Project Proposal – Educational Strategies for Highly Evolving Technology

Ryan Turner

June 2018

This project will conduct research on how people learn about software development. Specifically, how people who identify as developers (with some previous level of training or experience) go about learning dynamic and growing topics. This proposal will describe the research problem and its concerns, discuss the key deliverables of the work, and provide a coded work breakdown structure with a calendar. The intent of this proposal is to collect feedback and ultimately serve as a contract for this research as part of studies at Georgia Institute of Technology.

Research Problem

For creators of technology, educating users is often a challenge. Some creators tend to introduce formal courses, while some others rely on users referencing documentation and trial-and-error. In some, third parties have created learning materials like weblog tutorials or even reference examples.

When considering the traits of technologies, a pattern appears to emerge: those that are rapidly-changing like augmented reality and artificial intelligence seem to depend on informal education. For these, often documentation and a sample 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.

Rapidly-changing topics seem to be learned in fragmented, previously unmeasured ways. This research will focus on the following problem:

How do developers learn rapidly-changing frameworks?

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

Background literature

In this research, non-institutional learning methods will be considered. Informal learning is one of these methods (Marsick and Watkins 2001), which is defined as learning with little structure and often as the byproduct of some other activity (Marsick and Watkins 2001). Informal learning is "relevant to practice in many cultures and contexts" and "[takes] place wherever people have need, motivation, and opportunity for learning" (Marsick and Watkins 2001). While this method is often looked at from a business context (Marsick and Watkins 2001; Noe, Clarke, and Klein 2014), motivation could come as uncertainty of creating a new solution or resolving a bug in code. A social component is also defining for informal learning. For example, while the trigger may be considered locally 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.

In examining how developers learn rapidly-changing frameworks, we must start by reviewing their options. Developers have many solutions available for learning (see Research Problem). The next paragraphs describe the informal learning activities that take place within a subset.

In research of programmers use of tools, Murphy-Hill and Murphy (2011) show how peer interaction leads to more discovery. Discovery is more specifically called "the first stage of some kinds of learning." Murphy-Hill contrasts peer interaction from Marsick's definition of informal learning; however, when considered according to the definition above, the distinctions fade. In this work, Murphy-Hill 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. The same may apply to frameworks as well.

Hackathons are also proposed as "excellent informal learning platforms" (Nandi and Mandernach 2016). 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 Murphy-Hill and Murphy (2011)'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 (Nandi and Mandernach 2016). 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) (Yousef et al. 2014). 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 (Yousef et al. 2014). From anecdote, MOOCs appear to be a popular tool to learn frameworks.

The proposed research relates to a topic proposed by Noe, Clarke, and Klein (2014): "What antecedents and conditions facilitate continuous learning, especially informal learning, and knowledge sharing?" In

considering the form of learning, Noe, Clarke, and Klein (2014) 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 (Noe, Clarke, and Klein 2014; Bear et al. 2008). Yet, the future issue is still proposed: "What are the antecedents and consequences of informal learning?" [noe2014learning].

In the problem itself, the term "frameworks" is used. For the sake of this research, a framework is defined as "a form of software reuse that primarily promotes the reuse of entire architectures within a narrowly defined application domain" (Pasetti 2002).

Research methodology

A survey research methodology will be used for this work. In order to help control internal validity, the question was constrained to only researching the learning of frameworks.

Dependent variables for this research will include:

- Tools selected
- Time in which learning occurs
- Duration for the learning

Independent variables for this research will include:

- Prior knowledge and experience (both in the framework and in computer science generally)
- Learning style preference
- Access to learning tools

The dependent variables directly map to the research questions proposed; the independent variables will be recorded and included in the analysis.

In order to control for internal validity, the population surveyed will be limited to members of the React Native community. This population is picked simply as a matter of convenience for the author, however it ensures that all subjects are learning the same framework. Additionally, subjects will be asked to limit their recollection to only recent times (some number of months) so as to avoid bad reporting; this problem is anticipated based on findings from Murphy-Hill and Murphy (2011).

In order to control for external validity, it's important to determine what rapidly-changing means and show where this research may be applied. The author does not intend for this research to be inferred as applicable to mature frameworks, or even outside of software frameworks. Additionally, work must be done to establish the characteristics of the React Native framework and its community.

Responses to the surveys are the needed data for this research. Should they not be available, either the scope of the study will be reduced, or the surveys will be re-distributed with incentives and more prominent sponsors (community leaders, educators, etc). If they still are not available, the study will change to the "supply side," looking at the learning tools and teachers instead.

Key Deliverables

Status Reports

There will be five weeks of status reports provided; each will present the progress made, challenges encountered, and an updated project plan. This will quickly and clearly show whether the project is on schedule, as well as help determine if the project proposal is being adhered to.

Milestones

There will be two intermediate milestones, both largely focusing on the surveys. In the first milestone, both surveys will be presented. Background context will inform peer reviewers of the research topic. Feedback by the mentor and peers will be addressed and incorporated into the survey prior to being distributed.

In the second milestone, the results and analysis from the first survey will be presented. This will also in-

clude background context and the very beginnings of analysis. Peers will learn about the research, and feedback will be gathered on the initial direction of the paper.

Final Deliverables

The final project materials will be in a public repository with the structured results from the surveys. It will be posted to Github in an easily consumed, OSS format, such as markdown documentation and csv results. This repository already exists today with source for all assignments, the project plan, and an export of the plan with progress noted. It can be found here.

The final paper will present the findings with a format largely following this document's. The presentation will be a recording of the author presenting at a research and training session in his workplace.

Work Breakdown Structure

The following tasks will take place during the life of this project:

1. Weekly Status Checks
    1. Weekly Status Check 1
    2. Weekly Status Check 2
    3. Weekly Status Check 3
    4. Weekly Status Check 4
    5. Weekly Status Check 5
2. Intermediate Milestones
    1. Milestone 1
    2. Milestone 2
3. Final Deliverables
    1. Project paper
    2. Final project deliverable
    3. Project presentation
4. Broad Developer Survey
    1. Build survey
    2. Incorporate feedback
    3. Distribute survey
    4. Compile results, analyze
    5. Document process and conclusions
5. Detailed Developer Survey
    1. Build survey
    2. Incorporate feedback
    3. Conduct survey interviews
    4. Compile results, analyze
    5. Document process and conclusions

For more details of these tasks, like the duration, constraints, and scheduled times, please refer to the project export in the repository linked above.

Calendar

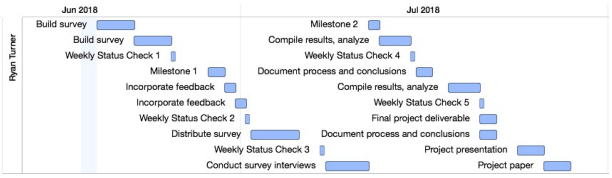

Figure 1: Snapshot of the resource timeline, taken from the repository

In the figure, a snapshot is displayed. Weekly status checks are labeled, and work completed prior to the status check is expected to be presented for that week. The same data is repeated below in a list fashion, with the tasks listed referencing their WBS code from the previous section. Tasks that are listed are expected to be complete and have an update present in that week's status check.

Week 1

- 4.1 Build survey
- 5.1 Build survey

Week 2

- 2.1 Milestone 1
- 4.2 Incorporate feedback
- 5.2 Incorporate feedback

Week 3

- 4.3 Distribute survey

Week 4

- 5.3 Conduct survey interviews
- 2.2 Milestone 2
- 5.4 Compile results, analyze

Week 5

- 5.5 Document process and conclusions
- 4.4 Compile results, analyze

Week 6

- 3.2 Final project deliverable
- 4.5 Document process and conclusions
- 3.3 Project presentation
- 3.1 Project paper

References

Bear, DJ, HB Tompson, CL Morrison, M Vickers, A Paradise, M Czarnowsky, M Soyars, and K King. 2008. "Tapping the Potential of Informal Learning: An Astd Research Study." *Alexandria, VA: American Society for Training and Development.*

Marsick, Victoria J, and Karen E Watkins. 2001. "Informal and Incidental Learning." *New Directions for Adult and Continuing Education* 2001 (89). Wiley Online Library:25–34.

Murphy-Hill, Emerson, and Gail C Murphy. 2011. "Peer Interaction Effectively, yet Infrequently, Enables Programmers to Discover New Tools." In *Proceedings of the Acm 2011 Conference on Computer Supported Cooperative Work*, 405–14. ACM.

Nandi, Arnab, and Meris Mandernach. 2016. "Hackathons as an Informal Learning Platform." In *Proceedings of the 47th Acm Technical Symposium on Computing Science Education*, 346–51. ACM.

Noe, Raymond A, Alena DM Clarke, and Howard J Klein. 2014. "Learning in the Twenty-First-Century Workplace." *Annu. Rev. Organ. Psychol. Organ. Behav.* 1 (1). Annual Reviews:245–75.

Pasetti, Alessandro. 2002. *Software Frameworks and Embedded Control Systems*. Vol. 2231. Springer Science & Business Media.

Yousef, Ahmed Mohamed Fahmy, Mohamed Amine Chatti, Ulrik Schroeder, M Wosnitza, and H Jakobs. 2014. "A Review of the State-of-the-Art." *Proceedings of CSEDU*, 9–20.