Third-Party Development for Multi-Contextual Services: On the Mechanisms of Control

Daniel Rudmark, University of Borås, Allégatan 1, 501 90 Borås, Sweden and
Viktoria Institute, Hörselgången 4, 417 56 Göteborg, daniel.rudmark@hb.se

Ahmad Ghazawneh, Jönköping International Business School, Gjuterigatan 5, 553 18
Jönköping, Sweden, ahmad.ghazawneh@ihh.hj.se

Abstract

The increasing adoption of nomadic devices and the associated use of information in numerous use situations pose new challenges for the ISD practice; handling the development of such multi-contextual services covering a broader vignette of users, devices and use situations than typically associated with ISD. Recently organizations have started tapping into development resources in large networks of third-party developers. Such development is enabled through the use of software platforms where developers through boundary resources, such as APIs, may access and extend functionality in new ways. Yet, studies on how organizations are able to control this type of development remains scarce. By synthesizing theory on control and boundary objects we aim at putting a new perspective and gain a greater understanding of how organizations attempt to control such development efforts. As an illustration, we draw upon a case study of a public transportation company which through deployment of a software platform is provided access to a large body of third-party developers. We use this case to study the measures taken to control development.

Keywords: Third-party development, control, multi-contextual services, boundary objects, innovation networks, software platforms, APIs.

1 Introduction

Previous reports have put attention on how the emergence of nomadic information processing elements affects information systems development (ISD) (Lyytinen and Yoo 2002). As a result, such information systems are able to co-locate with users in new settings. While lending a promise to support new everyday use situations, these services also pose new challenges to the construction of the information systems conveying such elements (Dey et al. 2001; Henfridsson and Lindgren 2010; Tuunanen et al. 2010). Compared to systems traditionally addressed within ISD research and practice (Henfridsson and Lindgren 2005), such challenges include, we argue, an unknown set of end-user computing devices, a broad and distant user base alongside a multitude of use situations. As noted in the IS literature, categorizing IS into classes (Markus et al. 2002; Clark et al. 2007) could be justified since such systems encompass similar challenges, which are partly or fully resolvable through a set of design principles (Markus et al. 2002). In this paper we address development of *multi-contextual services* (Henfridsson and Lindgren 2005; Henfridsson and Lindgren 2010; Lindgren et al. 2008), recognizing that multi-contextual services support users as they move through space, time and social contexts and that multiple use situations may co-exist. Consequently, the complexity associated with this class of services is rendering new development challenges, requiring new modes of development such as the inclusion of third-party developers.

An increasing number of organizations are adopting new strategies to reach a wider circle of *third-party developers* to participate in the development process. In so doing, firms can foster the development of software packages for different users and niches, reap productivity benefits from the provided R&D and potentially even become a standard of a large-scale innovative conduit. From a business point of view, firms would not have to bear the heavy cost of developers and programming labour but will rather be able to profit from the benefits of instead distributing, brokering and operating through third-party developer applications. All in all, firms' development strategies will transform from being a software producer into more of a facilitator and/or a distribution channel (Evans et al. 2006; Gawer and Cusumano 2002; Ghazawneh forthcoming; Messerschmitt and Szyperski 2003; Meyer and Seliger 1998). This transformation to a platform-based software development is supported by two types of strategic resources: (1) technical boundary resources such as APIs and SDKs, and (2) social boundary resources such as incentives and intellectual property rights (Ghazawneh and Henfridsson 2010). In this paper we use public transport as an illustration of how multi-contextual services manifest themselves and how third-party developers through boundary resources are made part of a new software ecosystem addressing the challenges identified above.

Exercising *control* (motivating people to work towards organizational goals (Kirsch, 2004)) is an important dimension of ISD. For organizations employing a third-party development model we argue that control entails implementing means working towards end-user applications aligned with organizational goals. Since, the controller-controlee relationship in such loosely coupled networks of actors is quite different than typically assumed within the control literature (Tiwana et al. 2010) the question of how to control applications constructed by third-party developers remains open. The research question addressed in this paper is therefore: *how and why does third-party development for multi-contextual services require adapted control mechanisms?*

The reminder of this paper is organized as follows. Section two reviews the extant literatures on the challenges facing developers of multi-contextual services and how these challenges could be addressed using third-party development. Section three outlines our theoretical basis. While section four presents the research methodology, section five outlines how a public transport company in Sweden employed a new strategy of using third-party developers to address the challenges of multi-contextual services and the how this development were controlled. Section six presents our analysis of the case, while the last section lays the conclusion and our future research directions.

2 Related Literature

2.1 Multi-Contextual Services

As previous studies show, designers of multi-contextual services must address variety in the spatial, temporal and social dimension (Henfridsson and Lindgren 2005). Further establishing user requirements when contexts are manifold and switching poses new challenges for the user involvement literature (Henfridsson and Lindgren 2010). Finally, Lindgren et al. (2008) contributes with how multi-conceptuality affects boundary spanning information systems.

Our literature reviews suggests that to develop services in this vein, designers need to address heterogeneity in three dimensions. 1) *Device heterogeneity*: whereas ISD is typically conducted towards a known set of end-user computing capabilities (Henfridsson and Lindgren 2005) this may not be the case for this class of systems (Lyytinen and Yoo 2002; Henfridsson and Lindgren 2010). Rather, as users of personal transport information systems move from home to work they may use a number of such devices (web sites, electronic signs, GPS navigators, mobile phones). In this type of personalized assemblage of computing devices overall, tasks are completed through user interactions with portfolios of artefacts rather than singularities (Carrol 2008). 2) *User heterogeneity*: in this class of systems, organizations are addressing users long out of their organizational reach (Tuunanen 2003) and eliciting requirements for such "a distant and unknown user collective" (Henfridsson and Lindgren 2010, p. 121) is problematic. This dilemma has faced the packaged software industry for some time (Sawyer 2000) where direct end-user interactions have shown to outperform those of intermediaries, for example system analysts (Keil and Carmel 1991). Not only is the personal transport user collective partly unknown and difficult to inquire, its needs are heavily distributed and deeply interconnected with the needs of everyday life (Yoo 2010). In fact, in a theorizing effort by Lamb and Kling (2003), the notion of "task" and "user" are seen as excluding the importance of workers being embedded in social networks and hence suggest the term "actor" instead. 3) *Use situation heterogeneity*: the usage of these devices by a large and diverse user collective embedded in an entanglement of social contexts (Lamb and Kling 2003) give rise to new and unanticipated use situations (Luff and Heath 1998; Perry et al. 2001). Yet, the understanding of how use contexts influence the potential for human action is paramount (Mallat et al. 2009).

2.2 Innovation Networks and Third-Party Development

By viewing innovation as a distributed phenomenon one may conceptualize innovation as a social-technical network spanning organizational boundaries (Boland et al.2007; Chesbrough et al. 2006; Van de Ven et al. 2008; Yoo et al. 2008; Yoo et al. 2009). Such networks consist of multiple actors and technology functions that both forms, facilitates and enables innovation (Van de Ven et al. 2008; Yoo et al. 2008; 2009). In particular, the formation of innovation networks coincides with the initiation of innovation processes in order to supply them with the required resources, knowledge and capabilities (Van de Ven et al. 2008). Yoo et al. (2008; 2009) distinguish four types of innovation networks, and examine their evolvement during innovation process. These new proposed innovation networks; singular, distributed, systemic, and doubly distributed networks, are classed by two dimensions (1) the homogenous verses heterogeneous nature of knowledge resources, and (2) the distribution of coordination and control over actors and resources in the network.

Innovation is mostly driven by knowledge from diverse disciplines and social worlds, which will be later integrated within the core platform where the proprietor firm exercises critical architectural control over the key elements (Yoo et al. 2008). Our studied case applied to the 'doubly distributed' form of innovation network. In this form of innovation network, the control of the process, structure and outcomes is distributed throughout the network, at the same time, the knowledge resources are highly heterogeneous and members operate under distributed control (Yoo et al. 2009). Such

innovation networks establish fruitful environments for third-party development on top of software platforms. We define a software platform *as a set of technology layers of interrelated interoperability specifications that form a common resource base from which derivative software can be developed and integrated. The resource base of such specifications enables a shift of design capability to external actors, who typically make their development of applications compatible with the platform*.

The software platform, the community of third-party developers, and the extensions and contributions they provide in the form of applications and services constitute a software platform ecosystem (Bosch and Bosch-Sijtsema 2010; Messerschmitt and Szyperski 2003). All interdependencies within such an ecosystem are underpinned by its software platform and operate through the exchange of information, resources, and artifacts (Jansen et al. 2009a; 2009b). The taxonomy of a software platform consists of two main dimensions. First, *the platform technology* which determines whether the software platform is desktop, web or mobile based. Second, *the platform layers* which defines whether developers build applications on top of an operating system (OS), through application extension or by developing extensions through a domain-specific language (DSL) (Bosch 2009; Jansen et al. 2009b). The value of software platforms mainly lies in the applications developed by third-party developers (Taudes et al. 2000). Those platforms and the developed applications have been studied by researchers who see them as drivers of innovation in multiple industries (Evans 2006; Messerschmitt and Szyperski 2003), turning firms into software leaders (Gawer and Cusumano 2002), and even initiate battles between competitors (Ferguson 2005; Gawer and Cusumano 2008). Examples of firms successfully adopted the concept and put it into work as a core strategy include Microsoft, Palm (Gawer and Cusumano 2002) Google, Amazon, Apple, and eBay (Tapscott and Williams 2006).

3 Theoretical Basis

3.1 Control Mechanisms

While third-party development using software platforms offer great promise for software development, the issue on how to steer development efforts towards organizational goals remains an open topic. Controlling Information Systems Development (ISD) has long been a central research topic which recognizes the importance of the execution of control mechanisms to ensure ISD project success. Researchers in the IS discipline have conceptualized control in various ways (Henderson and Lee 1992; Kling and Iacono 1984) and the growing body of literature also examined this subject from different practical angles (Kirsch 1997; Kirsch 2004; Choudhury et al. 2003; Levina, 2005).

ISD control research (Kirsch 1996; Kirsch 1997) exhibits four central control mechanisms. *Behaviour control* occurs when those controlled are rewarded when adhering to predefined procedures (e.g. an ISD methodology). *Outcome control* is exercised when articulated goals are met (e.g. a project deliverable according to plan). While these two forms of control are considered formal, other, less explicit (informal) control mechanisms exist. *Clan control* is exercised through a socialization process where new members gradually get "accepted" into the clan by acting on clan values (e.g. by using unarticulated development practices). Finally, by using *self control* individual members may set goals, regardless of any outside control mechanisms, and motivated by intrinsic rewards (e.g. by implementing a personal quality assurance system to ensure high-quality deliverables). In this paper, control is viewed as attempts to "motivate individuals to behave in a manner consistent with organizational objectives" (Kirsch 2004, p.374). Taking this broad view allows a study of exercising control in settings that are non-routine, complex and dynamic (Crisp 2002; Kirsch 1996), and that may not correspond exactly to theoretical conceptualizations (Mähring 2002). Traditionally control mechanisms has been exercised through phenomena such as IS development team control (Henderson and Lee 1992), intra-organizational politics (Kling and Iacono 1984), consultant-client relationships (Levina 2005) or even taken-for-granted standard ISD methodology control instruments such as user

requirements and testing (Kirsch 1996). However, many of these traditional control mechanisms within ISD are unavailable when drawing upon the types of large networks of third-party developers described in this paper. Hence, we see a need to theoretically assess the interface through which control may be exercised in such case. Considering the artefact-oriented coordination taking place in a software ecosystem, we next turn to theories on boundary objects (Carlile 2002; Star and Griesemer 1989) as an explorative instrument.

3.2 Boundary Objects Theory

A *boundary object* is defined as an artifact or a concept that is plastic enough to cut across multiple social worlds, and has enough structure to support several parties and their employed activities within separate social worlds (Bergman et al. 2007; Star and Griesemer 1989). In this paper we regard ISD as a distributed design practice incorporating a variety of knowledge specializations. The boundary object perspective has proven useful when studying the transfer of design knowledge across multiple social worlds and knowledge domains (Bergman et al. 2007) where different, but co-operating actors are tied together (Star and Griesemer 1989). Indeed, the transfer of such design knowledge is essential in reaching organizational goals using third-party development.

There are four types of boundary objects that have the ability to mediate different actors (Carlile 2002; Star and Griesemer 1989). First, *repositories* that supply a common reference point of data that are built to deal with problems of heterogeneity caused by differences in unit of analysis, they provide shared definitions and values for solving problems. Second, *ideal type*s which are simple or complex objects that do not accurately describe the details of any one subject, they can be observed and then used across different functional settings. Third, *coincident boundaries* represent common objects sharing mutual borders and dependences that exist between different actors but with different internal contents. These common objects enable different actors to use their different perspectives relatively autonomously and to share a common referent. Fourth, *standardized forms* devised as pre-defined methods of common communication across dispersed actors.

Despite these generic properties of boundary objects, the capacity to transfer knowledge is highly situated – artifacts working successfully as boundary objects in one setting can become a knowledge transfer roadblock in another. Carlile (2002) identified three characteristics of effective artifacts for knowledge transfer at a given boundary: (1) they establish a shared syntax or language for individuals to represent their knowledge – making knowledge transfer possible. (2) They provide a concrete means for actors on both sides of the boundary to share the meaning of the transferred knowledge, and (3) they facilitate a pragmatic process in which actors may need to renegotiate the meaning and thus redefine their internal knowledge specializations shaped by path-dependent trajectories. It has been argued that this is a iterative process where a renegotiation often is followed by a new shared syntax (Carlile 2004).

The boundary objects theory provides a basis for understanding how and why platform boundary resources such as SDKs and APIs enable coordination of activities across multiple knowledge resources where heterogeneous but co-operating actors are tied together (Bergman et al. 2007; Briers and Chua 2001; Star and Griesemer 1989). One of the most important type of boundary resources in third-party development is Application Programming Interfaces (APIs). An API is particular set of rules and specifications that an API consumer (third-party developer) can access and make use of the services and resources offered by an API producer (platform owner) that implements and publishes that API (de Souza et al. 2004). APIs have been seen as both organizational boundaries and contracts between the platform owner and the third-party developers. They create a level of trust between all parties engaged in a software platform ecosystem and enable third-party developers to develop applications based on the functionalities provided by the APIs.

4 Research Method

4.1 Research Context and Case Selection

As a result of an increasingly ubiquitous computing environment organizations are turning to explore new ISD models. In 2009 a public transportation company in Sweden opened a developer zone; a web site through which developers got access to APIs enabling them to build new type of end-user applications. This move was thought to primarily increase end-user services coverage and application variability, supporting the organizational journey towards increased usage of public transport.

We find this single case (Yin 2009) a relevant study object for two major reasons. First, personal transportation provides an excellent venue for research on multi-contextual services. The heterogeneous user base travelling to and from daily activities (such as work and education) typically moves over a variety of devices and co-existing use situations. Second, since the public transportation company (facing the challenges of multi-contextuality) employed a new distributed, platform-based system development model they also needed to deal with the control dimension of this new situation. Since we believe that control in this vein is theoretically underexplored we consider this case to carry significant theoretical relevance.

4.2 Data Collection and Analysis

The data presented in this paper has been collected over 14 months (2009-2010) and has been analysed in parallel (Miles and Huberman 1994). The data consisted of working group meetings, interviews, internal reports and publically available reports. All analysed meetings and interviews have been audio taped and transcribed. The data were collected in three phases. First, in order to better understand the system development challenges the initial data collected was interviews of explorative nature alongside working group meetings (to understand the current challenges). Second, the next set of interviews with these actors focused on previous projects and the unfolding of events. In this step internal and publically available reports mentioned in previous data were also collected. Thirdly, interviews with outside developers were conducted and data from the developers' zone were collected. All in all, this resulted in an analysis of a few hundred pages of written material.

Data source	Description	Number
Interviews Traffic Authorities	Interviews with representatives of traffic authorities. This includes both people currently working in these organizations, previous employees and consultants.	9
Working group meetings	These meeting typically lasted for 1,5-3 hours and concerned to a large extent the challenges of multi-contextuality and third-party development control.	4
Interviews Developers	Interviews with developers having used the labs web site of the public transportation company.	7
Web based material	Data from the Developer Zone. Of special interest was a discussion forum where developers asked questions to other developers and to the company.	n/a
Reports	Documents describing the anticipated transport situation in the region, what actions deemed necessary to achieve	5

	sustainability goals and a pre-study for exposure of API's.	

Table 1. Data Sources

All transcribed audio material and reports have been analysed using Atlas.ti, a data analysis software package. Utterances and report paragraphs addressing the scope of research have been identified and coded accordingly. In phase 1 (as described above) data was analysed inductively line-by-line (influenced by the methods presented by Strauss and Corbin (1990)). Using such an inductive approach to data analysis is motivated by the lack of control theory within this type of loosely coupled third-party development. Thus, in this stage it was important to understand the current state of affairs in the local practice without forcing too much researcher preconceptions onto the data while maintaining scientific integrity (Eisenhardt 1989). The relationships between codes were established and a detailed snapshot of the current struggles emerged (this corresponds to the three challenges of heterogeneity as explained above and the control challenges using third-party development). Analysis in phase two focused on the chronological unfolding of events (Langley 1999) and understanding how the organization dealt with control issues (Kirsch 1996) while the third phase investigated, through the eyes of third-party developers, how the APIs served as mediators of design capabilities (Bergman et al 2007; Carlile 2002) supporting the transportation company's goals.

5 Results

5.1 Third-party development as a response to multi-contextual challenges

The transfer and processing of traffic information from data gathering points out to travellers has long been the backbone of the public transportation company's information systems. During the 1990s and early 2000s the infrastructure used to spread this information (such as the display signs located at bus stops and train stations) was typically owned and controlled by the company. During the 2000s along with the massive penetration of the World Wide Web in average households, travel planning and travel update capabilities were extended into the homes and work places of most travellers.

During the autumn of 2008 there was a sudden increase in the usage of one of the public transportation company's servers. In 2002 the company set up a server hosting an xml-based web service, where third-party developers were given access to a selection of the company's data (such as travel planning capabilities). The existence of the server was not officially announced but was used third-party developers in many occasions. This included mostly students and start-up companies testing an idea related to public transport application development.

The server address along with necessary credentials to access the data became available on various web forums and hence known by many third-party developers. The reason for the sudden increase in traffic on the server was a release of the first iPhone application by a third-party developer using the data from the public transportation company. The release of this application marked a definite breakthrough for the use of mobile technologies in the history of the public transport company. Up to this point mobile phones had been causing a slim 1-1,5% of the total customer-initiated travel planning web traffic (the remainder was originating from the company homepage) whereas currently iPhone applications alone is producing some 12,5 % of the total amounts of hits

Although positive from an information diffusion perspective, this situation also faced the company with a dilemma - the iPhone application required a totally new and device-specific implementation of the mobile travel planner. Further, one year after the release of the iPhone application, Google released their mobile platform Android which was expected to work up a significant user base as well. The common denominator between the different platforms was a need for platform-centric implementation, which contrasted the previous standardized html-based and hence platform-independent capabilities for mobile phones. The question thus arose on how to develop information systems addressing this

emerging device heterogeneity. A business developer within Personal Transport Authority framed this challenge as:

"Because we cannot support every new mobile phone that enters the market, or all TV sets and you name it that is used to spread information. That is not our job, that is not what we're good at, and there are others that are much better than us at it."

5.2 Exercised control mechanism I : Attracting Developers

The public transportation company then decided to build a web site – a Developers Zone. On this web site developers could register, approve the terms of conditions and thereby get access public transport APIs. Further, the site contained a forum for developers to exchange ideas, propose solutions to problems; however, no direct support was offered by the public transportation company. The web manager added:

"Because we've always said that we do not give support. You have to solve problems yourself. You may ask a question on the forum and so hope that someone else helps, that's about how we think"

The APIs exposed to developers were identical to those used by the home page of the public transportation company themselves. The web manager indicated:

"The basic idea is really that we do not do anything special for [the developers]. These API's are those we need ourselves"

The APIs included: travel planning capabilities, commuter parking locations and disturbances in the traffic flow. Although the new Developers Zone was not marketed, its existence was spread among third-party developers and just three weeks after the launch, more than 70 third-party developers had registered on the web site.

Even though this third-party development strategy came along with some risks, for example, unpredictable web traffic volumes and the lack of control over how this information was interpreted and displayed by third-party developers, the benefits were still considered predominant. In other parts of Sweden where corresponding companies had been more hesitant to such strategy, "pirate APIs" had emerged. In these unsanctioned APIs, data were collected from public web sites and exposed through more programmer-friendly APIs - whose content then would move completely beyond the control of these organizations. This initiative along with its quite unrestrictive license terms was warmly welcomed by many developers, and became a mechanism to attract new developers. One engaged third-party developer explored the issue:

"I think it is a splendid initiative from the Public Transportation Company to give developers access to an interface as this to create their own applications. One can only hope that they function as an example to others, so that more will follow in the same direction"

5.3 Exercised control mechanism II: Announcing device non-coverage

Introducing the Developers Zone and the APIs enabled the public transportation company to have third-party developers construct end-user applications using their software platform. The major struggle which sparked the new ISD model was dealing with device heterogeneity and the company moved forward by making the following announcement on the Developers Zone:

"Do you have plans to build something for Android? Please let us know. We have no plans right now to develop a travel planner or other application for Android"

The intentional hope of the company was that other actors will support devices that run Android and works under its platform. This intended to motivate third-party developers to align with the organizational goal of supporting wider range of end-user devices. Directly after the announcement of

not supporting Android, a plenty of third-party developers responded by starting the implementation of Android travel planners[1] applications. The web manager commenting on that issue:

"We have gotten five or six responses to this call. The answers have ranged from 'yes we are building now and it is nearing completion', to somebody who said 'we have plans to possibly build and would like to engage in a business relationship''

5.4 Non-exercised control mechanism – boundary resource design

The applications typically built contained similar functionality as in the web site: travel planning combined with some device specific functionality, for example getting the nearest bus stop based on current location of the mobile phone through positioning technology. However, developing such applications was not without complications. The public transportation company applied various restrictions through their APIs. They were able to control third-party developers through the released APIs as they restrict what can be known and what can be done by these actors. For example, the public transportation company released an API that allows third-party developers to fetch all bus stops in a specific region without allowing them to store the fetched data on the device for more than 24 hours. One of the third-party developers commenting on that as:

"There's little more than ten thousand stops and there is a policy that if you make an application you cannot save stops forever,[...] I downloaded all the stops every time the application is started [...] But it does not work with Android since I got 'out of memory error' after a thousand stops."

Another recurring theme among the interviewed developers was the inability to fully support the diverse user group utilizing their applications. Along with the devices came not only the possibility to display data but also to use the information in new situations. One third-party developers noted:

"[Our user group] also pointed out to us that they got no assistance during their journey, they do not know just where on my journey I am at currently, which stop comes before my final stop, that kind of stuff they got no help with"

Using the location-based services such as GPS, it is possible to support the user based on current geographic position. However, this type of applications proved difficult to develop using these APIs. Another third-party developer explored this:

"Like, we make a real-time search on line 11 from this station but we can't tell just what bus stops the bus line will visit afterwards, or what prior stops it makes. [...] We had to hard-code 10 000 lines of bus stop data to make this work."

A similar, yet non-replied post in the forum reads:

"Which API should I query if I want to know what bus stops a bus line stop at? First and foremost I would like to know the lines that are available. Then the actual route the lines take. I have not found any API for any of it. Is there? Is it possible to arrange?"

Another innovative feature that a third-party developer wanted to implement was informing users about obstacles which suddenly arise during travel, which potentially might delay the current travel and suggest newer and faster roots. The inability to fully take advantage of the device was a result of the API design. A third-party developer exploring that:

"I mean, we are developing an application which the traveller uses right now, you are not at your computer [...] [rather] we should guide the user during travel: 'there is trouble ahead, jump off this bus and take this one instead'. That is not supported [...] The APIs are so bus stop oriented!"

[1] As of November 18, 2010, there were 6 applications in Android market implementing various parts of various APIs. The most popular had 10000-50000 downloads and a rating of 4,4 out of 5.0 (472 votes).

Further, the existing real-time disturbance information was deemed problematic; its location was limited to a specific municipality and its relevance was questioned. One of the third-partydevelopers exemplified how the boundary resources on particular use situation required the information to be delivered in a way:

"And you can just go in and view a traffic message on the web site, there are road works in [a local tunnel] which started 6 months ago, this is irrelevant real-time information […] and if you are out on the move it is even more important that the information is context-bound to here and now"

6 Discussion

In this paper, our objective was to develop an exploratory understanding of how and why third-party development for multi-contextual services does require adapted control mechanisms. In what follows, we discuss the exercised control practices, supported by evidences from our case study. We then conclude by presenting possible future research opportunities.

Nowadays, there is a wide variety of Smartphone devices being adopted and becoming a part of the information systems of many organizations. Most of those devices support development of third-party applications. This variety of devices, the operating systems they are running and associated development programs created a challenge for both the organization and third-party developers under study. In order for the public transportation company to employ a new innovation network, they provided the community of third-party developers a set of APIs. Hence, through this move, the public transportation adhered to the clan values of third-party developers and hence was able to much better tap in to these development resources.

To deal with device heterogeneity, the public transportation company employed a form of *outcome control*. The current literature on ISD control suggests that control are achieved through the reward of such sought after outcome (Android phone coverage) (Kirsch 1996; Kirsch 1997). In this case however, Android coverage was achieved through the announcement of a planned *non-coverage* from the company itself, and thereby opened up a niche for third-party developers. End-user platform coverage was however hampered by syntactic shortcomings in the boundary resource design (Carlile 2002) decreasing their efficiency as boundary objects.

Along with the emergence of nomadic devices, new use situations for a diverse user group emerged. These lay at the heart of multi-contextual services yet are very difficult to anticipate beforehand (Henfridsson and Lindgren 2005). In many ways the public transportation company were forced to rely on third-party self-control to deal with user and use situation heterogeneity[2]. However, to enable such self-control, the necessary degrees of freedom must be passed on to the controlee (Kirsch 1996). In this case, we argue, such degrees of freedom are mitigated through APIs.

First, APIs served as *Repositories* (Star and Griesemer 1989) by working as developers' reference point supplying specifications, data structures, object classes, protocols and so on. They helped developers solve immediate problems by providing them with shared definitions and values. Second, the APIs were possible to use due their *coincident boundaries* (Star and Griesemer 1989) as they offered third-party developers a common referent for using their different perspectives independently. Yet, here we also find implications for development of multi-contextual services. While the APIs served well for tasks before travelling, they displayed shortcomings in transferring design capabilities for support during travel. For example, the retrieved information was unable to efficiently map onto the devices consuming information while the user was on the move.

[2] In this case the use of the term *self-control* is grounded in the relationship between the platform owner and third-party developer. Third-party developers may consist of a group employing numerous control efforts to achieve their organizational goals.

Knowledge trading zones, such as the Developers Zone, gives the platform owner a potential to be triggered by feedback from the community of third-party developers. Hence recognizing insufficiency in current boundary resources, and further develop and introduce new and enhanced boundary resources. On the one hand, this will help the platform owner to constantly meet the expectations of developers, and on the other hand give an instrument to exercise control. However, our study shows that the public transportation company partially failed in supporting their introduced APIs, as their strategy was to let the community of third-party developers rely on itself. Thus, by sticking to path-dependent trajectories (expose what they already have) and not engaging in any third-party negotiations they also became unable to overcome some pragmatic boundaries (Carlile 2004) to better meet the challenge of use situation heterogeneity.

7 Implications and Conclusions

In this study we have explored the concept of *control* in third-party development settings. As noted in the literature, control in such open settings tends to be *bidirectional* (controller-controlee relationship) (Tiwana et al. 2010). Indeed, as illustrated by the case, the public transportation company (the controller) actually modified its behaviour in accordance with the clan values of the developers in order to more effectively employ these development resources. However, despite the lack of stipulated control (for example through formal contracting), the communication of non-coverage illustrates that exercising control can be quite powerful. Hence, we argue, employing successful control measures in this type of third-party development settings is thus two-sided: it includes both attracting developers, as well as motivating them to act in accordance with organizational objectives.

Moreover, by adding the complexity associated with developing multi-contextual services, from a control perspective, the design and management of APIs as boundary resources become essential. Since the usage of these resources will support a wide range of devices, users and use situations, the design of these boundary resources in our case points to that these resources should be both richer and more plastic to cut across the heterogeneity dimensions associated with multi-contextuality. Still, anticipating all potential uses of these resources beforehand is unfeasible. Thus, engaging in developer interaction and renegotiation of boundary resource design becomes an important means for influencing third-party development efforts.

The nature of this paper has been exploratory. Control has not, to our knowledge, been empirically assessed in this type of development settings. Since this research has been conducted as an in-depth single case study we believe that the preliminary findings in this paper should be investigated in other empirical settings for further insights into the phenomena. Given the challenges of multi-contextual services and how third-party development may be used to meet them, we see a need for more knowledge on how to conceptualise control in such settings.

References

Bergman, M., Lyytinen, K., and Mark, G. (2007) Boundary Object in Design: An Ecological View of Design."Journal of the Association for Information Systems, 8(11), 546-568.

Boland, R. J., Lyytinen, K., and Yoo, Y. (2007) Wakes of innovation in project networks: The case of digital 3-D representations in architecture, engineering, and construction. Organization Science, 18(4), 631–647.

Bosch, J. (2009) From software product lines to software ecosystems In Proceedings of the 13th International Conference on Software Product Lines (SPLC). Springer LNCS.

Bosch, J. and Bosch-Sijtsema, P. (2010) From integration to composition: on the impact of software product lines, global development and ecosystems. Journal of Systems and Software, 83(1), 67-76.

Briers, M. and Chua, W. F. (2001) The role of actor-networks and boundary objects in management accounting change: a field study of an implementation of activity-based costing. Accounting, Organizations, and Society. 26(3). 237- 269.-+96+96

Carlile, P. (2002) A pragmatic view of knowledge and boundaries: Boundary objects in new product development. Organization Science, 13(4), 442-455.
Carlile, P. (2004) Transferring, Translating and Transforming: An Integrative Framework for Managing Knowledge across Boundaries. Organization Science. 15(5), 555-568
Carroll, J, 2008. Theorizing the IT Artifact for Mobility: A Portfolio, Not a Singularity. ICIS 2008 Proceedings. Paper 64.
Chesbrough, H., Vanhaverbeke, W., and West, J. (2006) Open Innovation: Researching a New Paradigm. Oxford University Press, Oxford.
Choudhury, V., and R. Sabherwal. (2003). Portfolios of control in outsourced software development projects. Information System Research, 14(3) 291–314.
Clark, T. D., Jones M.C. and Armstrong C.P. (2007). The Dynamic Structure of Management Support Systems: Theory Development, Research Focus, and Direction, MIS Quarterly 31(3), 579-615.
Crisp, C. B. (2002). Control enactment in global virtual teams. Acad. Management Annual Meeting, Denver, CO (August).
de Souza, D. F. Redmiles, L.-T. Cheng, D. R. Millen, and J. Patterson, 2004. "How a Good Software Practice Thwarts Collaboration - The Mulitple Roles of APIs in Software Development," in Proc. Foundations of Software Engineering (FSE 2004), Newport Beach, CA, 2004, pp. 221-230.
Dey, A., Abowd, G., and Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact. 16(2), 97-166.
Eisenhardt, K. M.(1989). Building theories from case study research. Academy of Management Review, 14 (4), 532-550.
Evans, D.S., Hagiu, A. and Schmalensee, R. (2006) Invisible Engines: How Software Platforms Drive Innovation and Transform Industries. Cambridge MA: MIT Press.
Ferguson, C. H. (2005) “What’s Next for Google?,” MIT Technology Review, 108(1), 38-46.
Gawer, A., and Cusumano, (2002). M. Platform leadership: How Intel, Microsoft and Cisco drive industry innovation, Boston: Harvard Business School Press.
Ghazawneh, A. , “The Power of Platforms for Software Development in Open Innovation Networks,” *International Journal of Networking and Virtual Organisations,* forthcoming
Ghazawneh, A., and Henfridsson. O. (2010). “Governing Third-Party Development through Platform Boundary Resources”. In Proceedings of International Conference on Information Systems (ICIS), St.Louis, MO.
Henderson, J. C., and Lee. S. (1992). Managing I/S design teams: A control theories perspective. Management Science. 38(6) 757-777.
Henfridsson O., and Lindgren R. (2005). Multi-contextuality in ubiquitous computing: Investigating the car case through action research. Information and Organization, 15(2), 95-124
Henfridsson, O. and Lindgren, R. (2010). User involvement in developing mobile and temporarily interconnected systems. Information Systems Journal 20 (2), 119 – 137.
Jansen, S., Finkesltein, A., and Brinkkemper, S. (2009a) A sense of community: a Research Agenda for Software Ecosystems,” In 31st International Conference on Software Engineering, New and Emerging Research Track, 187-190.
Jansen, S., Brinkkemper, S. and Finkelstein, A. (2009b) Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems,” In Proceedings of the First Workshop on Software Ecosystems, 34-48.
Keil, M. and Carmel, E. (1995) Customer-developer links in software development. Communications of the ACM, 38, 33–44.
Kirsch, L. J. (1996). The management of complex tasks in organizations: Controlling the systems development process. Organization Science. 7(1) 1–21.
Kirsch, L.J. (1997). Portfolios of Control Modes and IS Project Management. Information Systems Research, 8(3), 215-239.
Kling, R. and Iacono, S. (1984). The control of information systems after implementation. Communications of the ACM, 27, 1218-1226.
Langley, A. (1999) Strategies for Theorizing from Process Data. Academy of Management Review. 24(4), 691-710.

Lamb, R., and Kling, R. (2003) “Reconceptualizing Users as Social Actors in Information Systems Research,” MIS Quarterly 27(2), pp. 197-235.
Levina, N. (2005). Collaborating on multiparty information systems development projects: A collective reflection-in-action view. Inform. Systems Res. 16(2) 109–133.
Lindgren, R., Andersson, M., and Henfridsson, O. (2008) Multi-contextuality in boundary spanning practices. Information Systems Journal. 18(6), 641-661.
Luff, P., and Heath, C. (1998). Mobility in collaborationProceedings of CSCW98, 305–314. Seattle, USA: ACM.
Lyytinen, K., and Yoo, Y. (2002). Research commentary: the next wave of nomadic computing. Information Systems Research, 13(4), 377–388.
Mähring, M. (2002). IT project governance. Ph.D. dissertation, Stockholm School of Economics, Stockholm, Sweden.
Mallat, N., Rossi, M., Tuunainen, V.K., and Öörni, A. (2009) The impact of use context on mobile services acceptance: The case of mobile ticketing. Information & Management. 46(3), 190-195.
Markus, M. L., Majchrzak, A., and Gasser, L. (2002). A design theory for systems that support emergent knowledge processes. MIS Quarterly, 26(3), 179–212.
Messerschmitt, D.G., and Szyperski, C. (2003). “Software Ecosystem: Understanding an Indispensable Technology and Industry”, MIT press, 2003.
Miles, M.B. and Huberman, A.M. (1994) Qualitative Data Analysis: An Expanded Sourcebook, 2nd edition, Thousand Oaks, CA: Sage Publications.
Meyer, M. H. and Seliger, R. (1998) Product platforms in software development. MIT Sloan Management Review. 40(1), 61-74.
Perry M, O’Hara K, Sellen A, Brown B, and Harper R (2001) Dealing with mobility: understanding access anytime, anywhere. ACM Transactions on Human-Computer Interaction. 8, 323–347
Sawyer, S. (2000) Packaged software: implications of the differences from custom approaches to software development. European Journal of Information Systems, 9, 47–58.
Star, S. L. and Griesemer, J. R. (1989) Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. Social Studies of Science. 19(3), 387-420.
Strauss A, Corbin J. (1990) Basics of qualitative research. Grounded theory procedures and techniques. Newbury Park: Sage Publications.
Tapscott D. and Williams A.D (2006). “Wikinomics: How Mass Collaboration Changes Everything,” New York (N.Y.): Portfolio.
Tiwana, A., Konsynski, B. and Bush, A. (2010) “Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics”. Information Systems Research (21:4), 675-687.
Taudes A., Feurstein M., and Mild A., (2000) “Options Analysis of Software Platform Decisions: A Case Study,” MIS Quarterly, (24:2), pp. 227-243.
Tuunanen, T, (2003) A New Perspective on Requirements Elicitation Methods. The Journal of Information Technology Theory and Application (JITTA), 5(3), 45-72
Tuunanen, T., Myers, M., and Cassab, H (2010). A Conceptual Framework for Consumer Information Systems Development. Pacific Asia Journal of the Association for Information Systems: 2(1)
Van de Ven, A. H., Polley, D., Garud, R., and Venkatraman, S. (2008) The Innovation Journey. New York: Oxford University Press
Yin, R.K. (2009) Case Study Research: Design and Methods, (4 ed.). Thousand Oaks: SAGE Publications..
Yoo, Y., Lyytinen, K., and Boland, R.J. (2008) Distributed Innovation in Classes of Networks. Proceedings of the Fourty-First Annual Hawaii International Conference on System Sciences (CD-ROM), January 7-9, Computer Society Press.
Yoo, Y., Lyytinen, K., and Boland, R. J. (2009) “Innovation in the Digital Era: Digitization and Four Classes of Innovation Networks,” Working Paper, Temple University.