App-Directed Learning: An Exploratory Study

Jonathan Sillito
University of Calgary
Calgary, Alberta, Canada
sillito@ucalgary.ca

Andrew Begel
Microsoft Research
Redmond, Washington, USA
andrew.begel@microsoft.com

***Abstract*—Learning a new platform is a common, yet difficult task for software developers today. A range of resources, both official resources (i.e., those provided by the platform owner) and those provided by the wider developer community are available to help developers. To increase our understanding of the learning process and the resources developers use, we conducted an interview and diary study in which ten developers told us about their experience learning to develop Windows Phone applications. We report on a preliminary analysis of our data viewed through the lens of *self-directed learning*. Using this lens, we characterize the learning strategies of our subjects as *app-directed*, and describe some of the particular challenges our subjects faced due to this strategy.**

I. Introduction

Software developers face a constantly changing set of platforms and technologies that they could or may need to learn to achieve their career goals. In the past decade, the effort of maintaining competence has changed from mastering desktop software development to web and mobile platform software development. One of the challenges of being a developer today is that life-long learning skills are required to enjoy a successful career.

Current mobile phone platforms, such as Google's Android, Apple's iOS and Microsoft's Windows Phone, participate in a two-sided (or arguably multi-sided) market [10]. Platform owners become successful not only by selling their phones to mobile phone consumers, but also by making their platform attractive to developers of mobile phone applications. To support these developers, platform owners ship Software Development Kits (SDK) for the developers to use, which provide them with development tools, APIs and associated documentation.

For example, developers of Windows Phone 7 (WP7) applications use developer tools integrated with Visual Studio, APIs in C#, and the Silverlight XAML language for defining an application's user interface (among other technologies), and a phone simulator for testing their applications under many form factors. Developers have access to thousands of pages of Microsoft-provided documentation, including reference documentation, tutorials and *how to* documents, which they can use to learn about the platform and its technologies.

Developers also exploit resources created by members of the broader development community unaffiliated with the platform owner. Examples include blogs, online forums, and question and answer sites. Recent research has explored how these resources are used to answer developers' questions [11], how effectively they cover features of a platform [9], and their value as a source of examples for developers [8], [12].

Building on this previous work, we focus our current research on understanding how developers learn to build an application on a platform for their own non-work-related side project. In this paper, we report on an interview and diary study that poses three research questions in the context of WP7 development: (1) How do people develop expertise on a new development platform?, (2) What challenges do they face? and (3) How can their learning challenges be overcome through more effective support?

This first formative analysis of our data (reported here) uses concepts from the self-directed learning research literature [4] as an analytic lens. We chose to use the self-directed learning model because our subjects' learning and development activities were self-directed, i.e., not coordinated by their managers or by a university instructor. We call our subjects' approach to learning *app-direct learning*, which we characterize as an instance of self-directed learning. We found that newcomers to the platform started by asking "what do I need to learn?" and "where should I start?" They found that picking an application to build helped answer these questions. In a sense, the application *defined* the syllabus for their own personalized course of instruction. The choice of application also provided structure for their learning while they shaped their learning goals and strategies, identified learning resources and evaluated the outcomes of their learning. Our contribution in this paper is to introduce app-directed learning and discuss its associated challenges in platform development.

II. Methods

In our interview and diary study, we recruited 10 Microsoft developers (P1...P10) who did not work for the Windows Phone product team to participate. We only accepted subjects who were actively developing WP7 applications on the side (i.e., not as part of their regular work) during the study. Subjects had an average of 9 years development experience, but were relatively new to WP7 development. Our one hour long semi-structured interviews focused on our subjects' experiences learning to develop on the platform. The subjects were also asked to keep a diary of their platform learning experiences. Each diary entry answered the same question: "What is one thing you have learned recently about the WP7 platform, and how did you learn it?" In total, we received 2 to 4 diary entries per participant for a total of 24 entries.

Knowles' self-directed model identifies five stages of learning: (1) diagnosing learning needs, (2) formulating learning goals, (3) identifying resources for learning, (4) choosing and implementing a learning strategy, and (5) evaluating the learning outcome. We used a Grounded Theory approach [1] to analyze our data and noted that motivation and persistence were powerful instigators and mediators of these five stages. Specifically, we used the following themes to formulate our interview questions and to guide our analysis.

A. *How do our subjects determine what to learn?*
B. *How are learning goals formulated by our subjects?*
C. *What resources do our subjects use and how do they find them?*
D. *What learning strategies are used by our subjects and how do they choose them?*
E. *How do our subjects evaluate the quality of their learning outcomes?*
F. *What motivates our subjects to start learning, and continue to learn, the platform?*

Our preliminary answers to each of these analytic questions are reported in the follow section.

III. Findings

At the interviews, we asked our subjects to describe and characterize their level of expertise with the WP7 platform, and explain how they achieved it. The diary entries explored their day-to-day learning experiences. The results of our analysis suggest that developing deep platform expertise requires the development of multiple applications over many years. Fortunately for the subjects and platform owners, acquiring the expertise to develop a first or second application on a platform does not require 10,000 hours of practice (as Gladwell suggests is necessary to be a world expert [2]), but can be accomplished in a significantly shorter time. Some aspects of our subjects' iterative learning processes are captured here.

A. How do our subjects determine what to learn?

Newcomers to a platform need a way to determine where to start in their learning and even need to learn "what are the capabilities the platform provides" (P7). Over time, developers also need to decide whether to continue learning. Selecting an application to build helped our subjects answer these kinds of questions. Subject P1 actually decided ahead of time what topics he wanted to learn and decided to build an application that "hit all of those topics."

For most of our subjects, however, the features of the intended application determined which parts of the platform were relevant to learn. These features, thus, defined for them a personalized "syllabus" that provided a procedural pedagogical scaffold. To illustrate this point, consider Subject P3. He developed an application that plays streaming music. His personal syllabus included learning the WP7 APIs related to web services, streaming data, and playing foreground and background audio. The audio APIs were not relevant to any of our other subjects.

Our key finding here is that rather than talking about how people learn the WP7 *platform*, it is better to talk about how people learn to implement *a particular set of application features* on the WP7 platform. One significant challenge with each subject's unique syllabus is that there is no single place that each can find everything he needs, and certainly not structured in the way that would propel the subject to most effectively learn them. From the platform owner's perspective, constructing a plan for developing effective platform learning materials and tutorials is almost impossible because every developer's learning needs are in fact so different.

B. How are learning goals formulated?

Building on the concept of the application as syllabus, the learning process described by our subjects interleaves learning activities with coding activities. Most of our subjects' learning arises when they find themselves stuck during application development. Their learning goal, then, is to figure out "Why is this not working for me? How am I going to accomplish what I need to accomplish?" (P8) and learn it well enough "to get something to work?" (P2).

One could classify the different kinds of development sessions experienced by our subjects based on the fraction of learning time versus programming time. For example, Subject P1's application threw an exception that he could not explain, so he spent two days trying to figure out the reason for his exception by coding workarounds, searching online, and looking at documentation. Finally, he came across a Stack Overflow post online that suggested to make a particular class public; a 10 second code change solved the problem.

When developers "get blocked" (P6), their very specific learning goals require narrowly focused and challenging information-seeking activities to realize. Thus, our subjects' most common obstacle, *learning how to use API feature X in my application,* proves to be harder than the more general and reasonably satisfied goal of *learning about API feature X*. Due to its context-specificity, this kind of learning goal is difficult for a platform owner to support. In a self-directed learning context, without any pre-arranged or pre-identified curriculum, identifying how to provide the developer the right help at the right time requires further research.

C. What resources are used, and how are they found?

The learning process triggered when Subject P9 was blocked was primarily about searching the web. "I just search online, and usually the first few results are Stack Overflow where questions are answered." Subject P8 judged how well he knew a topic by "how many lines of code" he could write before having to "go search for help." When programming, he kept both his development environment and his web browser open, so that he could "be working and searching."

The most common resources found through web search are not the references authored by the platform owner, but are more typically, for example, blog entries and Stack Overflow posts written by non-affiliated community members. Our subjects found that much of the official documentation was limited

because it was missing the collective wisdom accumulated in the experiences of the developer community. Subject P1 told us that he often found "ways to use it you may not have thought of, when not to use it, a-ha moments." Searching also helps foster a sense of community by identifying "someone else who has encountered this problem" (P8).

Code examples found online or distributed with the SDK play an important role in the learning process of our subjects. Sometimes examples taken from the web just did not work when subjects put them in their applications. They found it was time-consuming to find and try out multiple examples, just to discover the "one that works for me" (P10). Subjects usually started an application by "following the vanilla examples" (P3), and then adapting them to the desired application context. Though it takes a lot of effort, testing, incorporating, and adapting external code examples is an essential learning activity. Subject P6 agreed, saying that "customizing [the example] to your need would help you understand it a little more" (P6).

D. What learning strategies are used, and how are they chosen?

Our subjects learned as little as they could get away with when first learning about a topic, and would only learn more when revisiting the same topic later to fix issues or deliberately attempt deeper understanding. At first, the aim appears to be to learn a topic just well enough to get the feature working, as Subject P4 told us."I tend to do a bit of a rush through when doing a first pass of implementing new functionality," even if it is not completely integrated with the rest of the application. In general, however, the goal is aimed more at "go[ing] through it gradually" (P9), accomplishing one small task after another.

Of course, just getting something to work in an application is not enough to understand the technology completely, or even to know the *best* way to do it on that platform. "I know I've done it, I don't know if I've done it right, but it kind of works I think" (P2). Improvements to both the implementation of a feature in the application and to the developer's level of understanding arise when they run into technical problems like bugs or bad performance. Subject P3 became an expert on one aspect of the API by unknowingly "shipping with odd problems." Over time, he solved those problems "by figuring out how to handle all of the exceptions that c[ould] arise."

When enough time passes (less than you would hope) after learning a topic and moving on to other associated functionality, developers found that they did not retain even elementary knowledge of the feature. "I may learn it good enough, but then I forget it because I don't use it any more" (P2). This hurts them when they need to revisit the feature in order to improve to change the implementation because they have to relearn the information. Subject P8 said that if he learns something that he can not put into his application immediately, he will need to go "back and relearn that again." His learning strategy is to learn only as much as he can apply it to his application at that moment, and iterate on further learning and coding, as needed.

E. How do learning outcomes evaluated?

As mentioned above, we asked our subjects to characterize their level of expertise in order to learn what expertise means to them. We found that while their notion of expertise varies, it is consistently and closely tied to the relationship between the amount of time and learning required to actually build something with the platform. As the need to use online references during development work shrinks (though, never to zero), they start feeling that their expertise is growing.

More specific learning episodes are measured using their progress on the application. If a subject is most concerned with getting unblocked on a programming task, when he accomplishes that task, he will feel that he has learned enough about the subject. However, if a topic needs to be revisited later because of application issues, he recalibrates his self-assessment and learns to wait until his application is released "into the wild" (P8) before believing in his own mastery.

F. What motivates our subjects to learn the platform?

Mark Guzdial, in his Computing Education Blog, says "a key part of what a teacher does is to motivate the student to learn" [3]. In self-directed learning, the student is his or her own teacher, thus a developer's motivation, persistence, and self-efficacy play significant roles in the ultimate success of his learning process.

Our study subjects were initially motivated to learn to build an application as a long-term personal investment. Subject P8 said that "the entire application marketplace dynamic ... [it] is ... a long-term trend for how software is distributed." Additionally, he felt that learning a new platform would help them maintain relevancy in their careers. "When you have a side project, you are open to new technologies that you would not even know about in your day job, and ... bring them into your day job." In the past, developers' careers were focused on learning how to become desktop platform developers, but with the rise of mobile application marketplaces, their goals have become much more specific: to release an application on the platform, and "do well enough in the marketplace so that my hobby pays for itself" (P8). Subject P2 said that learning technologies gave him "nerd cred," and that:

> "I just want to learn so that when that next idea comes that actually will make money, I have all the tools in place, and I don't have to worry about learning how to do something."

Developers interpreted limited progress in developing their application as the platform being difficult to learn, or worse, perceiving themselves to be stupid. Subject P8 said "if I'm not able to solve my problems quickly ... it's a big morale deflater." Most of our subjects valued making concrete coding progress on their application as more valuable than spending time learning, and often mis-estimated the fraction of time they needed to spend on each. Their stated goal was to become expert as quickly as possible so they could program "in the zone." Similarly, Subject P2 said:

> "to not have to rely so much on looking up how to do something gives you a lot more time and less

> context switching; if I know exactly what I need to do, I can go on a coding binge and I don't even realize that time is going by."

The developer's perception of his own progress in building his application has a strong influence on self-efficacy (the belief in one's own abilities to complete a task or solve a problem). For participant P7 making (early) progress is important to maintain his motivation to build the application:

> "I just want to work on it, not learn the language, then do a lot of trial and errors to get to a solution because that would have cost me a couple weeks in learning."

Subjects P1 and P2 both described experiences in which they each gave up learning a new platform altogether when they found it was too hard to get started. We feel that the principle behind the design of the Logo programming language, namely a "low threshold and no ceiling," [5] should govern the design of platforms. The introductory learning experiences would then be gentle enough for novices to feel comfortable learning, without restricting an expert's ability to work on increasingly sophisticated projects.

IV. Summary and Future Directions

The organization of our subjects' learning around the development of an application is something we have called *app-directed learning*. Its defining aspects are that the features of applications that developers want to build define the topics they need to learn and that many of these topics are unknown to the developers until they encounter issues with them as they learn to build their application. The conscious goal of their learning was often only to learn as much as was needed to complete their immediate development goal. They could return to the topic later to gain a deeper understanding, if and when it was needed. Subjects typically found learning resources online via web search, and were more often supported by the experiences and wisdom of members of the development community from their blog posts and answers in Q&A forums, than by the platform owner's official documentation.

A consequence of this learning approach is that our subjects did not develop general expertise in the WP7 platform, but only with particular aspects. As Subject P7 said, "I have good breadth on what the platform provides me. There are some areas which I'm good at, and some I'm not." Developers assessed their learning by the completion of a feature or application. This measurement was vital for preserving a developer's motivation to continue learning and developing.

A secondary goal of our research is to identify ways that platform owners and other development communities can support developers in learning a platform. To this end, we have identified several key challenges and open questions that further analysis and followup studies will consider. What is the best way to structure and deliver learning resources when the needs of developers vary wildly? It is difficult to produce learning materials that account for a progression of knowledge when each learner wants to take a different path through the material, or skip it entirely for the next topic. At times, our subjects inadvertently released buggy code because the associated platform topics were more complex than they realized. Simply learning enough to "get it working" was not enough to avoid all of the pitfalls related to the particular use of a feature. Is it possible to provide documentation at an appropriate level of mastery without misleading novices into believing that they know everything that is required?

Likewise, how can one write application-oriented tutorials when each developer wants to build applications with their own unique set of features? Without documentation designed exactly for each developer's application, there can be no single place where a developer could find complete information about what they need to know, structured in a way that would enable them to learn it efficiently. This is in stark contrast to a typical university course in which the instructor defines the syllabus and learning materials for the entire class, and everyone follows it together in synchrony.

Our subject found that searching for learning resources is always related to what they are trying to do with the application or struggling with at that time. Their very dissimilar contexts make information-seeking difficult. If they find enough relevant resources (e.g., sample code), they then find that adapting them to fit into their application context is equally as hard. These challenges impede a developer's progress, weakening his self-efficacy and demotivating him so much that he might quit trying entirely. Is this situation similar to retaining women and minorities in undergraduate computer science [6]? Or is it closer to motivating effective end-user software developers [7]? We hope to draw inspiration from these neighboring research areas to help find solutions for app-directed learners.

References

[1] K. Charmaz. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. SAGE Publications, 2006.

[2] M. Gladwell. *The Tipping Point: How Little Things Can Make a Big Difference*. Little, Brown, and Co., 2006.

[3] M. Guzdial. It's not about the teachers, it's about the students: In MOOCs or Classroom. Blog, January 2013. http://computinged.wordpress.com/2013/01/07/its-not-about-the-teachers-its-about-the-students-in-moocs-or-classroom/.

[4] M. S. Knowles. *Self-Directed Learning: A Guide for Learners and Teachers*. Association Press, 1975.

[5] Logo Foundation. What is Logo?, 2011. http://el.media.mit.edu/logo-foundation/logo/index.html.

[6] J. Margolis and A. Fisher. *Unlocking the Clubhouse: Women in Computing*. MIT Press, 2003.

[7] B. A. Myers and M. Burnett. End users creating effective software. In *Extended Abstracts on Human Factors in Computing Systems*, pages 1592–1593, 2004.

[8] S. Nasehi, J. Sillito, F. Maurer, and C. Burns. What makes a good code example? In *International Conference on Software Maintenance*, 2012.

[9] C. Parnin and C. Treude. Measuring API documentation on the web. In *Web 2.0 for Software Engineering*, pages 25–30, Honolulu, HI, 2011.

[10] J.-C. Rochet and J. Tirole. Platform competition in two-sided markets. *Journal of the European Economic Association*, 1(4):990–1029, 2003.

[11] C. Treude, O. Barzilay, and M.-A. Storey. How do programmers ask and answer questions on the web? In *ICSE, NIER Track*, pages 804 –807, May 2011.

[12] A. Zagalsky, O. Barzilay, and A. Yehudai. Example overflow: Using social media for code recommendation. In *Recommendation Systems for Software Engineering*, pages 38–42, June 2012.