Learning Rapidly Changing Frameworks

Ryan Turner
Georgia Institute of Technology
rturner63@gatech.edu

ABSTRACT

Developers often face difficulties when learning new frameworks that are volatile – ones that are quickly changing over time and often have little long term support or feature deprecation periods. As teams work to improve developer experience, they must take into account how developers learn these frameworks and produce learning materials that best meet the developers needs. Informal learning is well understood, and many learning tools have been examined for use by developers and fields. This research seeks to answer details instead of how learners use these tools when given a specific problem. So how do developers learn rapidly-changing frameworks? 38 developers responded to a brief survey with details about how they learned React Native, a current example of a rapidly changing framework. This survey looked at the respondent's background, the tools they used in the past, and details of their preferred learning tool over the previous 90 days. The data was compared to normal distributions to look for preferences and patterns. Developers were shown to have favorite tools and commonly used tools, though they didn't directly correspond. Learning was immediately put to use in most cases. Developers most often did not complete consuming their learning tool, but they still had positive outcomes. Over one-third of developers learning during the period was a result of changes in the framework itself. While this research specifically examined React Native, many of the learner preferences may be applied to other frameworks and technologies. A learner behavioral pattern emerges that may be considered to build more effective educational material.

Author Keywords

just in time; framework; rapidly-changing

INTRODUCTION

Studying rapidly-changing technology like augmented reality and artificial intelligence most often takes the form of informal learning. For these, often documentation and samples are all the materials provided. Looking at more established topics like desktop computing and service management, the case is quite different. Corresponding certifications exist like CompTIA A+ and ITIL Foundation with accompanying curriculum and a network of professional teachers. The options vary greatly as the technology's maturity level changes. While computer science as a whole is a young practice compared to other kinds of engineering, certain topics pose bigger challenges to learners.

There is a wide body of knowledge on informal learning, as well as learning by developers. However, there has not been significant study on how learning occurs in various contexts. One such context is in the use of a framework, which is defined as "a form of software reuse that primarily promotes the reuse of entire architectures within a narrowly defined application domain" [9]. Couple that framework with rapid change, and learners show behaviors best summarized as "just in time learning".

How do developers learn rapidly-changing frameworks?

To answer this question, the "where, when, and why" are considered. Specifically, the three following research questions are investigated:

RQ1: What learning tools are preferred?

Understanding what tools are preferred helps to identify what traits developers prefer in their tools, as well as what content creators use. In pursuing this, various tools for learning will be considered. Some examples include institution-sponsored courses, massive open online courses (MOOCs), books, weblog tutorials, forum and knowledge market threads, workshops, and even hackathons. This can answer what best suits rapidly-changing frameworks. This detail represents **where** the learning occurs.

RQ2: Is their learning motivated by a pressing need?

When working with rapidly changing frameworks, developers may choose to learn just-in-time to solve a new problem. Other developers may instead respond to rapid change by closely following progress and proactively learning new materials. These two are enabled by different kinds of educational content: the former needs to be problem based and easily found, whereas the latter needs to be more engaging and use constructionism. This detail represents **when** the learning occurs.

RQ3: What scope of knowledge do developers pursue?

The scope of learning tools widely varies, from weeks-long MOOCs to ten-minute read weblog posts. This question asks whether the learner is studying to move on with their day or instead to better understand something they already practice. They may be using tools entirely, piece-meal, or just referencing code snippets. This answers **why** the learning occurs.

RELATED WORK

Learning of rapidly changing frameworks is primarily informal. Informal learning is defined as learning with little structure and often as the byproduct of some other activity [4]. It is "relevant to practice in many cultures and contexts" and "[takes] place wherever people have need, motivation, and opportunity for learning" [4]. While this method is often looked at from a business context [4,8], motivation could come as uncertainty of creating a new solution or resolving a bug in code. Informal learning is also often defined by a social component. For example, while the trigger may be local to an individual, it is often the result of an external change such as results of a tester's work or a request from a product owner. Informal learning takes place around all of us day to day.

Significant literature exists covering various kinds of informal learning tools and the way that they are consumed. Developers have many solutions available for learning. The next paragraphs describe the informal learning activities that take place within a subset.

In research of programmers tool usage, [6] show how peer interaction leads to learning and discovery. Discovery is more specifically called "the first stage of some kinds of learning." [6] contrasts peer interaction from Marsick's definition of informal learning; however, when considered according to the definition above, the distinctions fade. In this work, [6] demonstrates how situations like happenstance interaction, pair programming, and even change notification often result in peer observations and recommendations. These represent the "discoveries" of tools, but in some cases they also represent the teaching material itself: with pair programming, the peer interaction often creates an incidental situated learning experience.

Hackathons are presented as "excellent informal learning platforms" [7]. A hackathon is a "fast-paced event where competitors work in teams to go from an idea to working software or hardware within a single day or a weekend." Like in [6]'s work, the authors cite peer-learning as common place in this setting. Consistent with the definition of informal learning, at hackathons the problems create the need, the gamification create the motivation, and the industry mentors plus online resources create the opportunity. While the research focuses on learnings within teams, they also recognize that the learning environment created at a hackathon excels at producing industry-relevant learnings and skills. It is considered a "great opportunity to learn" by participants [7]. Frameworks may be introduced or brushed up on at these events, though the researchers did not consider this.

MOOCs present as popular online digital learning tools today. Popular examples include Coursera, edX, and Udacity. When considering learning theories within MOOCs, there tend to be either connectivism-driven MOOCs (cMOOCs) or extension MOOCs (xMOOCs) [10]. These are significantly different in their application of learning models, and more needs to be done to make use of informal, personalized, or professional learning on these platforms [10]. From anecdote, MOOCs appear to be a popular tool to learn frameworks.

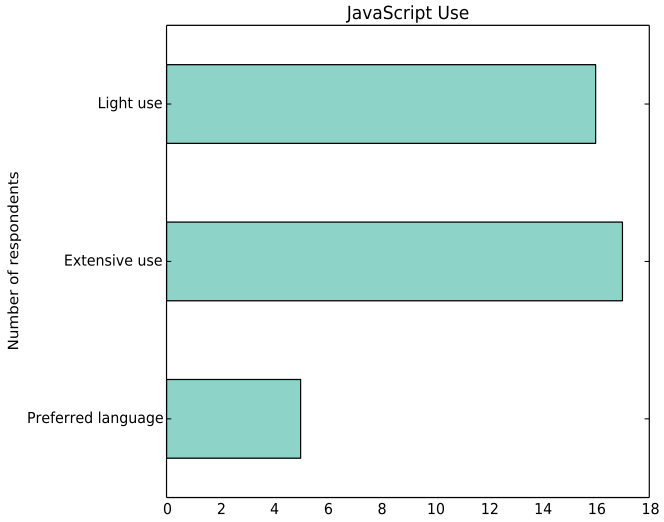

Figure 1. Respondents had varied experience levels with JavaScript.

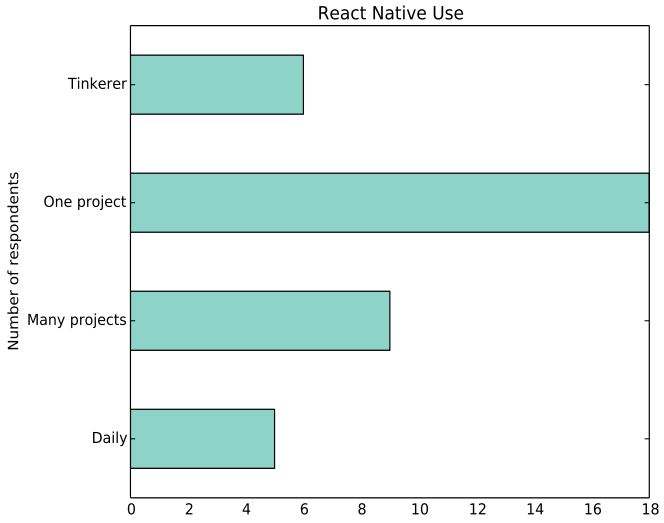

Figure 2. Respondents had varied experience levels with React Native.

Informal learning, specifically by developers about emerging topics, relates to a topic proposed by [8]: "What antecedents and conditions facilitate continuous learning, especially informal learning, and knowledge sharing?" In considering the form of learning, [8] mention that "informal learning may be equally important to or even more important than other forms of learning." In fact, informal learning is so prevalent in organizations that it may account for up to 75% of learning [1,8]. Yet, the future issue is still proposed: "What are the antecedents and consequences of informal learning?" [8].

METHODOLOGY

To better understand learning in this area, developers use of learning tools was reviewed. Ideally, developers would have been observed in their learning. Instead, a survey approach was taken. This is partly due to the feasibility to observing these interactions as well as the importance of capturing

responses from an international audience. Finally, this approach permitted developers to report on their perceptions of the learning: why did it occur, how successful was it, and how soon was it put to use.

For this research, the sample was limited to React Native practitioners and their use of the React Native framework. This helped the survey target a cohesive group of respondents, and the author's networking in the community aided in soliciting responses. React Native also is a widely used framework with the second highest contributor count on GitHub in 2017 [3]. Its varied population of users coupled with its prestable status makes for an excellent case as a rapidly changing framework to study.

The questionnaire survey was promoted through Twitter, chatrooms, mailing lists, and signs at a conference. Many respondents did not qualify for the survey and had their responses removed due to inexperience with the framework. Of those who remained, most were professional developers who had a medium-to-light level of experience with JavaScript (see figure 1). About half of the respondents were beginner React Native users, and the rest of the respondents were evenly split between completing a single project, completing multiple projects, and daily use (see figure 2).

The questionnaire survey starts by asking about the respondent's experience professionally, formal learning in programming, experience with JavaScript, and familiarity with React Native. React Native familiarity was used as a filtering question; if the respondent indicated no familiarity, the survey ended.

For respondents who were familiar with React Native, they were then asked questions about their learning and use in the previous 3 months. This was limited in order to avoid asking the respondent to describe experiences beyond recollection. First, the respondents were asked "What learning tools have you used? Check all that apply." 11 options for this question were presented, plus an unstructured other option. These options wildly varied between the formal and informal, small and large. Respondents were then asked to estimate how much of their time was spent learning as a percentage of their overall work. Next, they were asked what tool was the most useful to them. The remainder of the questions asked were limited to their use of that tool, seeking to understand when the learning took place, how it was used, and how beneficial it was.

The respondent data was collected and then reviewed. Python *pandas* was utilized in order to interpret the data and send it to *Scipy.stats* module and *matplotlib* for analysis and visual representation. Using this approach, some basic counts and pivot tables provide insights to respondent's preferences, and analyses like the chi square test and the Kruskal Wallis test were utilized to calculate statistical significance on the data.

RESULTS

Overall, 38 subjects provided qualified survey results. In this section, answers to the three research questions are presented, as well as some general trends identified in the data. At the end of each subsection, the data is briefly summarized.

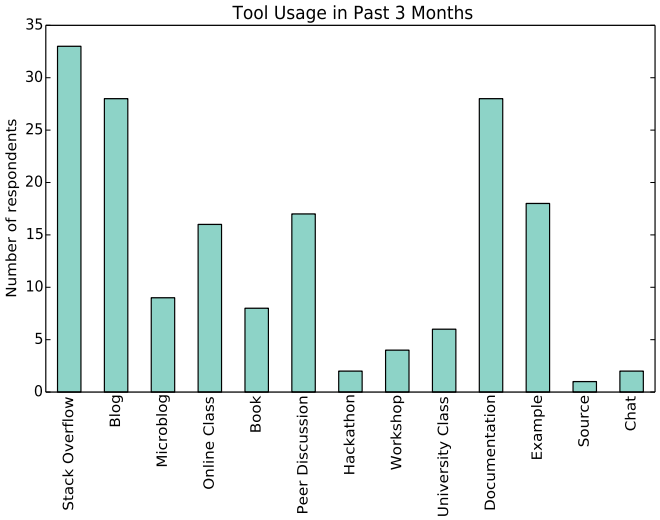

Figure 3. A clear preference for specific learning tools emerged.

Specific tools are preferred, but usage doesn't necessarily follow

Respondents indicated 172 different instances of tool usages in the past 3 months. Developers showed that specific tools were used more frequently than others. A chi-squared test comparing the results to a normal distribution resulted in $p = 5.80 \times 10^{-18}$. Clear leaders in tool usage included Stack Overflow, Blogs, and Documentation. Online classes, peer discussion, and examples were middle of the pack performers. The least commonly used tools included source code, hackathons, chat, workshops, university courses, books, and microblogs. The tool usage frequency is indicated in figure 3.

Each subject was asked to indicate a single preferred learning tool. A chi-squared test comparing the results to a normal distribution resulted in $p = 0.063$. There was a clearer split between winners and losers in this area; the winners were clearly stack overflow, blogs, online classes, books, and peer discussion; all other tools had significantly less learner preference. Documentation and examples, which showed average and above average usage occurrence, notably received no selections. The tool usage frequency is indicated in figure 4, with categories that received no selections omitted.

Some selections like hackathons, workshops, and microblogs were indicated as favorites by their only users. This is likely a result of users having an individual recent or highly significant learning session rather than a recurring learning pattern.

Most learning was applied the same day

In the general case, over the same 3 month time period respondents indicated how soon they used what they learned. 55.3% of respondents indicated that it was immediately put to use. The second most common delay was a day or so at 15.8%. The remainder of responses was nearly evenly split between many days, not yet, and it was to explain an existing usage. For specific occurrences, refer to figure 5.

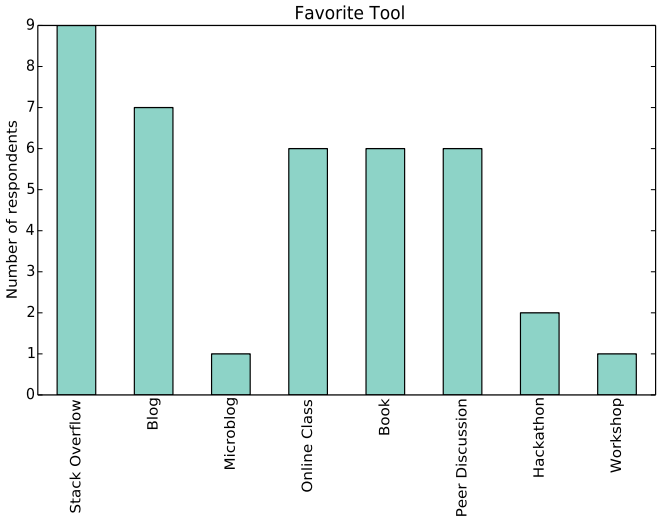

Figure 4. Developers indicated more polarized tool preference than usage.

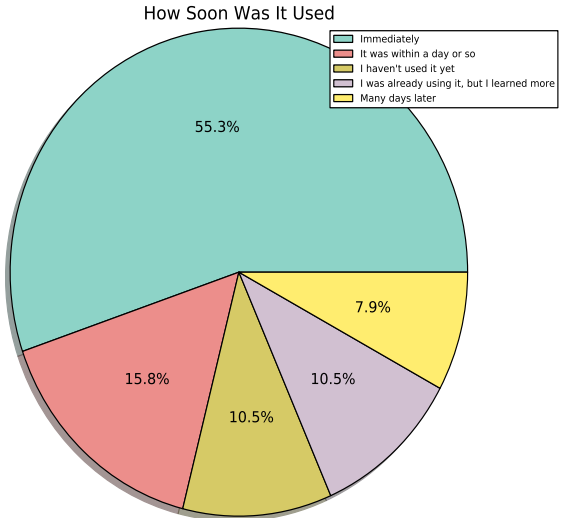

Figure 5. Nearly three-quarters of usage occurred within a day.

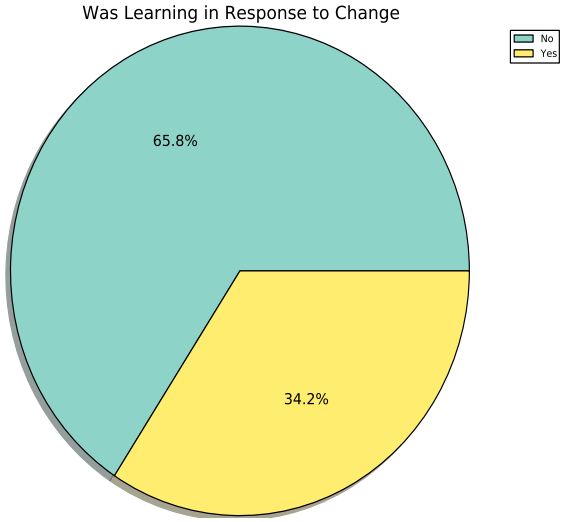

Figure 6. Over one-third of developers learning rapidly changing frameworks were doing so in response to a change in the framework.

While learners largely depending on just-in-time learning practices, the specific reasons for this weren't well understood. 34.2% of learners were responding to changes in the framework as indicated in figure 6. Other reasons like new feature needs, inexperience, and curiosity weren't evaluated but likely contributed.

Learners generally don't complete the full learning unit, but it's still useful

Learners generally did not complete their entire learning unit. TODO: add overall numbers.

Learning module completion did vary by tool. Tools with fewer than 5 responses were removed due to sample size being too small. With the resulting dataset, a Kruskal-Wallis test comparing the number of completions vs non-completions grouped by tool resulted in $p = 0.053$. Stack Overflow specifically showed an 8:1 ratio of non-completions to completions, while blogs showed 1:6. Online courses and peer discussions showed a 2:1 ratio, and documentation was even. Refer to figure 7 for a visual representation.

Despite generally not completing the full unit, user satisfaction was high. Figure 8 shows that the most popular selection was a 4 out of 5 on satisfaction.

LIMITATIONS

This research surveyed the community that uses React Native. Both the way the data was collected and the population surveyed present as limitations. While this work may apply to other web or mobile frameworks, some of the presumptions made may not match those for other tools.

In the sample used for this work, the social norms practiced by the developers may show favoritism to specific tools and methods of consumption. React Native was originally developed and is presently supported by Facebook [2], a social networking company. On the developer support page, Stack

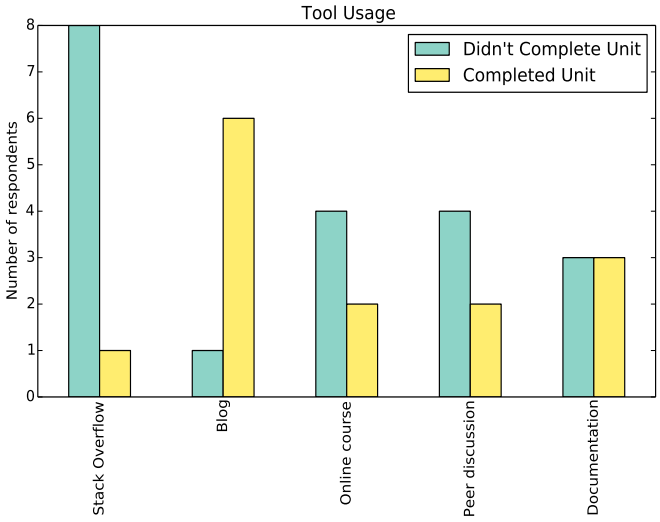

Figure 7. Learning unit completion varied highly by tool.

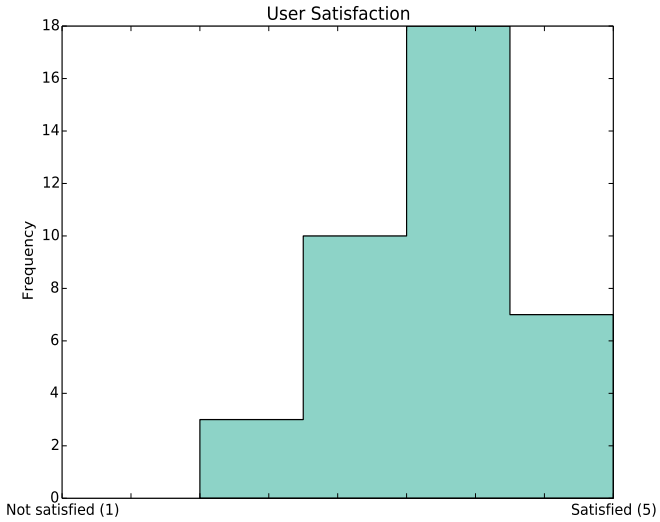

Figure 8. Learners are generally satisfied, though there is still more room to improve.

Overflow, forums, chat, and microblogs are all suggested as ways to get help [2]. As a result of the framework's recognition of these, other tools may be at a disadvantage. In fact, as of July 2018, there are no references to books or courses in the support documentation.

Additionally, React Native shares a lot in common with ReactJS, a JavaScript web frontend framework. Users of React Native often come from a web development background, which has many differences with software development. Detailed analysis of the difference can be found in [5], but of specific interest are differences in application characteristics, people involved in development, and social issues.

The challenges of React Native's mission also limit the application of this research. React Native ultimately creates iOS and Android mobile applications and relies on their native APIs to enable much of its functionality. The learner frequently is required to learn about these upstream projects, which expands the scope of learning beyond what other frameworks may require. This also puts users in the position of watching and anticipating changes in the parent APIs in order to better plan their work in React Native, such as recently with React Native's 0.56 release that responds to many previously announced changes, like new versions of Babel, iOS API, Xcode IDE, and Android SDK. Developers working on the React Native platform are sometimes forced to learn just-in-time with the framework's change even though it's highly anticipated; learners of frameworks with simpler dependency trees may not be forced into this pattern.

This research was conducted by surveys alone, which limited how the subjects were interrogated. To help increase response rate, the length of the survey was limited. Respondents were asked detailed questions only about their most useful tool; responses on tools that weren't highly favored by the respondent may reveal other usage patterns. Additionally, in making the survey straight forward for the respondents, some statistical analysis was complicated. More traditional survey design may lead to cleaner results.

Finally, some aspects of the survey asked about a 90 day time period. This is a long time for respondents to recollect, so there may be inaccuracies in respondent reporting. Additionally, within this time period learning may have occurred for a variety of reasons. Respondents were asked about a single occurrence, not the entire set. The problems faced may be related to current events in the framework, like a defect in the most recent release or an emerging trend in user interface design. These specific, widespread cases may skew the responses to the learning patterns that best address that issue rather than issues in general.

CONCLUSION

Software developers often have formal backgrounds, yet much of their technical contributions come from informal learning. While various kinds of informal learning have been studied, there is little understanding of how tools are used to learn emerging technologies. In fact, learning tools like MOOCs and peer discussion have been studied to understand their contributions to developer knowledge, but this has gen-

erally been oriented from the specific tool perspective or informal learning overall.

Developers using rapidly-changing frameworks often need to use areas that are new, or changes occur to existing areas they already use. In this use, learners have specific sets of tools they depend on day to day as well as specific tools that are their primarily preferences. Most knowledge is immediately put to use. Learners generally don't complete the full course, but that doesn't mean there are subpar learning outcomes. In general, learners reported high satisfaction.

Developers were polled about their learning React Native, which is a leading mobile development framework with many corporate sponsors. With monthly releases that have hundreds of contributions, there is a lot of regular change. Its developers generally are working in JavaScript, and some web technologies are used. This population may not be like others outside of the mobile and web development space. Additionally, this research depended on online surveys that asked respondents to speak about their experience in the past 90 days; this may have led to misrememberings.

FUTURE WORK

This study may be considered a pilot study. More data is needed and the survey needs updating to reach more substantial conclusions. Detail about tool specific learning outcomes may be possible, and correlations between learning tool usage and overall learning duration may also be possible. Additionally, future work with this survey could restructure answers to result in data that is more practically analyzed.

Originally, interview surveys were planned as well to collect qualitative data and investigate the motivation behind learning. Low survey participation prevented this from occurring. While surveys did not take place, a survey script was prepared and is available for future use in the repository.

References

1. DJ Bear, HB Tompson, CL Morrison, et al. 2008. Tapping the potential of informal learning: An astd research study. *Alexandria, VA: American Society for Training and Development*.

2. Facebook. 2018. React native a framework for building native apps using react. Retrieved July 25, 2018 from **https://facebook.github.io/react-native/help**

3. GitHub. 2017. GitHub octoverse 2017 | highlights from the last twelve months. Retrieved July 24, 2018 from **https://octoverse.github.com/**

4. Victoria J Marsick and Karen E Watkins. 2001. Informal and incidental learning. *New directions for adult and continuing education* 2001, 89: 25–34.

5. Emilia Mendes. 2014. Web development versus software development. In *Practitioner's knowledge representation*. Springer, 13–25.

6. Emerson Murphy-Hill and Gail C Murphy. 2011. Peer interaction effectively, yet infrequently, enables programmers to discover new tools. *Proceedings of the acm 2011 conference on computer supported cooperative work*, ACM, 405–414.

7. Arnab Nandi and Meris Mandernach. 2016. Hackathons as an informal learning platform. *Proceedings of the 47th acm technical symposium on computing science education*, ACM, 346–351.

8. Raymond A Noe, Alena DM Clarke, and Howard J Klein. 2014. Learning in the twenty-first-century workplace. *Annu. Rev. Organ. Psychol. Organ. Behav.* 1, 1: 245–275.

9. Alessandro Pasetti. 2002. *Software frameworks and embedded control systems*. Springer Science & Business Media.

10. Ahmed Mohamed Fahmy Yousef, Mohamed Amine Chatti, Ulrik Schroeder, M Wosnitza, and H Jakobs. 2014. A review of the state-of-the-art. *Proceedings of CSEDU*: 9–20.