SHOULD I STAY OR SHOULD I GO? THE EFFECTS OF CONTROL MECHANISMS ON APP DEVELOPERS’ INTENTION TO STICK WITH A PLATFORM

Abstract

Although control mechanisms have been widely studied in IS research within and between organizations, there is still a lack of research on control mechanisms applied in software-based platforms, on which the complex interactions between a platform owner and a myriad of third-party developers have to be coordinated. Drawing on IS control literature and self-determination theory, our study presents the findings of a laboratory experiment with 138 participants in which we examined how formal (i.e., output and process) and informal (i.e., self) control mechanisms affect third-party developers’ intentions to stick with a mobile app development platform. We demonstrate that self-control plays a significantly more important role than formal control modes in increasing platform stickiness. We also find that the relationship between control modes and platform stickiness is fully mediated by developers’ perceived autonomy. Taken together, our study highlights the theoretically important finding that self-determination and self-regulation among third-party developers are stronger driving forces for staying with a platform than typical hierarchical control mechanisms. Implications for research and practice are discussed.

Keywords: Software-based Platforms, Control Mechanisms, Platform Stickiness, Perceived Autonomy.

1 Introduction

Software-based platforms and their corresponding ecosystems have dramatically changed the software industry in the way how software is developed, distributed and managed. In recent years, platforms like Google’s Android operating system or Mozilla’s Firefox browser have experienced a massive growth in terms of third-party developers, offered applications (apps) and overall revenues. For example, Apple has reported six million app developers on their platform in 2013, with a strong increase of 1.5 million (i.e., 33 percent) since 2012 (Macstories, 2013). A software-based platform 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, p 675). The platform ecosystem includes numerous participants, namely the platform owner, third-party developers and platform users. This highly dynamic and rapidly evolving platform context is challenging existing theories and management methods alike, and thus presents a
significant opportunity for contributions in IS (Information Systems) research, as already highlighted in several calls for research (e.g., Beimborn et al., 2011; Manner et al., 2012; Tiwana et al., 2013).

In contrast to the traditional software development context in organizations, the boundaries of software platforms are in constant flux to integrate and interact with a myriad of third-party developers. A platform owner usually strives to attain a healthy and viable surrounding ecosystem, which means durability and growth for the platform (Hartigh et al., 2006). Health of an ecosystem is in turn driven by a combination of productivity, robustness and niche creation. Factors to contribute to these ends are for instance the total output and innovation on a platform along with a high survival rate, a persistent structure and a form of variety (Iansiti and Levien, 2004). In order to achieve platform health, a platform owner is advised to integrate and keep third-party developers in the corresponding ecosystem who regularly contribute their ingenuity and skills to the platform (Ceccagnoli et al., 2012). This is all the more important because out of 40-50 mobile platforms 97% of the total mobile market is held by seven platforms (Basole and Karla, 2011). Analogous to findings from the customer management literature, however, it is easier to keep developers than to acquire new ones (Hilkert et al., 2010; Schneider and Bowen, 1995). Consequently, it is reasonable for platform owners to create a sticky environment in which developers are willing to continuously develop apps for the platform and to engage themselves in the platform community. In addition, developers who stick to a platform can keep the need for additional switching costs to a minimum. Against this backdrop, the question arises of how to balance developers’ goals and activities with platform owners’ objectives and strategies.

In the traditional software development context, the principal-agent problem (or agency theory) has extensively been investigated (Eisenhardt, 1989) and there is a robust understanding about how information asymmetries lead to agency costs (Jensen and Meckling, 1976). The agency theory typically refers to a relationship between two parties (i.e., a principal and an agent) with incongruent goals and room for opportunistic behavior. The principal delegates tasks to the agent who fulfills these tasks upon a predefined contract. The principal-agent problem stems from hidden information and hidden actions on the agents’ side which the principal usually cannot observe (Eisenhardt, 1989; Jensen and Meckling, 1976). As such, and since the principal-agent relationship is typically coordinated through a formal contract, the principal has the right to apply 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 (Benlian and Hess, 2011). Shared goals may for example be to produce high quality apps to increase the platform’s installed customer base and to generate revenues (Bergvall-Kåreborn et al., 2010). Second, the relationship between a platform owner and third-party developers is typically less binding and less hierarchical. One reason is that the huge number of developers makes it very costly for platform owners to exercise tight control on each and every software app project. More importantly, however, is the fact that developers can basically make their own decisions on platforms in terms of their goals, projects requirements and activities (Tiwana and Keil, 2007; Tiwana and Keil, 2009) and therefore are more independent and enjoy a higher autonomy than in traditional software development contexts. At the same time, informal control modes have been found to fit better with more autonomous working relationships than formal control modes (Kirsch, 1996). Hence, this raises the question whether traditional formal control can be exercised in a platform context in an equal manner and with similar effects, or whether the use of informal control (in particular self-control) may be a more promising approach for software-based platforms.

Control modes have been widely investigated in IS research within and between single organizations to study, for example, the dynamics of control modes in IT projects (Kirsch, 2004), their differential effects on teams’ performance (Henderson and Lee, 1992), or in IT outsourcing projects.
(Gregory et al., 2013). However, there is still little understanding about the differential effects of control modes in platform contexts (Manner et al., 2012; Tiwana et al., 2010), particularly with regard to a comparison of formal and self-control mechanisms and their effects on developers’ intention to stick with a platform. Furthermore, despite the aforementioned importance of third-party developers’ independence in platform settings, the role of developers’ autonomy has been neglected so far in previous research. The objective of this paper is 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 higher platform stickiness from a developer perspective?

2. What is the role of third-party developers’ perceived autonomy in the relationship between platform control and platform stickiness?

The remainder of this paper is organized as follows: Section 2 begins by laying out the theoretical background of our research including related literature on platform governance and platform control. Then, the concept of platform stickiness is introduced and the role of developers’ perceived autonomy in platform environments is discussed. Section 2 ends with the formulation of our research hypotheses. In section 3, we describe the details of a laboratory experiment we conducted to examine our hypotheses. Section 4 presents the corresponding experimental results. The paper concludes with a discussion of our findings, implications for research and practice, limitations and avenues for future research.

2 Theoretical Background and Hypotheses

2.1 Platform governance and platform control

One of the main challenges for a platform owner is to harmonize the platform owner’s objectives and strategies with the developer’s goals and activities. 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’ innovation. Tiwana et al. (2010) define governance of a platform from a decision-making perspective including the allocation of decision rights between the platform owner and third-party developers, the ownership of the platform and its corresponding modules and, in particular, platform control. The latter governance mechanism is the focus of our study.

In general, 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. Precise performance targets are usually specified, subsequently monitored as well as evaluated, and according to the controlee’s performance rewarded or punished. Control targets can be formulated and implemented in many forms with diverse approaches, which represents the gist of control theory (Kirsch, 1996; Ouchi, 1979).

According to control theory (Kirsch, 1996; Ouchi, 1979), control mechanisms are typically segmented into two main approaches, which are formal control and informal control modes. Formal control, on the one hand, is divided into output and process (or behavior\(^1\)) control. For output control, output requirements and performance targets are specified as an objective, which have to be accomplished by the controlee. However, the actions to reach these objectives are arbitrary. In contrast, for process control, specific procedures and methodologies are pre-defined and have to be followed by the

---

\(^1\) In the control literature the terms ‘behavior’ (e.g., Kirsch, 1996) and ‘process’ (e.g., Tiwana et al., 2010) are used interchangeably. In this paper we stay with the term ‘process’ according to Tiwana et al. (2010).
controlee (i.e., are mandatory). However, the exact outcome and performance targets are not determined under process control. 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 controlee’s adherence to the methods and procedures prescribed by the controller (Kirsch, 1996). On the other hand, informal control is based on self-regulating theory and can be categorized into clan control and self-control (Ouchi, 1979). In clan control, members of a group commit themselves to mutual goals. The group members 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, individuals specify their own goals, evaluate themselves and decide on rewards or punishments based on their own performance (Ouchi, 1979). Currently, a wide variety of formal and informal control modes are observed in software-based platforms (Tiwana et al., 2010). Given that platform developers are most often independent from one another and do not organize themselves in clans, self-control is the most widely used informal control mechanism. Owing to the practical relevance of these three control mechanisms in platform settings, we will subsequently focus our study on process, output and self-control modes.

Previous IS research on control modes can be categorized in two main research streams, focusing on control modes either within or between organizations. Previous research within organizations has examined the choice of control in IT projects (Kirsch, 1996, 2004; Kirsch et al., 2002; Nidumolu and Subramani, 2003; Ouchi, 1979), the effects of formal control on software development innovation and teams’ performance (Cardinal, 2001; Henderson and Lee, 1992), and the effects of clan control in IT projects (Chua et al., 2012). Control research between organizations is related to the differences of control in internal and outsourced projects (Tiwana and Keil, 2009), choice and performance of control modes in outsourcing and offshoring projects (Choudhury and Sabherwal, 2003; Gregory et al., 2013; Rustagi et al., 2008; Srivastava and Teo, 2012; Tiwana and Keil, 2007) as well as the relationship of formal and informal control in outsourced projects (Tiwana, 2010). More recent studies have examined control in third-party development settings without, however, explicitly distinguishing between the effects of different control modes. Keil et al. (2013) for example have investigated the interaction effects between controls and risks, while Ghazawneh and Henfridsson (2012) have looked at the relationship of control and boundary resources.

Taken together, although control mechanisms have been widely studied in IS research within and between single organizations, there is still little knowledge about the differential effects of control modes in software platform contexts with a particular emphasis on the comparison of formal and self-control mechanisms and their effects on developers’ perceived autonomy and intentions to stick with a platform.

2.2 Platform stickiness and the role of perceived autonomy

As mentioned earlier, in order to attain and retain a healthy platform ecosystem, it is favourable for platform owners when third-party developers also stick with their platform over time (Ceccagnoli et al., 2012). 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. Given that platform stickiness can manifest itself in different forms, it can be measured with developers’ intention to continue to use the platform in the future (Hilkert et al., 2011), and conversely, with developers’ willingness to switch to another platform (Lin et al., 2012). In case developers continue to develop apps for the platform and to engage themselves in the platform community, they will contribute to the platform’s productivity, robustness and niche creation (Iansiti and Levien, 2004). In case developers switch to another platform for app development, they
destabilize the abandoned platform ecosystem and make it less innovative. Moreover, developers have additional costs for switching to another platform and rival platforms benefit from the experience and knowledge of the switching developers.

When developers stick with a platform, they become part of the overall ecosystem. Compared to the traditional software development context, however, the relationship between a platform owner and third-party developers is less hierarchical and less binding, 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 fulfil their work are nearly absent. Therefore, developers enjoy a higher degree of freedom in terms of their goals, project requirements and activities (Tiwana and Keil, 2007; Tiwana and Keil, 2009). In addition, because of the huge amount of third-party developers, it is very costly for platform owners to exercise tight control (e.g., including monitoring and correcting developers’ activities and outputs). As such, developer’s autonomy becomes critical for the overall success of the platform.

Autonomy has long been part of the job characteristics theory. According to Hackman and Oldham (1976, p. 258) autonomy defines “the degree to which the job provides substantial freedom, independence, and discretion to the individual in scheduling the work and in determining the procedures to be used in carrying it out.” Autonomy causes different psychological states like meaningfulness of the work, responsibility for outcomes and knowledge of the results (Hackman and Oldham, 1976). In addition, self-determination theory provides a frame for the study of human motivation and personality. Self-determination theory poses a formal theory that defines intrinsic and extrinsic sources of motivation. The individual’s experience of autonomy, competence, and relatedness promote forms of motivation and engagement for activities such as enhanced performance, persistence, and creativity (Deci and Ryan, 2002). Derived from self-determination theory (Deci and Ryan, 1987), autonomy is associated with higher intrinsic motivation, more creativity, higher self-esteem, less tension and higher job satisfaction. Given the important role of developers’ autonomy in a software platform setting, a critical question is how this concept will affect developers’ stickiness, which will be addressed in our hypotheses development.

2.3 Hypotheses development

The conceptual research model for this study is shown in Figure 1. In developing our hypotheses relating control modes, namely formal control and self-control, with platform stickiness (i.e., the direct effect mechanisms) via perceived autonomy (i.e., the mediation mechanisms), we build on control theory (Ouchi, 1979) and self-determination theory (Deci and Ryan, 2002).

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 objectives (process control). In contrast, self-control builds on self-regulation mechanisms and leaves the control work to the controlee. Based on this distinction of the way how control modes infringe on the controlee’s (i.e., in our case the developer’s) working environment, we argue that formal control will—all else being equal—lead to lower platform stickiness than self-control.

In order to execute formal control modes, controllers rely heavily on monitoring, evaluating and correcting the controlee’s outcomes and processes. Therefore, a controller usually requires evaluation information from controlees 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). First of all, this leads to additional efforts and costs to the actual development task on the controlee’s side. Further, there is evidence, that the exercise of such formal control modes with its evaluating and correcting activities is associated with feelings of oppression. Prescribed and regulated
requirements and methodologies are often likely to result in an unenthusiastic, purely compliant response (Ouchi, 1979), to stifle the controlee’s creativity (Amabile, 1998) and to impact the controlee’s capability to reflect on their own activities and decisions (Orlikowski, 1991). In contrast, self-control modes are by definition devoid of a controller’s monitoring, evaluating and correcting. Thus, there is no demand for evaluation information requirements and no infringement upon developer’s activities. Consequently, given this enhanced degree of freedom and lack of oppression, it is plausible that developers’ goals and activities will be more likely to coincide with a platform using self-control mechanisms. Therefore, developers will be more likely to keep contributing to the current platform (e.g., by submitting further apps or by engaging in the developer community of the platform) and are less willing to switch to another platform. Overall, these arguments suggest that self-control will lead to higher platform stickiness (i.e., higher continuance intentions and lower willingness-to-switch) than formal control (direct effect mechanisms).

**H1:** All else being equal, third-party developers will have a higher intention to continue contributing on a platform under self-control compared to formal control.

**H2:** All else being equal, third-party developers will have a lower willingness to switch to another platform under self-control compared to formal control.

---

**Figure 1. Conceptual Research Model.**

Given the high importance and prevalence of third party developers’ independence in platform ecosystems, we suggest that one major explanatory mechanism underlying the relationship between control modes and platform stickiness is the developers’ perceived autonomy. According to self-determination theory (Deci and Ryan, 1987), human beings’ perceptions of autonomy are generally related to higher intrinsic motivation, more creativity, higher self-esteem, less tension and, ultimately, higher job satisfaction. Moreover, feelings of autonomy help people better realize their own goals so that they are able to better identify themselves with the outcomes of their work. Any infringements on individuals’ endeavours from outside may thus 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 having a greater opportunity for individually developing 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 (mediation mechanisms). In other words, self-control will lead to higher platform stickiness than formal control because of higher perceived autonomy.

H3: The influence of control modes on intention to continue is mediated by third-party developers’ perceived autonomy.

H4: The influence of control modes on willingness to switch is mediated by third-party developers’ perceived autonomy.

3 Research Methodology

3.1 Experimental design, procedures and treatments

We conducted a laboratory experiment with a self-programmed mobile app mock-up development platform to test our hypotheses. We used electronic mail and posts on social networking websites to invite a group of subjects to participate in 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 and general questions on subjects’ experience that were presented in a pre-experimental questionnaire. In the second part, participants were instructed to step into the shoes of a platform developer to create a mobile app mock-up for a new mobile app platform. More specifically, the developers’ concrete mission was to design a hotel app for that platform with free-to-choose functionalities. However, the developers were exposed to different control modes which they had to follow to be able to submit their app to the platform’s app store. As another incentive, the best mobile app mock-ups would be entered into a raffle where they could win a tablet PC. As in other experimental studies (e.g., Benlian et al., 2012), providing extra incentives is very helpful in motivating subjects to participate and in increasing their involvement. After confirming the instructions presented at the beginning of the second part of the experiment, all participants were forwarded to a tutorial video that provided tips and hints on how to use the mobile app mock-up development platform. Thereafter, the developers were randomly assigned to one of our three experimental conditions (with varying control modes) in which they had to conduct the app mock-up development task. Participant were given 20 minutes to design the hotel app. Across all conditions, a pop-up window regularly reminded the participants about the remaining time. After 20 minutes, the developers had to end their development task and were forwarded to a post-experimental questionnaire. In this third part of the experiment, participants were asked about the study’s principal variables (i.e., platform stickiness and perceived autonomy) and manipulation checks. The average duration of the experimental sessions was 33.53 minutes ($SD = 3.37$).

<table>
<thead>
<tr>
<th></th>
<th>Self-control (n = 45)</th>
<th>Process Control (P) (n = 48)</th>
<th>Output Control (O) (n = 45)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No guidelines</td>
<td>Implementation sequence for the task was pre-given</td>
<td>Specific output criteria for the app mock-up were pre-given</td>
</tr>
<tr>
<td></td>
<td>Self-organized work</td>
<td>No output criteria for the app mock-up were specified</td>
<td>Implementation sequence was at the discretion of participant</td>
</tr>
</tbody>
</table>

Table 1. Experimental Design and Instructions.
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 by the developers’ mock-up. For example, subjects first should choose the background of the app, and then write a title for the app, 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 for output criteria, the apps should have a bluish background, an icon of a specific size to describe the app’s content, and so forth. In the self-control condition, participants received no specific targets but were rather instructed to organize themselves when creating the app mock-ups on the development platform. We performed a pretest with 10 participants to check the guideline wordings and to fine-tune 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 of the website. Typical design elements were, for example, background styles, text fields, buttons, a selection of icons and images as well as specific mobile elements. Inserted elements could be further modified by for example changing the color, style and size of the different elements.

![Figure 2. Experimental Website (Mobile App Mock-up Development Platform) with Instructions (Left Panel), Working Space (Middle Panel), and Design Kit (Right Panel).](image-url)
3.2 Variables measured in the questionnaire

Platform stickiness was measured with intention to continue and willingness-to-switch. The reports of intention to continue were based on the website stickiness construct by Dahui et al. (2006) and adapted to the platform context. Willingness-to-switch was measured with one item based on Lin et al. (2012). Users’ perceived autonomy was measured using three items derived from self-determination literature (Deci and Ryan, 2002). These items considered the dimensions of being independent, not feeling influenced by someone else and being able to choose one’s own course of action. In our sample, Cronbach’s alpha reliability of the intention to continue scale was .89 for the formal control and .71 for the self-control condition. The reliability of perceived autonomy was .81 for the formal control and .71 for the self-control condition. Discriminant validity of these two constructs was confirmed by high item loadings on their corresponding construct and low cross-loadings in principal components-based factor analysis. For our analysis, composite scores were calculated for intention to continue and perceived autonomy. As control variables, we collected age, gender, possession of smartphone, usage of smartphone, experiences in prototyping and computer aided design as well as in software and app development. We also included manipulation checks in our experiment for the three control modes. Items of the scales described above can be found in Table 1 of the Appendix.

3.3 Subjects

145 students from a large public university in Germany participated in our 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 contains 138 participants with an average age of 22.6 (SD = 4.5) years. 83 percent were male. The majority of participants (i.e., around 90 percent) had a technical background studying computer science, business informatics, or business engineering. Furthermore, about 75 percent (n = 106) of the participants reported that they had previous experience in software development, app development, prototyping or computer aided design.

4 Results

4.1 Manipulation checks and control variables

To confirm the random assignment of subjects to the different experimental conditions, we performed a multivariate analysis of variance (MANOVA). The overall effect of the MANOVA was not significant (λ = .94, F [10,103] = .65, p > .1). There were also no significant differences in gender (F = 0.01, df = 1, p > .1), age (F = 0.57, df = 1, p > .1), possession of a smartphone (F = 2.77, df = 1, p > .05), usage of a smartphone (F = 0.93, df = 1, p > .1), experience in software development (F = 1.31, df = 1, p > .1), experience in app development (F = 0.17, df = 1, p > .1), experience in computer-aided design prototyping (F = 1.35, df = 1, p > .1), and experience in image processing (F = 1.13, df = 1, 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 another MANOVA. We found a significant overall effect of the MANOVA (λ = .72, F [4,133] = 13.24, p < .001). Subjects in the formal control conditions indicated that they perceived significantly higher levels of control than subjects in the self-control treatment. In summary, these results indicate that the treatments were successfully executed.
4.2 Data analysis

A one-factor ANOVA (between subjects) revealed a significantly lower intention to continue (i.e., to contribute to this platform in the future) in the formal control conditions than in the self-control condition ($F = 8.02, df = 1, p < .01$) in support of H1 (see Table 2 and Figure 3). Conversely, in the formal control conditions, subjects’ willingness to switch was significantly higher than in the self-control condition ($F = 4.55, df = 1, p < .05$) in support of H2. We made similar findings when we compared intentions to continue and willingness-to-switch between the output, process and self-control conditions (i.e., when we split formal control into process and output control). Similar to intention to continue, perceived autonomy was rated significantly higher in the self-control condition than in the formal control conditions ($F = 8.23, df = 1, p < .01$). Taken together, participants in the self-control condition reported significantly higher platform stickiness and higher perceived autonomy (see Figure 3) than participants in the formal control conditions.

We tested for the mediation effect of perceived autonomy on willingness-to-switch using the three criteria suggested for a mediation model (Baron and Kenny, 1986). First, our independent variable, control modes, had a significant effect on willingness-to-switch ($\beta = -.180, T = -2.13, p < .05$). Second, control modes also significantly influenced perceived autonomy ($\beta = .239, T = 2.87, p < .01$). Furthermore, when we regressed willingness-to-switch on both perceived autonomy and control modes, the control modes coefficient became nonsignificant ($\beta = -.101, T = -1.23, p = .222$), while the effect of perceived autonomy remained significant ($\beta = -.330, T = -4.00, p < .001$). This indicates a full mediation of the effect of control modes on willingness-to-switch by perceived autonomy. We did the same mediation test for the second dependent variable intention to continue. We found that control modes had a significant effect on intention to continue ($\beta = .236, T = 2.83, p < .01$) and also on perceived autonomy ($\beta = .239, T = 2.87, p < .01$). When we regressed intention to continue on perceived autonomy and control modes, the control modes coefficient became nonsignificant ($\beta = .153, T = 1.89, p = .061$), while the effect of perceived autonomy remained significant ($\beta = .349, T = 4.33, p < .001$). This again indicates a full mediation of the effect of control modes on intention to continue. Overall, the results from the mediation analysis showed that perceived autonomy fully mediates the relationship between control modes and platform stickiness, supporting both mediation hypotheses H3 and H4.

<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>Intention to continue(^1)</td>
<td>3.53 (.95)</td>
<td>3.05 (.90)</td>
<td>1</td>
</tr>
<tr>
<td>Perceived autonomy(^1)</td>
<td>3.47 (.88)</td>
<td>3.03 (.81)</td>
<td>1</td>
</tr>
<tr>
<td>Willingness-to-switch(^1)</td>
<td>2.44 (1.20)</td>
<td>2.89 (1.14)</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: \(^1\) = 5-point Likert scale, anchored at (1) = strongly disagree and (5) strongly agree.

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

The main objective of this study was to investigate whether and why there are differential effects of formal control and self-control on third-party developers’ stickiness with a platform. Two key findings can be derived from this study. First, our findings show that, all else being equal, self-control mechanisms lead to higher intentions to continue contributing to a platform and a lower willingness to switch to another platform compared to formal control mechanisms. Conversely, third-party developers exposed to formal control are less likely to continue contributing to the platform, yet are more likely to churn to rival platforms. Given that such stickiness is a critical source for a platform ecosystem’s productivity and robustness (Hartigh et al., 2006; Iansiti and Levien, 2004) as well as innovation and knowledge (Ceccagnoli et al., 2012), the importance of third-party developers’ self-regulatory work thus cannot be overemphasized.

Second, our study shows that developers’ perceived autonomy serves as a mediator between control mechanisms and platform stickiness. More specifically, and because we found evidence for a full mediation, perceived autonomy provides a central explanatory argument for why self-control is superior to formal control in increasing platform stickiness all else being equal—it is superior because third-party developers perceive significantly higher autonomy under self-control than under formal control. These findings lead to interesting implications for theory and practice.

5.1 Implications for theory and practice

This study makes several contributions to IS research and practice. From a theoretical standpoint, this study offers a deeper understanding of the effect mechanisms relating control modes on software platforms with third-party developers’ stickiness. We show in our study that it is self-control—and not formal control—that is more conducive to beneficial developer perceptions in platform ecosystems. This is all the more important to understand because formal control modes have been the hitherto most widely studied control mechanisms in IS research that traditionally has focused on control relationships within or between organizations. Thus, the emergence of platform ecosystems and the specificities of how platform owners and myriads of third-party developers interact with each other make some types of control (i.e., self-control) more prominent than others (i.e., formal control). 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 third-party developers’ stickiness with a 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 in software platforms.

On a related note, we find that the typical principal-agent relationship and corresponding problems studied in (hierarchical) organizations do not hold for and cannot simply be transferred to software-based platforms. Given that coordination of software development in platform ecosystems differs considerably from traditional software development contexts and that it can be viewed as more demanding and complex, platform owners should obviously rely more on open control mechanisms to keep developers in their platform ecosystem. This is mirrored in the results of our study that highlights the theoretically important finding that developers’ self-regulation and self-determination are stronger driving forces for staying with a platform than classical formal control mechanisms. More broadly, our study points to a more balanced power and control relationship between platform owners and developers than a traditional principal-agent relationship would suggest.

Our results have also important implications for practice. Implications for platform owners are: (1) autonomy is a critical asset to third-party developers and (2) too much infringement on developers’ plans and endeavors can undermine their willingness to stick with a platform. Consequently, and consistent with previous recommendations (Yoffie and Kwak, 2006), we suggest that platform owners should increasingly embrace more soft power instruments—in contrast to classical hard power instruments (such as financial incentives or sanctions)—emphasizing to persuade developers to consider shared goals and to minimize the amount of formal control modes. In such a way, developers would enjoy the perceived autonomy to be crucial for their loyalty towards a platform and minimize the need for additional switching costs.

5.2 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. Compared to a real mobile app development setting, this 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 developers’ stickiness with a platform and thus on maximizing internal validity, future research is needed (e.g., a field experiment), preferably to verify our findings in a more realistic setting.

Second, 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 of third-party developers—as for example manifested in high prior experience with software and app development—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 professional and hobbyist developers to validate our results.

Finally, although we controlled for several confounding factors to increase our study’s internal validity, we acknowledge that there may be factors that could not be completely accounted for in our rather artificial lab experiment (e.g., the financial attractiveness of rival platforms, the competitive intensity between third-party developers). Future studies should include and control for these and other factors as alternative explanations for developers’ platform stickiness.

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’ beliefs about and contributions to software platforms.
Acknowledgements

The authors would like to thank the House of IT e.V. for supporting this research. Moreover we would like to thank W. Shakir for programming the mobile app mock-up development platform and A. Benlian for the valuable support and comments on this study.

References


Appendix

<table>
<thead>
<tr>
<th>Constructs</th>
<th>Indicators</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intention to continue</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>Willingness-to-switch</td>
<td>I would switch to a different platform in the future.</td>
<td>Lin et al. (2012)</td>
</tr>
<tr>
<td>Perceived autonomy</td>
<td>While designing the app, I felt that I was influenced. (R)</td>
<td>Adapted from Deci and Ryan (2002)</td>
</tr>
<tr>
<td></td>
<td>While designing the app, I felt independent.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>While designing the app, I was autonomous to decide on my course of actions.</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 following the procedures as in 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; Reverse coded items are marked with (R).

Table 1. Summary of Main Survey Items.