Co-creation and Fine-Tuning of Boundary Resources in Small-Scale Platformization

Anna Sigridur Islind, Tomas Lindroth, Ulrika Lundh Snis, Carsten Sørensen

Co-Creation and Fine-tuning of Boundary Resources in Small-scale Platformization

Anna Sigridur Islind [1], Tomas Lindroth [1], Ulrika Lundh Snis [1]and Carsten Sørensen [1,2]

[1] University West, Trollhättan, Sweden
[2] London School of Economics and Political Science, London, United Kingdom

{anna-sigridur.islind@hv.se, tomas.lindroth@hv.se,
ulrika.snis@hv.se, c.sorensen@lse.ac.uk}

Abstract. Most research on platform innovation studies the phenomena from a distance due to lack of access. This paper reports from within an action research case of platform development in a small-scale context. The case is based on a regional business initiative with the goal to establish an arena for mobile commerce and stimulate local industry growth. It was conducted in collaboration between researchers and third-party developers. The article shows how the initial phases of platformization are characterized by socio-technical arrangements, co-creation of boundary resources and intimate knowledge communication. The paper contributes to platform research by acknowledging a small-scale context for platform research. It further develops distributed tuning of boundary resources into an intimate fine-tuning process that we illustrate is valid for a small-scale context.

Keywords: Platform; Platformization; knowledge communication; fine-tuning; co-creation; boundary resources

1 Introduction

While research on mobile platforms in general has been the subject of research through the past decade and a half [1, 2] the recent explosive growth of smartphone apps on Apple and Google's app stores, along with a number of other notable examples, has elevated the issue of software and smartphone app platforms within the academic discourse. The iOS and Android app stores have jointly generated a total of 3 million different apps representing the development of around 150 billion lines of code since 2008 [3]. Apple alone has paid out $25 billion to its developer community the first six and a half years [4].

However, the examples of Apple and Google are far from the only ones, as the success of a range of services partly or entirely based on a generative Internet element [5] engages in complements from independent third-party developers through exposing Application Programming Interfaces (APIs), by providing Software

Development Kits (SDKs), by relying on standard Internet technologies, and through providing quality assurance support.

In previous research, the platformization is viewed as primarily an exercise of innovation at arms length through a variety of boundary resources [6, 7]. Tilson et al. [8] reported from successful examples with significant installed bases of code and coders. The studies of mobile platforms generally do not have access to primary data sources, but tend instead to study the phenomenon indirectly. This paper, however, takes an exclusively inside-out view of the nascent phase of platform establishment with a particular focus on the initial creation of the boundary resources that will constitute a mobile platform. This process of establishing a platform will in this paper be called *platformization*. This paper reports from platformization in a small-scale context, as a contrast to the previously reported, large scale platformization studies mentioned above. In large-scale platforms the process of developing boundary resources does happen in an unprompted manner from an uncoordinated large group of third-party developers. This is not likely to happen in a small context since there are no uncoordinated large groups of developers. In a small scale context there are at the utmost small development teams or local SME firms that will engage in platform development and co-creation through various types of projects.

This article addresses two research gaps. First, the lack of accounts on how small-scale mobile platforms is established and how boundary resources are developed in such platforms. Second, there has been relatively little attention paid to how the role of knowledge communication in platform innovation influences the creation and use of boundary resources among the different actor groups. These research gaps imply that today's knowledge of mobile platforms cannot be directly applied to a small-scale context. Drawing on this, we describe the initial phases of platformization carried out in a small-scale context.

The paper is based on an action research case where a consortium of academics and practitioners developed a platform for local apps and services aimed at becoming a mobile commerce hub of a local Swedish community. The paper describes the main findings from the development process through a post-hoc analysis based on diary entries, email thread analysis, interviews and project documentation. The empirical analysis explores the effects from choices made during the platformization process, and seeks to understand these effects in terms of the communication and transfer of knowledge between the constituents.

Initial findings suggest that communication and knowledge transfer calls for a more intimate knowledge communication process where the embeddedness of knowledge into codes and software modules must be reconsidered. Therefore, this paper asks the question: How can small-scale platformization be understood in terms of the creation of boundary resources and the knowledge communication?

The following section outlines the theoretical constructs for our analysis. Section 3 briefly presents the research approach leading to the empirical results in Section 4. Section 5 discusses the findings and synthesizes our contribution. Section 6 concludes the paper.

2 Theoretical concepts

2.1 Platforms and generativity

Most of the literature on platforms and platformization is grounded within innovation management and explores the shifting interrelationships between internal production arrangements, external partners and independent complementors [2, 9, 10, 11]. Within platform research, there is a distinction between technical and socio-technical platforms [9].

A growing body of literature concerns research regarding mobile- and software-based platforms where the main focus is on platforms as two-sided markets [3, 6, 7, 12, 13]. Much insight can be gained from studying this body of work, the successful platforms with a significant digital element tends to fall into the category of two- or multi-sided markets where the platform brings parties together in open co-creation arrangements [14]. Such platform arrangements are based on the exposure of APIs, sourcing of SDKs, and relying on standard Internet technologies for the development of new services.

One of the core platform characteristics often discussed is the generativity emerging from the digital characteristics supporting independent developers engaging in producing new and interesting boundary resource recombinations [8]. Zittrain [5] claims that: "Generativity denotes a technology's overall capacity to produce unprompted change driven by large, varied, and uncoordinated audiences", and further characterizes the following five dimensions of generativity: 1. Leverage in performing some task; 2. Adaptability and built upon with ease; 3. Ease of mastery for broad audience; 4. Accessibility of tools; and 5. Transferability of results.

This paper focuses primarily on the socio-technical platform, that is, not only the technical components of the platform, the APIs or the SDKs. These are important parts but they need to be understood together with the so-called social dimensions or contexts, i.e. the information requirements and knowledge communication among different actor groups. In regard to Zittrain's [5] five dimensions, which are primarily of social character, we consider the social aspects as equal important in platform research and thus we draw attention to how boundary resources are co-created and how knowledge communication would have been facilitated.

2.2 Boundary resources and distributed tuning

The research on software- and smartphone app platforms tends to explore features relating to the open and flexible models of collaboration facilitated by distributed development based on shared boundary resources, thus providing architectural leverage. To illustrate this, Ghazawneh and Henfridsson [6] introduce the boundary resources model.

Boundary resources are "the software tools and regulations that serve as the interface for the arm's length relationship between the platform owner and the application developer." [6 p.174]. There are two forms of specific platform boundary resources [6]. First there are technical boundary resources typically consisting of

software development kits (SDK) and application programming interfaces (APIs). Secondly there are social boundary resources, typically incentives, intellectual property and platform's documentation [9, 15]. These resources enable access to central modules of the platform [16]. This opens up for the necessary complementary innovation by third-party developers [17].

Generally, the third-party developers are those who develop applications that enrich a platform [6] and involve a heterogeneous group of app-developers (individuals or groups) who are disbursed and not a part of the same community as the platform owners. The platform owners are usually refereed to as the core management of the platform in terms of operating the platform and managing the boundary resources as assets for the third-party developers to a) use in its existing form and b) further develop the boundary resources.

Ghazawneh and Henfridsson's [6] model is primarily designated for large-scale platforms development contexts. The model emphasizes the distributed use of the boundary resources developed. Such processes are guided by principles of objectified and commodifying knowledge as well as the arms-length relationships.

The notion of tuning is originally built on Pickering [18] whereas Barrett et al. [19] have extended the concept of tuning. In Eaton et al [7] the concept of tuning is further developed into a platformization approach of distributed tuning. Based an an embedded case study analysis from Apple, their iOS service system and third-party developers they offer a process model that accounts for the power-oriented dynamics of using and designing boundary resources. "Distributed tuning emerges from on-going tensions among dispersed heterogeneous actors who deal with a set of technology artifacts in a network of dialectic interrelating as shown in all five themes." [7, p.235].

To understanding the creation of sociotechnical platforms, tuning is a robust analytical lens to understand "the dynamic nature of boundary resources in service systems" [7, p.221]. They also note that there is a power dimension in the relationship between the actors participating in a service system where they are not all equal in degree of agency over both material and other actors in the service system [7]. The relationships of tuning and influence function as fundamental characteristics of how boundary resources are resisted and accommodated [7]. They further highlight the distinction between boundary resources, auxiliary boundary resources and technology resources.

We apply Ghazawneh's and Henfridsson's model [6] for a detailed analysis of the nascent phases in the development of our small-scale platform and we specifically examine the platform development in terms of the relationship between platform owners, platform and third-party developers. The ambition is to argue for the communication of knowledge and capability to third-party developers in order to cultivate and contribute to the boundary resources of the platform.

2.3 Knowledge communication in innovation

If we then consider the social boundary resources and "management of knowledge at arms length" as a core function of design capability it highlights the importance of knowledge communication and learning for platform innovation. Much knowledge in

today's society in general, but in platform innovation in particular, is based on technical knowledge [20]. This knowledge remains in the heads of the experts, or in software developer or innovation teams. The knowledge and skills are more or less built-in and embedded in a particular organization's processes and systems. Hence, the communication of knowledge mainly emphasizes the economic exchange of knowledge objectified in the boundary resources, which implies a knowledge communication through product-based economic transactions [20]. A too great focus on technical knowledge can lead to an expert-driven approach that might not be appropriate for dealing with tacit and distributed knowledge among various actor groups [21]. An opposite approach is that of knowledge communication through social processes of professionalization [20]. Herein knowledge is rather co-created and shared among different groups of individuals. It fits well when routines and standards are not yet set and when on-going interaction does not depend of a robust structure of relationships [22]. In the nascent phases of platform innovation we regard this latter approach as highly relevant.

3 Research approach

The study is based on data collected over a 2.5 year action research (AR) project building a mobile app platform devoted to developing and harnessing local mobile commerce within a medium-sized Swedish city. The project was a regional business initiative with the goal of establishing an arena for mobile commerce and stimulating local industry growth. To reach the goal the project decided to design and build a socio-technical, mobile service cross-platform that should function as a bridge that should facilitate local entrepreneurship in mobile commerce.

The socio-technical platform was designed to support and manage the entire lifecycle of mobile app design, development and go-to-market strategies. In our case the socio-technical indicates a platform consisting of actors like programmers and managers, design processes, coding frameworks, SDKs and APIs, generic code modules as well as go-to-market and knowledge transfer strategies. App development processes were set up as a co-creative effort involving both platform owners, expert developers and local third-party developers.

To gain deeper insights conceptualized we applied qualitative approaches, conducted in close co-operation with industry. It was conceptually and methodologically framed using a variety of terms such as action research [23, 24, 25] and action case research [26, 27, 28].

The action-oriented activities in this research initiative were about the balance between involvement in the change process (the problem solving) and the research process [29]. We developed and conceptualized interventions in terms of introducing an artifact as a result of multiple perspectives on design, improvement, and use. The development process was organized between a group of platform owners including researchers and representatives from the municipality and third-party developers including technical consultants, teachers and local students, as further described in section four. The design process was constructed based on the principled outlined in the extant literature on action research, for example [30, 31, 32].

The action-oriented process generated a variety of data as the researchers (later on called platform owners) sought to be present in almost all activities concerning the project. Data was to a large extent captured through participatory observations at key points in the project cycles, such as: team meetings, discussion in on-line coordination tools, and education and training workshops. Furthermore, data was continuously collected through project documentation, such as, minutes and email conversations from project meetings and from ad-hoc interaction. At the end of the project, a series of 6 semi-structured interviews were conducted, 3 with expert developers and 3 with local third-party developers – in order to capture participant reflections on their experiences. It can, therefore, be argued that this data represents an inside-out perspective on platformization as opposed to the traditional outside observer perspective based on limited primary data or decontextualized secondary data.

For this paper, the data-set was re-analyzed specifically focusing on architectural choices, boundary resources, and knowledge communication within the project. Three main actor groups were involved, which will be explained more thorough in the result section. Three of the authors of this paper are a part of the group later called platform owners and the fourth, non-participant author of this paper served through this re-analysis the role of critical outsider seeking to challenge the analytical assumptions [33].

4 Results

In this section we describe the different roles and their activities in the project. The platformization process is visualized and exemplified with a senior citizen app. Additionally, the technological decisions are described as well as the so called knowledge transfer process from the third-party consultants to the local third-party developers.

4.1 The roles

There were several different actors in the project. Here we describe their main roles. The roles of platform owners and third-party developers are both inline with the theoretical underpinning. In Figure 1 the three major roles within the project are visualized.

Platform owners	Third-party developers	
Representatives from the local municipality and university including researchers.	Expert developers from a national IT-consultancy.	Local third-party developers: Faculty teachers and system developer students.

Figure 1: The different roles

The platform owners: The platform owners were researchers at the university that were involved in a research project (see Figure 2). The researchers contributed with

user experience, service design as well as system analysis competence. Part of the platform owners group were also members from the local municipality contributed with access to the local business setting.

Expert developers: In order to ensure the best possible design and execution of the platformization effort, a leading Swedish IT consultancy firm was hired to lead the technical development effort together with the platform owners. Consequently, the expert developers were employed at a large consultancy firm. The firm employed a great number of skilled management consultants and programmers with important platform related competences.

Local third-party developers: Third-party developers used the boundary resources for utilizing the platform's capabilities in developing apps. In this local setting, the third-party developers were primarily local entrepreneurs, businesses and students at the university. The plan was that this group would subsequently use, enrich and develop the platform after the contract with the expert third party developers ended. This linked the general ambition of stimulating entrepreneurship and regional business. Herein, *the project* is used when referring to all three groups simultaneously.

4.2 The platformization process

Phase A) Initiating and exploring platformization: The platform owners evaluated a range of different platform architectures from the researchers and expert developers point of view. A key consideration during this process was to come up with alternatives on what platform architecture and development arrangement to choose and design. The following options were available: (a) Open, web-based architecture with plain vanilla versions of HTML, CSS, JavaScript etc, allowing to build what in effect would be mobile websites partly cached on the handset, (b) Entirely native apps to be published across the two main smartphone platforms, iOS and Android, written using the respective SDKs and APIs (for example Objective C, Swift, etc), (c) using more advanced high-level generator tools to produce a variant of a), for example such as: PhoneGap, Angular.js, Telerik or Backbone.

The first choice to be made was between native apps or apps based on enriched HTML web-services. In this decision the consultants' standpoint was that the project should build hybrid (HTML) apps that would be modular and that the platform should be a self-supporting stand-alone entity. When it was clear that a native solution did not fit the project goals, other alternatives were considered. Different Platform as a Service (PaaS) solutions from Microsoft, Amazon and Google were evaluated and discussed.

Option (c) with a specific arrangement of Javascript environment and cloud provision was chosen. When the decision of (c) was delivered via e-mail written by the expert third party developers, to the platform owners, the reaction was a genuine surprise among the platform owners, who felt they had not been a part of that particular decision-making process as much as they would have liked. The motivation for this choice was that with (c), the project would be able to maximize the number of

modules that were built, and the degree to which these modules would be reusable. Both aspects were regarded as central to the generativity of the platform. As one consultant expressed it: *"Considering the skills needed and learning thresholds for a student, which is an important issue, we propose a PaaS-solution. That is, the environment and systems are operated and monitored by the cloud owner and not by us."* The argumentation from the consultant focuses on the PaaS choice. The chosen javascript framework and associated SDKs were downplayed both in the first e-mail that declared their choice and in the subsequent project presentation. The platform was, after these decisions, to be built in a cloud service called Cloudbees, which is based on Jenkins [34]. This particular service platform was set up to handle hybrid apps (HTML) with complex user interfaces and a Javascript library. This particular Javascript library is called Backbone and is fundamentally a Model View Controller (i.e. MVC) that enables the code to be modular and imposes calls to the server to be entirely done through RESTful API. This choice, the choice of Jenkins, Java and Backbone were never discussed, these were just silently implemented in the platform. An overview of the innovation platform is given in figure 1 below.

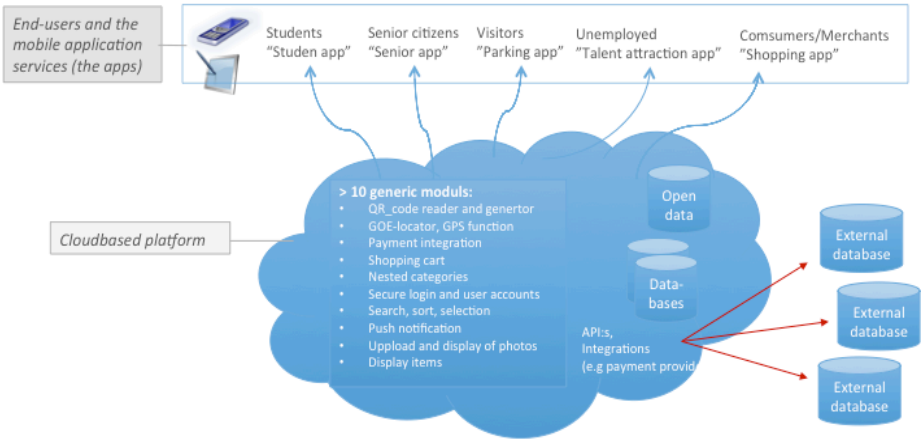

Figure 1. An overview of the cloud-based innovation platform

Phase B) Creating the apps and general modules: To get the platform working, the expert developers started developing and creating relevant modules. Initially, it was decided to develop a set of apps and services with associated code modules that later could be combined and further developed by the third-party developers. The generic modules were common features regarding secure login, GPS-features and payment options, among others. In addition to these modules the expert developers developed three apps as proof of concepts in order to show the generativity of the general modules: a) senior citizens app b) talent attraction app c) parking app. See Figure 3 for an overview of the development process. The proof of concept apps is marked with orange boarders in Figure 2.

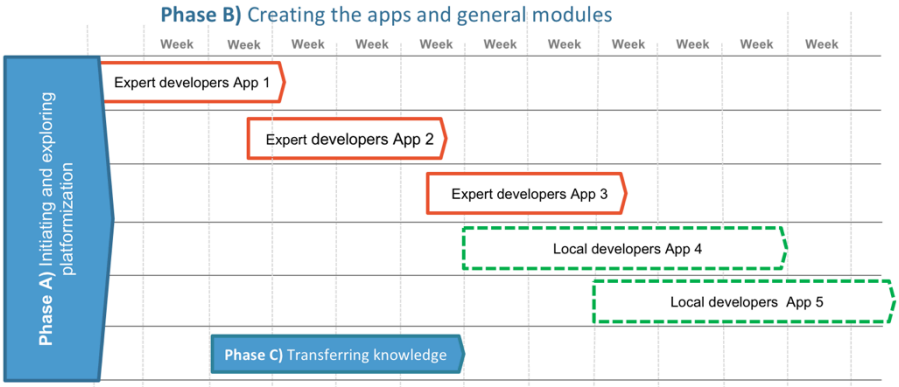

Figure 2: The platformization process: phase A, B and C

Later, in the development process it was time for the local third-party developers to develop their first apps in the established platform (Green in Figure 3). The generic modules were then used in several apps, which was in line with the general strategy to reuse the modules and code which were now ready for them to re-use as lego pieces. The plan was that this pwould be done in Scrum as well, and be a rather fast process. See Figure 4 for an overview of the platform architecture, its content and the end-user groups of the developed apps.

The experiences of developing apps in the platform were mixed. A local third party developer working within the platform felt that: *"The advantages of the platform was that many modules and solutions from previous apps were available. It was also very modular, making it easy to reuse functionality."* But on the other hand, the third party developers also thought *"Cloudbees rarely gives the user feedback control. The feedback was not kind. You receive an error code which was about 700 lines, only of exception failures from Java"*. The module-based architecture was supposed to speed up the programming process. Instead, when the expert consultants successively handed over the platform and the programing to the local third-party developers, the time to create an app increased, as seen in Figure 2.

Phase C) Transferring of knowledge: In order for the platform to thrive and reach generativity it was central to have a knowledge transfer strategy from the platform owners and expert developers to the local third party developers. The aim was to lower the entrepreneurial threshold by providing the necessary prerequisites and components to cost-effectively build and manage apps and thereby stimulate industry growth.

The knowledge transfer phase occurred in parallel with the development phases as illustrated in Figure 2. The third-party developers were able to sit at the consultancy and be present while the expert developers developed the boundary resources for the platform. To ensure the knowledge transfer for the local third party developers, there was a knowledge transfer series with lectures in how to develop new generative modules as well as in using the generative modules that were already in the platform. The knowledge transfer series was in a top-down manner where the technology was

introduced and pushed onto the local third party developers. During these series, the consultancy wanted to ensure that they had done their part in the knowledge transfer obligation and to do that, they flew in and had lectures.

The students craved more of focused training: *"The visits of {the consultancy} made the work with {the project} feel like a real job. The training was quite okay, what I remember was that the focus was on delivery deadlines during the time I was with them. So there was more focus on the project than on the training and learning."* A teacher felt that the process became so oriented towards limiting waste, that making mistakes was scaled down to the minimum, which left almost no space for the training to evolve into a real learning experience and competency.

As the local third-party developers consisted of students and teachers as a resource in developing apps for the platform, the chosen architecture implied a too steep learning curve due to the complex architecture of the platform. The students were overwhelmed with the technology framework, the rapid software development and knowledge transfer tempo.

The aspect of time, was therefore an important one, to overcome the barriers of the platform. However, due to the rapid sprints, there was no time available. The students and teachers' thoughts on the matter was the platform was too big, the platform interface was too messy and that other programming frameworks would have been a better fit. As one of the student said: *"So I think that the platform had too many templates and settings for what {the project} was doing, which made it messier."* Another student has a similar comment: *"The worst limitations that I experienced with Cloudbees was the lack of control."*

Pushing the expert developer's choice of technology onto the relatively inexperienced local third-party developers was not good, but pushing the compressed and front loaded learning cycles onto them as well with no time for reflection due to the time press and agility of the project, was worse. The lectures came to be in a monolog form where the consultancy (expert developers) fed the third-party developers information during a specific timeslot and there was no time for reflection. The third-party presence at the consultancy and the lecture series both happened before the actual handover of the programming responsibility to the local developers. When they were in the middle developing their first apps, they were on their own. At that time the knowledge transfer process had ended and the consultants had moved on to another project.

5 Discussion

As we stated in the introduction, platform innovation is a complex arrangement of *both technical and social boundary resources* that need to be critically considered [6], [9]. Diverse stakeholders are involved and different technical boundary resources become important such as application platforms, cloud services, code bases, programming knowledge, etc. Zittrain's [5] principles of generativity, where the capacity of general-purpose technology is utilized as boundary resources at arms length by groups of third-party developers, was backgrounded in this research initiative. Through the nascent phases in this platformization study, architectural

leverage was not easy to master as it turned out to have a number of consequences for the communication of knowledge and skills requirements. There were several vital decisions made by the expert developers, which possibly were motivated by their pre-existing set of technical knowledge. In general, these were not bad decisions, the technical platform is a modern, state-of-the-art development suit, but it turned out to be a poor match of requirement and skills for the local third party developers and the project in its nascent phase.

The more socially oriented resources, like knowledge about platform innovation remained in the heads of the experts, innovation teams as well as organization's processes and systems [20, 21]. Furthermore, the social relations were not coordinated towards the overall purpose of the actual project. The platform owners were interested in seeing the local context blossoming from a two-sided market perspective [14], whereas the expert developers interest was to produce as many generative modules as possible [5].

As [7] imply, at an arm's length relationship, much knowledge is embedded into the APIs and SDKs as commoditized knowledge aimed to be directly used by distant third-party developers. But in the small-scale setting, there were no critical mass of developers that could act as the other side in a two-sided market. The genuine co-creation process of boundary resources did not occur as the communication and sharing of knowledge was hard to engage in for an outsider, i.e. a local third-party developer. There was no ease of mastery for accessibility or "reuse" [5] for the third party developers.

Hence, the architectural leverage was not yet achieved, as the tension between the communication of knowledge and the commodification of knowledge was not considered carefully [3], [20], [21]. How knowledge is produced and how skills are being trained for the purpose of platform building was one such important aspect in the context of this case. This meant, in a small-scale project, aiming for generativity is less essential than getting the right skills and competencies to match the technical choices made and to the boundary resources that needs to be established. In the forefront was the primary challenge of coordinating the social relations and knowledge communication, which in retrospect is key consideration that should have been done more carefully with a gentler fade out compared to the mid project "prompt stop" in fifth week, as visualized in Figure 3, phase C.

Thus, for the nascent stages of platformization, establishing the foundation and negotiating best choices should all activities be governed by social arrangement and knowledge communication in accordance with [20] notions of professionalization and sedimentation.

5.1 Towards an intimate fine-tuning approach

This actual platform innovation initiative, with scarce resources and diverse participants, demanded a more advanced type of knowledge communication process that can bridge multiple types of actors in an open and efficient manner. Facilitating a more intimate climate for tuning the knowledge communication was in hindsight the most critical challenge. In retrospect, we can realize that the tuning process in a small-scale platform project cannot be of a distributed character with arms-length

relationship [7]. For a small-scale platform, the relationship should instead be characterized by intimacy. With intimacy we mean close collaboration over an extended period of time during the whole development process. We argue for a *fine-tuning process* where intimacy and knowledge communication in a co-creation process are key characteristics. In the following we summarize this discussion in more detail.

In a small-scale context, a platform cannot simply be brought in from the outside and placed within the local setting in terms of commodified boundary resources [6] and expected to grow; it takes intimacy and nurturing. Seeing code as value, as well as seeing knowledge as value is vital. There is a need for fine-tuning knowledge communication to accommodate the intimacy of professionalization and sedimentation. Frameworks, code and developed modules do not have an intrinsic value of their own especially if there are not skills and competency on how to enhance them. In a small scale platform context co-creation needs to be a goal, and offshoring or outsourcing may not be a viable alternative. Thus, the platform boundary resources needs to be co-created and locally legitimized in order to achieve architectural leverage and make it be cultivated in the local platform ecology. Without this co-creation and communication of knowledge, the code and modules lacks value [14], [35].

The code-base was from the expert developers point of view the main knowledge needed to be progressively transformed into new, commoditized mobile application services. The human craft of socially produced knowledge out of code bases was neglected. This stands in contrast to what Scarbrough [20] points out when stating that technological innovation needs social communicability and that the commodification of hardware and software knowledge has actually "provided a platform for increasing demand for human expertise." [20, p. 1000]. De Reuver and Bouwman [35] point out the importance of balancing dependencies and power in the development phase. At the same time, they state the challenges for "too many small parties" without any power-based governance often prevents entering the implementation phase. We experienced the strong dependency to the consultancy firm (the expert developers) as a resource to be used more intimately, as in alignment to [21]. The local third party developers involved called for more training and knowledge communication, rather than the delivery of modules. There was a clear need for professionalization aligning the remote expertise and architectural choice to the local expertise and competency. Viewing the students and teachers as boundary resources and boosting their knowledge, might have lead to a less dependency with the expert developers. The arm's length relationship was not valid. Involving the students and teachers more in the selection of architecture and taking into account the students and teachers pre-existing programming skills and the teachers background in selection of architecture, would in retrospect, have delivered a different outcome for the local setting. Lean approaches also played a role in this case. Both focus on short sprints and limiting waste is a central aspect. With longer sprints and with a more forgiving process regarding mistakes both teachers and students could have played a more central role in the development process.

Following Zittrain's [5, p.1980] notion of generativity: "Generativity denotes a technology's overall capacity to produce unprompted change driven by large, varied, and uncoordinated audiences" might need a re-consideration. In this case, as a small-

scale platform establishment, it is much better characterized in terms of the opposite context, namely, prompted change through a small, possibly homogenous, and coordinated set of socio-technical arrangements [9, 14] to establish the platform and its complements. The communication of knowledge emphasizes a fine-tuning process of intimacy in professionalization, sedimentation of skills and competency but also a different view of time as well as goals. This fine-tuning approach is fitting for a small-scale platformization, rather then the arms-length relationship that fits for a large-scale platformization discussed by Eaton et al [7]. However, we acknowledge that as a platform grows from small to large the importance of social arrangement and knowledge communication decrease in favor for commoditized knowledge into code, SDK's and API's.

6 Conclusion and further work

The aim of this paper was to shed light on the initial phases of platformization in a small-scale context. We have argued for acknowledging small-scale contexts as a part of the platform research. The models presented in prior research do not fit the purpose of the platformization processes observed in our case. The biggest challenge regarding the models is concerning the arms-length relationship between the platform owners and the third-party developers, which is the key in a large-scale platform. Through our research, we have illustrated that in a small-scale platformization, the relationship needs to be formed differently. The intimacy in the relationship between the platform owners and the third-party developers is the core. From an evolving small-scale platformization perspective, this relationship must be supported by an intimate and co-creative process of knowledge communication, herein called fine-tuning.

From this, we call for further research regarding fine-tuning of boundary resources for similar small-scale contexts. Research tends to cover platformization efforts once they have become successful and by definition are past their nascent phase. We would, however, argue that the first steps in the life of a platform are just as essential as its later race for global domination, and that perhaps we can learn from those baby-steps that result in falls rather than running.

References

1. Ciborra, Claudio U. (1996): The Platform Organization: Recombining Strategies, Structures, and Surprises. Organization science, vol. 7, no. 2, pp.103-118.
2. Gawer, Annabelle and Michael A. Cusumano (2002): Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation Harvard Business School Press.

3. Sørensen, Carsten, Mark de Reuver, and Rahul Basole (2015): Mobile Platforms and Ecosystems. Journal of Information technology, vol. 30, no. 4, pp. Forthcoming Special Issue Editorial.
4. Ranger, Steve (2015): iOS versus Android. Apple App Store versus Google Play: Here comes the next battle in the app wars. ZDNet. http://www.zdnet.com/article/iosversus-android-apple-app-store-versus-google-play-herecomes-the-next-battle-in-the-app-wars/
5. Zittrain, Jonathan (2006): The Generative Internet. Harvard Law Review, vol. 119, pp.1974-2040.
6. Ghazawneh, Ahmad and Ola Henfridsson (2013): Balancing Platform Control and External Contribution in Third-Party Development: The Boundary Resources Model. Information Systems Journal, vol. 23, no. 2, pp.173-192.
7. Eaton, Ben D., Silvia Elaluf-Calderwood, Carsten Sørensen, and Youngjin Yoo (2015): Distributed Tuning of Boundary Resources: The Case of Apple's iOS Service System. MIS Quarterly: Special Issue on Service Innovation in a Digital Age, vol. 39, no. 1, pp.217-243.
8. Tilson, David, Carsten Sørensen, and Kalle Lyytinen (2012): Change and Control Paradoxes in Mobile Infrastructure Innovation: The Android and iOS Mobile Operating Systems Cases. In 45th Hawaii International Conference on System Science (HICSS 45), Maui, HI.
9. Gawer, Annabelle, ed. (2009): Platforms, Markets and Innovation. Cheltenham: Edward Elgar.
10. Gawer, Annabelle (2014): Bridging differing perspectives on technological platforms: Toward an integrative framework. Research Policy, vol. 43, pp.1239-1249.
11. Thomas, Llewellyn, Erkko Autio, and David Gann (2014): Architectural Leverage: Putting Platforms in Context. The Academy of Management Perspectives, vol. 28, no. 2, pp.198-219.
12. Tiwana, Amrit, Benn Konsynsky, and Ashley A. Bush (2010): Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics. Information Systems Research, vol. 21, no. 4, pp.675-687.
13. Wareham, Jonathan, Paul B. Fox, and Josep Lluís Cano Giner (2014): Technology Ecosystem Governance. Organization science.
14. Boudreau, K.J. and K.R. Lakhani (2009): How to Manage Outside Innovation. MIT Sloan Management Review, vol. 50, no. 4, pp.69-75.
15. Ghazawneh, A., and Henfridsson, O. (2011). Micro-strategizing in platform ecosystems: a multiple case study.
16. Yoo, Youngjin, K. Lyytinen, R. J. Boland, and N Berente (2010): The Next Wave of Digital Innovation: Opportunities and Challenges, A Report on the Research Workshop of "Digital Challenges in Innovation Research". http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1622170
17. Gawer, A and MA Cusumano (2008): How companies become platform leaders. MIT Sloan Management Review, vol. 49, no. 2, pp.28. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=pubmed&cmd=Retrieve

&dopt=AbstractPlus&list_uids=794440609
6486535532related:bBGH5EAzQG4J
18. Pickering, A. (1993). "The Mangle of Practice: Agency and Emer- gence in the Sociology of Science," American Journal of Sociology (99:3), pp. 559-589.
19. Barrett, M., Oborn, E., Orlikowski, W. J., and Yates, J. A. 2012. "Reconfiguring Boundary Relations: Robotic Innovations in Pharmacy Work," Organization Science (23:5), pp. 1448-1466.
20. Scarbrough, Harry (1995): Blackboxes, Hostages and Prisoners. *Organization Studies*, vol. 16, no. 6, pp.9911019.
21. Sørensen, C, and Lundh-Snis, U. (2001), Innovation through knowledge codification, *Journal of Information Technology*, 16(2), 83-97.
22. Hislop, D. (2002), Mission impossible? Communicating and sharing knowledge via information technology, *Journal of Information Technology*, 17, pp. 165-177.
23. Avison, D., Lau, F., Myers, M., and Nielsen, P. A., (1999). Action research. Communications of the ACM, (42:1): 94-97.
24. Baskerville, R. (1999). Investigating Information Systems With Action Research. Communications of the Association for Information Systems Research (2): 1-32.
25. Baskerville, R., and Myers, M. D., (2004). Special issue on action research in information sys- tems: Making IS research relevant to practice-foreword. MIS Quarterly, (28:3): 329-336.
26. Braa Kristin, and Vidgen Richard (1997), Balancing interpretation and intervention in information system research: the action case approach, In Information systems and qualitative research, pp. 524-541, Springer US.
27. Braa Kristin, and Vidgen Richard (1999), Interpretation, intervention, and reduction in the organizational laboratory: a framework for in-context information system research, In journal of Accounting, Management and Information Technologies, vol 9:1, pp 25-47.
28. Braa Kristin, and Vidgen Richard (2000), From observation to intervention, In Planet Internet, pp 252-276, Studentlitteratur, Lund, Sweden.
29. McKay, J. and Marshall, P. (2001). The dual imperatives of action research. Information Technology & People, Vol. 14 Iss: 1, pp.46 - 59.
30. Baskerville R and Wood-Harper AT (1996) A critical perspective on action research as a method for information systems research. Journal of Information Technology 11, 235-246.
31. Hevner, Alan R., Salvatore T. March, Jinsoo Park, and Sudha Ram (2004): Design Science in Information Systems Research. MIS Quarterly, vol. 28, no. 1, pp.75-105.
32. Mathiassen, Lars, Mike Chiasson, and Matt Germonprez (2012): Style Composition in Action Research Publication. MIS Quarterly, vol. 36, no. 2, pp.347-363.
33. Lee, Allen S. and Geoffrey S. Hubona (2009): A Scientific Basis for Rigor in Information Systems Research. MIS Quarterly, vol. 33, no. 2, pp.237-262.
34. De Loof, Nicolas (2013): Cloud development and deployment with CloudBeesPackt Publ.

35. De Reuver, M. and H. Bouwman (2012): Governance Mechanisms for Mobile Service Innovation in Value Networks. Journal of Business Research, vol. 65, no. 3, pp.347-354.