Mobile Application Quality and Platform Stickiness under Formal vs. Self-Control — Evidence from an Experimental Study

Completed Research Paper

Tobias Goldbach
Technische Universität Darmstadt
Hochschulstraße 1,
64289 Darmstadt, Germany
goldbach@ise.tu-darmstadt.de

Viktoria Kemper
Technische Universität Darmstadt
Hochschulstraße 1,
64289 Darmstadt, Germany
kemper@ise.tu-darmstadt.de

Alexander Benlian
Technische Universität Darmstadt
Hochschulstraße 1
64289 Darmstadt, Germany
benlian@ise.tu-darmstadt.de

Abstract

Although control modes have been extensively studied in IS research, minimal research attention has been directed towards understanding how different control mechanisms operate in software-based platforms. Drawing on self-determination theory and IS control literature, we conducted a laboratory experiment with 138 participants in which we examined how well third-party developers contribute to a mobile app development platform in terms of output quality and whether they are willing to stick with this platform under formal (i.e., output and process) and informal (i.e., self) control. We demonstrate that self-control has consistently stronger effects on application quality and platform stickiness than formal control modes. We also shed light on perceived autonomy as explanatory mechanism through which the control modes’ effects are mediated. Taken together, our study highlights the theoretically important finding that self-determination among third-party developers is a stronger driving force than typical hierarchical control mechanisms. Implications for research and practice are discussed.

Keywords: Software-based platforms, Control mechanisms, Formal and self-control, Perceived autonomy, Mobile application quality, Platform stickiness
Introduction

Platform-based software ecosystems have dramatically changed the software industry in the way how software is developed, distributed and managed. In recent years, platforms such as Apple’s App Store or Facebook’s App Center have experienced a massive growth in terms of third-party developers, offered applications (apps) and overall revenues. Apple, for example, has reported six million registered app developers, more than one million active apps and revenues of 10 billion dollars from their App Store sales in 2013 (Apple 2014; Macstories 2013). Facebook recently announced that more than nine million apps are available on its social network contributed by more than two million third-party developers (Techcrunch 2011). A software platform is a software-based product or service that serves as a foundation on which outside parties can build complementary products or services and is defined as “the extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they interoperate” (Tiwana et al. 2010). The ecosystem surrounding such platforms includes numerous participants, namely the platform owner, third-party developers and platform users who are typically mutually dependent on one another.

The utility of almost any platform is increasingly shaped by the ecosystem that surrounds it. Take for example Apple’s record-breaking iOS platform that includes the iPhone, iPod, and iPad. Its value to its myriads of users comes largely from over 1 million complementary apps over which Apple has little ownership. Unlike traditional software development, platforms are designed to leverage the expertise of a diverse developer community—with ingenuity, skills, and an appreciation of user needs that platform owners might not possess (Ceccagnoli et al. 2012). The goal of platforms is thus to rapidly develop new capabilities and foster innovations that go well beyond the original product portfolio of the core. For a software-based platform to flourish, a large base of actively engaged and loyal third-party developers that produce high-quality applications is indispensable (Hartigh et al. 2006). Iansiti and Levien (2004) for example found that productivity, robustness and niche creation are crucial performance indicators of ecosystem health which are in turn strongly driven by an active developer base that not only produces high-quality applications but also remains loyal to the platform ecosystem. Constantly attracting new developers and keeping them on their platforms while ensuring high output quality are thus important goals for the platform’s long-term viability and success. Failing to constantly leverage third-party developers as a source of innovation may otherwise lead to negative consequences, as can be witnessed by the demise of several once-dominant platform protagonists such as Nokia or Blackberry that missed—among other problems—to create and manage a persistent pipeline of high-quality apps (Tiwana 2014). Given the need to maintain a prosperous platform ecosystem, the question arises of how to balance platform owners’ objectives and strategies with third-party developers’ behaviors and goals.

Traditional software development contexts are typically shaped by principal-agent relationships, where a principal delegates tasks to an agent who fulfils these tasks upon predefined contracts (Jensen and Meckling 1976). Because of incongruent goals and room for opportunistic behavior, the principal usually applies different formal control mechanisms (e.g., output or process control) to make the agents’ activities or performance more transparent (Ouchi 1979). Software-based platforms, however, are distinct from traditional software development contexts for several reasons: First, the interests and goals of participants in platform ecosystems are not necessarily incongruent (Yoffie and Kwak 2006). Shared goals may for example be to produce high-quality apps to increase the platform’s installed customer base and to generate revenues (Bergvall-Käreborn et al. 2010; Tiwana 2014). Second, the relationship between a platform owner and third-party developers is typically less hierarchical and less compulsory. One major reason is that the huge number of mostly self-employed freelancers and hobbyists makes it prohibitively costly and time-consuming for platform owners to exercise tight control on each software app project (Benlian et al. 2015). More importantly, however, is that the fact that developers can predominantly make their own decisions on platforms in terms of their activities, project requirements and products (Tiwana 2014) and therefore are more independent and enjoy a higher autonomy than in traditional software development contexts. Nevertheless, control modes are a substantial part of platform governance and different forms are implemented on actual platforms (Tiwana et al. 2010). Hence, this raises the question whether traditional formal control can be exercised in a platform context equally and with similar effects, or whether facilitating informal control leads to more promising results for software-based platforms.

In the IS domain, control modes have been extensively examined in internal as well as in outsourced and open-source IT projects to study, for example, the dynamics of control modes in IT projects (Kirsch...
2004), their differential effects on team performance (Henderson and Lee 1992), or the balancing of control models in IS development offshoring projects (Gregory et al. 2013). However, there is as yet little understanding about the differential effects of control modes in software platform contexts, particularly with regard to a comparison of formal and self-control mechanisms and their effects on both work-related (i.e., work effort and output quality) and platform-related (i.e., loyalty to or stickiness with a platform) developer outcomes, even though several scholars have called for investigating such theoretically underexplored distinctions (e.g., Tiwana et al. 2013; Wareham et al. 2014). Moreover, previous research has largely focused on the nature and conditions of appropriate control modes in different contexts rather than on the differential effects of such modes on key developer performance outcomes (Tiwana and Keil 2009). In the few cases in which the effects of control modes have been investigated (e.g., Keil et al. 2013), the explanatory mechanism of why control modes influence developer performance outcomes remained largely implicit. Given the aforementioned importance of third-party developers’ independence on software-based platforms, shedding light on the role of autonomy in explaining the relationship between platform control and developer behaviors may thus be helpful in opening this black box. The objective of this paper is therefore to address these gaps, guided by the following research questions:

(1) Which mode of platform control—formal control by the platform owner or self-control by third-party developers—leads to better outcomes in terms of development effort, app quality and platform stickiness?

(2) What is the role of third-party developers’ perceived autonomy in the relationship between platform control modes and these outcomes?

Our study offers three noteworthy contributions to both research and practice. First, it contributes to the body of knowledge in the IS control literature—which is widely advanced in organizational and project-related contexts but relatively limited for software-based platforms (e.g., Tiwana et al. 2010)—by examining the differential effects of control modes on app developers’ intentional and actual behaviors. Second, our study significantly adds to IS control research by identifying perceived autonomy as a critical explanatory mechanism through which control modes impact developers’ platform- and work-related decisions, which goes beyond previous empirical studies that treated the relationship between control modes and performance outcomes largely as a black box (e.g., Keil et al. 2013). Third, by examining which and how control modes affect app developers’ platform- and work-related decisions on software-based platforms, this study provides platform owners with valuable insights about the critical role of developers’ autonomy in affecting their performance and loyalty, which speaks in favor of embracing soft power instruments (e.g., Yoffie and Kwak 2006) on software platforms to foster cooperation.

Theoretical Background

Platform Governance and Control Modes

One of the main challenges for software platform owners is to align their own objectives and strategies with the developer’s activities and goals. This corresponds with the view of Tiwana et al. (2010) who see the central governance challenge for platform owners in finding the balance between retaining control to guarantee the platform’s integrity and relinquishing control to foster developers’ innovative capacity. Tiwana (2010) defines governance of a platform from a decision-making perspective including the partitioning of decision rights, the allocation of the ownership of the platform and its corresponding modules and, in particular, platform control. In this study, we focus on the latter governance mechanism.

According to control literature (Kirsch 1997; Ouchi 1979), control refers to a controller’s attempts by which he or she influences an individual or an individual group (the controlee) to act in accordance with the objectives of the controller. Control is exercised with specific mechanisms, such as rules, regulations and incentives, which, when adopted by the controlee, result in activities and outcomes that are in line with the controller’s objectives and goals. Control targets can be formulated and implemented in many forms with diverse approaches. As per control theory, two main categories of control mechanisms are typically distinguished, which are formal control and informal control modes. Formal control, on the one hand, is divided into output and process (or also called behavior) control. In terms of output control, controllers pre-specify output requirements and performance targets as objectives, which are then monitored, evaluated and accordingly rewarded. However, the specific actions to reach these objectives
can be arbitrarily chosen by the controlee. By contrast, under process control, no specific outcomes are pre-determined and therefore free to be chosen by the controlee. Instead, specific procedures and methodologies are pre-defined and have to be followed by the controlee (i.e., are mandatory). For both types of formal control, evaluation information is required from the controlee, particularly on intermediate or final outputs (e.g., deliverables at predetermined milestones) or about controlees’ adherence to the methods and procedures prescribed by the controller (Kirsch 1996).

On the other hand, informal control can be categorized into clan control and self-control (Ouchi 1979). In clan control, members of a group of controlees commit themselves to mutual goals and are monitoring, evaluating and correcting each other in accordance with their goals and their shared norms and values. Thus, the group members tend to adopt comparable processes and to produce outcomes with similar performance and quality. Under self-control, the individual controlees specify their own goals, evaluate themselves and decide on rewards or punishments based on their own performance. Control is thus not exercised by a group of controlees, but lies in the hands of each single controlee. Currently, a wide variety of formal and informal control modes are observed in software-based platforms (Wareham et al. 2014). Given that third-party developers are most often independent from one another and do not organize themselves in clans, self-control is the prevailing informal control mechanism on software platforms, in particular on consumer-oriented platforms (Tiwana 2014). Owing to the practical relevance of these control modes in platform settings, we will subsequently focus on process, output and self-control modes.

Previous IS research on control modes can be categorized in three major research streams, studying control modes either within organizations in internal IT projects, at the interface between organizations in IT outsourcing/offshoring relationships, or in more open settings such as open-source or software platform contexts. Prior studies that examined control modes within organizations have for example looked at the antecedents and choice of different control modes in IT project settings (Kirsch 1996; Kirsch 2004; Kirsch et al. 2002; Nidumolu and Subramani 2003), the effects of formal control on software development innovation and team performance (Cardinal 2001; Henderson and Lee 1992), and the role of clan control in IT projects (Chua et al. 2012). IS control research that shed light on inter-organizational settings has, for example, investigated the effectiveness of control modes in internal and outsourced projects (Tiwana and Keil 2009), the configuration of control portfolios in IT outsourcing and offshoring projects (Choudhury and Sabherwal 2003; Gregory et al. 2013; Rustagi et al. 2008; Srivastava and Teo 2012; Tiwana and Keil 2007) and relationship between formal and informal control in outsourced projects (Tiwana 2010). Control research on open-source projects has mainly focused on how control can be leveraged to influence developers’ motivations and behaviors (e.g., Roberts et al. 2006). More recent studies have started to examine control also in platform ecosystem settings (e.g., Goldbach and Kemper 2014). Ghazawneh and Henfridsson (2013), for example, have looked at the relationship between control and boundary resources on Apple’s iPhone platform, whereas Wareham et al. (2014) investigated the tension between control and autonomy in a business software ecosystem. While all of the mentioned studies have laid the groundwork for understanding the nature, antecedents and conditions of control modes in traditional contexts, research on the differential effects of such control modes on software platforms has remained scarce. Moreover, previous studies have treated the impacts of control modes on performance outcomes largely as a black box without explicating why such effects are at work, thus leaving fundamental questions about control modes’ effect mechanisms unanswered.

Taken together, although control mechanisms have been widely studied in IS research, there is still little understanding about how and why different control modes affect controlees’ (i.e., third-party developers’) behaviors and work outcomes in software-based platforms. Given the importance of third-party developers’ independence on software-based platforms, we now cast light on the role of autonomy for developers’ motivations to engage in, contribute to and stay with a platform, before we further elaborate on developer behaviors and performance outcomes that are crucial for software platforms.

**Developer Autonomy on Software Platforms**

As noted earlier, unlike traditional software development, the relationship between a platform owner and third-party developers is less hierarchical and less binding on software platforms, with only limited power of the platform owner to direct and instruct third-party developers. Formal individual contracts with detailed sanction mechanisms and requirements on how developers have to fulfill their work are thus less dominant. Hence, developers enjoy more freedom in determining their goals, project requirements and
activities (Tiwana and Keil 2009). In addition, because of the huge amount of third-party developers, it is prohibitively costly for platform owners to exercise tight control on developers’ activities and outputs. As such, developer’s autonomy becomes a critical lever for the overall success of a software platform.

Self-determination theory provides a psychological framework for understanding the role of autonomy for individuals’ intrinsic motivations (Deci and Ryan 2002). Autonomy refers to “the degree to which a job provides substantial freedom, independence, and discretion to the individual in determining the procedures to be used in carrying it out” (Hackman and Oldham 1976) and strongly builds on the principles of self-governance and self-regulation highlighting the importance of free will and choice when deciding on work procedures and outcomes. According to self-determination theory, it is particularly individuals’ experience of autonomy that promotes intrinsic motivation for activities that lead to outcomes such as enhanced performance, quality, persistence, and creativity. Studies drawing on self-determination theory also found that autonomy is associated with higher intrinsic motivations, more creativity, higher self-esteem, less tensions and higher job satisfaction (Deci and Ryan 2002).

Self-determination and autonomy have been shown to play a crucial role in the empowerment of individuals, especially in work settings, that positively affect their work performance. According to Fernandez and Moldogaziev (2012), empowerment can be understood as a management strategy to share authority, resources, information and rewards with employees in order to make them feel more powerful. Two effects of empowerment are commonly distinguished in research literature that lend themselves to be applied to a software platform context: First, empowerments’ direct effects refer to work-related outcomes encompassing individuals’ effort and productivity (i.e., how hard they work) and employee’s willingness to find newer and better ways to work, resulting in high-quality outcomes (i.e., how smart they work). Second, empowerments’ indirect effects are associated with job-related outcomes that refer to individuals’ more general perceptions of job satisfaction as well as trust in and loyalty towards their employer. Motivated to study third-party developer behaviors and performance outcomes on software platforms under formal and self-control modes, we adapt empowerment’s distinction of direct and indirect effects to our research context by focusing on both work-related (i.e., application quality and development effort) and platform-related outcomes (i.e., developers’ willingness to stick with a platform).

**Developer Behaviors and Performance Outcomes on Software Platforms**

**Work-Related Outcomes: Application Quality and Development Effort**

Two major work-related performance outcomes of third-party developers on software platforms are the effort they put into their work (input) and the resulting quality of the developed applications (output).

Effort is defined as the amount of energy and time a developer devotes to his development tasks relative to other developers. This conceptualization of effort is in line with previous organizational research wherein effort is characterized by the force, energy and activities which they put into their work (Churchill et al. 1985). Further, effort is also seen as part of work engagement that encompasses both individuals’ willingness to work (i.e., their involvement and dedication) and their capability to work (i.e., their effort and vigor) to accomplish a task. High development effort on a platform is, for example, manifested in how often apps are released, updated and improved. Strong efforts on third-party developers’ side are thus highly likely to strengthen a platform ecosystem in terms of its productivity, persistence and growth.

According to IS and website quality research (e.g., Benlian et al. 2011; Loiacono et al. 2007), the quality of applications refers to users’ evaluations of an application’s functionality, usefulness, ease of use and design. That is, when judging the quality of an application, users usually draw on different system indicators—related to a system’s purpose, behaviors, accessibility and aesthetics—that together form the perception of the system’s overall quality. In the context of software platforms, application quality is a particularly critical performance indicator of software platforms because the end-user market usually rewards high-quality output with strong sales and penalizes low-quality output with poor sales (Tiwana 2014). High-quality applications usually attract the lion’s share of user attention, while poor quality applications most often disappear in the long tail of a platform’s distribution of provided applications (e.g., Ghose et al. 2012). Consequently, a constant supply of high-quality applications contribute to an ecosystem’s health through promoting positive self-reinforcing network effects that foster fast innovation cycles and strong user growth, while an overabundance of poor quality applications may even jeopardize a platform’s survival.
Platform-Related Outcome: Platform Stickiness

As a crucial platform-related outcome, third-party developers’ willingness to constantly participate in and stick with a platform over time has also been shown to maintain a healthy platform ecosystem (Ceccagnoli et al. 2012; Wareham et al. 2014). Accordingly, platform stickiness is a crucial performance factor that indicates a persistent relationship between a platform and its developers. Based on Zott et al. (2000), we define platform stickiness as the ability of a platform to retain third-party developers on their platform, which is reflected in developers’ intentions to continue contributing to a platform ecosystem. When developers continue to develop and update apps for the platform and to engage themselves in the platform community, they usually contribute to the platform’s productivity, robustness and innovative capacity (Iansiti and Levien 2004). By contrast, when developers switch to a rival platform, they are likely to destabilize the abandoned platform ecosystem through the drain of knowledge and expertise. Even worse, rival platforms and their customer bases may benefit from the influx of developers and their apps, which may ultimately overturn network and evolutionary effects in favor of rival platforms.

Hypotheses Development

In this section, we develop the theoretical rationale for our research model shown in Figure 1. We first present the hypotheses related to the differential effects of control modes on third-party developers’ perceived autonomy, and on their work- and platform-related performance outcomes to address the question of whether and which control modes favorably affect developers’ perceptions, behaviors and performance outcomes. This is followed by hypotheses related to the mediating role of perceived autonomy that may explain how and why control modes may affect such developer outcomes. Finally, we hypothesize the relationship between development effort and output (i.e., app) quality.

Differential Effect Hypotheses

As mentioned before, formal control mechanisms are used by a controller to pre-specify either what the controlee should accomplish in a certain project (output control) or how the controlee should accomplish the project’s objectives (process control). In contrast, self-control builds on self-regulation mechanisms and leaves much of the control work to the controlee. Based on this distinction of how control modes empower or constrain controlees in a given working relationship on software platforms, we argue that self-control will—all else being equal—lead (1) to higher perceived autonomy, (2) stronger work-related performance outcomes (i.e., more effort and higher application quality), and (3) higher platform stickiness than formal control, which will be theoretically developed next.
According to self-determination theory, an essential feature of autonomy is individuals’ freedom to set their own goals and to work according to their own plans (Deci and Ryan 1987). In this regard, human beings’ perceptions of autonomy have been found to be related to higher intrinsic motivation, less tension and higher work satisfaction (Deci and Ryan 2002; Pearce et al. 2003). That being said, any infringements on individuals’ goals and endeavors from outside may thus reduce their autonomy and satisfaction with their work. Applied to the software platform context, we argue that compared to formal control modes, self-control modes will create a working environment that will be more favorable in supporting third-party developers’ desire to work autonomously in the pursuit of their self-interests. This is mainly because self-control modes help developers strengthen their intrinsic motivations and thus self-regulatory behaviors to contribute apps to a platform, whereas formal control modes rather act as externally imposed requirements that are likely to hamper developers from perceiving and exercising autonomy in their daily work. Based on self-determination theory, we thus hypothesize that:

H1a: All else being equal, third-party developers will perceive higher autonomy under self-control than under formal control.

Under formal control conditions, controlees have to comply with pre-specified behaviors and/or outcomes. Controlees are urged to regularly verify and align their current activities and preliminary results with the given requirements. This may result in distractions and a lower awareness for their work procedures and outcomes. Even worse, pre-specified and tightly regulated work is often likely to result in an unenthusiastic, purely compliant response that produces inertia rather than pro-active endeavors (Ouchi 1979). In this regard, formal control often creates the impression of dependent controlees who are not able to achieve the best results on their own. This may also increase the need for closer supervision leading to even more compliant behaviors. As a result, tight controls usually stifle controlees’ creativity and often interfere with their capabilities to reflect on their own activities and decisions (Amabile 1998; Orlikowski 1991), so that they are less able to find better ways in doing their jobs, resulting in less creative and more compliant work outcomes. In contrast, given the absence of pre-specified instructions on goals and procedures under self-control, controlees are able to be more reflective about their work and thus to think through their activities and goals. Moreover, they are more likely to perceive their outcomes as depending on their own efforts, initiatives and decisions, which naturally motivates them to invest more time and effort into their activities (Wang and Netemeyer 2002). Applying these arguments to the software platform context, we stipulate that in the pursuit of their vested self-interest, third-party developers will put more effort into their work practices and generate higher-quality applications under self-control than under formal control. This is largely because self-control modes create working conditions that are more in line with developers’ personal goals and working practices. Accordingly, we hypothesize that:

H1b: All else being equal, third-party developers will produce higher quality apps under self-control than under formal control.

H1c: All else being equal, third-party developers will exert more effort in developing apps under self-control than under formal control.

The main steps to exercise formal control modes are in formulating the goals, and then monitoring, evaluating and correcting controlees’ outcomes and processes according to these goals. In order to perform these steps, controllers heavily rely on evaluation information form controlees on intermediate and final outcomes (Kirsch 1996). From a control theoretical perspective, there is evidence that the exercise of such formal control modes with its evaluating and correcting activities is associated with feelings of oppression that reduce controlees’ self-confidence and sense of belonging (Ouchi 1979). By contrast, self-control modes are by definition devoid of such infringements by the controller. Instead, given the enhanced degrees of freedom through self-regulation and the lack of tedious obligations, controlees are more likely to continue working even under negative experiences (Wang and Netemeyer 2002). These arguments suggest that third-party developers will have a higher tendency to keep contributing to and stick with a software platform’s ecosystem (e.g., by submitting further apps or updates to platform) under self-control than under formal control.

H1d: All else being equal, third-party developers will have a higher intention to stick with a platform under self-control than under formal control.
Mediation Hypotheses

Given the high importance and prevalence of third-party developers’ self-interest in platform ecosystems, we suggest that developers’ perceived autonomy is one essential explanatory mechanism underlying the relationship between control modes and third-party developer behaviors.

Higher autonomy and self-guidance typically manifest themselves in a significantly higher intrinsic motivation of individuals to pursue tasks (Campbell and Pritchard 1976). Intrinsically motivated behaviors are usually considered more enjoyable, energizing and self-fulfilling than extrinsically motivated behaviors and often lead to greater persistence in increasing the quality of the behavior’s and task’s performance outcomes (Omodei and Wearing 1990). Furthermore, previous studies have shown that higher self-confidence evoked by higher autonomy positively affects the amount of effort employees invest into their work and into improving their performance outcomes (Bandura 1997). Employees are also more likely to focus longer on a task and to go the extra mile, when they feel autonomous and intrinsically motivated (Spreitzer 1995). Transferring these arguments to software platforms, we argue that developers’ perceived autonomy plays a mediating role between control modes and developers’ work-related outcomes. That is, when third-party developers work under self-control, they will first recognize and appreciate a more autonomous working environment that is characterized by self-regulation and self-directed decisions. This feeling of autonomy will then empower developers to put more effort into their activities and improve the quality of their outputs. Therefore, we propose that perceived autonomy will carry over the effects of control modes on developer effort and application quality. Put differently, we suggest that self-control, compared to formal control, will result in higher quality apps and higher developer effort because of higher perceived autonomy.

H2a: The effects of control modes on application quality are mediated by third-party developers’ perceived autonomy.

H2b: The effects of control modes on third-party developers’ effort in their work are mediated by developers’ perceived autonomy.

As mentioned above, self-determination theory predicts that human beings’ perceptions of autonomy are related to higher intrinsic motivation, more creativity, higher self-esteem, less tension and, ultimately, higher work satisfaction (Deci and Ryan 1987). Feelings of autonomy thus help people better realize their own goals, so that they are able to better identify themselves with the outcomes of their work. Any infringements upon individuals’ endeavors from outside may reduce their autonomy and satisfaction with their work. Previous empirical research has also shown that employees feel more satisfied with their working place in an organization, when they are not under permanent scrutiny, but have greater leeway in developing and reaching their personal achievements (Hackman and Oldham 1976). Since third-party developers are less urged to deliver evaluation information to the platform and feel fewer infringements on their freedom to act independently under self-control than under formal control, they will perceive higher autonomy. This higher autonomy, in turn, is likely to translate into higher platform stickiness, given that developers feel more satisfied and comfortable with the platform under such circumstances. We thus suggest that perceived autonomy will mediate the relationship between control modes and platform stickiness. In other words, self-control will lead to higher platform stickiness than formal control because of higher perceived autonomy.

H2c: The effects of control modes on third-party developers’ intention to stick with a platform are mediated by developers’ perceived autonomy.

The Relationship between Development Effort and Application Quality

Previous research studies could demonstrate a positive association between work effort and output quality (e.g., Christen et al. 2006). When employees immerse themselves in a task and invest time and effort to probe different techniques and solutions, they usually come up with more creative and higher-quality outcomes than when they would spend less effort (Johnson et al. 2003). In line with that logic, we argue that third-party developers who put more effort into their work will produce higher-quality apps. Spending more effort into app development will likely result in more iterations to test and compare
different functionalities and designs and will thus incrementally improve the application’s overall quality. We therefore propose that:

\textit{H3: Third-party developer’s effort in their work is positively associated with the quality of their apps.}

Research Methodology

\textbf{Experimental Design, Procedures and Treatments}

We conducted a laboratory experiment based on a self-programmed mobile app mock-up development platform to test our hypotheses. We used electronic mail and posts on social networking websites to recruit subjects from a public university for our experiment in exchange for a small participation fee (5€). After arriving in our lab, participants were asked to take a seat in front of a computer and were given all information to understand the three parts of the experiment. The first part covered socio-demographic questions that were presented in a pre-experimental questionnaire. In the second part, before the subjects were assigned to one of the three experimental conditions (see Table 1), each subject was exposed to the same baseline set-up of our development platform which served as a benchmark based on adaptation theory (Helson 1964). In this way, participants were able to establish a common frame of reference ensuring that the context and background of their experimental experiences were homogeneous across conditions and the disparities across different conditions were caused only by our treatments.

<table>
<thead>
<tr>
<th>Table 1. Experimental Design and Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informal control</td>
</tr>
<tr>
<td>Self-Control (n = 45)</td>
</tr>
<tr>
<td>• No guidelines</td>
</tr>
<tr>
<td>• Self-organized work</td>
</tr>
<tr>
<td>Process Control (n = 48)</td>
</tr>
<tr>
<td>• Implementation sequence for the task was pre-given</td>
</tr>
<tr>
<td>• No output criteria for pp mock-up were specified</td>
</tr>
<tr>
<td>Output Control (n = 45)</td>
</tr>
<tr>
<td>• Specific output criteria for the app mock-up were pre-given</td>
</tr>
<tr>
<td>• Implementation sequence was at the discretion of participant</td>
</tr>
</tbody>
</table>

After presenting the baseline platform (including a tutorial video), the subjects were randomly assigned to one of three experimental conditions and were instructed to step into the shoes of a platform developer to create a mobile app mock-up for a new mobile platform. More specifically, the developers’ concrete mission was to design a hotel app for that platform with free-to-choose design and functionalities. As incentive, the subjects were informed that the best app mock-ups would be entered into a raffle where they could win a tablet PC. Based on results of a pre-test with 10 subjects, in which we varied the length of development time, participants were given 20 minutes to design the hotel app. A pop-up window regularly reminded the subjects about the remaining time. While the subjects designed their app mock-ups, we collected clickstream data including the number of clicks, the number of elements added to and removed from the mock-up screens. After 20 minutes, the subjects were forwarded to a post-experimental questionnaire. In this third part of the experiment, subjects were surveyed on questions regarding platform stickiness and perceived autonomy. We also included questions on manipulation checks in this survey. The average duration of the experimental sessions was 33.53 minutes (SD = 3.37).

In our experiment, we employed control modes (i.e., output, process and self-control) as treatments in a 3 (control modes) x 1 between-subjects design (see Table 1). In order to obtain experimental groups of approximately the same size, we implemented a blocked randomization process (Kirk 2012). Participants were assigned to a group according to randomly permuted lists of the three groups. In the process control mode, and consistent with previous studies (Kirsch 1996), participants were required to follow a specific sequence of tasks to create their mobile app mock-up. However, no specific output criteria (e.g., about the final design and appearance of the mock-up) had to be fulfilled. For example, subjects first should choose the background color of the app, and then write a title, and the like. In contrast, participants in the output control condition were indicated to fulfill specific output criteria (e.g., about the design and layout of the app). In this control mode, however, the sequence of implementing these output standards was not pre-given but could be freely chosen by the participants. As an example, the apps should have a bluish
Formal vs. Self-Control on Software-based Platforms

background or navigation buttons on the bottom of the app screens. In the self-control condition, participants received no specific prescriptions but were rather instructed to organize themselves when creating the app mock-ups on the development platform. Based on the above mentioned pre-test with 10 subjects, we refined the wording of the instructions and fine-tuned the experimental procedure.

Figure 2 depicts a screenshot of the self-programmed mobile app mock-up development platform. While the experimental instructions were presented at the left of the website, the screens of the smartphone apps that had to be designed were displayed in the center. Elements for the app could be filled in via drag-and-drop by using app elements from a design kit at the right hand side. Typical design elements were background styles, text fields, buttons, a selection of icons and images as well as specific mobile elements. Inserted elements could be further modified by changing the color, style and size of the elements.

**Subjects**

145 students from a large public university in Germany participated in our controlled laboratory experiment. Seven cases had to be dropped from the sample because of the following reasons: two participants had technical problems and failed to complete the second part of the questionnaire. Five participants were dropped because of inconsistencies in their responses. Our final sample thus contained 138 participants with an average age of 22.6 (SD = 4.5) years. 83 percent were male. The majority of participants (i.e., 90 %) had a technical background studying computer science, business informatics, or business engineering. Furthermore, about 75 % (n = 106) of the participants reported that they had previous experience in software development, app development, prototyping or computer aided design.

**Measurement Characteristics**

Measures for platform stickiness were adapted from the website stickiness construct of Dahui et al. (2006) and adjusted to the platform context. Participants’ perceived autonomy was measured using a single-item derived from self-determination literature (Deci and Ryan 2002) and based on findings that the predictive validity of single items is comparable to multi-item measures (e.g., Bergkvist and Rossiter 2007; Sarstedt and Wilczynski 2009). Subjects’ effort during their app mock-up development activities were measured using subjects’ objective clickstream data collected during the experiment. We measured developer effort by averaging the number of elements added to and removed from the app mock-up during the experiment because this reflected the intensity and engagement with which the subjects developed and refined their app mock-up. We also collected control variables (i.e., age, gender, possession and usage of smartphone, subjects’ general profit-making orientation, experiences in prototyping and
computer aided design, and in software and app development) and included manipulation checks. All items of the study’s constructs and their sources are shown in Table 1 of the Appendix.

Post-Hoc Assessment: Expert Ratings of Mobile App Quality

To evaluate the quality of the 138 app mock-ups designed by the subjects, we created a web-based survey for an expert rating. In the survey, the app mock-up screens were displayed together with a 6-point rating scale (anchored at (1) = strongly disagree to (6) = strongly agree). We measured quality of the app mock-up with a single-item (“The app-mockup has a high overall quality”) adapted from Wells et al. (2011) based on the notion to capture an overall perception of app quality and not specific facets of it. Given the high number of app mock-ups that had to be evaluated, a one item assessment was an acceptable compromise to balance expert rating efforts with measurement accuracy. Previous studies comparing single-item and multiple-item measures could show that the predictive validity of single items is comparable to multi-item measures (e.g., Bergkvist and Rossiter 2007; Sarstedt and Wilczynski 2009), in particular, when the rating object (in our case the app) and the rating attribute (in our case app quality) are sufficiently concrete (e.g., when raters are supported through a visual display of the objects to be rated and receive a common definition of the rating attribute). Five experts (i.e., two IS academics with a research focus in app development and three IS practitioners with long experience in the mobile business) were invited to conduct the expert ratings from a customer’s point of view. In order to obtain a common baseline understanding of the mock-ups’ quality, the experts were shown 20 mock-ups in a random order before they could start with their ratings, and the meaning of app quality was defined and clarified based on examples. Then, the experts were asked to assess the quality of the 138 app mock-ups that were presented randomly to each rater. The inter-rater reliability among the five experts resulted in a Cronbach’s alpha $\alpha = .85$ indicating high consistency between the raters (e.g., Krippendorff 2004). We thus used these ratings as measurement items for app quality in our data analysis.

Results

Manipulation Checks and Control Variables

To confirm the random assignment of subjects to the different experimental conditions, we performed several one-way ANOVAs. There were no significant differences in gender ($F = 0.01, p > .1$), age ($F = 0.57, p > .1$), possession of a smartphone ($F = 2.77, p > .05$), usage of a smartphone ($F = 0.93, p > .1$), subjects’ general profit-making orientation ($F = 0.89, p > .05$), experience in software development ($F = 1.31, p > .1$), experience in app development ($F = 0.89, p > .05$), experience in computer-aided design prototyping ($F = 1.35, p > .1$), and experience in image processing ($F = 1.13, p > .1$) among the experimental conditions. These results indicate that participants’ characteristics and experience were not the cause of the differences in their perceptions and intentions. To check the manipulation of the different types of experimental conditions, we ran a MANOVA. We found a significant overall effect of the MANOVA ($\lambda = .72, F[4,133] = 13.24, p < .001$). Subjects in the formal control conditions indicated that they perceived significantly higher levels of (external) control than subjects in the self-control treatment. In summary, these results indicate that the treatments were successfully executed.

Comparison of Control Mechanisms’ Differential Effects

A MANOVA test was conducted to test the differential effects of the control mechanisms. MANOVA test statistics included Pillari’s trace, Wilks’ lambda, Hotelling’s trace, and Roy’s largest root. The $p$-values of these statistics were found to be significant ($p < .05$). Therefore, further ANOVAs were conducted on the four dependent variables. The subsequent ANOVA tests (see Table 2) revealed a significantly higher platform stickiness in the self-control condition than in the formal control condition ($F = 8.02, p < .05$). Similarly, perceived autonomy was rated significantly higher in the self-control condition than in the formal control condition ($F = 9.43, p < .01$). We found similar results for the expert ratings of app quality. Experts rated those app mock-ups significantly higher regarding app quality that were developed in the self-control condition compared to those created in the formal control condition ($F = 8.40, p < .01$). In contrast to the findings above, we could not find any significant differences between the experimental conditions in terms of developer effort ($F = .23, p > .05$). Based on these results, H1a, H1b and H1d could
be supported, while H1c had to be rejected. We made qualitatively similar findings when we compared the four dependent variables between the output, process and self-control conditions individually.

Table 2. ANOVA Results and Group Means (SD) for Self-Control vs. Formal Control Conditions

<table>
<thead>
<tr>
<th></th>
<th>Self-control (n = 45)</th>
<th>Formal control (n = 93)</th>
<th>ANOVA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M (SD)</td>
<td>M (SD)</td>
<td>Df</td>
</tr>
<tr>
<td>App Quality</td>
<td>3.62 (.74)</td>
<td>3.11 (1.00)</td>
<td>1</td>
</tr>
<tr>
<td>Effort</td>
<td>19.09 (5.20)</td>
<td>18.63 (5.32)</td>
<td>1</td>
</tr>
<tr>
<td>Platform stickiness</td>
<td>3.53 (.95)</td>
<td>3.05 (.90)</td>
<td>1</td>
</tr>
<tr>
<td>Perceived autonomy</td>
<td>3.91 (.93)</td>
<td>3.34 (1.06)</td>
<td>1</td>
</tr>
</tbody>
</table>

**Test of the Research Model**

For testing mediation and the fit of the theoretical model with the data, we used structural equation modeling (SEM) package Mplus 5.2 (Muthén and Muthén 2010) with maximum likelihood estimation and bootstrapping (95% bootstrap confidence interval (CI)). The advantage of SEM over regression analysis is the simultaneous consideration of several dependent variables in one unified model and the inclusion of error terms on the measurement item level.

**Measurement Model Assessment**

The psychometric properties of the measurement models were assessed by examining individual item loadings, internal consistency, convergent validity, and discriminant validity of all latent constructs. Convergent validity for latent constructs was evaluated using three criteria recommended by Fornell and Larcker (1981): (1) all measurement factor loadings must be significant and above the threshold value of .70, (2) the composite reliabilities should exceed .80, and (3) the average variance extracted (AVE) by each construct must exceed the variance due to measurement error for that construct (i.e., AVE should exceed 0.50). All loadings of the measurement items on their respective factors were all significant ($p < .05$) and above the recommended threshold values (Chin and Todd 1995). Composite reliabilities of the constructs were $\alpha = .89$ for platform stickiness, and $\alpha = .87$ for app quality. The values for AVEs were .74 for platform stickiness, and .56 for app quality. Thus, these constructs met the norms for convergent validity. In addition, for satisfactory discriminant validity, the square root of the AVE from the construct should be greater than the variance shared between a construct and other constructs in the model (Fornell and Larcker 1981). All of the square roots of the AVE also exceeded the corresponding inter-construct correlations, providing strong evidence of discriminant validity. Hence, the latent constructs in this study represent concepts that are both theoretically and empirically distinguishable.

**Structural Model Analysis**

Results from the theoretical model analysis show a nonsignificant $\chi^2$ statistic of 53.80 ($p > .05$, df = 39). Fit indices indicate an adequate fit with CFI = .98, RMSEA = .052 and SRMR = .049. Overall, these results indicate a good fit of the theoretical model with the data (e.g., Hu and Bentler 1999). Results of the model are shown in Figure 3 including path coefficients and the variance explained ($R^2$ values). The model successfully explained a considerable portion of variance in app quality ($R^2 = 0.39$), and platform stickiness ($R^2 = 0.22$), while explaining just a low amount of variance in perceived autonomy ($R^2 = 0.7$) and development effort ($R^2 = 0.04$). When leaving perceived autonomy out of the model (i.e., our mediating variable), the results were consistent with our ANOVA findings. While there were positive and significant effects of control modes on app quality and platform stickiness (both $p < .05$), we could not identify a significant effect of control modes on development effort ($\beta = -.05; p > .01$). However, in support of H3, we found a positive significant effect of effort on app quality ($\beta = .43, p < .001$).

Our mediation hypotheses were assessed based on bootstrapping procedures recommended by Preacher and Hayes (2008). In addition to a positive significant direct effect of control modes on app quality
(β = .24, p < .01) after introducing perceived autonomy into our model, we found a significant indirect effect of control modes on app quality via perceived autonomy (β = .07, p < .05). Perceived autonomy thus significantly and partially mediated the relationship between control modes and app quality, in support of H2a. Furthermore, we found a significant and fully mediated indirect effect (β = .11, p < .01) from control modes to platform stickiness via perceived autonomy (i.e., the direct effect of control modes on platform stickiness became non-significant), supporting H2c. Finally, we found no significant indirect effects from control modes to development effort (β = .05, p > .05). Thus, H2b could not be supported.

Discussion

The main objective of this study was to investigate whether and why formal control and self-control exert differential effects on developers’ work- and platform-related outcomes in platform ecosystems. Three key findings can be derived from our study. First, our findings show that, all else being equal, self-control is superior to formal control in strengthening developers’ platform stickiness, while simultaneously ensuring higher application quality. Conversely, third-party developers exposed to formal control are less likely to contribute high-quality apps to a platform, yet are more likely to churn to rival Second, our study shows that developers’ perceived autonomy serves as an important mediator between control mechanisms and crucial developer performance outcomes. More specifically, and because we found evidence for significant mediation mechanisms, perceived autonomy provides a central explanatory argument for why self-control is superior to formal control in increasing platform stickiness and application quality—it is superior because third-party developers perceive significantly higher autonomy under self-control than under formal control. Third, while we could not show that formal and self-control were differentially related to developers’ work efforts, perceived autonomy was found to significantly affect development efforts that, in turn, had a strong influence on app quality. These findings lead to interesting implications.

Implication for theory and practice

From a theoretical standpoint, this study offers a deeper understanding of the effect mechanisms relating control modes on software platforms with developers’ work- and platform-related outcomes. We show in our study that it is self-control—and not formal control—that is more conducive in platform ecosystems to both developers’ subjective perceptions (i.e., platform stickiness) and objective behavioral outcomes (i.e., application quality). This is all the more important to understand because platform ecosystems cannot exist or prosper without the contributions of highly motivated developers who are willing and able to continuously contribute high-quality applications. Previous studies of control effects in IS development have only scarcely considered whether control modes differentially relate to developer performance outcomes. However, it is important to understand whether all types of control modes affect developer
behaviors equally or in the same way, because some control modes may strongly affect developer performance whereas others may not be as salient. Based on our study findings, the emergence of platform ecosystems obviously makes some types of control more prominent than others. Our study is, to the best of our knowledge, the first to establish the heightened importance of self-control and perceived autonomy as intrinsic motivational drivers for developers’ stickiness with and high-quality work on a software platform. It thus contributes to IS control literature by studying control mechanisms in a yet underexplored context and advances platform governance literature by highlighting the shifting balance between formal and informal control mechanisms on software platforms. More broadly, our study’s findings point towards a more balanced and less hierarchical power relationship between platform owners and developers and away from traditional principal-agent relationships.

A second contribution of this study relates to the explanation of why control modes affect developers’ behaviors and performance on software platforms. While previous control studies in IS research have treated the relationship between control modes and performance outcomes largely as a black box, our study identified perceived autonomy as a crucial explanatory mechanism through which control modes impact developers’ behaviors and performance. This study thus contributes to an advanced understanding of why control modes differentially affect developers’ behaviors and their performance outcomes.

Finally, contrary to beliefs that self-control mechanisms attract masses of third-party developers who produce just poor quality applications due to the lack of direction and integrity (e.g., Hagiu and Halaburda 2010), our study’s findings show that self-control seems not only to increase stickiness compared to formal control, but also to facilitate that higher-quality applications are contributed to a platform. This finding underscores the importance of self-regulatory forces in directing developers’ work and shows that, stated colloquially, letting a thousand flowers grow can be positive for the health of platform ecosystems (Boudreau 2012). Although we constrained our focus in this study to pure forms of control modes without considering control portfolios that could eventually have more favorable impacts, examining the distinct effects of self-control vs. formal control adds to existing IS control literature by providing a more nuanced understanding of single control modes’ relative effectiveness.

Given that platforms gaining in importance as internet-based business models (Veit et al. 2014), our results have also important implications for practice. First, in platform ecosystems, it is important to understand whether all types of control modes affect developer behaviors equally. Thus, for platform owners who are trying to attract and motivate developers to contribute to their platform or to sustain their level of participation, it is imperative to understand which types of control modes are likely to generate more (or less) developer participation and output (i.e., app) quality. Our study suggests that self-control mechanisms are superior to formal control modes in generating higher platform stickiness, while simultaneously motivating developers to deliver higher-quality apps. Platform owners may therefore benefit from this study by carefully testing and monitoring the relative effectiveness of different self-control mechanisms vis-à-vis formal control modes on their software platforms. Examples for such self-control mechanisms may include open access to developers’ contribution and performance statistics or the availability of IT tools helping developers manage themselves and their work in a self-reliant way.

Second, our experimental findings underscore the need for platform owners to recognize and leverage developers’ autonomy as a critical asset, because—as we found in our study—too much infringement on developers’ plans and endeavors can undermine their willingness to stick with and contribute high-quality apps to a platform. Consequently, and consistent with previous recommendations (Claussen et al. 2013; Yoffie and Kwak 2006), we therefore suggest that in contrast to classical hard power instruments such as financial incentives or sanctions, platform owners should increasingly embrace more soft power instruments to bring developers onto a common path. Such soft power instruments may emphasize the use of intangible resources (e.g., private market intelligence or information about future plans) to build legitimacy and trust, and persuade developers to consider shared goals and a compelling platform vision.

**Limitations, Future Research and Conclusion**

As with any study, there are some limitations that provide opportunities for future research. First, based on our self-programmed mobile platform, we simulated an app development process and platform control mechanisms in a laboratory setting, which is rather artificial and thus limits the external validity of our study. Although our explicit focus of this study was on establishing a causal link between control modes and developer behaviors on a platform and thus on maximizing internal validity, future research is needed
to verify our findings in a more realistic setting and with a longer-term focus. Second, to isolate the distinct effects of formal control vs. self-control modes, we treated these two control mechanisms as dichotomous, mutually exclusive phenomena. However, we are aware that in reality, a variety of different controls are simultaneously selected from both formal and informal options as part of control portfolios. A fruitful avenue for future research may thus be to study the complementary or substitutive effects of combining different control modes on developer behaviors on software platforms and how these effects vary over time. Third, our lab experiment was conducted drawing on university students as participants. Although we believe that the students participating in our study had technical backgrounds and skills very similar to our target population, one should be cautious to generalize our results to a real-life setting. Further research is encouraged to conduct lab and field studies with a more representative mix of third-party developers. Fourth, although we controlled for several important confounding factors (e.g., development experience, profit-making orientation), we acknowledge that there may be factors that could not be completely accounted for in our rather artificial lab experiment (e.g., multi-homing scenarios or the competitive intensity between developers). Future studies should include and control for these and other factors (e.g., specific intrinsic or extrinsic motives, trust, and satisfaction) as alternative explanations for why and how well developers participate in a platform ecosystem. Finally, future work may extend our research model by including other developer-related outcome variables that are important for the success and prosperity of platform ecosystems and may be influenced by different control modes, such as developers’ creativity or innovativeness and architectural choices.

To conclude, we believe that examining control mechanisms in software-based ecosystems is a rich avenue for future research, especially given that hitherto under-researched control modes such as self-control are gaining in importance. We hope this study gives fresh impetus to researchers to refine our understanding about third-party developers’ behaviors on and contributions to software platforms.

Acknowledgements

The authors would like to thank the House of IT e.V. and Software AG, Darmstadt for supporting this research.

References


Appendix

<table>
<thead>
<tr>
<th>Constructs</th>
<th>Indicators</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceived Autonomy</td>
<td>While designing the app, I was autonomous to decide on my course of actions.</td>
<td>Adapted from Deci and Ryan (2002)</td>
</tr>
<tr>
<td>Platform stickiness</td>
<td>If available, I would expect my use of such a platform including similar instructions to continue in the future.</td>
<td>Adapted from Dahui et al. (2006)</td>
</tr>
<tr>
<td></td>
<td>If available, I would intend to continue using such a platform including similar instructions in the future.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If available, I would plan to keep using such a platform including similar instructions in the future.</td>
<td></td>
</tr>
<tr>
<td>Manipulation checks</td>
<td>The instructions on the app development platform indicated me to</td>
<td>Developed by the authors, based on Benlian and Hess (2007)</td>
</tr>
<tr>
<td></td>
<td>... follow a specific sequence of tasks/steps to create the app.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>... fulfill specific output criteria regarding the app’s design.</td>
<td></td>
</tr>
</tbody>
</table>

Note: All items were measured with five-point Likert scales and anchored with (1) strongly disagree and (5) strongly agree.