner, 2015). In fact, how-to videos have become widely accepted in the open-source community, to a point where open-source projects have started to make instructional screencasts an integrated part of their documentation (Moslehi et al., 2018). For example, WordPress[5] encourages its users and contributors to create new how-to videos[6] that describe certain use-case scenarios or features. Also, users have started to enrich their bug reports with screencasts[7] to exactly demonstrate how to reproduce a bug.

While some research (Storey et al., 2014; Storey, 2015; Barzilay et al., 2013; Parnin et al., 2012; Ahmad et al., 2018) investigated the use of social media in software engineering, little research exists on understanding the creation and usage of different types of screencasts for software engineering. Among the earliest work is MacLeod et al. (2017), who conducted a user study to gain insights on how and why software developers use programming screencasts for documenting software. Their study focused on the creation of crowd-based screencasts for programming purposes and highlighted the need for understanding how and why screencasts are used by developers and software engineers.

In our research, we conducted a user survey[8], to gain insights on how relevant instructional screencasts and other types of documentation are for the new generation of software engineers and how they use these documentation types for performing different software engineering tasks (e.g., understanding program features, modifying their source code, using features of a GUI application, learning new concepts). The main contributions of our paper can be summarized as follows:

- Conducting a user survey, which focuses on the use of multimedia documentation by the new generation of software engineers.
- Identifying source(s) of documentation that are preferred by software engineers depending on their personal (e.g., experience) and work context (e.g., help with a particular software engineering task, to improve their current skills).
- Identifying components or information sources of tutorial (how-to) videos that are important to the user when searching for screencasts that are useful and relevant to a specific task.
- Providing insights on how software engineers use tutorial videos that showcase the use of application features.
- Providing insights about benefits, disadvantages, and room for improvement in using instructional screencasts and their role as documentation in software engineering.

Our findings show that screencasts and Stack Overflow are among the most important crowd-based information sources used by the new generation(s) of software engineers to perform their tasks. We also found that depending on their level of experience, position, or reason for resorting to a documentation source, software engineers will rely on different information sources.

The remainder of this paper is organized as follows: Section 2 motivates our work and defines the research questions. Section 3 compares our work with related work. Section 4 describes our user survey set up, demographics of the survey respondents, data preparation, and our approach of analyzing open-ended questions. Section 5 presents the study results and answers to our research questions obtained through analyzing the survey responses. Section 6 discusses the lessons learned as well as recommendations and future directions for the adoption of screencasts in software engineering. Section 7 provides a discussion of potential threats to the validity of our approach followed by Section 8 which concludes the paper.

1 https://www.edx.org/.
2 https://www.coursera.org/.
3 https://www.youtube.com/.
4 https://vimeo.com/.
5 https://wordpress.com/.
6 https://en.support.wordpress.com/video-tutorials/.
7 https://youtu.be/Am9SNUhSz4w.
8 https://1drv.ms/b/s!ArPyXNcsMbKOg-gwo1O1Jm4jefdL2A?e=hvt0NW.

2. Motivation

Organizations have started to recognize that their traditional processes and workflows no longer match the changing learning and work habits of the younger workforce (Eisner, 2011), who prefer to use digital content and social media tools over traditional print or textual documents (Storey et al., 2010). Providing such work environment has become an essential aspect in not only improving productivity but also in attracting and retaining future employees (Storey et al., 2010; Reisenwitz, 2009).

A significant body of work (Eisner, 2011; Turner, 2015; Reisenwitz, 2009; Lissitsa and Kol, 2016; Krug, 1998; Bolton et al., 2013) exists that shows different generations of the population have different signature products and different preferences in their learning and communication style and usage of technologies. Based on this, different generations of the population and their characteristics are categorized as follows:

- Gen X, also referred to as the generation of digital immigrants, are people who did not grow up with digital media during their youth (e.g., printed documents) but learned to adapt to these new technologies and use of digital media (e.g., e-mail) as a communication medium (Reisenwitz, 2009).
- Gen Y, also known as millennials[9] and the generation of tech-savvy or digital natives, were exposed to digital technologies early in their lives (either as a youth or young adult). Gen Y is known for being visual learners who search online resources for hands-on information rather than using textbooks; a generation that prefers image- and video-based communication and social networks over textual documents (Eisner, 2011; Reisenwitz, 2009; Bolton et al., 2013).
- Gen Z, often also referred to as the generation of technoholics, is a generation that depends since their youth on the use of electronic devices and digital content to manage their lives. For this generation, ubiquitous computing plays an increasing role, with boundaries between individual devices disappearing. While data mining and context awareness (e.g., location awareness), which enable services (e.g., recommendations and search results on the Internet) that are tailored to individual user's needs, have emerged as an important aspect of their daily lives (McCrindle and Wolfinger, 2009).

Furthermore, research by Freund (2015) has also shown that personal factors such as level of professional experience, level of familiarity with technologies and products, and the role of a software engineer affect their information seeking behavior. At the same time researchers (e.g., Storey (2015), Eghan et al. (2020,?), Turner (2015), Moslehi et al. (2018)) have started to develop tools and approaches that integrate crowd-based documentation (e.g., YouTube screencasts and Stack Overflow question and answers) with other software artifacts (source code, vulnerability descriptions, etc.).

In our previous work, we focused on providing ubiquitous access to textual documentation (i.e., use-case scenarios) extracted from speech component of screencasts to enrich project documentation (Moslehi et al., 2016), and the linking of application features shown in how-to screencasts to their corresponding source code implementation (Moslehi et al., 2016). Also, in Eghan et al. (2020), we introduced a Sematic Web-based approach for linking screencasts to security vulnerabilities.

The goal of the user survey presented in this paper is to provide further evidence that screencasts are often used by the new generation of software engineers for different tasks. We also provide recommendations on how to improve the further adoption of screencasts in the software engineering domain. In our survey, we were interested in general in the use of crowd-based multimedia documentation and social media. More specifically **we focused on the use of instructional screencasts** for software engineers and how various factors (e.g., user demographics, the task being performed) affect the use and perception of such screencasts. Given the increasing role of crowd-based multimedia resources and screencasts in practice as well as the challenges that exist in their usage, the goal of our survey was to gain a better understanding of the usage preferences of instructional screencasts in a software engineer's task context.

We therefore argue that mining crowd-based multi-media content and its integration in existing software engineering workflows can help provide younger or less experienced software engineers with easier access to a documentation they are more familiar with. To validate our assumption, we conducted a survey and by analyzing the responses, we try to answer the following research questions:

- **RQ1**. What type of documentation source is preferred by our survey participants to perform their software engineering tasks based on their level of experience or age?
- **RQ2**. How frequently and why do software engineers watch instructional screencasts?
- **RQ3**. What are the important components of instructional screencast according to its users?
- **RQ4.** Are software engineers interested in tracing content from instructional screencasts to software artifacts?
- **RQ5**. What are the advantages and disadvantages of using instructional screencasts over written documentation in the software engineering domain?

3. Related work

The adoption of social media tools within the software engineering community has been previously investigated by Black et al. (2010) and Storey et al. (2010, 2014). Black et al. (2010) found that social media tools are useful in a sense that they improve the speed of communication and facilitate communication among colleagues. Storey et al. (2014) found that while traditional and face-to-face communication is still an important means of communication among developers, social media channels have been widely accepted as valuable tools for collaboration and communication within the software engineering community. The use of such social media channels has resulted in the emerge of social programmers (Storey, 2015) who leverage social media tools to participate and contribute to the creation of crowd-based resources. However, this requires a level of social media literacy (Storey, 2015) to be able to effectively leverage the tools and interact with the community. Furthermore, Storey et al. (2014) found that, despite all the benefits of using social media in software engineering, there are also many remaining challenges, such as: (a) issues with trustworthiness, the quality and reliability of the content, (b) trouble finding information related to a specific version of a project, (c) finding out-of-date information, (d) switching between multiple media channels and lack of links from these channels to design, algorithm, and programming artifacts. This implies the importance of linking and integrating such documentation to their corresponding software artifacts to facilitate the maintenance of software artifacts and the provision of documentation for software engineers.

Storey et al. (2010, 2014) also showed that, aggregating social media channels (e.g., GitHub) or integrating them into the IDEs can support collaborative activities (e.g., version control, release

[9] https://en.wikipedia.org/wiki/Millennials.

management, etc.), coordination, and communication. Such integration can improve the reliability and traceability between the socially produced content and code artifacts. For example, Subramanian et al. (2014) proposed a method for linking code examples posted on Stack Overflow to API documentation using their tool *Baker*. This tool links code snippets to Java classes and methods or JavaScript functions, with an observed precision of 97%. Jiau and Yang (2012) measured in their work the inequality of crowdsourced API documents on Stack Overflow and found that a larger proportion of existing documents addresses a smaller portion of topics. They leveraged this inequality and proposed a method that projects a crowdsourced documentation into a concept domain to recover traceability links based on the reusability of the concept domain. Barzilay et al. (2013), developed a code search tool called *Example Overflow*. This tool is implemented on top of Stack Overflow and extracts high quality code examples. As part of their empirical analysis, they studied the type of questions posted on Stack Overflow and how these questions could be answered by their tool.

Different exploratory studies exist that analyze the characteristics of textual crowd-based documents (e.g., Parnin et al. (2012), Campbell et al. (2013), Pham et al. (2013)). Parnin et al. (2012) investigate in their work the use of crowd-based documents to replace traditional software documentation. They measured in their work the extent to which blog posts cover API methods. Their study shows that social media posts can cover 87.9% of the API methods and therefore could potentially replace traditional documentation. Campbell et al. (2013) combined Stack Overflow questions and PHP and Python projects' documentation by applying LDA. They found that topics in Stack Overflow did not overlap the topics on projects' documentation and that many topics covered on Stack Overflow but not by traditional project documentation, are related to external project documentation or tutorials. Pham et al. (2013) study if a common testing culture can be extracted to address challenges that social coding sites introduce to testing behavior. They conducted a survey among GitHub users to identify such challenges and their impact on current testing practices. In their study they also suggest strategies that, if applied by software developers and managers, can positively affect the testing behavior. Li et al. (2013) conducted two user studies to gain insights on how programmers seek help during different software engineering activities, the challenges programmers face, and the limitations of existing tool support. Their studies showed that help seeking is an integral part of software development and it should be better supported by existing software development processes. They further suggest that cloud-based communities-of-practice may provide opportunities to help developers, by being able to take advantage of the wisdoms of crowds. Nasehi et al. (2012) analyzed code examples of Stack Overflow that are voted by users as being good code examples to identify the characteristics that, if applied, can improve the development and evolution of API documentation. Brandt et al. (2009) conducted two studies to investigate how programmers use the Web in general to solve programming problems. In their first study, they conducted a controlled experiment observing how their study participants use online resources to solve the specific programming task assigned to them. Their findings from the study indicate that programmers use the Web for just-in-time learning and to clarify and extend their existing knowledge. However, no specific analysis was conducted on the use of tutorial videos. In their second study they analyzed existing Web search logs of a programming portal and the observed results support their observation from their first study that the Web in general is being used by programmers for learning and refreshing existing knowledge.

Escobar-Avila et al. (2019) conducted a survey involving 205 participants to investigate the use of different online resources for learning. More specifically, they considered in their survey the format of resources the study participants prefer to use when learning a new topic, their information seeking behavior and aspects the participants like or dislike using current online resources. They observed that their participants prefer a combination of visual and auditory resources. The study also revealed that in situation when participants have to search for information related to learning programming, they prefer written tutorials and documentation over video tutorials. Other key findings from their study are that their study participants appreciate the large amount and diversity of information available online but dislike the fact that only limited search capabilities are available to find high quality content. These results are further supported by evidence, which we present in this paper.

Wells et al. (2012) reported in their work that tutorial videos can not only be used as a learning aid for traditional lectures but that they can also significantly improve students' performance in education. In Mohorovičič (2012) Mohorovičič compared tutorial videos with written text, and the analysis showed that screencasts can provide a clearer explanation of a subject matter while being accessible through different devices. In the same work (Mohorovičič, 2012), the author also reported that visual learners and students benefit from screencasts by being able to learn at their own pace, whenever and wherever they want.

More specific to the software engineering domain, MacLeod et al. (2015, 2017) conducted a user study on the **creation** of programming tutorial screencasts that contain live coding and are created by the crowd and posted on YouTube. Based on their analysis of 20 tutorial screencasts and interviewing 10 developers/YouTubers they identified benefits and challenges of using and creating screencasts. They observed that developers often create screencasts to generate and maintain an online identity, to learn by teaching to others, and to share their knowledge with the community. Their research also shows that screencasts are commonly created by "doing by showing", where screencast creators demonstrate the execution of a use case in the program while performing live coding to help people understand the data flow and expected output of the program. Such mapping of execution to code helps the audience to better understand where in the code certain program features are implemented. They also performed another study (MacLeod et al., 2017) using Ruby on Rails screencasts, where they compare screencasts published through free platforms (i.e., YouTube) with screencasts that are specifically designed for a specialized paid platform (i.e., RailsCasts). As part of this work, the authors also introduced some guidelines for screencast creators on how to produce better videos.

Amongst the earliest research in the domain of linking videos to other software artifacts is the work by Bao et al. (2015a, 2017), who developed a video scraping tool, *scvRipper*, to automatically extract developers' behavior from screencasts. They extract user actions in screencasts, by employing key point-based template matching using visual cues in an image (e.g., icons that appear in a window). In (Bao et al., 2015b) Bao et al. also proposed an approach to track user activities and developed a tool called *ActivitySpace* that supports inter-application information needs of software developers. The tool reduces the effort required by developers for locating documents and recalling activities performed in their daily work. In Bao et al. (2019), Bao et al. introduced a tool called *VT-Revolution* that captures workflows in programming tutorials and enables timeline-based browsing of these tutorials as well as accessing the API documentation of a selected code element. In their approach, they record low-level Human-Computer Interaction (HCI) data extracted from the tutorial creators while working with screencast authoring tools. Khandwala et al.'s *Codemotion* (Khandwala and Guo, 2018) uses a computer vision algorithm to extract source code and dynamic

code edit intervals from tutorial screencasts. The tool uses a video player UI to enable code search and navigation. Zhao et al. (2019) introduced in their work a content-centric analysis of screencasts that uses a deep learning-based computer vision technique to automatically recognize developer actions in screencasts. They apply first image differencing techniques to detect the change regions between two consecutive video frames. The detected change regions are then fed into a Convolutional Neural Network model to extract abstract image features, which they then classify to predict user actions that caused the change. In our previous work (Moslehi et al., 2016), we extract documentation from the speech component of tutorial screencasts that describe how to use features found in a GUI based application. As part of our work, we conducted a case study that illustrated how our approach can be used to extract usage scenarios from WordPress screencasts. The case study showed that the approach was capable to extract usage scenarios from the selected WordPress screencasts, with a median precision and recall of 83.33% and 100%, respectively.

Other work (MacLeod et al., 2015, 2017; Ponzanelli et al., 2016; Ponzanelli, 2019) addresses the need for designing concise video tutorials that contain less noise and describe more concise what viewers expect to find in a video (Ponzanelli, 2019). Ponzanelli et al. (2016), Ponzanelli (2019) developed an approach to extract relevant code fragments from software development tutorial videos and link them to Stack Overflow discussions by mining the (captioned) speech and GUI content of the video tutorials. Yadid and Yahav (2016) developed an approach to extract code from programming video tutorials to enable deep indexing of these videos. Their approach consolidates code across multiple image frames of the videos and uses statistical language models to make corrections on the extracted code. Another work by Ott et al. (2018) presented an approach that uses deep learning, and more specifically convolutional neural networks and autoencoders, to identify source code examples in image frames of a large data set of videos. Escobar-Avila et al. (2017), Parra et al. (2018) presented text retrieval-based tagging approaches to help users to identify whether the content of a video tutorial is relevant to their needs. Poche et al. (2017) summarize and classify tutorial video comments using Support Vector Machines (SVM). Ellmann et al. (2017) used a frame similarity approach (i.e., cosine similarity and LSI) to identify and distinguish development screencasts from other types of videos on YouTube. They also link these screencasts to their relevant API documentation using only the audio transcript of the videos. In our previous work (Moslehi et al., 2018), we proposed an LDA-based approach of locating source code artifacts that are related to GUI features of a software application that are being demonstrated in a screencast. We (Eghan et al., 2020) also proposed a Sematic Web-based approach of linking screencasts that demonstrate how to exploit security vulnerabilities to their relevant vulnerability descriptions on National Vulnerability Database (NVD)[10] and establish indirect links from such screencasts to their relevant project dependency information from Maven Central.

In general, the presented related research addresses: (a) how **developers** create documentation using social media, (b) **creating** videos to document and share their **programming** knowledge, (c) the use of crowd-based programming tutorial videos as input to extract source code or programming documentation from the videos to **link** them afterwards to their corresponding application source code or to other software artifacts, (d) the use of crowd-based GUI screencasts to extract documentation or for source code retrieval. To complement this existing work, we are motivated to conduct a survey to better understand how **software engineers** with different backgrounds (e.g., level of professional experience, generation) and roles (e.g., software designers, developers, software testers, students, researchers, etc.) **use** social media and more specifically **screencasts** for different types of tasks (e.g., highly technical or less technical) and to **provide recommendations** on how to potentially improve further adoption of screencasts in the software engineering domain.

4. User study

In this section we describe the design and setup of our user study questionnaire and provide some insights related to the demographics of our survey respondents, data preparation and, how we processed open-ended questions.

4.1. User study setup

We conducted a survey[11] consisting of 33 questions including 29 multiple choice, five- and six-level Likert scale[12] questions (e.g., Always, Often, Sometimes, Rarely, Never) and 3 open-ended questions. We designed the questionnaire such that it can be completed in approximately 15 min. We used Google forms, which is a widely used platform to create free online surveys, to design and create the questionnaire. The survey was also approved by the Concordia Research Ethics committee. We distributed the link to the survey on LinkedIn,[13] Facebook,[14] Twitter[15] and invited software engineers and anyone with experience or expertise in programming to participate in the survey. We also distributed the link to software engineering researchers in the MCIS[16] and ASEG[17] labs, and a learning platform[18] that is accessible to both graduate and undergraduate students enrolled in a software engineering course at Concordia University.[19] Participants were assured that we intend to publish the aggregated results of the research and it will not be possible to identify individual participants in the published results.

The questionnaire includes four main parts which we briefly describe here. The first part is used to collect background and demographic information of our survey participants (e.g., gender, birth year, occupation, years of professional experience as a software engineer, programming languages that they use and their open-source experience).

The second part of the questionnaire addresses questions such as: (1) what type of documentation (digital, non-digital, and social media) is preferred by the participants, to get help with their software engineering tasks (e.g., using an application feature, using an API, etc.) or learn something (e.g., a programming language, how to use a tool, etc.), (2) the importance of different social media channels (e.g., GitHub, Stack Overflow, YouTube, etc.) in completing their software engineering tasks and, (3) the type of media and the format that they prefer to use to learn new concepts or use as documentation. The answers to these questions are not mutually exclusive and multiple answers are possible.

In the third part of the questionnaire, we want to gain insights on how frequently and why software engineers watch instructional screencasts. We included questions asking participants

10 https://nvd.nist.gov/.

11 https://docs.google.com/forms/d/e/1FAIpQLScNU4QDfcXzVUfc6d0P1-g95iMAh_ev9Fl6TWvSrIBLPFiiLQ/viewform.

12 https://www.surveymonkey.com/mp/likert-scale/.

13 https://www.linkedin.com/.

14 https://www.facebook.com/.

15 https://twitter.com/.

16 http://mcis.polymtl.ca/.

17 https://sites.google.com/view/juergenrilling/home.

18 https://moodle.concordia.ca/moodle/.

19 https://www.concordia.ca/.

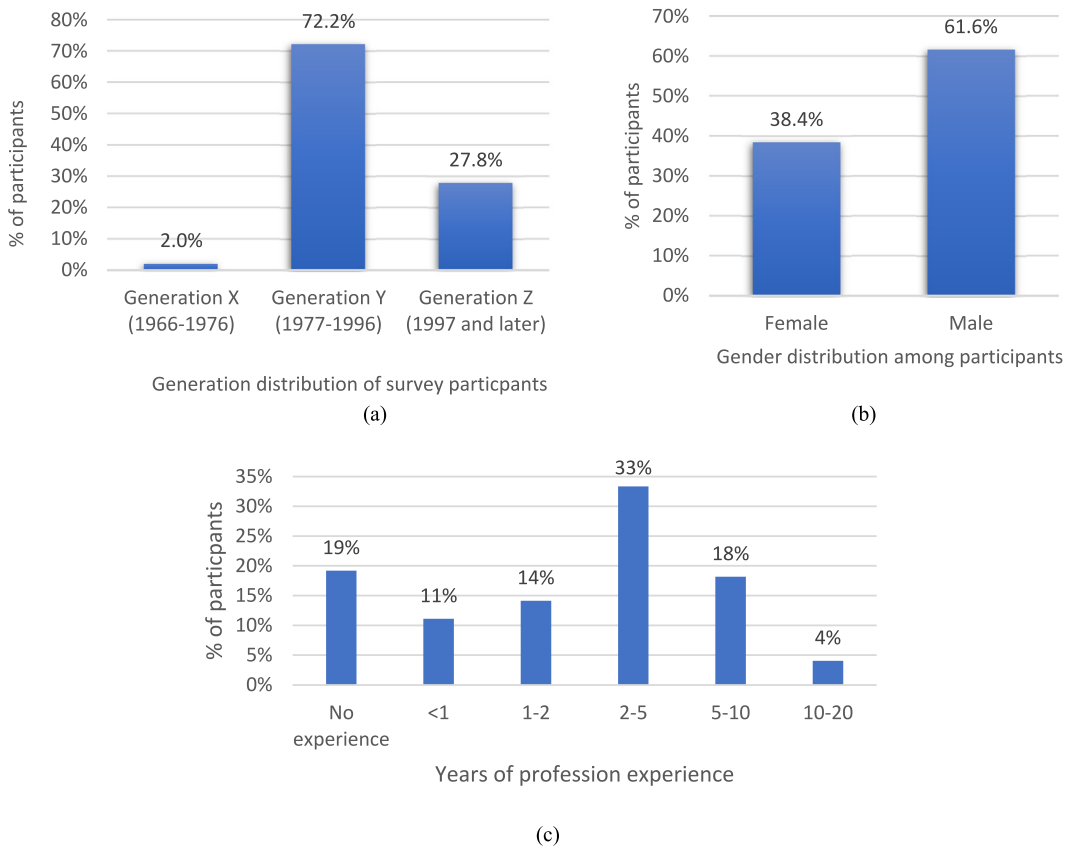


Fig. 1. Demographic information of survey participants.

why they watch software engineering-related tutorial screencasts, how often they view tutorial screencasts as a learning resource and how often they find such tutorials effective and their content useful. We also included a set of questions to help us identify what users consider the most important aspects of a tutorial screencast, including their level of satisfaction with the audio and video quality of screencasts, and what criteria they use for selecting a tutorial video (e.g., author, content, speech, HD quality, number of views, number of likes, title).

We also provide two sample scenarios, one of which describes a software engineer who wants to learn how to perform a specific task related to a high-level usage scenario of an application and a more technical scenario describing an implementation issue of this software application. We then asked survey participants what type of documentation (Stack Overflow, the official online documentation, a tutorial screencast, or all of them) they would use to perform a given task. In addition, to better understand what our participants consider as advantages and disadvantages of using screencasts compared to text-based documentation, participants were asked to complete open-ended questions.

In this fourth part of the questionnaire, we asked participants how they would instructional screencasts as a software artifact, when contributing to an open source project they are unfamiliar with. With this assumption in mind, we asked them how frequently they watch screencasts related to a software project whose source code they want to customize or how often they try to locate the source code artifacts of a feature that is being demonstrated in a screencast. We also asked them two questions, where they had to specify which information source they would prefer if they could find the relevant answer in different resources (e.g., screencasts, online textual documentation, security issues or Stack Overflow Q&As).

4.2. Demographics of the participants

We received 99 responses from participants with different software engineering backgrounds. Figs. 1 to 4 provide a general overview of the demographics of the survey participants. Fig. 1(b) shows that the majority of the survey respondents are male (61.6%) compared to 38.4% being female. Fig. 1(a) shows the age distribution of the survey participants, with 72.2% of the participants being born between 1977 and 1994 (therefore considered to be Gen Y), 27.8% being born after 1994 (Gen Z) and only 2.0% being born between 1966 and 1976 (Gen X). Most of the survey participants (62%) are graduate students, software developers (43%) or researchers (24%)- see Fig. 2, with 19.2% of participants having no prior professional experience as a software engineer (Fig. 1(c)).

Fig. 3 provides an overview of the programming languages most used by our participants, with Java (82%), SQL (54%), and Python (49%) being the top three languages. Fig. 4 provides an overview of how experienced the participants are in terms of contributing to open source or commercial projects. Among the participants, 26% have never contributed to an open source or commercial projects and 37% have worked on 2 to 5 projects and 16% have worked on more than 5 projects.

4.3. Data preparation

To answer our research questions, we preprocessed our collected data. In case of our six-level Likert scale responses, we provided the option of "Not sure" to the respondents so that when they do not have a clear answer for a question, they can select this option. During preprocessing, for each question, we removed the "Not sure" responses. For instance, for a question that asks about the level of preference of a person using a certain type

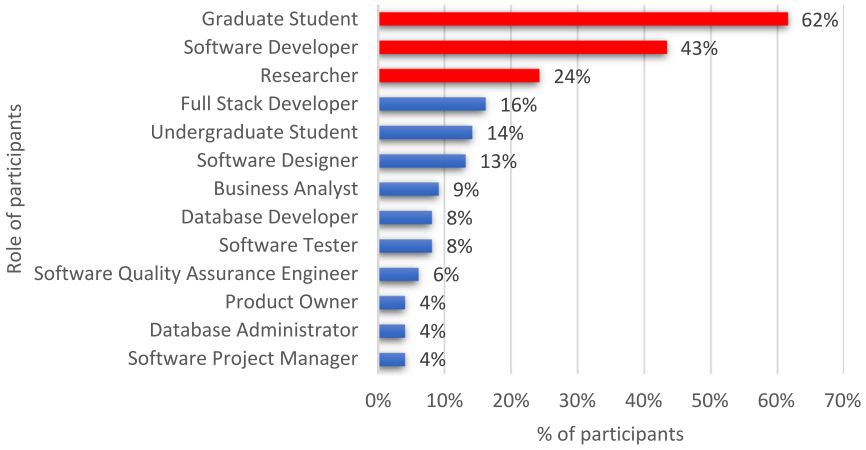


Fig. 2. Role of the participants. Options that were selected by more than 20% of the participants are colored in 'red'.

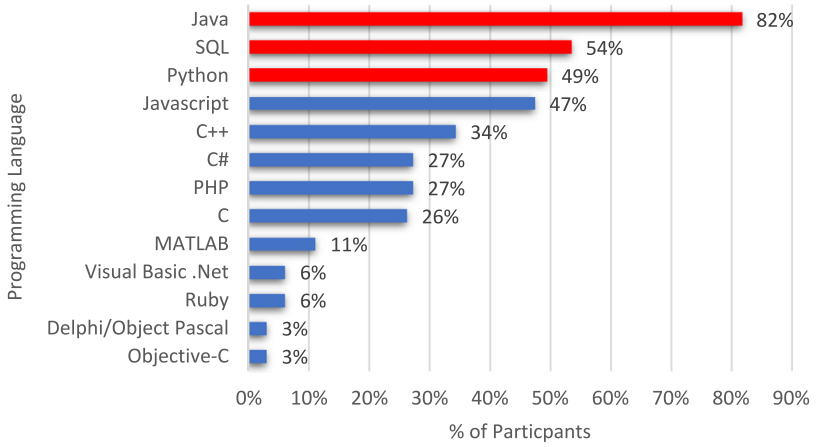


Fig. 3. Programming languages used by participants for development or maintenance tasks. Top 3 languages are colored in "red".

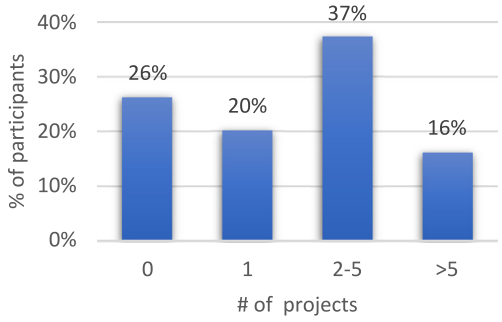


Fig. 4. Number of open source or commercial projects participants actively contributed.

of documentation for a particular type of task, we provided Likert scale levels such as: "Highly preferable", "Moderately preferable", "Somewhat preferable", "Not very preferable", "Not preferable at all", "Not sure". In this case we removed all respondents that selected the "Not sure" from this specific question, since it did not provide any additional insights. Our survey[20] contains 7 questions with "Not sure" answers (questions 8, 10, 11, 20, 21, 23, 26), for a total of 35 possible "Not sure" answers. Fig. 5 shows the frequency the "Not sure" option was chosen by our participants. The figure includes only participants who chose the option "Not sure" at least once.

It should be noted that the response by participants P14 and P40 (highlighted in red) showed an over proportional large number of "Not sure" responses. Based on this observation, we concluded that their responses to other questions may also not be informative and removed all their responses from our further analysis. We also merged the levels of professional experience (Fig. 1(c)). Since most of our participants are graduate students and we also have undergraduate students in our respondents (Fig. 2) we observed that most of our participants (78%) have less than 5 years of professional experience in software engineering. Therefore, based on the observed distribution of the data (Fig. 1(c)), we merged "No experience" and "<1 year" into "<1 year" and also we merged "5–10 years" and "10–20 years" into ">5 years".

In case of our participants' age groups, we had two respondents (Fig. 1(a)) who were part of the Gen X age group (born between "1966–1976"). As mentioned before, as part of our data preparation we removed two participants which responded with an over proportional large number of "Not sure" responses, with one of these removed participants (P14), being of Gen X. We, therefor decided to remove the remaining single Gen X (born 1966–1977) participant from our dataset, resulting in a dataset with 96 participants of which 72.9% belong to Gen Y and 27.1% to Gen Z.

20 https://docs.google.com/forms/d/e/1FAIpQLScNU4QDfcXzVUfc6d0P1-g95iMAh_ev9Fl6TWvSrIBLPFiiLQ/viewform.

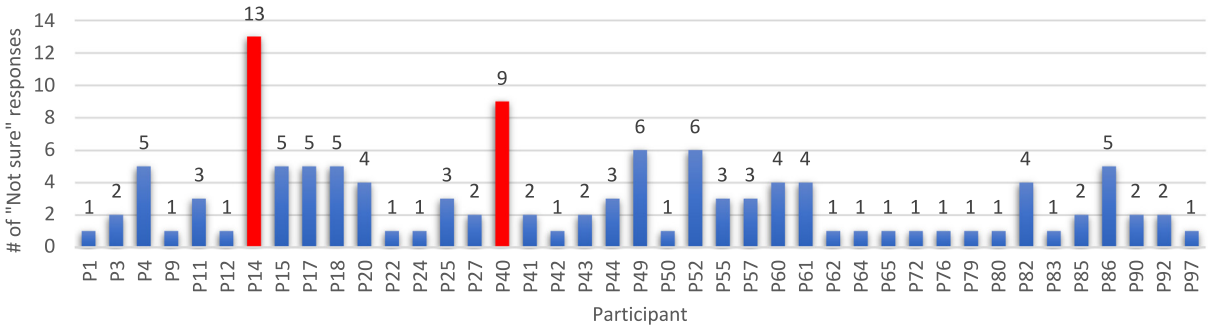


Fig. 5. Frequency of "Not sure" options selected by individual survey participants.

Table 1
Interpretation of Kappa statistics.

Kappa Statistic	Agreement
< 0	Less than chance
0.01 – 0.20	Slight
0.021 – 0.40	Fair
0.41 – 0.60	Moderate
0.61 – 0.80	Substantial
0.81 – 0.99	Almost perfect

4.4. Analyzing open-ended questions

In our questionnaire, we asked three open-ended questions. In this section, we explain how and using what approach we analyzed the answers to these questions. In our analysis we followed a grounded theory approach (Miles et al., 2018) and one of the authors with the help of a Ph.D. student took the following steps to code the answers:

1. One of the raters identified the main categories in the respondents' answers and shared the topics with the other rater.
2. Both raters independently reviewed and tagged the respondents' answers to identify codes for each category derived in the previous step.
3. The raters shared with each other the codes that they identified in the second step and had a discussion to agree on their identified codes and built a codebook.
4. Then both raters tagged the answers once again using the codebook that was created in step 3.
5. Finally, we calculated inter-rater reliability using Cohen's kappa (Sim and Wright, 2005) to measure the degree of agreement among the raters. Table 1 shows how we can interpret the Kappa statistics to find the degree of agreement.

5. Study results

In this section, we analyze our survey responses to answer the research questions introduced in Section 2.

RQ1. What type of documentation source is preferred by our survey participants to perform their software engineering tasks based on their level of experience or age?

Fig. 6(a) provides an overview of the respondents based on their years of professional experience and their preferred source of developer knowledge they typically use for completing software engineering tasks. Participants could select between "Digital" (e.g., Slashdot, Sourceforge, Visual Studio documentation, Eclipse documentation, mailing lists, emails), "Non-digital" (e.g., telephone, face2face, project workbook, documents, books) or, "Social media" (e.g., blogs, Twitter, LinkedIn, YouTube, Vimeo, Stack Overflow, Slack, Facebook, GitHub). Among the responses for "Non-digital" resources, one participant and for the "Social media" preference, also one person responded, "Not sure". These responses were removed from our analysis since they are not informative. Our survey results show that **"Non-digital" knowledge resources are the least preferred** among participants. The results also show that **social media resources gain in popularity among the more experienced** software engineers.

We also observed that respondents with different level of professional experience have different preference in using "Digital" knowledge resources. This observation is statistically significant, with a p-value = 0.02 for Chi-squared statistical test of significance. However, the result of the same test for the "Non-digital" and "Social media" knowledge resources were not statistically significant.

Classifying the respondents based on their age and their preference of using "Digital", "Non-Digital" and, "Social Media" as their information resource shows that **as participants get older, they rely more frequently on "Social Media" sources** (Fig. 6(b)). However, the results are not statistically significant.

Based on the received responses, **"Stack Overflow", "GitHub", and "YouTube" are the three social media resources** that our participants consider to be the most important "to complete their software engineering-related tasks". Again, we removed the "not sure" responses from our analysis as being non informative. The analysis (Fig. 7) shows that, as developers become more experienced, they find "Stack Overflow" to be more important than other media types and the use of "YouTube" significantly drops among more experienced professionals with p-value = 0.025 for Chi-squared statistical test. Also, more experienced (>2 years) software engineers rely less frequently on "GitHub" compared to the less experienced professionals (<2 years). The same statistical test of participants with different levels of experience having different levels of preference in using Stack Overflow and GitHub did not show any statistical significance.

We further asked our participants to rank their preference of using "Books", "Blog posts", "Images", "Online written documentation", "Podcasts", "Question and answer sites", and "Videos" for learning new skills/concepts. The survey results show that **"Videos" are the most preferred (i.e., "Highly preferable" or "Moderately preferable") and "Podcasts" the least preferred (i.e., "Not very preferable" or "Not preferable at all")** media types. "Question and answer sites" and "Online written documentation" are also among the common sources that respondents choose to learn new skills/concepts.

We also analyzed the responses to see whether the years of professional experience have an impact on the preferred media for learning new skills/concepts. Fig. 8 shows that **as participants gain work experience, they rely less on "Videos" and instead prefer "Question and answer sites"** as their primary source for

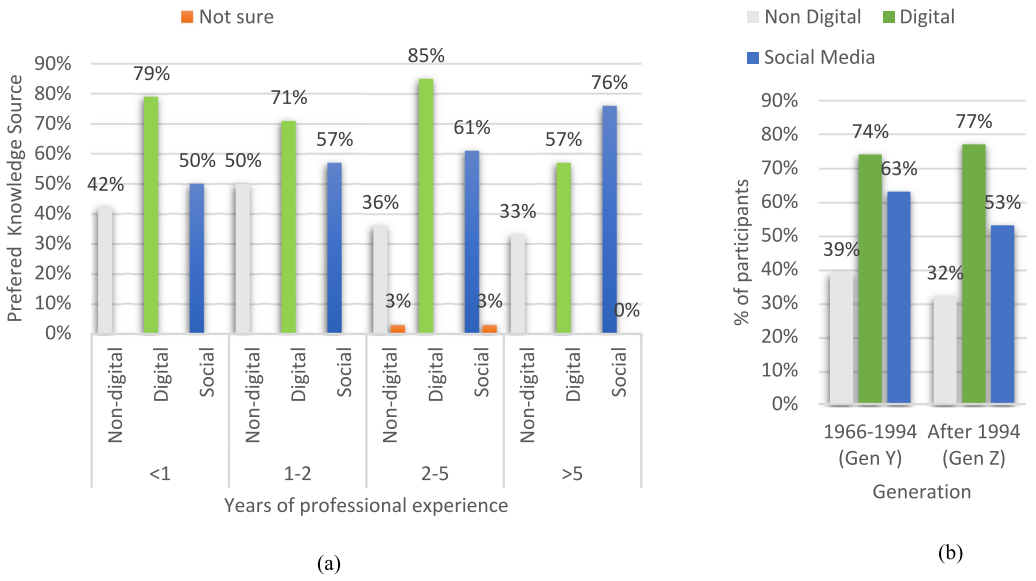


Fig. 6. Preferred **source of developer knowledge** for software engineering related tasks based on: experience (a) and age group (b) of the participants.

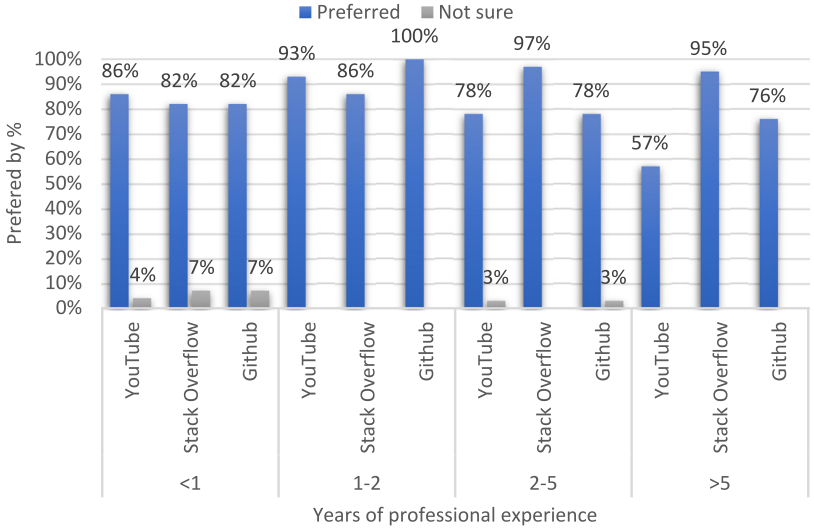


Fig. 7. Importance **of different media types** for completing software engineering tasks based on professional experience.

learning new skills/concepts. However, the results are not statistically significant. This is due to the proportion of the participants who prefer "Videos" as a learning source that remains above 80% for participants in all categories (i.e., levels of experience).

Conclusion: While "YouTube" or "Videos" are the most popular information sources used by less experienced software engineers (less than 2 years of experience) to complete their tasks or learn new skills, more experienced professionals prefer "Question and answer sites" such as "Stack Overflow" to get help or improve their current skill set.

RQ2. How often and why do software engineers watch instructional screencasts?

Our survey shows that participants who consider themselves as "Software Tester", "Software Quality Assurance Engineer", "Graduate Student" or "Business Analyst" are among the top four categories who identified instructional screencasts found on "YouTube" as an important source for completing software engineering tasks (Fig. 9). While "Full stack Developers" and "Software Designers", which constitute 12% of our survey participants, found such "YouTube" screencasts to be a less important resource when completing a work task. One possible reason for this is that more experienced software engineers have already the technical expertise required for completing their assigned work tasks.

Our survey also shows that **half of the participants (51%) watch instructional videos as a learning resource on a regularly (weekly) basis** (Fig. 10(a)). Further analysis of our data shows that **software engineers watch most commonly screencasts to "Learn a programming language" or "Get familiar with a software application user interface"** (Fig. 10(b)), while watching videos for "Bug fixing", "Learn how to customize open-source projects' source code", and "Finding workarounds for other problems" are fewer common reasons (i.e., selected by fewer than 50% of the participants).

Among the reasons for watching tutorial screencasts are both technical and programming-related topics such as "Bug fixing", "Finding workarounds for other problems", "Get development tips and tricks", "Learn a programming language", and "Learn how to customize open source projects' source code", as well as more high-level (less technical) topics such as: "Learn new concepts", "Learn how to set up an application", "Get familiar

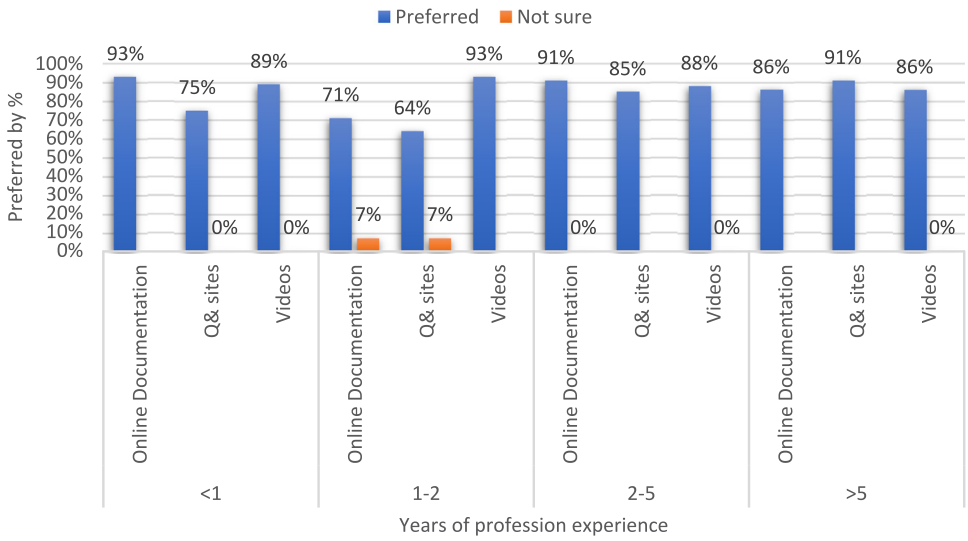


Fig. 8. Preferred use of **information sources** to learn new skills/concepts based on professional experience.

	Business Analyst	Database Developer	Full Stack Developer	Graduate Student	Researcher	Software Designer	Software Developer	Software QA Engineer	Software Tester	Undergraduate Student
Important	0.78	0.75	0.56	0.78	0.75	0.62	0.72	0.83	0.88	0.77
Somewhat important	0.22	0.25	0.31	0.15	0.17	0.31	0.16	0.17	0.12	0.15
Not important	0	0	0.12	0.07	0.08	0.08	0.12	0	0	0.08

Fig. 9. Proportion of participants with different role/occupations and how important screencasts found on YouTube are for them to complete their software engineering tasks.

with a software application user interface", and "Learn how to use specific features of a software application".

We further analyzed our survey data to identify why users watch screencasts. We base our analysis first on how frequently they watch tutorial screencasts (Fig. 11), and then analyzed if the work context of a person affects their tutorial watching behavior (Fig. 12). To gain further insights into the results we partially ordered the reasons for watching tutorial screencasts based on the proportions of participants that fall in each category (Figs. 11 and 12).

In addition, we analyzed the responses where the participants indicated that they "Almost never" watch tutorial screencasts as a learning resource (Fig. 11). We noticed that this group of users, if they watch tutorial videos, they watch these videos to "Get familiar with a software application's user interface" (71%) or to "Learn how to use specific features of a software application" (71%). Among the survey participants who indicated that they watch tutorial videos on a daily basis (18%), they use them mostly to "Learn a programming language" (94%), to "Get familiar with a software application's user interface" (94%), or to "Learn new concepts" (88%).

We next studied how the work context of a person affects their tutorial watching behavior (Fig. 12). For the analysis, we only considered roles (job descriptions) that were selected by 5 or more of our participants (Fig. 2). Our analysis shows that while **"Bug fixing" is overall among the least popular reasons for viewing videos, 83% of the "Software Quality Assurance Engineers" and 50% of the "Software Testers" indicated that they specifically watch videos for this reason**. Also, **"Graduate Students" and "Researchers" are more likely to rely on tutorial videos as a learning resource compared to "Undergraduate Students"**. This might be indicative that undergraduate students are often provided with resources that are specifically designed and provided for their courses and assignments, while graduate students and researchers will often have to find their own resources to learn new material.

We also analyzed if a software engineer's professional experience plays a role for watching how-to videos (Fig. 13). **While participants with less than of one-year work experience mostly watch videos for learning a programming language, more experienced users watch videos to familiarize themselves with an application or to learn how to setup an application**.

The **majority of our survey participants find tutorial videos an effective learning tool for software engineering** (i.e., 49.5% "Very effective" and 34.3% "Moderately effective") and the information presented in these videos to be useful (i.e., 32.3% "Very useful", 48.5% "Moderately useful").

In addition, we studied the responses to see whether tutorial videos are a popular learning resource for technical and non-technical tasks. As part of our survey, we provided the participants with the following two scenarios using WordPress,[21] a well-known content management system: 1. How to add a blog post to WordPress (less technical) and 2. How to restrict

[21] https://wordpress.com.

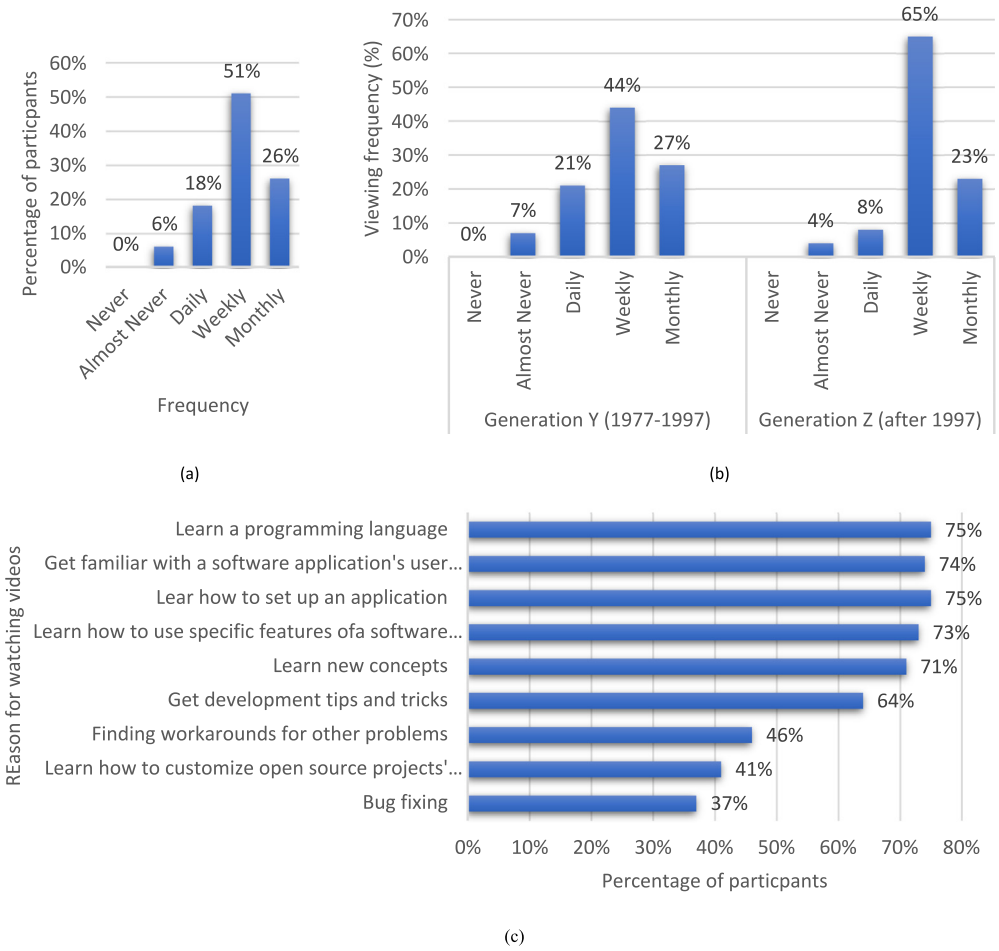


Fig. 10. Frequency of watching software engineering-related tutorial screencasts by participants (a), Frequency of watching software-related tutorial screencasts classified by generation and their motivation for watching them (c).

	Daily	Weekly	Monthly	Almost Never
Bug fixing	0.41	0.38	0.4	0.14
Find workarounds for other problems	0.76	0.41	0.48	0.14
Get development tips and tricks	0.82	0.75	0.52	0.14
Learn a programming language	0.94	0.76	0.8	0.14
Learn how to customize open source projects' source code	0.71	0.35	0.4	0.29
Learn new concepts	0.88	0.75	0.68	0.29
Learn how to set up an application	0.76	0.82	0.68	0.57
Learn how to use specific features of a software application	0.82	0.78	0.68	0.71
Get famiiar with a software application's user interface	0.94	0.74	0.68	0.71

Fig. 11. Proportion of participants classified by frequency and purpose of watching tutorial screencasts.

WordPress site access by IP (technical). We then asked the survey participants to indicate which type of documentation they prefer to use for each of the two scenarios. Our survey results (Fig. 14) show, while the two top choices for both types of tasks are "Tutorial video" and "Stack Overflow", **for the less technical task, users prefer to watch "Tutorial video" as the most popular learning source** (Fig. 14)**, while for the technical task "Stack Overflow" is the first choice** (Fig. 14). Further analysis, which classified the use of different documentation types not only based on task type but also the age group a participant belongs to (Fig. 15), showed that Gen Y users will resort for learning how to perform both, technical and non-technical tasks (technical task 39%, non-technical task 43%), more frequently to tutorial screencasts compared to participants that are part of the Gen Y category (technical task 19%, non-technical task 23%). We provide a detailed discussion of possible reasons for the low adoption rate and how they can be addressed in the Discussion section (Section 6) of this paper.

Conclusion: Our survey shows that tutorial screencasts are considered by most users to be useful for comprehending and learning new tasks/concepts. Tutorial screencasts are frequently used by all our survey participants for less technical software engineering-related purposes. Our survey also showed that while fewer people rely on screencasts for technical and coding tasks

	Business Analyst	Database Developer	Full Stack Developer	Graduate Student	Researcher	Software Designer	Software Developer	Software QA Engineer	Software Tester	Undergraduate Student
Bug Fixing	0.44	0.38	0.38	0.46	0.42	0.31	0.26	0.83	0.5	0.21
Learn how to customize open source project's source	0.56	0.25	0.56	0.52	0.38	0.54	0.33	0.67	0.38	0.20
Finding workarounds for other problems	0.44	0.38	0.5	0.56	0.5	0.46	0.33	0.83	0.75	0.35
Learn new concepts	0.56	0.75	0.88	0.79	0.71	0.77	0.7	1	0.88	0.42
Get development tips and tricks	0.56	0.62	0.81	0.64	0.62	0.77	0.7	1	0.88	0.78
Get familiar with a software application's user interface	0.89	1	0.88	0.64	0.62	0.77	0.67	1	1	0.57
Learn a programming language	0.89	0.75	0.88	0.79	0.79	0.77	0.74	0.83	0.75	0.56
Learn how to set up an application	0.89	1	0.69	0.82	0.62	0.77	0.7	1	0.88	0.63
Learn how to use specific features of a software application	0.89	0.88	0.69	0.8	0.71	0.77	0.72	0.83	0.88	0.78

Fig. 12. Proportion of participants with different role/occupation based on their motivation for watching screencasts.

	<1 year	1-2 years	2-5 years	>5 years
Bug Fixing	0.47	0.29	0.39	0.27
Learn how to customize open source project's source	0.46	0.36	0.48	0.32
Finding workarounds for other problems	0.66	0.36	0.48	0.27
Learn new concepts	0.76	0.5	0.79	0.73
Get development tips and tricks	0.76	0.71	0.64	0.55
Get familiar with a software application's user interface	0.69	0.86	0.73	0.86
Learn a programming language	0.86	0.5	0.79	0.77
Learn how to set up an application	0.76	0.57	0.88	0.68
Learn how to use specific features of a software application	0.82	0.79	0.7	0.68

Fig. 13. Participants' likelihood for watching software engineering-related how-to screencasts based on their years of professional experience.

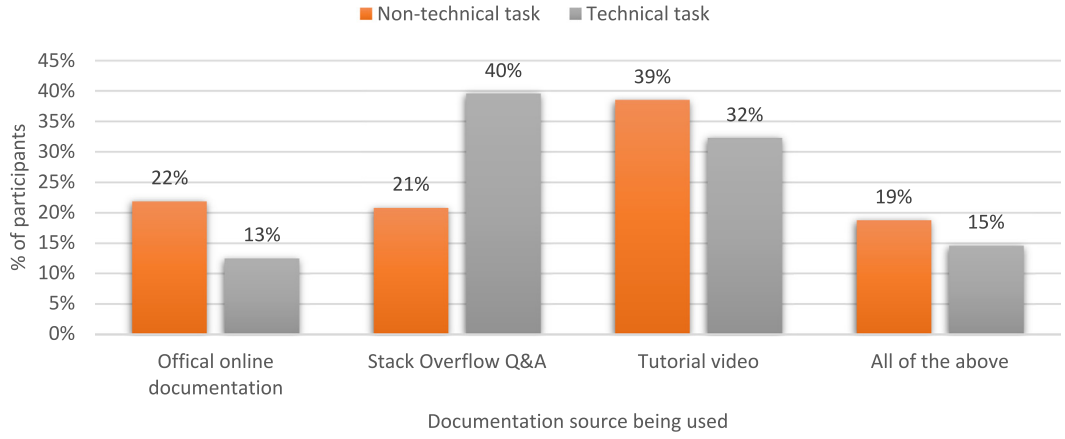


Fig. 14. Type of documentation used to learn how to perform technical and non-technical tasks.

among the ones who rely on screencasts for this type of tasks are mainly less experienced users and software testers/QA engineers.

RQ3. What are the important information sources of a screencast?

While tutorial videos are the most popular source for learning, **27% of our participants indicated that they "Often" are unable to find a tutorial that clearly meets their specific needs** and, 43% indicated that they "Sometimes" face this issue (Fig. 16).

We were further interested in identifying what components (video, audio, closed caption) of a tutorial video are the most important to the viewers. For this, we analyzed responses to survey questions related to the quality of the different information sources associated with a video and how important these sources are when users watch a video. As shown in Fig. 17(a), **most of our respondents (52.5%) indicated that they are "often" satisfied with the audio and visual quality of a tutorial video, and 13.1% indicated that they "always" are satisfied**.

A relevant tutorial video can be located using different criteria such as content (by watching part of a video) or using meta data, such as title, description, author, number of likes/dislikes, number of views. We therefore asked our participants how important each of these information sources (including the length of videos) are when selecting a video. **Most respondents (94%) consider the content of a video as an important (i.e., highly important or moderately important) aspect for choosing a tutorial video to watch. The second and third most important criteria are title (84%) and content of the speech transcription (69%) of a tutorial video**. While the number of likes (35%), number of views

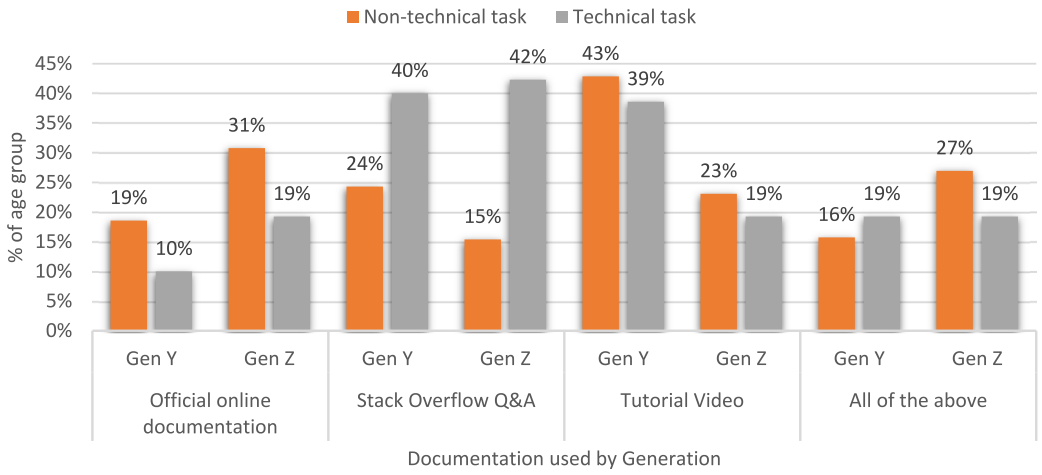


Fig. 15. Type of documentation used to learn how to perform (a) a less technical task; (b) when performing a technical.

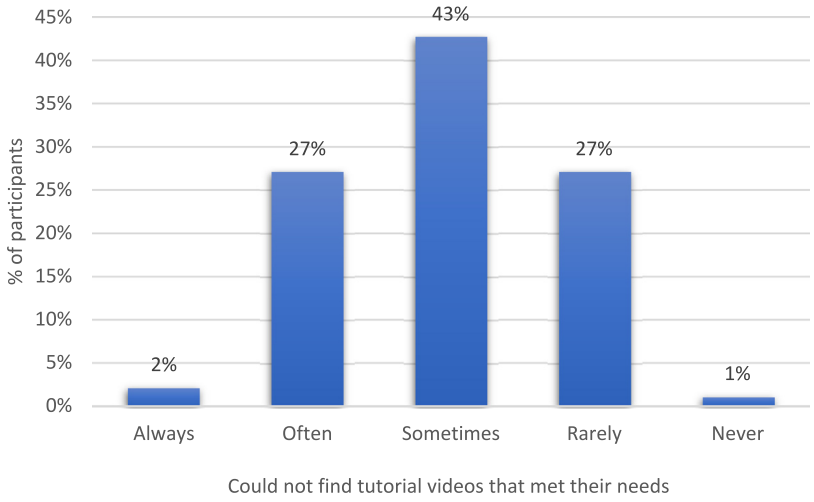


Fig. 16. The frequency of participants being unable to find videos that clearly describe what they need.

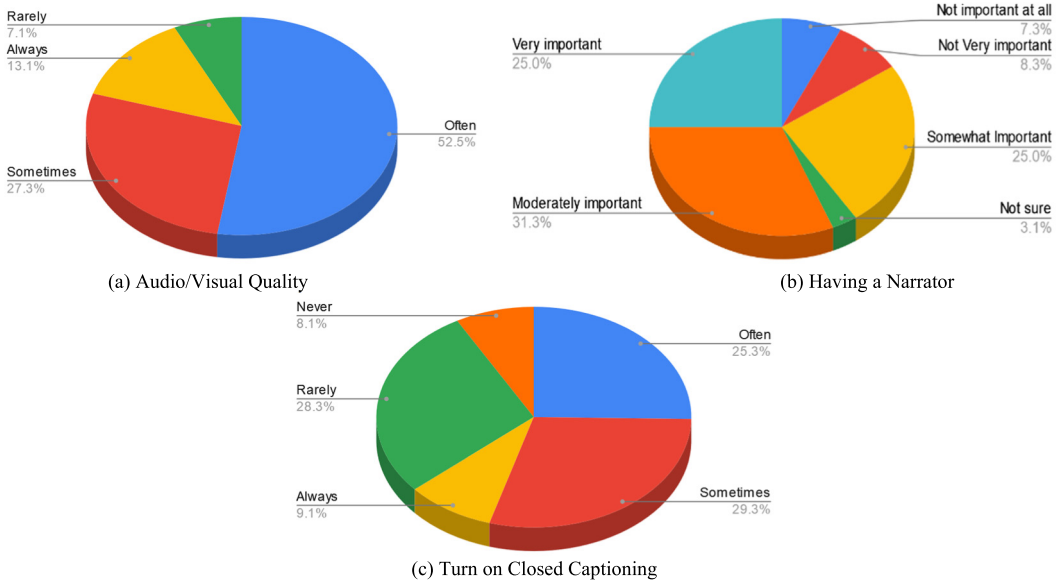


Fig. 17. How often are you satisfied with the audio and visual quality of a tutorial video? (a), How important is having a narrator for tutorial videos? (b), How often do you turn on closed captioning (subtitles) while watching a tutorial video? (c).

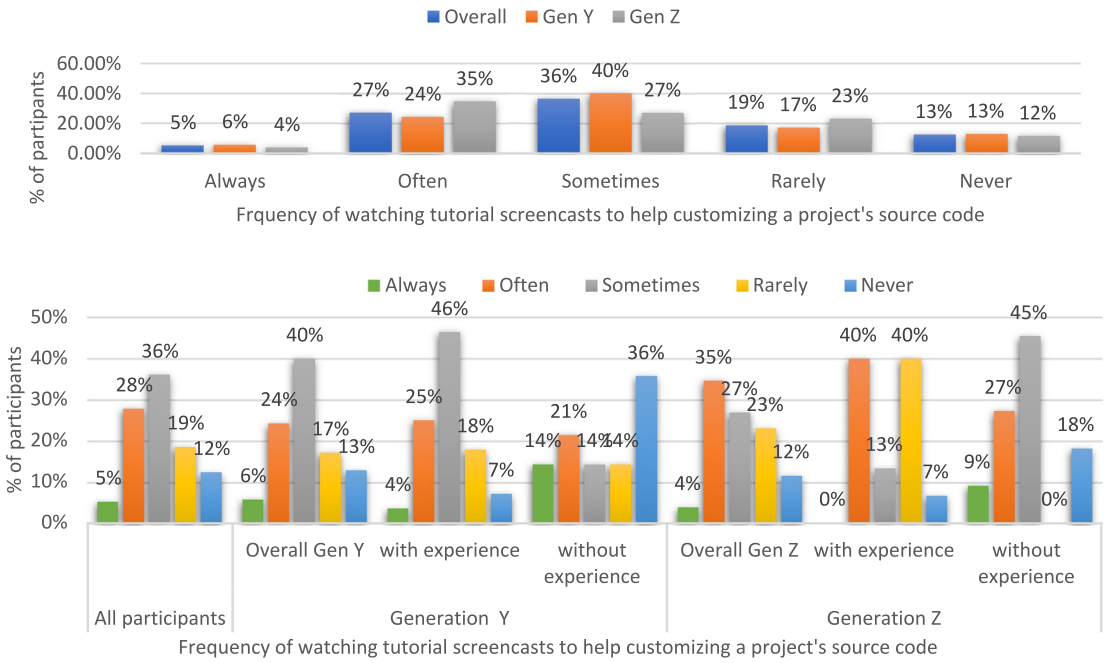


Fig. 18. Participants that would watch tutorial screencasts to help them customize source code (top figure by age group, bottom figure by age group and if participants have previous experience with open source projects or not).

(45%) and author (30%) of a video are less relevant (i.e., Not very important, Not important at all) for selecting a video.

Also **providing tutorial videos with narrators (and therefore also a speech component) is an important aspect for selecting a video**. The results show that, 56% of our respondents consider this as very important or moderately important factor (Fig. 17(b)) for selecting a video. Furthermore, **34.4% of our respondents indicated that they often or always turn on the closed captioning while watching a tutorial video** (Fig. 17(c)).

Conclusion: Overall, our respondents are mostly using the content (i.e., visual or speech) and title of a tutorial video for selecting which video to watch, while other metadata (i.e., video descriptions, author information and number of likes/dislikes/views) are less frequently used during the selection process.

RQ4. Are software engineers interested in tracing content from instructional screencasts to software artifacts?

For our scenario where a participant (without any prior knowledge of project's source code) watches an instructional screencast to help them customize the source code of an open source project, **28% of the participants indicated that they *often* try and 5% responded that they *always* try to do so when they contribute to a project** (Fig. 18). For the scenario where participants were asked if they have tried to locate the source code shown in a screencast, **27% of our participants stated that they *often* decide to watch a screencast of a software project whose source code they want to customize and 4% declared that they *always* try to do so** (Fig. 19). In both cases, when performing Chi-squared statistical tests, no significant difference could be observed between the generations (p-value of 0.55 for Fig. 18 and p-value of 0.82 for Fig. 19).

Since not all participants have prior experience with working on open source projects, we performed an additional analysis classifying the responses based on their prior experience on working with open source projects (Fig. 18 bottom and Fig. 19 bottom). While some differences between participants with and without prior experience in contributing to open source projects could be observed, these differences among the two generations were not statistically significant (p-value of 0.35 for Fig. 18 and a p-value of 0.20 for Fig. 19).

When asked what technical topics they would find the most interesting to watch as screencasts, our survey participants responded (Fig. 20), "Videos relevant to build configurations" and "Videos relevant to security issues" are the technical topics that are the most interesting to them.

From the results of the last three questions (Figs. 18, 19 and 20) **there is an implication that users can benefit from having bi-directional links between tutorial screencasts** (related to: GUI features, build configurations, security issues) and other software artifacts (especially source code). Establishing such traceability links will allow for recommending related documents or artifacts to improve system, subject, or feature comprehension and therefore facilitate performing a change/task.

We also included an open-ended question for participants to share any suggestion about the role or usefulness of screencasts in software engineering. Using the open coding approach introduced in Section 4.4, the following main topics were identified in the answers: (a) content design, (b) accessibility and availability, (c) traceability and, (d) non-relevant answers (i.e., answers not relevant to the question that we asked). We then created a codebook for each of these topics and calculated the inter-rater reliability using Cohen's Kappa (Sim and Wright, 2005) for each of the codes (Table 2). In what follows, we present our analysis for the codes, where the Kappa statistics show *substantial* or *almost perfect* agreement among the coders (marked in bold text in Table 2). For the "Accessibility and Availability" category we present the results for the codes whose Cohen's Kappa statistics show *moderate* agreement between the coders (marked in italic text in Table 2) since this is the highest agreement we have between our coders for this category.

Content Design: Nine of our survey respondents suggested to **design "*short*" tutorial screencasts that provide "*to the point*" information that is "*relevant to the topic*" of the tutorial and does not contain any noise or irrelevant information**. For example, one of the participants stated that *"Tutorial videos should be short and crisp in order to be more understandable"*. Another participant mentioned that, *"… One only thing is video should be up to-point and should have proper content as written*

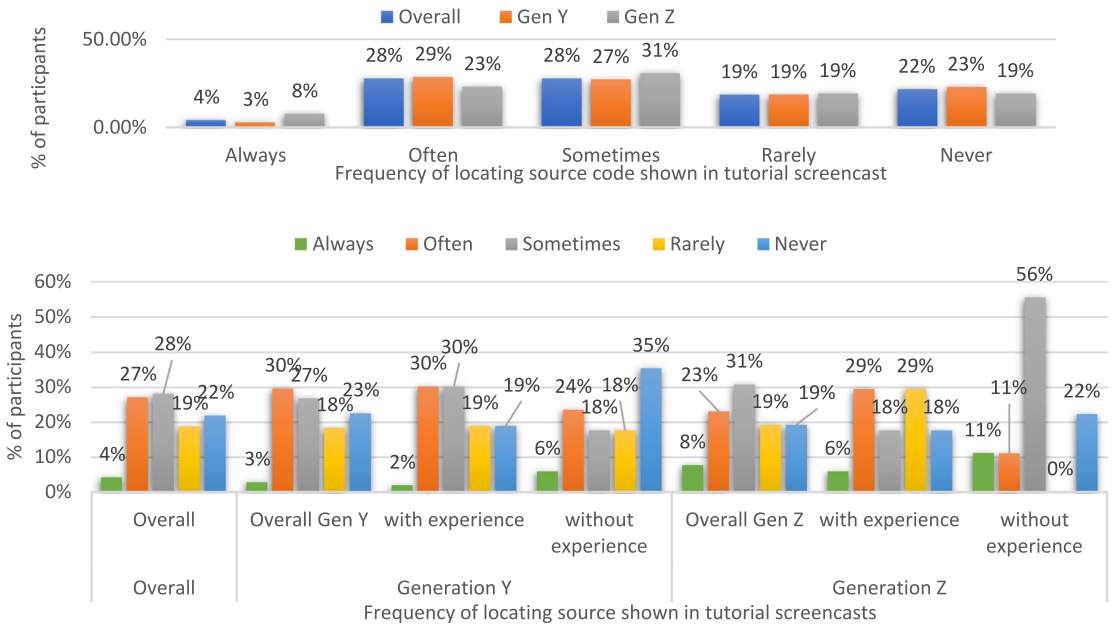


Fig. 19. Survey participants who would try to locate source code artifacts of a feature demonstrated in a screencast. (top by age group, bottom figure by age group and if participants have previous experience with open source projects or not).

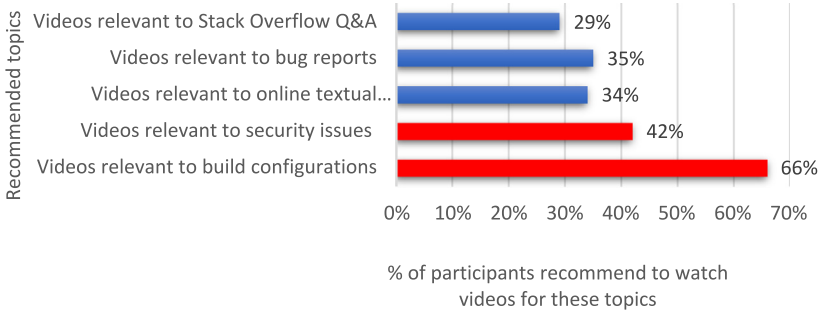


Fig. 20. The type of screencast that our participants find most interesting/relevant and would recommend watching.

Table 2
Topics, codes and, inter-rater reliability of participants' suggestions about the role or usefulness of how-to tutorial videos in software engineering domain.

Topic	Code	Cohen's Kappa
Content design	Step by step	0.56
	Audience expertise	0.45
	To the point	0.68
	Interesting	0.39
	Interactive	0.32
	Thorough	0.85
	Visual	0.39
	Good quality	0.85
	Narrator	0.48
Accessibility and availability	Find relevant screencasts	0.37
	Dedicated platform	0.48
	Create more screencasts	0.48
Traceability	**Keep screencasts up-to-date**	1.0
	Complement screencasts with other documents	0.78
Non-relevant	**Answer is not relevant to the question**	0.61

in the description".. Three of our participants believe that **information presented in tutorial screencasts that explain a certain task should have no missing/skipped information, with all steps explained in the video to be complete and "*thorough*"** (e.g., "*… Sometimes in videos the steps are skipped*") along with clarifications and explanations (e.g., "*Concept followed by explanation.*"). Other participants also mentioned that **having "*good quality*" audio/visual/content was something that they believe has a positive effect on the usefulness of tutorials in software engineering**.

Accessibility and Availability: This category addresses the issue of creating more videos and making them easier to find. For the following codes our raters had a *moderate* agreement. Two participants suggested to **consider a dedicated platform that contains tutorial screencasts related to only software engineering**. For example, one participant mentioned: "*A platform with high quality curated videos would be very useful (provided it is available for free or a minimal fee)*". Two other respondents mentioned that **there should be more tutorials created and made available online** (e.g., "*…it is essential that information in this form should be available as it makes the learning process interesting*".).

Traceability: Suggestions in this category are related to creating traceability links between screencasts and other types of software artifacts (e.g., source code, textual documentation, etc.). Two of our participants mentioned that **one issue with using tutorial screencasts is that the version of the software or source code that is being presented in the screencast is often outdated**. For example, one participant stated that "*… the problem is that keeping them up to date is hard because of the sequential*

nature of them", while another one mentioned that "*It is quite important in the sense that these videos are updated as the relevant software are updated. Else, documentation is more beneficial*". Screencasts are mostly short and provide visual learning that sometimes lack details and in-depth knowledge about a subject matter. Therefore, eight participants suggested to **complement software engineering-related tutorial screencasts with other types of documentation (online textual documentation, source code, PowerPoint, etc.) or use screencasts as a documentation that is complementary to other documentations**. For instance, one of the respondents' suggestion was: "*It is nice if the video publishers reference to other sources for more details on that particular topic*". Another respondent stated that "*Some people learn faster and better visually; tutorial videos should complement original documentation to provide better user coverage*". Providing well maintained traceability links between screencasts and other software artifacts could resolve some of this ambiguity and further improve the interpretation of screencast content.

Conclusion: Our survey not only shows that tutorial screencasts are a popular documentation and learning medium used by many software engineers, but users often follow up on what they have seen in screencasts by trying to trace video content they were watching to the corresponding source code implementing the viewed video content. Our survey participants also provided suggestions on how screencasts can be improved in terms of their content design (i.e., to the point, thorough, good quality) and traceability (to keep screencasts up-to-date or complement screencasts with other documents).

RQ5. What are the advantages and disadvantages of using tutorial videos over written documentation in the software engineering domain?

We asked our participants in two separate questions to share their experience in terms of *benefits/challenges* related to the use of tutorial videos when compared to written documentation. Our two raters analyzed the responses to these questions using again the open coding approach. The two raters identified the major answer categories for each question and created the corresponding codebooks. The categories, their related codes and their inter-rater reliability results (i.e., Kappa's statistics) are shown in Tables 3 and 4. In what follows we analyze the responses to these questions and the codes whose Kappa's statistic shows a *substantial* or *almost perfect* agreement between our raters (also marked in bold text in Tables 3 and 4). We first analyze the responses to the question about the *benefits* of using tutorial screencasts over written documentation (Table 3).

Content Design: The multimedia format of tutorial screencasts allows video creators to present content in a way that is more "*interesting*" compared to written documentation. Three respondents indicated that, **screencasts are "*more enjoyable*", and "*less boring*" to watch compared to traditional documentation.**

Ubiquitous and Ease of Access: Crowd-based tutorial screencasts are available in most cases online for free. Therefore, software engineers who are seeking help can easily access and use them by streaming them directly from video portals. Two of our participants mentioned that **"*easy access*" and being "*more accessible*" are two of the main benefits of using tutorial screencasts**. Two other participants stated that, **they can "*watch it anywhere and anytime*" and "*access anywhere with Internet facility*".**

Understandability: Responses that were coded as "*easy understanding*" are the most cited (41% of our respondents) benefit of using tutorial screencasts over written documentation. Based on the received responses, **tutorial screencasts are "*much easier to follow*" and "*easier to understand as you can watch the same problem being solved with your eyes*".** Also, we found that people who want to learn a new concept, use screencasts as their preferred learning source, especially for **visual learners** since **"*seeing things in action could result in better understanding of concepts*".** Furthermore, our survey shows that **screencasts help people with remembering and retaining new concepts in memory** since, as mentioned in one sample response, "*learning visually helps us to remember things easily*".

Table 3
The categories, codes and, inter-rater reliability of participants' responses about the benefits of using tutorial screencasts over written documentation.

Category	Code	Cohen's Kappa
Content design	**Interesting**	0.85
	Interactive	0.31
	Visual	0.32
Accessibility	**Ubiquitous and easy access**	0.71
Understandability	Avoid mistakes	0.49
	Easy understanding	0.61
	Fast learning	0.91
	Explanatory	0.64
Negative opinion	**Had negative opinion about screencasts**	0.85

The visual format of screencasts was stated by 39% of the participants as being **beneficial for "*fast learning*", especially "*for people who are slow-readers*"**. Also, our participants find using screencasts "*time saving*" since "*they convey more comprehensive knowledge in shorter time*" which makes learning "*faster and more efficient*" than when using written documentation.

Another benefit mentioned by five participants for using tutorial screencasts is that **they are easier to understand and more "*explanatory*"** compared to written documentation. One of the main reasons is that the multimedia format of screencasts allows screencast creators to leverage both audio (narrator) and visual (graphical presentations) components to convey their content. For example, participants stated that **"*tutorial videos give a better explanation of the tools/applications being worked with. Also, they show a step by step process on how things are done…*" and that "*videos use more clear words to describe the content than documentation*".**

Negative Opinions: Although this open question explicitly asked about the *benefits* of using tutorial screencasts compared to written documentation, we also received three coded responses with only negative opinions towards using tutorial screencasts. These **three participants prefer "*textbooks*" over screencasts because, as one of participants mentioned, videos lack "*…concrete information. Besides, it is impossible to search videos*"** and, another respondent indicated that **"*while videos focus on specifics, documentation is more detailed and structured. In the long run, reading the documentation has more benefits*".** In what follows, we elaborate more on these challenges of using screencasts over written documentation resulted from analyzing the four major categories of responses (Table 4) identified in our survey.

Reliability: The "*poor quality*" of screencasts is considered by three of our participants as one of their main disadvantages. For example, one respondent mentioned that **videos "*often could not cover the title and lack quality*"**. About 21% of the survey respondents listed "*lack of details*" as a disadvantage of using screencasts. For example, as **two of the respondents stated, "*written can sometimes provide more details over videos*" and "*videos might skip some parts*"**. This may be because video creators try to fit a larger topic in a short video. For instance, two of the participants indicated that, (a) "*The video tutorials are usually an overview of the topic. Detailed tutorials are rarely available. Mostly because the topic is too complex to fit in a video*" and, (b) "*… because of the time constraint, videos most often leave out information*".

Table 4
The categories, codes and, inter-rater reliability of participants' responses for related to disadvantages of using tutorial screencasts over written documentation.

Category	Code	Cohen's Kappa
Reliability	Lack of trust	0.43
	Poor quality	0.85
	Lack of details	0.65
	Version compatibility	0.80
Searchability/Sequential	**Long to watch**	0.73
	Waste of Time	0.38
	Not Searchable	0.19
	Hard to follow	0.66
	Not markable	0.66
Convenience	Internet	0.80
	Accent	0.65
	Not easy to use	0.66
	Not copy-pasteable	0.80
Effectiveness	**Not practicing**	1.0

In addition, two of the responses are concerned with "*version compatibility*" between the version of the software being presented in a screencast and the version which is used by the person viewing the screencast. Participants indicated **"*most of the time they are not version based…*" and "*it is very difficult/impossible to update a video, whereas it is easy to update text*"**.

Searchability: The sequential nature of screencasts limits the ability to directly search for content within the videos. As a result, users prefer short videos over longer videos, since they allow viewers to scan faster through the video to find relevant content. Also, when locating content, users will often skip non-relevant content, which is easier done for written documentation compared to watching or fast-forwarding through a tutorial screencast. Around 10% of our participants declared that tutorial **screencasts are "*long to watch*", meaning that "*they take too long to watch, and they include too much irrelevant information*" and "*time gets wasted*" especially since sometimes "*…they are super time consuming and they do not have a content that actually we want*"**.

Two participants were concerned with tutorial screencasts **being "*hard to follow*" since (a) we "*may miss some things in between if we do not listen carefully*" and, (b) "*… for a video, you have to go to the pace of the video, but with text you can go at your own pace*"**. Furthermore, two other participants indicated that **screencasts are "*not markable*" and there is "*difficulty in recording points*"**.

Convenience: Based on the received responses, we also noticed that respondents sometimes find it more convenient to obtain required information from written documentation due to convenience reasons. For example, two participants mentioned **the need for "*Internet*" access/usage as an inconvenience for watching tutorial screencasts since (a) "*access to high speed Internet is not available everywhere*" and, (b) watching videos requires "*more Internet usage*"**. Also, four of our respondents indicated that **they have difficulty with understanding the "*accent*" of some narrators: "*sometimes the accent of the tutor is hard to understand*"**. One participant mentioned that **screencasts are "*not easier to use than documentation*" and two other participants indicated that screencasts are "*not copy-pasteable*" especially for code snippets and commands**.

Effectiveness: Finally, one participant stated that **watching screencasts and "*not practicing*" makes them less effective: "*you might end up in tutorial purgatory, just watching without actually practicing*"**.

Conclusion: Our open-ended questions revealed that content, easy access, easy understanding, fast learning, and being explanatory are among the benefits of tutorial videos over written documentation. Our survey also showed that given the popularity of "YouTube" and "Videos" their benefits outweigh some of the disadvantages of screencasts over written documentation, such as content quality, convenience, version compatibility, and lack of searchability.

6. Discussion

In this section, we summarize and discuss our findings and lessons learned from the survey results before drawing some conclusions and propose future directions on how to improve tutorial screencasts in software engineering. It should be noted, since our survey participants were mostly younger and less experienced (fewer than five years of professional experience), the following discussions and findings are target for this new generation of software engineers.

6.1. Lessons learned

Our study confirms observations made by Eisner (2011), Wells et al. (2012) that the new generation of developer is well adapted and familiar with the use of social media resources and grew up with new learning/teaching styles, such as *blended-learning* (Garrison and Kanuka, 2004) as an integral part in teaching and prefer multimedia based communication and social networks over textual documents (Wells et al., 2012). From a software engineering perspective, our study shows that the use of information found in social media depends on the level of experience, the role in the project and the specific type of work context the participants are involved in. We further observed that the most popular usage of screencasts is for less technical tasks and as a resource for comprehending and learning new tasks and concepts. This is while less experienced participants as well as Software Testers and QA engineers are among the most common users of screencasts.

While screencasts are a popular source of documentation, users do not like the fact that screencast content is not searchable, requiring them to watch a screencast and spend time to locate relevant parts (if they exist at all). Based on our findings, integrating crowd-based screencasts with other software artifacts or documentation through establishing traceability links is mentioned as a key requirement by the survey participants. Such integration of crowd-based screencasts with other artifacts especially source code can help overcoming many challenges (e.g., version compatibility, switching between multiple media channels, and code change (Storey et al., 2014)). Given that screencast users in our study were among the less experienced software engineers, this can considerably speed up their maintenance tasks. These findings are further supported by our own previous works, where we mined and linked screencasts that showcase GUI features of a software application (Moslehi et al., 2018, 2016) or linked screencasts to security vulnerabilities of an API. These observations are also supported by other researchers (Ponzanelli et al., 2016; MacLeod et al., 2017; Ponzanelli, 2019), who studied the creation of programming screencasts as a documentation or linking programming screencasts with other relevant software artifacts (e.g., source code, Stack Overflow question and answers). We believe that this line of research can lead to the development of new tools that provide not only a better integration of screencasts with other software artifacts but can also provide software engineers who might have a different background and expertise, within a work environment they are more familiar with.

Based on the feedback and suggestions that we received from our survey participants, which also highlights both advantages and disadvantages of using tutorial screencasts over written documentation, we are able to draw some recommendations that can be used to improve the integration and adoption of screencasts in the software engineering domain.

6.2. Recommendations for the adoption of instructional screencasts

In what follows, we do present some general recommendations that, if applied, can further improve the automated integration of instructional screencasts with other software documentations. Many guidelines exist for creating video and screencast content. YouTube already has a dedicated section[22] that provides guidelines for video creators to boost their channel subscriptions and keep their audience engaged and satisfied with the content presentation, quality, and design, these guidelines are generic to be applicable to any genre (e.g., education, beauty, music, gaming). Other guidelines, which are introduced to build educational channels[23] are more closely related to tutorial videos that we studied in this work. However, these guidelines do not fully address the specific needs of software engineering-related technical videos (e.g., versioning information, source code linking). Also, video production guidelines[24] on YouTube lack guidance for creating high quality **screencasts,** which includes screen recording. Also, the provided guidelines for YouTube search and discovery[25] do not consider the content (of GUI and speech) in (screen captured) videos since YouTube's search works based on metadata, title, and thumbnails.

The findings from our survey provide insights on how users with different background and experience levels perceive different forms of documentation. Such insights can be used to recommend documentation media that matches more closely users' needs and preferences. For instance, depending on the level of experience, occupation, age, and problems being solved, the results from our survey can be used by project managers and organizations to include new media types as part of the software lifecycle artifacts to improve the quality, usefulness and overall acceptance of system documentation and increase the productivity of their workforce. In this work we specifically focused on how screencasts are used by software engineers. Screencasts are the type of documentation that is perceived differently depending on the work and user context. While previous work (e.g., MacLeod et al. (2017)) has identified best practices on how to create and design screencasts to document code, the objective of our recommendations is to provide a more comprehensive framework of recommendations for creating **non-programming and programming** related software engineering screencasts.

Our results support the observations and suggestions made in MacLeod et al. (2017) about designing short videos that are to the point (Ponzanelli et al., 2016; MacLeod et al., 2017) and of high quality. In addition, our survey also highlight that screencast viewers would like to see screencasts to be complemented with other types of documents and that locating content and promoting videos are important issues for them.

Here we complement these guidelines by our findings and recommendations (items that are marked by an * are also presented in YouTube Creator Academy):

(1) *Provide a subtitle. Screencast creators should provide a (English) transcript of their video and its content (even if it is in a language other than English[26]), eliminating the dependency on automated transcription services, which are still prone to errors.
(2) Recording high definition (HD) videos that have a readable text size (MacLeod et al., 2017), which can improve the visual quality of screencasts.
(3) Reducing noise in videos by showing only the features that are related to the topic that is being presented in the video and by minimizing the amount of time spent on showing non-relevant subject on the screen. This can be done by providing links to other screencasts that cover other topics (MacLeod et al., 2017).
(4) *Providing complementary documents (MacLeod et al., 2017) such as written documents or slides that can further improve the linking results.
(5) *Annotating or tagging screencasts using keywords that can help to clearly distinguish one video that covers a topic from a video that covers another topic (MacLeod et al., 2017).
(6) How-to screencasts should include information about the version of the software application whose features are being demonstrated to improve the location of the correct source code version.

Furthermore, based on these findings, we can now provide screencast creators and organizations with more specific recommendations that can further improve the quality and potential user acceptance of software engineering screencasts.

Build a dedicated platform: Currently there are platforms that are paid[27] or unpaid,[28] or platforms that provide Massive Open Online Courses (MOOCs)[29] by universities[30] that provide content that is designed and created by university professors or professionals from well-established companies (and not by the crowd). However, to serve a broader range of content creators and users (i.e., crowd), providing platforms dedicated to crowd-based software engineering tutorials, that replace YouTube which is used for multiple purposes, can provide better **monitoring and enforcement of quality standards** related to audio and visual quality of the videos. Such platforms can also support offline viewing of screencasts and provide a dedicated location for hosting specialized screencasts, which will improve accessibility and availability of screencasts. In addition, such platforms can benefit the overall content quality of hosted videos on these specialized platforms, by promoting videos with high quality content and removing non-relevant videos.

Make content searchable and support annotations: As our survey confirmed, the content of a tutorial screencast is the main deciding factor for users to select a screencast. As some of our survey participants stated, current video portals and players lack (semantic) search or index functionality, which would allow them to search for specific content or verify that a video contains the expected content without having to view the full video. In addition, current video players and portals lack the ability to annotate and save user annotated video content. Providing such more advanced viewing and annotation features could greatly benefit further use of screencasts in the software engineering community.

Traceability links: Traceability links are not only an essential aspect in software engineering, but they can also be used to accommodating people with different documentation media preferences and needs. Such traceability links allow to identify and make relevant complementary documents and software artifacts directly available to users. Traceability links between screencasts

22 https://creatoracademy.youtube.com/page/home.
23 https://creatoracademy.youtube.com/page/course/educational-channel?hl=en.
24 https://creatoracademy.youtube.com/page/home.
25 https://creatoracademy.youtube.com/page/lesson/discovery?hl=en.
26 https://creatoracademy.youtube.com/page/lesson/global-format#strategies-zippy-link-3.
27 Such as https://www.udemy.com/.
28 Such as http://mooc.org/.
29 https://www.classcentral.com/help/moocs/.
30 https://www.classcentral.com/universities.

and other software artifacts can also address or highlight potential inconsistency of versions shown in screencasts and software artifacts and tools used by the viewers to avoid potential misleading or missing information. Integrating screencasts by linking them to other software and documentation artifacts can also benefit development processes where creating and updating formal documentation is not a priority.

Consider target audience's level of expertise: As our survey shows, the level of professional experience plays a significant role in selecting screencasts as information resource for software engineering tasks. Among the reasons why software engineers do not watch tutorial screencasts is that viewing videos is time consuming and the viewed screencasts might even not be relevant or useful. For example, less experienced software engineers rely more often on screencasts to provide them with some additional and animated explanation of a subject matter to fully comprehend it. The content of screencasts should therefore be adapted to the level of expertise of the potential audience by: (1) adjusting the pace accordingly while describing or demonstrating a topic so that the audience can easily follow the information flow presented in the videos (e.g., for less experienced developers) or quickly allow them to find information they need (more experienced audience), (2) In addition, provide references to complimentary (ideally matching the expertise level of the viewer) to other videos related to the topic.

Promote videos for the right community: Our findings also illustrate that level of experience, occupation, and age of people does affect why users are viewing tutorial screencasts. Video creators should consider this information when promoting their videos or selecting portals where they publish their videos. For instance, visual learners (Gen Y and Z) prefer to watch videos when learning new concepts. However, experienced software engineers prefer Q&A sites to receive their technical information. For screencasts covering non-technical topics, the target audience and therefore content delivery should be geared more towards Gen Y and Z viewing preferences, while for technical videos they should match more closely the level of experience of their target audience.

7. Threats to validity

As previously discussed, the general motivation for conducting our survey is to provide insights on how and why screencasts are used by software engineers. The analysis of the responses from our user survey show that tutorial screencasts are a popular information source, especially among less experienced software engineers. Screencast content is an important information factor for many viewers when deciding which tutorial to watch. However, there are some threats to external, internal and construct validity regarding the user study and its results that need to be addressed in future work.

External Validity. For our survey we received 99 responses, which were collected by sharing our questionnaire on different social media platforms and mailing lists. Most of our participants (around 60%) were graduate students. This is mainly because we shared the questionnaire on different social media and mailing lists as well as Moodle[31] that is only accessible to graduate and undergraduate students enrolled in a software engineering course. A potential limitation of our study is that our survey results may not be generalizable to non-students, even though 55.5% of the surveyed students had more than two years of professional experience. In order to mitigate this threat, future work needs to attract a more diverse set of participants with different levels of expertise and work background, to further improve the generalizability of the results and to provide additional insights on how screencasts are used among software engineers in an industrial setting.

Construct Validity. We designed a questionnaire consisting of 34 questions and our main objective was to identify how and why software engineers use tutorial screencasts. For the survey, we used multiple choice, Likert scale, and open-ended questions to gather information from our participants. Our survey allowed us to analyze how screencasts are used by software engineers therefore capturing and revealing useful results. Future work should extend our survey setting to allow for attracting software engineers to participate in a study that also includes interviews for additional clarification and follow-up questions, and/or allows for user studies, where participants are applying different types of information and documentation sources on concrete case studies. Such user studies could provide additional insights on how users are applying screencast tutorials or other types of documentation in real work contexts. However, such user studies require significant resources and preparation time, including the preparation of pilot (feasibility) studies to improve the reliably of the study. As a result, it is often not possible to conduct such user studies for a large number of participants. Furthermore, future work should emphasize the latest experiences which participants had with screencasts and other types of crowd-sourced and multi-media based software documentation, since recent experiences can be remembered more clearly and can lead to receiving even more reliable responses.

Internal Validity. Having mostly graduate students as our participants, may have resulted in results that a more applicable towards less experienced software engineers. Although our analysis results provide meaningful insights on how and why screencasts are used by software engineers, additional survey participants with more diverse backgrounds would be required to obtain a more representative sample and to increase the confidence in statistical significance of the study results. Another threat to internal validity in our study are related to the causality of our results, since in some cases it is possible that our result might be influenced by the criterion validity of our questionnaire. For example, the decision of a participant to enable closed captioning while watching a screencast might be influenced by the fact that the participant is not a native speaker of the language in which they watch tutorials in. A possible mitigation approach would require to further compare participants' responses with objective measures related to the specific criteria (e.g., language used in the screencast and a participant's native language).

8. Conclusion

In this work we conducted a user study to understand how and why software engineers with different roles and backgrounds use multimedia documentation and more specifically tutorial screencasts to get help in performing their software engineering tasks. We also aimed to find out how mining crowd-based multi-media content and linking it with other artifacts and documentation can help provide different generations of software engineer with easier access to a documentation they are more familiar with. We received 99 responses by sharing our questionnaire on different media platforms. Our findings revealed useful information and insights into the usage of screencasts for different purposes as well as what are other preferred media resources used for software engineering related tasks by our participants.

Among our main findings are that the new generation software engineers with less than two years of professional experience are the ones who rely more on "YouTube" or "Videos" to complete their software engineering tasks or learn new concepts/skills. Also, we found that more experienced developers'

[31] https://moodle.concordia.ca/moodle/.

first choice of getting help is “Question and answer sites” such as “Stack Overflow”. Our other finding is that screencasts are more used for getting help with less technical tasks while “Question and answer sites” are more suitable to be used to fulfill technical tasks in software engineering.

We analyzed our received responses to better understand the criteria used by our participants to decide when and why they watch a tutorial screencast. Our findings show that a viewer’s level of experience, occupation and therefore also the task they are performing, play a significant role in selecting tutorial screencasts as a resource to support and complete their software engineering-related tasks.

In our study we also investigated what are important information sources that our respondents consider when selecting a tutorial video to watch. Based on our findings, high quality content and having a narrator are among most important criteria. Also, being able to search video content could vastly increase the accessibility and usefulness of screencasts as a source of software documentation. We also received interesting suggestions on the role and usefulness of tutorial screencasts in software engineering domain.

As part of our future work we plan to increase the number of survey participants with a more diverse expertise and conduct additional user studies that will involve follow-up interviews and additional application scenarios.

CRediT authorship contribution statement

Parisa Moslehi: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft. **Juergen Rilling:** Validation, Formal analysis, Writing – review & editing, Supervision. **Bram Adams:** Validation, Formal analysis, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We thank Amine Barrak for his help with the open coding approach.

References

Ahmad, A., Feng, C., Ge, S., Yousif, A., 2018. A survey on mining stack overflow: question and answering (Q&A) community. Data Technol. Appl. 52 (2), 190–247.

Bao, L., Li, J., Xing, Z., Wang, X., Xia, X., Zhou, B., 2017. Extracting and analyzing time-series HCI data from screen-captured task videos. Empir. Softw. Eng. 22 (1), 134–174.

Bao, L., Li, J., Xing, Z., Wang, X., Zhou, B., 2015. Reverse engineering time-series interaction data from screen-captured videos. In: 2015 IEEE 22nd Int. Conf. Softw. Anal. Evol. Reengineering, SANER 2015 - Proc. pp. 399–408.

Bao, L., Xing, Z., Wang, X., Zhou, B., 2015. Tracking and analyzing cross-cutting activities in developers’ daily work (N). In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering ASE. pp. 277–282.

Bao, L., Xing, Z., Xia, X., Lo, D., 2019. VT-Revolution: Interactive programming video tutorial authoring and watching system. IEEE Trans. Softw. Eng. 45, 823–838.

Barzilay, O., Treude, C., Zagalsky, A., 2013. Facilitating crowd sourced software engineering via stack overflow. In: Finding Source Code on the Web for Remix and Reuse. Springer New York, New York, pp. 1–19.

Black, S., Harrison, R., Baldwin, M., 2010. A survey of social media use in software systems development. In: Proc. 1st Work. Web 2.0 Softw. Eng. pp. 1–5.

Bolton, R.N., et al., 2013. Understanding generation Y and their use of social media: A review and research agenda. J. Serv. Manag.

Brandt, Joel, Guo, Philip J., Lewenstein, Joel, Dontcheva, Mira, Klemmer, Scott R., 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. In: CHI '09, Association for Computing Machinery, New York, NY, USA, pp. 1589–1598. http://dx.doi.org/10.1145/1518701.1518944.

Campbell, J.C., Zhang, C., Xu, Z., Hindle, A., Miller, J., 2013. Deficient documentation detection: A methodology to locate deficient project documentation using topic analysis. In: IEEE Int. Work. Conf. Min. Softw. Repos. pp. 57–60.

Eghan, E.E., Moslehi, P., Rilling, J., Adams, B., 2020. The missing link – a semantic web based approach for integrating screencasts with security advisories. Inf. Softw. Technol. 117.

Eisner, S.P., 2011. Managing generation Y. IEEE Eng. Manag. Rev. 39 (2), 6–18.

Ellmann, M., Oeser, A., Fucci, D., Maalej, W., 2017. Find, understand, and extend development screencasts on youtube. In: Proc. 3rd ACM SIGSOFT Int. Work. Softw. Anal. - SWAN 2017. pp. 1–7.

Escobar-Avila, J., Parra, E., Haiduc, S., 2017. Text retrieval-based tagging of software engineering video tutorials. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion ICSE-C. pp. 341–343.

Escobar-Avila, J., Venuti, D., Di Penta, M., Haiduc, S., 2019. A survey on online learning preferences for computer science and programming. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training, Montreal, QC, Canada. ICSE-SEET, pp. 170–181. http://dx.doi.org/10.1109/ICSE-SEET.2019.00026.

Freund, L., 2015. Contextualizing the information-seeking behavior of software engineers. J. Assoc. Inf. Sci. Technol. 66 (8), 1594–1605.

Garrison, D.R., Kanuka, H., 2004. Blended learning: Uncovering its transformative potential in higher education. Internet High. Educ. 7 (2), 95–105.

Highsmith, J.A., Highsmith, J., 2002. Agile Software Development Ecosystems. Addison-Wesley.

Jiau, H.C., Yang, F.-P., 2012. Facing up to the inequality of crowdsourced API documentation. ACM SIGSOFT Softw. Eng. Notes 37 (1), 1–9.

Khandwala, K., Guo, P.J., 2018. Codemotion: expanding the design space of learner interactions with computer programming tutorial videos. In: Proceedings of the Fifth Annual ACM Conference on Learning at Scale - L@S '18. pp. 1–10.

Krug, J., 1998. Understanding generation x. J. Manage. Eng..

Li, H., Xing, Z., Peng, X., Zhao, W., 2013. What help do developers seek, when and how? In: 2013 20th Working Conference on Reverse Engineering, Koblenz. WCRE, pp. 142–151. http://dx.doi.org/10.1109/WCRE.2013.6671289.

Lissitsa, S., Kol, O., 2016. Generation X vs. generation Y - A decade of online shopping. J. Retail. Consum. Serv.

MacLeod, L., Bergen, A., Storey, M.-A., 2017. Documenting and sharing software knowledge using screencasts. Empir. Softw. Eng. 22 (3), 1478–1507.

MacLeod, L., Storey, M.-A., Bergen, A., 2015. Code, camera, action: how software developers document and share program knowledge using youtube. 2015 IEEE 23rd Int. Conf. Progr. Compr..

McCrindle, M., Wolfinger, E., 2009. The ABC of XYZ : Understanding the Global Generations/ Mark McCrindle with Emily Wolfinger. UNSW Press Sydney.

McLure Wasko, M., Faraj, S., 2000. ‘It is what one does’: why people participate and help others in electronic communities of practice. J. Strateg. Inf. Syst. 9 (2–3), 155–173.

Miles, M.B., Huberman, A.M., Saldana, J., 2018. Qualitative Data Analysis: A Methods Sourcebook. SAGE Publications.

Mohorovičič, S., 2012. Creation and use of screencasts in higher education In: MIPRO 2012-35th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. - Proc. pp. 1293–1298.

Moslehi, P., Adams, B., Rilling, J., 2016. On mining crowd-based speech documentation. In: Proceedings of the 13th International Workshop on Mining Software Repositories - MSR ’16. pp. 259–268.

Moslehi, P., Adams, B., Rilling, J., 2018. Feature location using crowd-based screencasts. In: Proceedings of the 15th IEEE Working Conference on Mining Software Repositories MSR. pp. 192–202.

Nasehi, S.M., Sillito, J., Maurer, F., Burns, C., 2012. What makes a good code example?: A study of programming Q & A in StackOverflow. In: 2012 28th IEEE International Conference on Software Maintenance ICSM. pp. 25–34.

Ott, J., Atchison, A., Harnack, P., Bergh, A., Linstead, E., 2018. A deep learning approach to identifying source code in images and video. In: Int. Conf. Min. Softw. Repos. pp. 376–386.

Parnin, C., Treude, C., Grammel, L., Storey, M.-A., 2012. Crowd Documentation: Exploring the Coverage and the Dynamics of API Discussions on Stack Overflow. Georg. Tech Tech. Rep..

Parra, E., Escobar-Avila, J., Haiduc, S., 2018. Automatic tag recommendation for software development video tutorials. In: Proceedings of the 26th Conference on Program Comprehension - ICPC ’18. pp. 222–232.

Pham, R., Singer, L., Liskin, O., Filho, F.F., Schneider, K., 2013. Creating a shared understanding of testing culture on a social coding site. In: 2013 35th International Conference on Software Engineering ICSE. pp. 112–121.

Poche, E., Jha, N., Williams, G., Staten, J., Vesper, M., Mahmoud, A., 2017. Analyzing user comments on youtube coding tutorial videos. In: 2017 IEEE/ACM 25th International Conference on Program Comprehension ICPC. pp. 196–206.

Ponzanelli, L. others, 2019. Automatic identification and classification of software development video tutorial fragments. IEEE Trans. Softw. Eng. 45, 464–488.

Ponzanelli, L., et al., 2016. Too long; didn't watch!: extracting relevant fragments from software development video tutorials. In: Proceedings of the 38th International Conference on Software Engineering - ICSE '16. pp. 261–272.

Reisenwitz, T., 2009. Differences in generation X and generation Y: Implications for the organization and marketers. Mark. Manag. J.

Seemiller, C., Grace, M., 2017. Generation Z: Educating and engaging the next generation of students. About Campus 22 (3), 21–26.

Sim, J., Wright, C., 2005. The kappa statistic in reliability studies: Use, interpretation, and sample size requirements. Phys. Ther. 85 (3), 257–268. http://dx.doi.org/10.1093/ptj/85.3.257.

Storey, M.-A., 2015. Selecting research methods for studying a participatory culture in software development. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering - EASE '15. pp. 1–5.

Storey, M.-A., Singer, L., Cleary, B., Figueira Filho, F., Zagalsky, A., 2014. The (R) Evolution of social media in software engineering. Proc. Futur. Softw. Eng. - FOSE 2014. pp. 100–116.

Storey, M.-A., Treude, C., van Deursen, A., Cheng, L.-T., 2010. The impact of social media on software engineering practices and tools. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. pp. 359–364.

Subramanian, S., Inozemtseva, L., Holmes, R., 2014. Live API documentation. In: Proceedings of the 36th International Conference on Software Engineering - ICSE 2014. pp. 643–652.

Turner, A., 2015. Generation Z: Technology and social interest. J. Individ. Psychol.

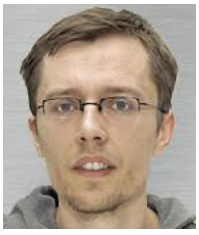
Wells, J., Barry, R.M., Spence, A., 2012. Using video tutorials as a carrot-and-stick approach to learning. IEEE Trans. Educ. 55 (4), 453–458.

Yadid, S., Yahav, E., 2016. Extracting code from programming tutorial videos. In: Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software - Onward! 2016. pp. 98–111.

Zhao, D., Xing, Z., Chen, C., Xin, X., Li, G., 2019. Actionnet: Vision-based workflow action recognition from programming screencasts. In: IEEE/ACM 41st International Conference on Software Engineering, Montreal, QC, Canada. ICSE, pp. 350–361. http://dx.doi.org/10.1109/ICSE.2019.00049.

Parisa Moslehi is currently a data scientist and machine learning engineer at Group Dynamite working in a team to deliver innovative, cloud-based AI and machine learning solutions to the enterprise. She received her Ph.D. in Computer Science from Concordia University, Montreal, QC, Canada in 2020. She was a research assistant in Ambient Software Engineering Group (ASEG) lab at Concordia University, and the lab on Maintenance, Construction and Intelligence of Software (MCIS) at Polytechnique Montreal. Her research area includes mining software repositories, software evolution, software maintenance, information retrieval, and recommender systems.

Juergen Rillingis a Professor in the Department of Computer Science and Software Engineering at Concordia University, Montreal, Canada. He obtained a Diploma degree in Computer Science from the University of Reutlingen, Germany, in 1991 and a M.Sc. in Computer Science from the University of East Anglia, UK, in 1993. He received his Ph.D. from the Illinois Institute of Technology, Chicago, USA, in 1998. The general theme of his research over the last 19 years has been on providing software maintainers with techniques, tools, and methodologies to support the evolution of software systems. His current research focus is on supporting the modeling and analysis of global software ecosystems. He has published over 100 papers in major refereed international journals, conferences, and workshops. He also serves on the program committees of numerous international conferences and workshops in the area of software maintenance and program comprehension and as a reviewer for all major journals in his research area.



Bram Adams is an associate professor at Queen's University. He obtained his Ph.D. at the GH-SEL lab at Ghent University (Belgium). His research interests include mining software repositories, software release engineering and the role of human affect in software engineering. His work has been published at premier software engineering venues such as EMSE, TSE, ICSE, FSE, MSR, ASE and ICSME. In addition to co-organizing the RELENG International Workshop on Release Engineering from 2013 to 2015 (and the 1st IEEE Software Special Issue on Release Engineering), he co-organized the SEMLA, PLATE, ACP4IS, MUD and MISS workshops, and the MSR Vision 2020 Summer School. He has been PC co-chair of SCAM 2013, SANER 2015, ICSME 2016 and MSR 2019.