ABSTRACT

Title of dissertation: INSTITUTIONAL LOGICS, INDIE SOFTWARE DEVELOPERS AND PLATFORM GOVERNANCE

Yixin Qiu, Doctor of Philosophy, 2013

Dissertation directed by Professor Anandasivam Gopal
Professor Il-Horn Hann

Department of Decision, Operations & Information Technologies, R.H.Smith School of Business

This two-essay dissertation aims to study institutional logics in the context of Apple's independent third-party software developers. In essay 1, I investigate the embedded agency aspect of the institutional logics theory. It builds on the premise that logics constrain preferences, interests and behaviors of individuals and organizations, thereby determining the appropriate and legitimate decisions and actions of actors. In the meantime, most social actors operate in fields characterized by multiple institutional logics where contradictions exist, allowing individuals and organizations with opportunities for negotiation and change through exploitation or management of these contradictions. I specifically study two competing institutional logics: professional and market logics when they are experienced simultaneously by independent iOS app entrepreneurs. Using participant observation and semi-structured interviews, I delineate the ways in which logic tension is reconciled through mechanisms of *logic synthesis* in three entrepreneurial areas – app ideation, app execution and app marketing, and conditions which facilitate or inhibit logic synthesis.

In essay 2, I study the emergence and evolution of field-level logics in the context of Apple's desktop developers – Mac indies. Following the cultural

emergence model of field-level logics in Thornton et al. (2012), and the argument that "field-level logics are both embedded in societal-level logics and subject to field-level processes that generate distinct forms of instantiation, variation, and combination of societal logics" (p148), I particularly examine the relationship between resource environment and the emergence and evolution of field-level logics. Taking advantage of a critical change in developers' resource environment – Apple's opening of the iOS App Store and subsequently the Mac App Store, and hence its governance model shifting from mainly a technological platform to a platform that includes a market exchange place, I identify developers' logics before and after the change, namely, the software ecosystem logic and platform ecosystem logic. Two ideal types are constructed for the logics along elemental categories, and a content analysis demonstrates the logic shift pattern as resource environments change. A further analysis of the two logics suggests that the software ecosystem logic and platform ecosystem logic are in contestation at this early stage of institutional change.

INSTITUTIONAL LOGICS, INDIE SOFTWARE DEVELOPERS AND PLATFORM GOVERNANCE

by

Yixin Qiu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:

Professor Anandasivam Gopal, Co-Chair
Professor Il-Horn Hann, Co-Chair
Professor Ritu Agarwal
Professor Brent Goldfarb
Professor Erve Chambers *(Dean's Representative)*

Dedicated to my parents: Qianzheng Gu, Yong'an Qiu and my husband Tao Shen.

ACKNOWLEDGMENTS

This dissertation would not have been possible without many people's contribution and support. My deepest gratitude is to my co-advisers: Dr. Anand Gopal and Dr. Il-Horn Hann. It has been a unique and unforgettable experience of conducting a study on institutional logic using qualitative interpretive methodology with two economists. I am truly indebted and thankful to them for having faith in my work, and pushing me to think deep on the theoretical front. They have urged me to document the methodology more systematically and organize the writing in a more structured way. I have been very fortunate to work with Anand and Il-Horn. Dr. Ritu Agarwal has been one of the best academic role models I have ever known. Her constructive criticism on my conceptual framework has been enormously thought-provoking. I would also like to thank Dr. Kislaya Prasad for his time in advising the weekly IS student presentation series. I have benefited a great deal from his comments on my work. I am grateful to Dr. Brent Goldfarb for a great class and his interest in my work. His class on entrepreneurship is one of my favorite in the doctoral program. I have forever learned to keep a critical mind in evaluating the causal inference in any academic research. Brent's work on Dot Com Era and technology commercialization has been great inspirations in my study on app entrepreneurs. I am also immensely appreciative of Dr. Erve Chambers for his guidance since I was in the Applied Anthropology program. It is his class and books that introduce me to the world of culture and fieldwork, which I have continued to use till this day in carrying out my dissertation research. I wish Erve all the best after his retirement.

I would also like to thank other professors who have been crucial in making my graduate study experience unforgettable. First and foremost is Dr. Kate Stewart. I am thankful to her for recruiting me to the doctoral program in the Smith School, for

which my life is forever changed. I am indebted to Dr. Kay Bartol for invaluable intellectual stimuli during the research meetings. I am grateful to Dr. Siva Viswanathan for pushing me to work harder on my research and presentation skills, and to Dr. Samer Faraj for his class, which introduced me to the world of IS research.

I am extremely grateful to the fellow PhD students in the Smith School, many of whom have become professors in top-tier universities. I would like to especially thank Jie Mein Goh for her time in listening to my struggle and frustrations during the dissertation process and offering her insights, Sherae Daniel for her encouragement, Sri Kudaravalli for his help in polishing my research ideas, and Jiban Khuntia for the many memorable conversations we have had. I am also grateful for the weekly IS student presentation series, from which my research has received tremendously helpful feedback.

I am thankful to all the app developers who have agreed to my interview request, the MoDevDC meet-up group organized by Pete Erickson, where I conducted most of my field observations, and ZerionSoftware, for providing me with an internship opportunity from which I gained invaluable experience for my research.

Last but not the least, this dissertation would not have been possible without the support of my family: my parents, in-laws and my husband, Tao Shen. Their love, care and tolerance have helped me overcome constant frustrations during my research and the writing process.

TABLE OF CONTENTS

Essay1: On Synthesizing Professional and Market Logics in Nascent Entrepreneurship: A Study of iOS App Entrepreneurs

Abstract

Research in institutional logics indicates that professional and market logics are two competing institutional logics which often lead to conflict due to differences in their key attributes such as sources of legitimacy, norms and definitions of success. In this paper, we study how these conflicting logics are experienced simultaneously by independent application (app) developer entrepreneurs on the Apple App Store. Subsequently, we investigate how these app entrepreneurs resolve these conflicts through a process of logic reconciliation that we term *logic synthesis.* Using a practice-based qualitative approach, we identify the logic anchors for a set of app developers on three entrepreneurial areas – app ideation, app execution and app marketing. Subsequently, we identify the key processes by which app developers find ways to resolve the conflict between market and professional logic within these entrepreneurial areas so that they may achieve a balance between the two logics at the level of the entrepreneurial area, i.e., achieving logic synthesis. Since synthesis does not occur in isolation, we also identify and discuss a key set of factors that hinder or encourage the process of synthesis. Using a set of interviews and field work methodology, we provide a nuanced model of logic anchoring and synthesis in the context of the nascent entrepreneur on the App Store, thereby contributing to the literature on institutional logics, platform ecosystems and nascent entrepreneurship.

***Keywords*:** independent application developers; institutional logic; qualitative research; nascent entrepreneurship; platform ecosystems

1. Introduction

The software development industry has undergone many transformational changes in the last decade as platforms and customer needs have evolved. The latest transformation in this industry is the emergence and growth of the app economy, characterized by small applications that are developed specifically to be implemented on mobile or hand-held platforms and sold through virtual retail platforms (MacMillan, Burrows and Ante 2009, Wasserman 2010), the most popular of which is the Apple App Store. The App Store has been credited with revolutionizing the mobile apps consumption and production market by "democratizing" digital innovation (Boudreau 2012, Yoo et al. 2010). Specifically, the App Store has allowed independent software developers ("indie" developers) to enter the marketplace for apps by providing access to customers that were previously available only to larger and more established firms. In effect, the introduction of the App Store has allowed nascent app entrepreneurs who have identified the needs of niche segments to directly reap the benefits of their innovative activity.

Nascent entrepreneurs are defined as those individuals who decide to commit their own time and resources to founding a new firm in the form of a start-up (Wagner 2007). This represents conception, the first stage of a four-stage entrepreneurial process, of which the other stages are gestation, infancy and adolescence. Using this definition, many of the indie developers who offer apps on the App Store are nascent entrepreneurs since their firms are usually founded by 1-2 persons. These indie developers perform all the activities related to software engineering, including software design and development, quality assurance and security in order to meet App Store's criteria for quality and functionality. At the same time, they are also responsible for strategically positioning their apps in the marketplace, reacting to

competition and marketing their apps effectively. In effect, app entrepreneurs are required to handle the *business* practices needed to be competitive in the marketplace as well as the *software engineering* aspects of application design and development. It is often the case that the market demands are in conflict with the statutes of software development and design principles for such indie developers. Indeed, this conflict is enshrined in the ideal type construction of professional and market logic proposed within the institutional logics perspective (Thornton 2002, 2004, Thornton, Ocasio and Lounsbury 2012). A natural question arises – if such conflicts from contrasting logics are systematic in the environment that these app entrepreneurs operate in, how then do they *resolve* these conflicts on an ongoing basis? This central question forms the focus of this paper.

The theory of institutional logic builds on the premise that society consists of a set of interdependent and yet contradictory institutional logics and is rooted in seminal work by Friedland and Alford (1991), followed by Thornton and her colleagues (1999, 2002, 2004, 2005, 2012). These logics include logic of the family, community, religion, state, market, profession and corporation. These logics dictate actions and decisions that are considered legitimate and rational, depending on the context of a particular institutional order. Understanding these logics is important because of the two levels of influence they wield on social actors (firms, individuals or collectives). First, logics constrain preferences, interests and behaviors of individuals and organizations, thereby determining the appropriate and legitimate decisions and actions of actors. Second, most social actors operate in fields characterized by multiple institutional logics where contradictions exist (Dunn and Jones 2010, Yeow and Faraj 2011), allowing individuals and organizations with opportunities for negotiation and change through exploitation or management of these contradictions.

Building on this literature, in this paper we focus on the dynamics between logic of the *profession* and logic of the *market* in particular, and examine how they both shape and are appropriated by indie app developers on the Apple App Store. On one hand, app developers follow the professional logic rooted in software engineering principle as well as the accepted norm of user-directed innovation in the software profession (Von Hippel 1986). The focus here is on developing apps that meet their own needs and provide the most elegant or satisfying engineering solutions, regardless of cost or revenue implications (Wasserman 2010). On the other hand, the logic of the market within the App Store requires developers to be highly responsive to the platform owner's control in the market place and the extremely competitive and volatile market environment, wherein the focus is on adaptation to platform policies, first-mover advantages, operational efficiency, and an emphasis on customer needs. Clearly, these two competing logics experienced by individual nascent entrepreneurs on the App Store create "conflicting pressures on their cognitive and behavioral capacities" (Thornton et al. 2012, p. 57), which need to be reconciled in a systematic manner (Jain, George and Maltarich 2009, Eikhof and Haunschild 2007, Tschang 2007).

Our study delineates the antecedents and the processes by which this logic reconciliation is achieved. We build on the argument that particular circumstances experienced by individuals will trigger different goals and schemas and shape their behaviors (Thornton et al. 2002, Ross and Nisbett 1991). Our work here also answers the call issued by Thornton et al. (2012) for more research at the individual-institutional-field levels. To that end, we conduct a qualitative study of app developers in the Mid-Atlantic region and address the following research objectives. First, we identify what attributes constitute professional and market logic for app developers

and how these logics are manifested in app developers' specific entrepreneurial practices. Second, we identify factors that influence the manner in which app developers *reconcile* conflicting logics as part of their working process. Finally, we identify the specific processes that app developers use in managing logic conflict. From our qualitative data, we are able to provide a richer and more comprehensive model of logic conflict and reconciliation carried out by app developers on the fast-moving, competitive App Store.

Our study contributes to the institutional logics literature in three significant ways. First, we delineate specific attributes of the professional and market logic in the context of App entrepreneurs. The conceptualization of market logic here is particularly interesting. Because the App Store market is created and governed by Apple as the platform owner, the instantiation of market logic and its implications for legitimacy and strategic behavior by app developers differs considerably from descriptions of market logic seen in the literature where there is no such central controlling figure (Thornton 2001, 2002, 2004). Thus, our description of market logic is a marked departure from extant literature and is rooted in contexts where software platforms dominate (Parker and van Alstyne 2008).

A second contribution of our work is to identify contexts where professional logic is not sacrificed in favor of market logic. Most of the extant literature on dynamics between professional and market logic emphasizes the notion that in the presence of conflict, professional logic tends to lose out to market logic. However, it is possible that some agents sacrifice elements of market logic in favor of professional logic in order to attain success. In the App Store context, we observe that both logics influence individual practices, and that conflict is reconciled in both directions. We term this "logic synthesis", a more balanced view of interaction between market and

professional logic that does not start with the bias of one-directional movement from professional to market logic yet allows a more expansive discussion of antecedents of logic reconciliation. Our study also advances theorizing on the microfoundations of institutional logics (Thornton et al. 2012) by describing how the presence of multiple logics trigger and enable individual behaviors and practices on the App Store.

Finally, our work contributes to the literature on nascent entrepreneurs (Davidsson and Honig 2003, Baron and Ensley 2006) by studying a growing area of nascent entrepreneurship, i.e., app entrepreneurs. Through our qualitative approach, we identify three major types of tasks that app entrepreneurs engage in, and provide a more nuanced analysis of how logic synthesis strategies are developed for each of these tasks as these entrepreneurs move from conception to gestation and infancy (Wagner 2007). We start with a review of the relevant theoretical arguments in professional and market logic in the next section.

2. Theoretical Background – Conflicts between Professional and Market logic and Logic Synthesis

Institutional logic is broadly defined in the literature as "the socially constructed, historical patterns of material practices, assumptions, values, beliefs, and rules by which individuals produce and reproduce their material subsistence, organize time and space, and provide meaning to their social reality" (Thornton and Ocasio, 1999, p. 804). In other words, institutional logics provide acceptable and legitimate guidelines for behavior for entities within societal, organizational or individual contexts. The literature has examined varying competing logics such as the logic of religion, corporation and market, which act on individuals or groups depending on the context. Logics are characterized by a set of factors such as the sources of legitimacy, signals of authority and the existence of unifying norms, each of these help identify the specific logic (Thornton 2004). In this paper, we specifically address the competing

logics that manifest in the context that we study – professional and market logics, and their conflicts.

Conflict between professional and market logic has been examined in the literature at societal, organizational and individual levels. At the societal level, prior work has used a typology of logic ideal types to describe how factors such as sources of legitimacy, authority, identity, norms, strategy and economic systems vary between market and professional logics (Thornton 2004, Thornton et al. 2012). At the organizational level, Thornton (2001, 2002) argues that professional logic emphasizes personal capitalism with a focus on factors such as personal reputation, personal networks and organic growth. Alternatively, the market logic here revolves around market capitalism, where market position, corporate structure, acquisition growth and capital committed to market return are emphasized. Different firms may choose different elements within these contrasting logics but in many cases, the industry tends toward market logic as the dominant logic over time (Thornton 2002).

At the individual or small group level, these logics are individually manifested in professions that use economic value as legitimizing factors versus professions that use alternative (often artistic) values as legitimizing factors. Conflict is driven more by organizational and individual identity and less by alternative forms of capitalism. For example, Bourdieu's (1990) theories of artistic and economic logic in practice contend that "the economic logic of practice is followed when individual benefits are gained from exchanging goods and services via markets, such as product markets, capital markets or labor markets. In comparison, artistic logic of practice is marked by the desire to produce art for art's sake, where art is an abstract quality that surfaces in specific aesthetics or individual reactions by the recipient, and needs no external legitimization" (Bourdieu 1990, Eikhof and Haunschild 2007, p. 526). Because of the

contradictions embedded in these two contrasting logics, interactions between individuals subscribing to one of these two logics tend to engender conflict. In documenting the 1996 Atlanta Symphony Orchestra Strike, Glynn (2000) shows how musicians embrace artistic creativity and excellence as symbols of success, while management and the Board consider financial return as success, leading to conflict. In a different context, Nag, Corley and Gioia (2007) examine the attempt by a high-tech R&D organization to transform into a market-oriented organization by grafting new, non-technological knowledge. Extant knowledge in the firm is driven by scientists and engineers possessing a pure technology-push mentality focused on developing cutting-edge technology, often without an obvious commercial application. In contrast, the new knowledge that was grafted in the organization is represented by executives who take a market-pull orientation and aim to make the organization market-driven and customer focused, thereby creating conflict in the firm.

When conflicting logics are experienced by the same individual, negotiations between the two logics undergo a more intricate process. In the microfoundations model of institutional logics, Thornton et al. (2012) argue that "humans have multiple, loosely coupled, and often contradictory social identities and goals. Specific social situations and interactions shape which social identities and goals get triggered. And individuals learn multiple contrasting and often contradictory institutional logics through social interactions and socialization" (p. 80). Applying this argument to the dynamics between professional and market logic suggests that when both logics are available and accessible to individuals, the logic that is more salient depends on the immediate situational characteristics. Furthermore, how individuals deal with the inherent conflict between professional and market logic is affected by social interactions with others and the socialization that follows. The literature shows that

professionals indeed vary in the degrees to which they are susceptible to making changes induced by the market logic. For some, professional identities and norms are more salient even though adapting to the market logic could bring in monetary benefits. For example, Stern (2004) shows how scientists pay a compensating differential to participate in science. Studies on open source software suggest that many developers contribute to writing software out of intrinsic instead of extrinsic motivations (Roberts, Hann and Slaughter 2006, Shah 2006). Alternatively, other professionals choose to *reconcile* the two logics to reach a logic balance (Lampel, Lant and Shamsie 2000, Tschang 2007). This is especially observable when professionals establish business based on their domain knowledge and expertise, or when they experience career transitions. Their role identity may change as skills, behaviors, and patterns of interaction adapt to meet the demands of the new role (Jain et al. 2009, Ebaugh 1988, Louis 1980).

Professionals reconcile competing logics through different methods. One method involves *revising* their beliefs and behaviors, sometimes involuntarily. For instance, Eikhof et al. (2007) document how German theatre artists invest extensive effort in strategic networking (a market logic strategy) to ensure positions in future plays as a response to idiosyncratic and subjective staffing decisions by stage managers. While gaining job security, artists compromise part of their artistic passion for economic benefits. A second method involves not significantly compromising on professional logic but *acquiring* elements of market logic to achieve balance. Nag et al. (2007) show that rather than adopting practices championed by business development and marketing professionals, R&D engineers and scientists adapted a previous technology problem into a market problem they could solve. However, they preserved their professional identity while addressing the logic conflicts through modified

professional knowledge practice. Finally, *collaboration* to acquire market logic can also accomplish logic reconciliation. Jain et al. (2009) show that professors involved in technology transfer activity delegate commercialization activity to those who possess related skills while preserving cherished values associated with being an academic.

Taking this argument one step further, we also find some evidence for when market logic is compromised in favor of professional logic. Voronov and De Clercq (2007) study the Ontario wine industry where success could be driven by both commercial strategy and a degree of artistic and authentic appeal. The authors find that in many cases, the dominant logic is one emphasizing artistic authenticity while concealing their practices on the commercial aspects of the business. Many vineyard owners sacrifice market logic to gain professional logic under some circumstances.[1] A similar fluidity of identity and logics was observed in Elsbach and Flynn's study (2008) of toy designers in a large US corporation, where most designers defined their creative approach as being "flexible" rather than being excessively market-driven.

These arguments imply that it is likely that professionals in fields with a strong market and engineering logic may use two approaches to logic balance: one where professional logic will be sacrificed in pursuit of market logic, and the other where professional logic will still dominate but may have elements of market logic grafted on. As it is possible for professionals within the same profession to have diverse identities, similarly it is conceivable that logic balance may be achieved from either end-point of the continuum, which is a significant deviation from extant literature. Building on these arguments, we propose a working definition of *logic synthesis* for achieving logic balance. We propose: *For logic synthesis to occur, it is necessary to*

[1] The paper positions these contrasts under the framework of Bourdieu's (1993) "field of large-scale production" (FLP) versus "field of restricted production" (FRP).

concurrently consider both logics of the profession and of the market. The process of logic synthesis can occur from either focal logic in an individual facing logic tension. Logic synthesis therefore involves focal logic compromise, opposite logic extension, or both, such that a balancing point can be found where the best proportion of the two logics is achieved for the individual in the specific context.

3. Research Methods

Given the limited theory pertaining to app entrepreneurs' motivations, strategies and processes available in the literature, we chose to study logic synthesis through an inductive, ethnographic study, with a focus on entrepreneurs' actual practices (Orlikowski 2000, Schultze and Orlikowski 2004, Levina and Vaast 2005, Levina and Vaast 2008). Inductive studies are especially useful for developing theoretical insights when research focuses on areas that extant theory does not address well (Ozcan and Eisenhardt 2009). The ethnographic approach is especially effective in grasping the culture of an emerging group of population in an open-ended manner. It is therefore a good fit for the research questions at hand.

We restrict our sample to include developers who have published at least one app on the App Store. Since we are interested in nascent entrepreneurs, we focus only on those who build bootstrapped ventures with no external financing support. This can include either full-time or part-time businesses. We exclude hobbyists who publish apps on the App Store only for fun and not for revenue-generating purpose. Since the thresholds to publishing an app on the App Store are low, entrepreneurs with heterogeneous technical backgrounds and logics are present. The potential to attract entrepreneurs identifying with both professional and market logics concurrently makes the mobile app industry well-suited to studying logic synthesis in a cross-

sectional manner, as opposed to prior work that studies logic shift over time in a focal industry (Thornton 2002).

Data collection for the study began in November 2009 from multiple data sources: field and online observations, semi-structured interviews as well as participant observation for triangulation. Over 90 hours of field observations were conducted in the 2009-2011 time-period. In addition, the first author attended multiple Mac and iOS developers' meet-ups and events for mobile entrepreneurs in the Middle Atlantic Region. During these visits detailed field notes were taken on developer presentations and interactions. The meet-ups provided a rich understanding of the ecosystem around the app developer community in addition to access to specific app entrepreneurs who were then identified as potential interview candidates. In total, 26 face to face semi-structured interviews were conducted with 19 app entrepreneurs. Each interview, lasting between 1 to 2 hours, was recorded and transcribed. Extensive memos were taken during the transcribing process. Resources such as company websites, blogs, user forums and Facebook fan pages of the 19 entrepreneurs were examined for triangulation. Theoretical sampling (Glaser and Strauss 1967) was used when choosing new participants in order to maximize variant instances of the two logics and synthesis practices. The first author also conducted participant observation through a 10-month internship at an iOS mobile software company. Through day-to-day work activities and interactions, a deeper understanding of the lifecycle of a mobile app and the decision-making process of developers was gained. This experience was used to validate internal and face validity of the model developed (Adler and Adler 1987). Table 1 provides an overview of all collected data.

Our analytic approach followed an iterative process of theory development and analysis. The analysis consisted of three steps: first, we identified major areas that an

app entrepreneur needs to make decisions on during the development and marketing of an app. After drawing up a list of identified practices, we condensed the set of entrepreneurial activities into three stages that echo extant literature: app ideation, app execution and app marketing. Given that app development is a form of new product development, the first two stages are consistent with the rational plan perspective of the new product development process, which maintains that "a product that is well planned, implemented, and supported by senior management will be a success" (Brown and Eisenhardt 1995, p. 348). These two stages are also consistent with the literature on where interactions between the two logics occur (Voronov et al. 2007, Baron et al. 2006, Tschang 2007, Nag et al. 2007, Lampel et al. 2000, Glynn 2000) which shows that logic conflict occurs especially during period of conceiving a new idea and the stage of delivering a product, a service or a performance. Additionally, the literature has addressed the need for specific forms of marketing practices in the context of software products (O'Mahony and Bechky 2008, Meeteren 2008) that warrants the inclusion of app marketing as the third stage of entrepreneurial activity. We discuss these three stages in detail later in the paper.

During the second stage of the analysis, our goal was to identify factors that shape professional and market logic for indie developers, based on information provided by the developers during the interviews. As part of this stage of analysis, we aimed to identify the dominant logic anchor for each developer on the three entrepreneurial stages identified above. Thornton et al. (2012) argued that individuals have multiple and often contradictory goals and therefore can dynamically access conflicting logics according to specific situations. Building on this argument, we do not presuppose that developers will display the same dominant logic anchor across all three entrepreneurial stages but allow these to differ, thereby departing from the strict

ideal-type approach observed in the logics literature (Thornton 2002, 2004). Thus our unit of analysis is the entrepreneurial practice rather than the individual. We iteratively identified the logic anchor and the corresponding practices through careful cross-developer and cross-entrepreneurial activity analysis, as well as an examination of similarities and differences in logics and practices (Miles and Huberman 1991).

In the third phase of analysis, we coded app entrepreneurs' logic synthesis. Whenever entrepreneurs conveyed a sense of change, shift, integration and compromise related to either of the two logics, we coded these as synthesis practices. We also concurrently developed working hypotheses about factors that drive or inhibit logic synthesis in any given situation. We constantly compared the emerging themes and hypotheses in subsequent data collection, analysis and extant literatures. Gradually, our codes reached a level of saturation where they were mutually exclusive and comprehensive (Miles and Huberman 1991). At the conclusion of the analysis, we arrived at the theoretical model, shown in Figure 1, incorporating the three entrepreneurial stages of activity, the two competing logics and synthesis practices. We discuss these findings in more detail next.

4. Findings

We first describe the general environment shaping professional and market logic for indie app entrepreneurs. Then we discuss the two logics enacted by app entrepreneurs within the three entrepreneurial areas. The structure and examples of this discussion are shown in Table 2. Next, we discuss the factors that inhibit logic synthesis. Lastly, we describe in detail the factors that promote logic synthesis in addition to specific processes of synthesis observed in our fieldwork and interviews.

4.1. Factors Shaping Professional and Market Logic for Indie App Entrepreneurs

The manifestation of professional logic in the indie app community is driven by factors similar to those observed in the software engineering profession. These include a focus on user-led innovation, training in software engineering principles and an emphasis on peer recognition. Software developers like to tinker and work on hobby projects at their spare time, similar to user innovators in open source communities who enjoy the freedom and creativity inherent in picking their own projects to work on (Von Hippel 1986, Shah 2006). Working on their own projects enables developers to avoid time pressures present in organizational projects where task deadlines exist and "shortcuts" may be adopted (Austin 2001). While personal interest determines what the developer chooses to work on, software engineering training dictates how the developer goes about writing this software. The IEEE Computer Society's Software Engineering Body of Knowledge defines software engineering as the "application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" (Abran, Moore, Bourque and Dupuis 2004:1-1). In addition, a focus on quality metrics, such as performance, capability, usability, and reliability (Kekre, Krishnan and Srinivasan 1995) become integral to professional logic. When consumer-oriented software is developed, issues related to software aesthetics (how the application looks, feels and sounds) (Garvin 1987) as well as user interface design become crucial, and hence relevant to professional logic, particularly for developers on the App Store since Apple is known for its emphasis on app graphics. Finally, an important element of professional logic is legitimacy that accrues from the peer community. Like many professional groups, such as editors and artists, where peer recognition and approval denote achievement

and status (Thornton 2002, Elsbach et al. 2008), the developer community values meritocracy and status attainment through community recognition (O'Mahony et al. 2008, Stewart and Gosain 2006).

In comparison to professional logic, factors shaping market logic for indie app entrepreneurs stem from two sources - fundamentals of market-based economies as well as App Store-specific market conditions. The fundamental elements of market logic in most market-based economies include developing customer and market knowledge, a focus on solving customer problems (Nag et al. 2007, Baron et al. 2006), understanding and managing budgets and financials (Glynn 2000, Nag et al. 2007), generating positive cash flow and operating within acceptable levels of risk (Baron et al. 2006).

The more profound impact on market logic here however, comes from the App Store environment. In a marketplace with around 700,000 apps (AppShopper, accessed October 2012), indie developers face tremendous competition on the market. Furthermore, the additional level of control exerted by Apple as the platform owner strongly influences market logic. Like owners of other software platforms, Apple develops the operating system and provides the software development kit (SDK) and APIs for third-party developers. What is different about Apple is that it also owns the exclusive distribution channel: the App Store. Thus, as a gate keeper, Apple determines which apps can be sold through its review process. While reviewing is intended primarily for quality assurance, at times it is also enacted for political reasons, leading to the process being perceived as inconsistent and nontransparent[2]. In direct contrast to the negative fallout from the review process, the platform-designed top charts and promotion features on the App Store help boost apps' sales

[2] http://techcrunch.com/2009/07/27/apple-is-growing-rotten-to-the-core-and-its-likely-atts-fault/

tremendously. Thus, app entrepreneurs enacting market logic are acutely aware of these App Store-specific levers and actively incorporate them in their entrepreneurial decisions, described next.

4.2. Professional and Market Logic Practices within Entrepreneurial Areas

App Ideation: Personal Needs vs. Mass Market Needs

App developers' practices and selected quotes in accord with professional and market logics in the three entrepreneurial areas are presented in Table 2. In the app ideation phase of the entrepreneurial process, developers make decisions on the types of applications to be developed and features to be included. A recurring theme from developers who identify with professional logic is the choice to write software that fulfills their own needs or relates to their own passion. In our interviews, entrepreneurs state that when they work on familiar domains or on apps they personally need, they tend to use the app more often and constantly improve it. Specifically, developers design the user interface they are satisfied with, create the flow of the app that best matches their use habits and also improve the innate engineering quality of the software by finding and fixing bugs since they interact with the apps frequently.

In contrast to serving personal needs, strategies aligned with market logic focus on developing apps with a mass market appeal. The rationale is that these apps are more likely to achieve wide adoption and subsequently climb onto top charts, which feed more downloads. App developers influenced by market logic also spend considerable time keeping track of the top-chart apps and researching current apps on the platform; using this information to drive the choices of apps they develop. Some developers would also practice "copycatting," i.e., learning from and mimicking existing hit apps. On the App Store, it is common to see "cheats" and "walkthroughs"

for top-ranking games, and apps with deceivingly similar names and content to hit apps. These strategies clearly conflict with professional logic, which stresses personal needs as a source of ideation and therefore might not result in apps that appeal to the mass market. In addition, further conflict arises when professional logic emphasizes personal use and continuous improvement while market logic is characterized by ideation driven by customer and market trends, which may not necessarily lead to continuous quality improvement. Thus, these two contrasting logics in app ideation each offer some benefits to nascent app entrepreneurs but conflict by emphasizing vastly different elements.

App Execution: Pursuit of Quality vs. Pursuit of Efficiency

The second stage of activity, app execution, is concerned with development effort on app release, maintenance, updates and app portfolio strategy. Developers anchoring on professional logic set high engineering standards before shipping apps. Participant DC stated:

"Our standards are pretty high. We believe in having well thought out, well tested apps with excellent ease of use and good documentation. An app is ready to ship when it is feature complete, well tested, and has no known serious bugs."

While software training institutionalizes the engineering aspect of the software, Apple's platform characteristics and culture concurrently influence developers' decisions on software design. Apple provides design guidelines for 3rd-party app developers and recognizes well-designed software with their annual design awards. Developers also look up to Apple products for design inspiration. These principles also influence software decisions, as described by developer NS:

"*When we release an app, we say, it's gotta have a beautiful user interface, it's gotta be intuitive, and it's gotta be crisp. When someone looks at it, they have to say "this looks nice". You also have to look at what Apple produces, and just say is this something Apple would release, does this look good enough to be an Apple product."*

Although quality is the focus of attention for developers influenced by professional logic, those identifying with market logic embrace the idea of efficiency and time to market. When fighting for visibility on the App Store is extremely challenging, many developers engage in quick development and try every means to claim a spot in a new platform or a less crowded app category. The rapid rate of change on the platform provides many opportunities for the agile developer to exercise a "rush" strategy. For instance, the introduction of the iPad allowed entrepreneurs the chance to quickly port existing apps to the new platform, even if the design of the app was better suited to the mobile phone and not the tablet. Furthermore, the unpredictability of app sales on the App Store nurtures the practice of excessive experimentation as a strategy; this involves frequent app launch and limiting the level of effort on each app launched. As a consequence, these apps could suffer from low reliability and usability. With regard to design, instead of creating graphics or user interface with attention to detail, which follows professional logic, market logic-influenced developers tend to purchase stock images for app icons and make do with crude aesthetics and UI design.

These app execution level strategies again highlight the tension between practices aligned with the two logics. While professional logic suggests polished app design and development as well as continual quality improvement, the uncertainty of sales on the App Store could easily render this practice costly and unrewarding. On the other hand, market logic execution suggests many quick apps, jettisoning apps that get no traction in the marketplace and quick updates, each of which saves valuable developer time and effort. These strategies however tend to fall short on core software engineering standards as well as Apple's design guidelines for third-party developers.

App Marketing: Organic Word of Mouth vs. Hit and Consumer-Oriented Marketing

Within the third stage, app marketing, developers make strategic decisions that concern spreading awareness of developed apps as well as focusing effort on supporting and communicating with end-users. Developers influenced by professional logic value peer recognition, i.e., they endorse each other's apps through social media channels such as blogs and Twitter, with a focus on word of mouth to enhance awareness. In the meet-up group where many of our observations were conducted, the organizer makes an effort to create a culture of supporting indie app developers. Developers demo new apps in face to face meetings and make announcements on mailing lists about new releases. Developers enacting professional logic also tend to extend such peer-acceptance marketing strategy to the seller-consumer relationship. Assuming that end users have the ability to discover a good product on their own, developers believe that customer acquisition will automatically follow. This leads to a relatively passive marketing approach. Developers thus rely on organic product discovery, and refrain from initiating communication with end users unless specific support needs arise. They also resist the idea of consumer marketing ideologically, interpreting it as superficial compared to more tangible work like programming and coding. Participant NS provided this viewpoint:

"My experiences have led me to believe that most of the time marketing is not helpful to a company. It feels all gimmicky to me, and maybe it's just as a developer, I'm susceptible to those things more than other people, but I notice gimmicks, like I can see all this person did is tweet this contest again where they were giving out a Macbook Air, great. Another person flooding my stream on Twitter of content I don't care about. We're trying not to be gimmicky, and we try to just produce an awesome product that hopefully people will love, and that's like our philosophy."

Contrary to organic word of mouth and passive user communication, developers influenced by market logic value market recognition, i.e., they make full use of platform policies in ensuring app visibility. Essentially, the objective here is to

introduce an app and achieve a “hit”, i.e., obtaining visibility through the top charts within the first few days, and extending the app’s top chart lifetime. In order to achieve this, developers aggressively use competitive pricing, versioning and advertising to start with. Subsequently, they release a stream of constant updates to the apps to encourage positive customer reviews. Finally, they try to increase the discoverability of the app through techniques such as in-app advertising, App Store Search Engine Optimization and cross-app promotions.

Another significant component of the approach used by market logic-based developers here is to actively communicate and establish rapport with users, overcoming challenges posed by design of the platform. Our interviewees stated that Apple’s governance mechanisms tend to limit direct access to consumers who download the apps, i.e., developers are not provided with user information such as email addresses. In addition, end-users in general lack knowledge about the relationships between third-party developers, the platform and additional service providers. Thus, it is not uncommon for app developers to receive negative reviews of their apps that are, in fact, due to issues with the operating system or other service providers. Recognizing this and realizing that direct communication with end-users may remedy user confusion and enhance trust, entrepreneurs have creatively found ways to circumvent the platform’s governance. These include directing consumers from the app description to their websites, social networking sites and influential blogs. Developers are also active participants in user forums, responding to and providing support to consumers directly. A positive outcome of such open communication is that relationships between developers and consumers reach a value co-creation stage (Nambisan and Baron 2007, Di Gangi and Wasko 2009) wherein

consumers suggest feature updates, evaluate such updates and even issue clarificatory information on behalf of the developer to other consumers.

For these marketing level strategies, the tension between the two logics is clear in terms of the ideology espoused by developers who subscribe to differing logics as well as the effects of these activities on eventual app downloads. The "organic word of mouth" approach observed within professional logic might not be sufficient in pushing apps to App Store top charts, which is critical for app sales. On the other hand, the "hit" marketing approach does not provide the developers with adequate legitimacy in the developer community. Consumer-oriented marketing is not popular among developers who value peer opinions, thereby leading to further conflict for developers who may try to balance the two approaches.

4.3. Logic Synthesis – Practices and Strategies

In this section, we move from describing logic practices to the process by which synthesis occurs. Recall that we proposed a broad definition of synthesis in the theory section. Here, we refine that definition. We propose that for logic synthesis to occur in the App Store context, either or both of the following conditions are met: first, app entrepreneurs are willing to engage in tasks they did not think were necessary or did not identify strongly to start with; second, entrepreneurs give up partly on activities that they identified strongly with. Synthesis, as per our arguments, occurs from both focal logics. For developers focused on professional logic, synthesis occurs in two ways. First, when concessions are made on practices consistent with the professional logic. Second, when elements of market logic are grafted onto existing practices. In both these cases, the developer is viewed to move to a more hybrid position with respect to the focal logics. A similar set of moves characterize synthesis for developers enacting market logic. We will discuss how developers synthesize

conflicting logics in the three entrepreneurial areas using this working definition. However, not all developers perceive the need to manage logic conflict in equal measure, so before delving into synthesis practices, we first delineate factors that drive or inhibit app developers' synthesis practices.

4.3.1. Antecedents of indie app developers' logic synthesis

Factors Inhibiting Logic Synthesis

We first discuss factors that *inhibit* logic synthesis, since these prevent eventual balance across logics, which is arguably important for success on the App Store. The literature indicates that *strong identification with professional beliefs and norms* inhibit reconciliation (Glynn 2000, Nag et al. 2007). In addition, our analysis uncovers that *relational capital from the Mac developer community* and developers' *labor market status* also tend to curb synthesis.

The first factor that inhibits synthesis is the extent to which the developer is invested in his or her focal logic, leading to disagreement or disapproval for the opposing logic. Developers thus become less likely to compromise on the focal logic and/or integrate components from the opposite logic. This effect is particularly strong in app execution. For instance, developers with strong logics in product substance and quality see no value in compromise. *"Releasing an app with just a couple of wallpapers?"* - developer TM was dismissive of a wallpaper app, which cost 99 cents without much substantive content or originality. In contrast, market-logic focused developers in app execution do not understand why some developers invest effort in one single app. Developer DS responded:

"I think there are a lot of people are more design purist, or engineering purist. Their goal is to make something beautiful, or make something cool, like technically, the code is really cool inside the way they do it. My goal in this is not like aesthetic, my goal is to make a living and support my family and support my employees' family

on the apps we make, so that changes our priorities, that's why we release things so quickly, we write simpler apps. It's not just developing for developing's sake" (DS).

Beliefs in focal logic also prevent developers from synthesis in app marketing. Developers who strongly believe in product quality and peer-marketing dislike the quick development cycle and the hit marketing strategy on the App Store. As developer JV stated:

"I don't want to have anything to do with it. I don't think it's common, not even possible to make 3 or 4 apps a year. I don't want to be dependent on the hits; you have to have shorter cycle. Hopefully you will make a good hit out of every cycle. If I take a year to develop an app, if it's not a hit, then I'm screwed."

Developer NS similarly opposed the idea of frequent feature update, a part of the hit marketing strategy, in favor of organic word of mouth based on substantial feature changes:

"I wanna be able to provide the features that people talk about. So on QB, a new version is coming up, 'oh did you see QB added Skype', 'no I didn't see that, it's great, I use Skype all the time', so that's part of the philosophy. And the other part of the philosophy is the burden of updating is on the user really. And if they see your app coming up every week, they'll be like, 'what's the point, why is it coming up every week or two?' like nothing is changing."

On the other hand, developers believing in market recognition and in-App Store marketing do not understand why some developers would market apps among other app developers. Participant JS shared his opinion:

"I've noticed people buy advertising on sites like iPhoneDevSDK. *I never understood that, unless you're making a product that's oriented towards developers. I don't see a lot of value in doing promotion of the applications to the developer community itself. Developers more than anybody are keenly aware of what's out there on the App Store, what's doing well and what's not. So I think it's way more important to be promoting a product to consumers than to developers."*

The second factor that inhibits logic synthesis is iOS developers' relational capital in the Mac developer community; the effect of this factor is most salient in app execution and app marketing. The Mac and iOS shares the foundation of the operating

system, so many Mac developers write iOS apps, too. The Mac indie software market is still largely dominated by personal capitalism, an instantiation of professional logic (Thornton 2001), where personal networks and personal relationship are valued in the business process (Meeteren 2008). Therefore, although it is advisable for Mac developers to compromise a little on product quality and adopt App Store-specific marketing techniques, they can afford not to. This is because the effort in pursing superior product design and quality is paid back with status in the Mac indie community rather than on the App Store. Tapping the mature network of this community is potentially rewarded with financial returns on iOS apps, as Developer RR indicated:

"The first thing I did is that I announced it in the Chicago C4 Conference in front of about 200 people. One of the tech bloggers for the Mac Space John Gruber was at this conference, he saw it, he linked it to his site, just from this guy, and at this weekend, I got 2000 hits for my site. So I think there are two different ways, there's I want to build a bunch of buzz about my app, trying to get in the top 10 of the App Store, and ride that wave out. And there's another that no, I'm going to focus my attention in building a good quality product. The idea is that if you focus on building something that's tight, that's quality, that has a good crafted experience, all you need to do is get one of these people in these big sites to use it, love it, and then write about it. And in that sense you're not propositioning people, and they are doing it for you."

The final factor that inhibits synthesis reflects the growing opportunities in the labor market for app developers. The App Store environment allows many software developers to create apps on a part-time basis, while still maintaining full-time positions in software development, similar to the open source model (Roberts et al. 2006). The rapid growth of the app economy, combined with the relatively high levels of glamour attached to the mobile platforms, has enhanced the status of visible and successful app developers significantly. These factors have created significant app consulting opportunities for app developers, especially nascent entrepreneurs who

have boot-strapped their own firms. These potential opportunities for consulting tend to dampen the need to synthesize conflicting logics, especially for developers enacting professional logic in app marketing. Developer JR, who was occupied with consulting work, lacked the time in improving communication with end users about his popular app WZ and stated:

"Now on the App Store it only shows the reviews for the current version. So for developers you really have to stay on top of that. WZ, last time I checked, there are 2 reviews for the current version. People are disgruntled because it lacks something that it doesn't claim to have. So people see those 2 reviews and it really impacts the downloads. If I were a better businessman, I would be pushing out a new update as soon as possible to get those bad reviews off, I would be doing everything in my power to get some reviews. I need to get those two off that page, I need to manage that. I'm not, and it's a problem".

Factors Facilitating Logic Synthesis

With regard to factors *facilitating* logic reconciliation, the literature identifies the effect of *entrepreneurial pressures* and the desire to *achieve economic success* (Jain et al. 2009, Eikhof et al. 2007, Tschang 2007). In addition, our analysis reveals that *entrepreneurial learning* and *knowledge acquisition need from community* also enhance synthesis. Entrepreneurial learning addresses entrepreneurs' adaptations in beliefs, practices and routines incrementally in response to feedback about outcomes over time (Levitt and March 1988, Huber 1991). Entrepreneurs acquire knowledge through direct experience – trial-and-error experimentation, or learning by doing process. Our fieldwork suggests that app entrepreneurs develop learning through interactions with the market, the platform and end-users, which induces synthesis. Community knowledge acquisition need denotes the communal knowledge-sharing characteristics of app entrepreneurs, viewing software developers as a community of practice (Brown and Duguid 1991), which again induces synthesis. In the next section,

we embed these factors in the discussion of logic synthesis across the three entrepreneurial areas. Our discussion follows the relationships shown in Figure 1.

4.3.2. The Effective Mechanisms of Synthesis

Synthesis in App Ideation: Inner / Outer Evaluations & "Eating Your Own Dog Food"

In the app ideation phase, logic tension manifests around whether developers release apps fulfilling personal needs or addressing market needs. Logic synthesis, by our definition occurs from both focal logics. For professional-logic driven developers, entrepreneurial pressure and entrepreneurial learning push them towards logic synthesis in app ideation, through the form of "inner" and "outer" evaluations. Market-logic driven developers synthesize, alternatively, through "eating your own dog food".

Inner evaluation is for idea selection, either by being self-analytical or through peer critiquing. In doing so, synthesizing developers give up their preferred views on design and engineering quality while reducing the potential risk of developing apps with no market appeal. Entrepreneurial pressures forced developer TM to think twice what to develop before starting the project:

"I think everyone has ideas. You need to be able to filter out your own ideas. I had A LOT OF ideas. And I look at them and I go, wait a minute, that one wouldn't have mass appeal, why would I do that. I mean I've definitely thought and definitely heard of ideas that I thought were good ideas, but you know, would a lot of people buy it, if I say no, then I just don't think it's really worth the effort." (TM)

Developer JN's first two apps did not gain as much traction as he had expected. He recognized that this may be due to his lone-ideation approach. This learning experience made him change his strategy to incorporate collective wisdom for future apps:

"For my next app, I'm going to get a couple of people, I think you can fool yourself about what's the best way to do things if you're the only one deciding that.

And if other people think your ideas aren't great enough to wanna work on them, they probably aren't that great. In any case, they're not gonna put in the extra hour. Examined by more than one peer advice will be really helpful."

Outer evaluation synthesis consists of leveraging users' input through testing before releasing the app to the general public. Engaging users early on also requires developers to modify the software if needed, leading to changes in the execution plans. However, user feedback at this stage can improve software usability and gain customer loyalty from early adopters. Developer RR shared this viewpoint:

"Testing is really the counterbalance between developer's needs and consumers'. Think about who you wanna sell it to, and get it in front of those people as soon as possible, because you won't be able to guess what they want, they will always surprise you. You want this, gotta be this way. They will tell you the other way....feedback is the key."

For developers enacting market logic on the other hand, entrepreneurial pressure drives synthesis. Several participants emphasized that while it is critical for an app to address a market need, it is also important to build something that they like or find useful. This synthesis is an instance of grafting elements from professional logic. Doing so leads to systematic use of the app by the developer himself (similar to a user), which allows bug and usability fixes early to improve quality. Developers refer to this as "eating your own dog food", as stated below:

"We're doing a Calendar replacement. I'm using this as my main app for calendar, obviously looking for bugs and stuff, but I'm kind of living out the bugs so that my customers eventually don't have to." (KY)

Synthesis in App Execution: Emotional Detachment with Own Technology & Increased Emphasis on Design

In the app execution phase, developers face the conflict of emphasizing their business based on efficiency or quality. Following our two-way synthesis definition, we found that developers anchoring on professional logic synthesize through

detaching emotions with their own technology, and those anchoring on market logic achieve synthesis through increasing attention on app design.

Entrepreneurial pressure and entrepreneurial learning motivate developers identifying with professional logic to synthesize, and this synthesis is manifested in app development, app release and app update decisions. During app development, synthesis occurs around developers' decisions to build apps from ground up versus leveraging existing technologies. The latter option helps increase development efficiency, allowing developers to achieve synthesis through grafting elements from market logic. Developer TM learned the lesson on his very first app. He wrote the building blocks of the app all by himself when he could have saved effort and time using frameworks provided by Apple. In some cases, writing every aspect of an app from scratch is short-term efficient since adopting new APIs or functionality requires mastering new material. However, not leveraging well-maintained and widely adopted technologies can incur high in-house maintenance costs later on. Logic synthesis here indicates that the entrepreneur relinquish the "not-invented-here" mentality and leverage technologies from the larger developer community to achieve business efficiency.

During the software release stage, developers face the question of when to stop development and publish the app on the market. Professional logic would suggest that the app be published when it is polished and of high quality. However, this logic ignores the financial cost of working on the software beyond the optimal point of release. This question is common in product development contexts, where there may be differences between manufacturing and marketing on product release dates (Tatikonda and Montoya-Weiss 2001). In the App Store context, synthesis occurs

when developers opt to forego new features in order to ship the app sooner, as stated by developer NL below:

"Like the WM application, it was pretty much done, and I was just working on supporting rotations, trying this way, this way, it doesn't rotate correctly. I spent like 2 weeks working on it. At the end, I'm like, you know we had to cut out rotation …I wanna ship things that are good, but sometimes you have to see the forest through the tree. Sometimes you focus so much on this one little thing, and then you look back and you just spent thousands of dollars trying to get this little one thing right. So maybe think bigger, more strategically."

While NL's synthesis in cutting features was driven by entrepreneurial pressure, developer SJ was driven by learning from a trademark dispute he had on an earlier app. The dispute was finally settled with SJ selling his app name, but the incident took 6 months to resolve. Recognizing the potential risks of losing the app name and being dragged into a trademark dispute again, for the next app, SJ decided to leave behind features that were not as important and get the app on the market fast:

"Basically the first person who has it in commerce, they are the first person who gets the name. And if somebody names their app PP tomorrow and I hadn't put it out, and also I gotta change my name and everything. So, and again it's all because of the trademark experience I had with MQ, and so I was like, you know what, I gotta put it out. It doesn't matter what it looks like. I just gotta get the name. So what I did, I ripped off all the voice-over stuff, and there are some screens, they didn't look so good going out, but they didn't need to be there, so I ripped out whatever didn't look good, just to kind of put it out, and people loved it."

In addition to software development and release, synthesis also takes place during software update decisions. Here, synthesis generally indicates a "move-on" attitude, wherein developers opt to stop enhancing an existing app with limited market demand, even though professional logic would suggest these improvements. Developer JN shared his experience:

"I spent a lot of time to get network work properly after the initial release, because now people can compete head to head, and it's something people will like. And it was really difficult... so I spent A LOT of time getting it to work right, and it works really well, but almost no pump in sales, something I thought people are going

to appreciate, they don't really appreciate that much...was very large time investment for very little return."

Developers identifying with market logic also engage in synthesis practices in app execution, although for different reasons: their experience with platform policy changes and user interactions. Specifically, they move away from the "rush" strategy and invest resources in product design, detailed UI and the aesthetics of the app, most of which would not be necessary under pure market logic. Developer TM's original strategy "*was to release as fast as you can, and update as fast as you can."* He stated that "*If I had a few bugs, I'd just send it up there; I would cut corners and constantly add features."* However, in November 2009, Apple changed a key policy on the App Store: it began to only allow the first version of the app to appear on the release chart[3] instead of every new version. This change put crudely engineered apps at risk since user response to the first version became more critical than ever for apps' subsequent performance. Interviewees in this study could not emphasize enough that post-policy change, their apps only got one shot at being viewed on the new-release chart. For market-logic driven developers, the new policy necessitated a process of synthesis, as indicated by developer TM: "*I do more initial develop; you need to spend more time, polish it up, test it, market it, appropriately."* Whereas previously the goal was constant appearance among consumers, now TM aims at making a splash on the initial launch. Placing emphasis on design has its own positive impact on app download, achieved through platform featuring:

"I mean everything Apple features is something that has a lot of polish on it. They go above and beyond the bare minimum. And I've produced apps at bare minimum, and they never get featured. And then this GB was the first game ever featured of mine. I had a lot of games, but this is the first time I actually spent all my time on polish, making it pretty, adding in dancing, the glowing balls are dancing to

[3] http://www.iphonedevsdk.com/forum/business-legal-app-store/32936-dark-tidings-updates-may-not-appear-release-date-list-anymore.html

music; that's not necessary, that's just the polish that people like. And I think that's what Apple looks for." (TM)

Besides platform influence, end-user interactions are another source of learning that motivates developers solely focused on business outcomes to consider software quality, design and aesthetics, thereby enabling synthesis. Developer KR's recording app is popular thanks to its mass market appeal and powerful features. However, the design of the icon and user interface attracted a number of user complaints. As a result, KR decided to give the app a new look:

"*I did everything one my own. Now that I'm making more money, I think I'm gonna hire a graphic artist to clean up the user interface, change all the graphics and design better icons. I've got several reviews saying that the user interface is kind of clunky."*

To some extent, the App Store environment itself is dialectic. On one hand, competition and sales unpredictability prevent app developers from investing in too much polish. On the other hand, the platform and its users reward effort on apps' aesthetics and design. Therefore, a broad synthesis strategy at least for app development and app release, is to make *"simple yet polished apps"* (developer TM), or *"a fully-polished product that doesn't include all of the features you expect to include in the product eventually"* (developer JS).

Synthesis in App Marketing: Peer Partnership, Niche Marketing & Peer Community Contribution

In the app marketing stage, developers face the tension of whether to count on organic word of mouth for marketing or adopt the "hit" and consumer-oriented marketing strategy. Again, as synthesis occurs from two directions, our fieldwork and interviews uncovered that professional logic-driven developers synthesize through engaging peer partnership and niche marketing, and market logic-driven developers synthesize through peer community contribution.

For professional-logic oriented developers, entrepreneurial pressure highlights the importance of within-App Store marketing to achieve visibility and sales. One synthesis strategy employed is partnership with peers who have popular apps on the App Store. In doing so, developers still rely on support from the developer community – an instantiation of logic of the profession, however, the partnership allows their product to reach the mass market on the App Store, which is in accord with logic of the market:

> *"One of the things we partner with FS company who has a significant presence in the market place is they have a network of head-to-head play games, so by partnering with them, we can sort of use their existing distribution network, and their network of players, as long as the game is of similar quality, and meets the interests of their customer base, we should be able to be successful, should be able to step on their foundation." (DC)*

Synthesis in app marketing is influenced by entrepreneurial learning as well. Developers whose apps compete in content with Apple's native apps face legitimacy issues among both the developer community and the general consumer base. App Store legitimacy becomes particularly problematic among peers who identify strongly with Apple's culture, or those who view stepping on the platform's turf as risky behavior. Developer RH wrote a music play-list app intended to replace native functionality of Apple's iPod. This move led to dissenting voices in the developer community, suggesting that the developer *"doesn't get the platform"* and that *"You don't redesign Apple's stuff."* When sharing the app with his developer peers, RH did not receive the support he expected. Peer recommendation turned out not to be a viable marketing tool in this case. Based on this experience, RH decided to target his app to hardcore music fans only. Even though the niche marketing reduced the sales potential, the app received very positive reviews on the App Store. In general, while the platform encourages open innovation, the professional logic-driven developer is reminded that there are limits to openness, especially in garnering support from peers

who identify strongly with the platform. An alternative consumer-oriented niche marketing strategy is preferred here.

Market-logic driven developers engage in synthesis during app marketing through peer community contribution. This allows developers to integrate peer recognition and feedback with their logic's emphasis on market recognition. Software development as a profession relies on teams and communities of practice (Faraj and Sproull 2000, Von Krogh et al. 2003, Crowston et al. 2006) for knowledge sharing due to the range of technologies involved and the pace of technological advancement. Several participants in our study mentioned that the indie work style does not allow colleagues to exchange knowledge on a regular basis, therefore joining a peer community and accessing peer knowledge becomes particularly critical for business success. A good example of this type of synthesis is Developer DS, who pursues a hit-oriented marketing strategy by focusing on App Store rankings and in-store visibility. In the meantime, he contributes extensively to the iOS community locally and virtually. He holds "Office Hours" to invite local developers to join his office space to chat about work and documents his iOS development and business experience on his blog. Thus, DS does not compromise on market logic in marketing but extends his practices in accord with professional logic. Although DS's philosophy on app marketing differs considerably from his peers, he earns positive word of mouth for his contribution to the community, thereby increasing his marketing effectiveness.

5. Discussion and Conclusion

How do nascent app entrepreneurs address the conflicts they experience from trying to balance their professional practices as software developers, on one hand, and the need to effectively manage a competitive marketplace, on the other? How are these conflicts dealt with in different entrepreneurial areas? These questions formed

the basis for the analysis we present in this paper. The conflict between professional and market logics has been studied in the literature over time, as the institutional discourse shifts (Thornton 2002, 2004), or between social actors with different group and organizational identities (Glynn 2000, Nag et al. 2007). In our context, we study how these conflicts are contemporaneously managed within the same app entrepreneur via the concept and process of logic synthesis. Through a qualitative study, we develop a deeper understanding of how both logics can occupy an app entrepreneur's decision portfolio and compete for the entrepreneur's attention. We show the different mechanisms of and conditions for the *two-way logic synthesis*. Our work thus contributes to the literature on the relationship between individuals and institutional logic, as well as nascent entrepreneurship.

Two broad sets of factors shape the professional and market logic guiding decisions and strategies for indie app entrepreneurs. They are 1) software developers' professional training, their interest in tinkering and the nature of community of practice, as well as 2) Apple's new organizing form as both a technological platform and an exclusive distribution channel. Since both of these logics act on the same indie app developers, the setting allows us to investigate these unfolding dynamics of logic synthesis at a level of granularity that is rare in the institutional logics literature.

Our study identified three entrepreneurial areas in which logics get expressed: app ideation, execution and marketing. We find that practices in accord with professional logic include "ideating" new apps through personal needs and passions, pursuing high engineering standards and employing peer recommendation to market their apps. Practices in accord with market logic, on the other hand, entail reacting to mass-market needs and trends, following an efficiency-oriented strategy as well as a hit and consumer-oriented marketing strategy. Our findings suggest that the tension is

stronger in app execution and app marketing than in app ideation for app developers because of the markedly different meanings attached to app quality and variety of ways in which an app may be discovered, sought or marketed on the App Store. While app ideation is important, the market-oriented activities such as marketing and development provide more observable instances of logic conflict that need synthesis.

We observe that most developers have a starting logic: that of professional or market, in all three entrepreneurial areas at the inception of writing apps for the App Store. Then depending on specific circumstances the entrepreneur experiences, some aspects of the logic portfolio become more salient than others, and thus s/he can synthesize in one area but not others or in none at all. Although engaging in logic synthesis would be ideal for all entrepreneurs, we do not claim it happens in all circumstances, nor do we observe this in our fieldwork. We found that three hurdles are present for logic synthesis: beliefs in focal logic, relational capital from Mac developer community and indie developers' labor market status. In contrast, three other factors: entrepreneurial pressure, entrepreneurial learning, and knowledge acquisition need in community of practice facilitate logic synthesis. These synthesis drivers, to some extent, echo the socialization mechanism necessary for individuals to learn multiple contrasting and often contradictory institutional logics (Thornton et al. 2012). Entrepreneurial pressure and entrepreneurial learning are especially relevant to socialization, because they serve as two conduits that help developers understand and familiarize the different requirement and characteristics of the App Store market place, the platform culture and software engineering principles. Without this constant pressure to achieve success or to learn from the marketplace, most entrepreneurs would find it very hard to operate in the fast-moving App Store environment. Thus, these two factors are of particular importance to app developers.

The concept of logic synthesis proposed in our study entails either or both of *focal logic compromise* and *opposite logic grafting*. We extend the literature by identifying practices of two-way logic anchoring and synthesis, whereas extant literature tends to mostly recognize reconciliation only from professional to market logic. Our findings suggest that for indie app developers, the key to success on the App Store is to constantly consider strategies congruent with both the professional and market logic and synthesize from both directions. Besides, since conditions on the App Store constantly change either due to changing competition or platform policy changes, different entrepreneurial activities can become salient at different times, which shape developers' goals and actions. This suggests that indie app developers need to engage in logic synthesis in a dynamic, rather than static fashion, in order to remain relevant and stay competitive.

Synthesis practices carried out by developers in our study are informative to other entrepreneurs who are considering entry into the competitive app economy. While low entry barriers and the lure of "hits" increases competition, our results suggest that a balanced and dynamic approach to both institutional logics on the App Store is a better position for success. Findings from our research also have implications for nascent entrepreneurs in other professional arenas building businesses around a central platform. Examples include photographers selling photos through *istockphoto.com* and artists selling craftwork on *etsy.com*. While the professional logic and market logic vary for each of these markets, the underlying relationship between the two logics and logic synthesis strategies identified in this research can provide a framework for studying professions beyond software development.

Our work also points to several directions for future work in this context. When we delineated entrepreneurial learning as a driving force for developers' synthesis

practices, we did not include learning from peers as one aspect. Social interactions with other developers could be an important influence on developers' decisions and formation of synthesis practices, which we aim to incorporate in future research. We have argued for the value of logic synthesis but whether this is true across Apple's App Store, or indeed on any mobile platform, i.e., the Android, is an empirical question. It is also possible that one of the antecedents to synthesis is competition – competitive categories on the platform likely will incentivize synthesis compared to categories with lesser competition. While our codes did not address this directly in our study, competition remains a potent driver of entrepreneurial behavior. It is often assumed that larger firms active in the marketplace have resolved logic conflict by empowering market logic (Thornton 2002); however, is this necessarily true in environments where there is a powerful gate-keeper such as Apple? Our focus here is on nascent entrepreneurs but there are several large corporations also competing on the App Store. How does existing research on logic conflict translate to these firms on platforms? There are many such questions that can be addressed on mobile technology platforms and it is our hope that our work will contribute to and spark off more interest in the topic of institutional logics in the technology platforms context.

References

Abran, A., J. W. Moore, P. Bourque, R. Dupuis. 2004. *Guide to the Software Engineering Body of Knowledge*. SWEBOK. IEEE Computer Society Press

Adler, P. A., P. Adler. 1987. *Membership Roles in Field Research*. Sage. Thousand Oaks, CA

Austin, R. D. 2001. The effects of time pressure on quality in software development: An agency model. *Inform. Systems Res.* **12**(2) 195-207.

Baron, R. A., M. D. Ensley. 2006. Opportunity recognition as the detection of meaningful patterns: Evidence from comparisons of novice and experienced entrepreneurs. *Management Sci.* **52**(9) 1331-1344.

Boudreau, K. 2012. Let a thousand flowers bloom? An early look at large numbers of software app developers and patterns of innovation. *Organ. Sci*. **23**(5) 1409-1427.

Boudreau, K. 2010. Open platform strategies and innovation: Granting access vs. devolving control. *Management Sci*. **56**(10) 1849-1872.

Bourdieu, P. 1990. *The Logic of Practice*. Polity Press, Cambridge, UK.

Bourdieu, P. 1993. *The Field of Cultural Production*. Columbia University Press, New York, NY.

Bourdieu, P. 1996. *The Rules of Art: Genesis and Structure of the Literary Field,* trans. Susan Emanuel. Polity Press, Cambridge, UK.

Brown, J. S., P. Duguid. 1991. Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation. *Organ. Sci.* **2**(1) 40-57.

Brown, J. S., P. Duguid. 2001. Knowledge and organization: a social-practice perspective. *Organ. Sci.* **12**(2) 198-213.

Brown, S. L., K. M. Eisenhardt. 1995. Product development: Past research, present findings, and future directions. *Acad. Management Rev.* **20**(2) 343-378.

Crowston, K., K. Wei, Q. Li, J. Howison. 2006. Core and periphery in Free/Libre and Open Source software team communications. *Proc. 39th Hawai'i International Conference on System Science,* Poipu, Kauai, HI, USA.

Cyert, R. M., J. G. March. 1992. *A Behavioral Theory of the Firm*, *2nd edition*, Blackwell, Cambridge, UK.

Davidsson P., B. Honig. 2003. The role of social and human capital among nascent entrepreneurs. *J. Business Venturing* **18**(3) 301-331.

Dunn, M. B., C. Jones. 2010. Institutional logics and institutional pluralism: the contestation of care and science logics in medical education, 1967-2005. *Admin. Sci. Quart.* **55**(1) 114-149.

Ebaugh, H. R. F., 1988. *Becoming An Ex*. University of Chicago Press, Chicago.

Eikhof, D. R., A. Haunschild. 2007. For art's sake! Artistic and economic logics in creative production. *J. Organ. Behavior* **28**(5) 523-538

Elsbach, K. D., F. J. Flynn. 2008. Issues of identity in collaborations among creative professionals: a study of toy designers. Working paper. UC Davis.

Faraj, S., L. Sproull. 2000. Coordinating expertise in software development teams. *Management Sci.* **46**(12) 1554-1568.

Freidson, E. 2001. *Professionalism: The Third Logic.* University of Chicago Press, Chicago, IL.

Friedland, R., R. R. Alford. 1991. Bringing society back in: Symbols, practices, and institutional contradictions. W. W. Powell, P. J. DiMaggio, eds. *The New Institutionalism in Organizational Analysis*, University of Chicago Press, Chicago, IL, 232-263.

Garvin, D.A. 1987. Competing on the eight dimensions of quality. *Harvard Business Rev*. November-December 101-109.

Glaser, B., A. Strauss. 1967. *The Discovery of Grounded Theory: Strategies of Qualitative Research.* Wiedenfield and Nicholson, London.

Glynn, M. A. 2000. When cymbals become symbols: Conflict over organizational identity within a symphony orchestra. *Organ. Sci.* **11**(3) 285-298

Huber, G. P. 1991. Organizational learning: the contributing processes and the literatures. *Organ. Sci.* **2**(1) 88-115.

Jain, S., G. George, M. Maltarich. 2009. Academics or entrepreneurs? Investigating role identity modification of university scientists involved in commercialization activity. *Res. Policy* **38**(6) 922-935.

Kekre, S., M. S. Krishnan, K. Srinivasan. 1995. Drivers of customer satisfaction for software products: Implications for design and service support. *Management Sci.* **41**(9) 1456-1470.

Lampel, J., T. Lant, J. Shamsie. 2000. Balancing act: learning from organizing practices in cultural industries. *Organ. Sci.* **11**(3) 263-269.

Levina N., E. Vaast. 2005. The emergence of boundary spanning competence in practice: Implications for implementation and use of information systems. *MIS Quart*. **29** (2) 335-363.

Levina N., E. Vaast. 2008. Innovating or doing as told? Status differences and overlapping boundaries in offshore collaboration. *MIS Quart*. **32**(2) 307-332.

Levitt, B., J. G. March. 1988. Organizational Learning. *Ann. Rev. Sociol.* (14) 319-340.

Louis, M., 1980. Surprise and sense making: What newcomers experience in entering unfamiliar organizational settings. *Admin. Sci. Quart*. **25**(2) 226–251.

MacMillan, D., P. Burrows, S. E. Ante 2009. Inside the app economy. *Bloomberg Businessweek* October 22, 2009.

Meeteren, M. V. 2008. Indie fever: the genesis, culture and economy of a community of independent software developers on the Macintosh OS X platform. *Bachelor thesis*. Human geography, University of Amsterdam.

Miles, M., A. M. Huberman. 1991. *Qualitative Data Analysis, 2nd edition*, Sage, Beverly Hills, CA.

Nag, R., K. G. Corley, D. A. Gioia. 2007. The intersection of organizational identity, knowledge, and practice: attempting strategic change via knowledge grafting. *Acad. Management J.* **50**(4) 821-847.

O'Mahony, S., B. A. Bechky. 2008. Boundary organizations: enabling collaboration among unexpected allies. *Admin. Sci. Quart.* 53(3) 422-459.

Orlikowski, W. J. 2000. Using technology and constituting structures: a practice lens for studying technology in organizations. *Organ. Sci.* **11**(4) 404-428.

Ozcan, P., K. M. Eisenhardt. 2009. Origin of alliance portfolios: entrepreneurs, network strategies, and firm performance. *Acad. Management J.* **52**(2) 246-279.

Roberts, J. A., I.-H. Hann, S. A. Slaughter. 2006. Understanding the motivations, participation, and performance of open source software developers: a longitudinal of the Apache projects. *Management Sci.* **52**(7) 984-999.

Ross, L., R. E. Nisbett. 1991. *The Person and the Situation*. McGraw-Hill, New York, NY.

Schultze, U., W. J. Orlikowski. 2004. A practice perspective on technology-mediated network relations: the use of Internet-based self-serve technologies. *Inform. Systems Res*. 15(1) 87-106.

Shah, S. K. 2006. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Sci.* **52**(7) 1000–1014.

Srinivasan, A., F. Suarez. 2010. First mover advantages in hyper-dynamic environments: a study of the iPhone ecosystem. *Presented at Acad. Management Conf.* August 6-11, Montreal, Canada

Stern, S. 2004. Do Scientists Pay to Be Scientists? *Management Sci.* **50**(6) 835-853

Stewart K., S. Gosain. 2006. The impact of ideology on effectiveness in open source software development teams. *MIS Quart.* **30**(2) 291-314.

Tatikonda, M. V., M. M. Montoya-Weiss. 2001. Integrating operations and marketing perspectives of product innovation: the influence of organizational process factors and capabilities on development performance. *Management Sci.* **47**(1) 151-172

Thornton, P.H., W. Ocasio, M. Lounsbury. 2012. *The Institutional Logics Perspective: a New Approach to Culture, Structure and Process*, Oxford University Press, USA.

Thornton, P. H., W. Ocasio. 2008. Institutional Logics. Greenwood, R., C. Oliver, R. Suddaby, K. Sahlin-Andersson, eds. *The Sage Handbook of Organizational Institutionalism.* Sage Publications Ltd., 99-129

Thornton, P. H., C. Jones, K. Kury. 2005. Institutional logics and institutional change in organizations: transformation in accounting, architecture, and publishing. C. Jones, P. H. Thornton, eds. *Transformation in Cultural Industries (Research in the Sociology of Organizations)* Emerald Group Publishing Limited, 23 125-170

Thornton, P. H. 2004. *Markets from Culture: Institutional Logics and Organizational Decisions in Higher Education Publishing*. Stanford University Press, Stanford, CA.

Thornton, P. H. 2002. The rise of the corporation in a craft industry: conflict and conformity in institutional logics. *Acad. Management J.* **45**(1) 81-101

Thornton, P. H., W. Ocasio. 1999. Institutional logics and the historical contingency of power in organizations: Executive succession in the higher education publishing industry, 1958–1990. *American J. Sociol.* **105**(3) 801–843.

Thornton, P. H. 2001. Personal versus market logics of control: a historically contingent theory of the risk of acquisition. *Organ. Sci.* 12(3) 294-311.

Tschang, F. T. 2007. Balancing the tensions between rationalization and creativity in the video games industry. *Organ. Sci.* **18**(6) 989-1005.

Von Hippel, E. 1986. Lead users: a source of novel product concepts. *Management Sci.* **32**(7) 791–805.

Von Krogh, G., S. Spaeth, K. R. Lakhani. 2003. Community, joining, and specialization in open source software innovation: a case study. *Res. Policy* **32**(7) 1217-1241.

Voronov, M., D. De Clercq. 2007. When art and commerce unite: from separate worlds to blurry boundaries and impression management. Working paper. Brock University.

Wagner, J. 2007. What a difference a Y makes-female and male nascent entrepreneurs in Germany. *Small Business Economics*. **28**(1) 1-21.

Wasserman, T. 2010. Software Engineering Issues for Mobile Application Development. *Proc. FSE/SDP Workshop on Future of Software Engineering Research*, ACM, New York, NY, 397-400.

Yeow, A., S. Faraj. 2011. Microprocesses of healthcare technology implementation under competing institutional logics. *32nd Internat. Conf. Inform. Systems (ICIS), Assoc. Inform. Systems,* Shanghai, China

Yoo, Y., K. J. Lyytinen, R. J. Boland Jr., N. Berente. 2010. The next wave of digital innovation: Opportunities and challenges: A report on the research workshop "Digital Challenges in Innovation Research." Working paper, Temple University, Philadelphia. http://ssrn.com/abstract=1622170.

Table 1 Data Sources

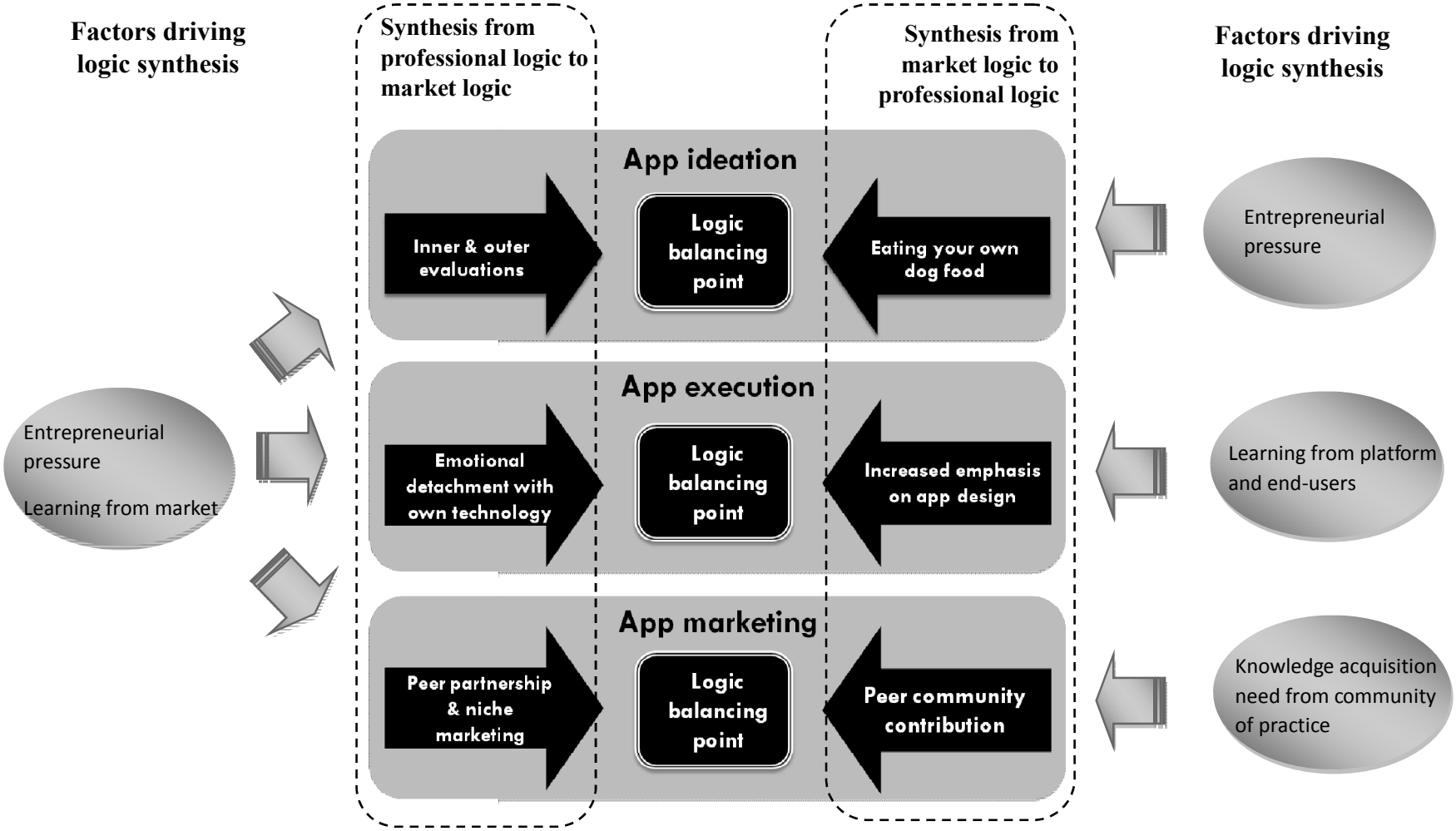
Data source	Description
Field observations	29 visits to multiple meet-ups of Mac and iOS developers and events for mobile entrepreneurs in the Middle Atlantic Region, totaling over 90 hours
Interviews	26 semi-structured face to face interviews and multiple informal chats with 19 iOS indie app entrepreneurs who have released at least 1 app on the App Store
Online resources	Company websites, blogs, user forum, Facebook fan pages of the 19 entrepreneurs
Participant observation	10-month internship at a small mobile app company; working on tasks related to market research, marketing and technology support

Table 2 Professional and Market Logic Practices of iOS App Entrepreneurs

Entrepreneurial areas	Practices of professional logic	Examples	Practices of market logic	Examples
App ideation	• **Building apps reflecting personal needs and passion**	*"There is this one person says that not everybody is gonna design your way, maybe you should survey how different designers do work, you should come up with some mixture of process. I say no. I feel that to build a quality product, I have to build something that I wanna use, and I love using it…"(RR)* *"That application was developed by me and for me, and that has been tremendously successful. Caz that's my way I bicycle, I use it all the time, interval, music, for the intensity I want, so it's fulfilling a need." (JB)*	• **Addressing mass market needs** • **Following market trend**	*"We had a good number of downloads from our alarm clock app, because people want that vs. some niche little app, that maybe can get big traction, but it's harder to determine if that's the case" (KY).* *"For NT, it came from the fact that I saw the MT being number 1 for quite a while, so I wanted to create a knockoff. The idea was I wanted to look for something that's really high in the charts and really easy to implement (KR).*
App execution	• **Pursuing engineering and design quality**	*"I consider myself as an engineer. Engineers are trained in a much more systematic way to produce code and solve problems. I've had a couple of encounters with people that I consider to be coders and not engineers. It sounds a little bit elitist, but it's just a matter of training." (JN)*	• **Time to market** • **Excessive experimentation**	*"There's not that many iPad apps now, there's still a rush to get what's called universal apps or ipad apps, so you can get visibility again" (TM)* *"So if we launch something and it doesn't do great, I will just move on. These are just like we spend a week or we spend 4 days on something. If it doesn't take off, it's fine, I learned to build something new that I didn't know how to build before, I'll try the next thing" (DS).*

App marketing	• **Peer recommendation** • **Passive consumer marketing**	*"The biggest thing that I do is I write official blog entries to make it convey that there are serious developers behind this, so anybody that does come to the website, it looks like a professional website." (JR)*	• **In-store, in-app, hit-oriented marketing** • **Winning over users from the platform**	*"We found the most effective approaches are the approaches that are directly actionable by users, where they could click and download the app" (JS).* *"Once you build an app, it kind of turns into a monster. It's popular. It's your baby until you release it. And it's the world that decides what you should do with it. And if they want something, you gotta put it in there" (TM).*

Figure 1 Independent iOS App Entrepreneurs' Two-way Logic Synthesis



Essay 2: From invisible hand to visible hand: platform governance and institutional logic of independent Mac developers, 2001-2012

Abstract

In the research of institutional logics, field-level logics have continuously gained interests among institutional scholars. A cultural emergence model of field-level logics was proposed in the latest development of the institutional logics perspective (Thornton, Ocasio and Lounsbury 2012). This study aims to validate a section of the model: the relationship between resource environment and emergence and evolution of field-level logics, and do so in the context of Apple's desktop developers – Mac indies. I examine a critical platform governance change from Apple – its opening of the iOS App Store and subsequently the Mac App Store, and hence its role shifting from mainly a technological platform to a platform that includes a market exchange place, and dissect the content of Mac developers' institutional logic before and after the change in resource environments. Through a qualitative interpretive study, and a combination of narrative and content analysis, I show that a software ecosystem logic prevailed for Mac developers prior to the opening of the App Store, and a platform ecosystem logic emerged after that. For software developers, two layers of resource environments are present – platform governance and developers' own economy. Together, they influence software developers' institutional logic through both material practices and symbolic meanings. Two ideal types are constructed for the logics along elemental categories, and a content analysis demonstrates the logic shift pattern as resource environments change. A further analysis of the two logics suggests that the software ecosystem logic and platform ecosystem logic are in contestation at this early stage of institutional change. This study has implications for research in institutional logics and platform governance.

Keywords: field-level institutional logics; resource environment; platform governance; independent software developers; qualitative research; ideal types

Introduction

During the past two decades, the institutional logic theory has undergone substantial theoretical and empirical development. Originally established in the seminal work by Friedland and Alford (1991), it has been a great tool in responding to the critique on the neo-institutional theory by DiMaggio and Powell (1983, 1991). The criticism on the latter is that the main concept of the theory: institutional isomorphism and organizational homogeneity "proffers a rather monolithic or unitary concept of the environment and the legitimacy of institutional myths, and with this an implicit overtone of the legitimacy of conformity to change" (Townley 1997: 262). Institutional logic perspective in contrast, argues that institutional environments are pluralistic, and the society is composed of a set of interdependent and yet contradictory interinstitutional logics, each with differing belief systems and sources of rationality (Friedland and Alford 1991). Broadly conceptualized as the "socially constructed organizing principles that shape individual preferences and organizational interests as well as the repertoire of behaviors by which interests and preferences are attained" (Friedland and Alford 1991: 232), institutional logic has been instrumental in explaining the heterogeneity in organizational decisions and routines (Thornton and Ocasio 1999, Thornton 2002, 2004, Lounsbury 2002, 2007), contestation and resistance behavior at organizational and individual levels (Townley 1997, 2002, Reay and Hinings 2005, Marquis and Lounsbury 2007, Qiu, Gopal and Hann 2012), and change and evolution of organizational fields (Nigam and Ocasio 2010, van Gestel and Hillebrand 2011).

Following the characteristic of nested multi-levelness of the theory, Thornton and colleagues developed the concept of institutional logic at the level of the industry or field

(Thornton and Ocasio 1999, 2008, Thornton 2002, 2004). In the recent advancement, Thornton, Ocasio and Lounsbury (2012) articulate a cultural emergence model of formation of field-level institutional logics, and explicate the mechanisms through which cross-level effects operate. Taking a linguistic approach, the model illustrates how societal level and external logics as well as resource environment form field-level logics through shaping both the material practices and symbolic representations in the field. While the theoretical relationships and processes are documented in the model, more empirical studies are needed for model validation. In addition to logic emergence, the model also theorizes the change and evolution of the field-level institutional logic, and the authors urge that great care must be exercised in future research in expressing what aspects of the field-level practices are changing and how changing practices reflect changing symbolic meanings and institutional logic (Thornton et al. 2012: p169).

The objective of this study is to particularly investigate the effect of resource environment on the emergence and change of field-level institutional logic in the field of consumer software industry, and particularly of Apple's desktop platform and its independent third-party developers. Resource environment, according to Thornton et al. (2012) can include market and other forms of governance, such as governments, corporations, and information networks (p157). Scholars have identified that market conditions such as consumer demand and resource competition (Thornton 2002, Rao 1998), and public regulatory practices such as state or provincial public policies (Reay and Hinings 2005, 2009, Townley 1997, 2002), and professional associations (Greenwood, Suddaby and Hinings 2002) are forms of resource environment, which provides opportunities for the emergence of new institutional logics. Although literature

has implied that resource environment can constitute multiple forms or events for an institutional field (e.g. Hoffman 1999); in the field of software industry there is a hierarchical relationship between different parts of the resource environments, and an understanding of that can provide us with better knowledge about formation and change of field-level institutional logics.

Third-party developers are software platform's external innovative assets, who build complementary applications based on platform's core technologies (Tiwana, Konsynski and Bush 2010, Gawer 2010). Together the collection of the platform, third-party developers, their complementary creations for the platform, and the users formulate the ecosystem around a given platform (Tiwana et al. 2010, Gawer 2010, Cusumano 2010). Most software platforms exercise governance and control over developers (Linux being an exception) to various degrees through technological and non-technological means (Tiwana et al. 2010). Thus, the platform constitutes key resource environment for developers. At the same time, developers operate in a free market and create their own economy together with other market participants. This market is influenced by the platform, in terms of its installed base and customer characteristics; however it is still a self-contained economy with its own internal coherence. The market conditions therefore form the second part of the resource environment for developers. How then do resource environments lead to the emergence of field-level logic of independent Mac developers? This is the first research question I intend to answer in this study. In addition to formulation of institutional logics, literature has documented the effect of change in resource environments, such as environmental jolts, shocks, or critical events on institutional change (Nigam et al. 2010, Sine and David 2003, Hoffman 1999). For third-

party developers, changes in their resource environment include direct ones from the platform, such as platform governance change, as well as indirect ones, which are affected by the platform governance change and occur in developers' own economy. So the second research question I intend to explore is: how do changes in resource environments impact third-party developers' institutional logic? For the study, I examine a critical platform governance change from Apple – its opening of the iOS App Store and subsequently the Mac App Store, and hence its role shifting from mainly a technological platform to a platform that includes a market exchange place, and dissect the content of Mac developers' institutional logic before and after the change in resource environments. Regarding the outcome of field-level institutional logic change, literature has illustrated various forms of change based on their direction and extent (Thornton et al. 2012), such as settlement of a new dominant logic (Thornton 2002, Rao, Monin, and Durand 2003), stable co-existence of competing logics (Dunn and Jones 2010, Reay and Hinings 2009, Purdy and Grey 2009, Lounsbury 2007), or ongoing logic change (van Gestel et al. 2011). What is under-explored is the relationship between resource environment and logic change outcome. Therefore, the third research question this study intends to answer is what the dynamics are between the incumbent logic and the new logic of third-party Mac developers and how resource environments impact such dynamics.

Leveraging online archival data sources and combining narrative and content analysis (Nigam and Ocasio 2010), I trace the logic formulation and evolution for independent Mac developers, or Mac indies as called by themselves. Usually composed of one or two people, these micro-sized software firms write apps for Apple's desktop computer platform and sell them directly to customers mainly through the Internet. The

findings reveal that a software ecosystem logic prevailed for Mac developers prior to the opening of the App Store. During that time, Apple as resource environment provides the hardware and operating system technologies as well as development tools and design guidelines to developers. It also conveys a sense of coolness and the innovative spirit, and infuses an artistic pride in software and personal computers. Resource environment within developers' own economy comprises the Internet as a distribution channel, proliferation of infrastructure service providers and software technologies, and a customer base mainly composed of power users and Mac enthusiasts. This resource environment not only facilitates Mac indies to run viable software business, it also helps developers to build a strong identity of independence. Together, these resource environments shape the software ecosystem logic of Mac indie developers. After Apple changed governance mechanism by entering the software distribution domain in addition to being a technology platform, a different logic: platform ecosystem logic emerged. New rules and regulations from Apple ensue, and these send a strong message of control and bureaucracy to Mac indie developers. The governance change has also brought about new market dynamics and different profiles of fellow developers and end users; hence it changes the economic conditions that developers previously have operated in. The change in the market conditions serves as a catalyst for developers to redefine their software valuation and relationship with the platform. All these changes in the resource environments thus help create a new type of institutional logic for developers. A further analysis of the two logics suggests that the software ecosystem logic and platform ecosystem logic are in contestation at this early stage of institutional change, and the

organizational learning practices from one of the resource environments – the platform, reinforces this contestation for Mac developers.

This research first of all deepens our understanding about the emergence and evolution of institutional logics in the field of consumer software industry through examining the cross-level processes. Secondly, I illustrate in detail the characteristics and effect of resource environments in a field with structure different than what has been traditionally studied as an institutional field – the resource environments in the software industry exhibit a hierarchical and two-layer characteristic. Thirdly, I identify the temporal logic shift pattern evidenced in the changing field-level practices and symbolic meanings through a content analysis. This research also contributes to the software platform governance and software ecosystem literature by emphasizing the role of third-party developers in the software ecosystem. Through an institutional field conceptualization, I identify and explicate the material practices and symbolic representation of platform governance and third-party developers. I aim to construct software developers as institutional actors, who do not just write compatible software for a platform, but in fact constantly interpret and make meanings of governance mechanisms from the platform, which explains their subsequent practices and strategies. This study hence adds a symbolic and cultural lens to the software platform governance literature, which currently is mainly composed of perspectives on architecture-modular design (Baldwin and Clark 2000, Baldwin and Woodard 2009), economic explanations (Katz and Shapiro, Rochet and Tirole 2003, 2006, Parker and Van Alstyne 2005, 2012, Eisenmann, Parker and Van Alstyne 2009), and organizational considerations (Gawer and Henderson 2007, Cusumano and Gawer 2002). Apple's context is unique for the research

questions in the study. In recent years, the consumer IT industry witnesses a growing number of platforms adopting the "App Store" model, such as in software (mobile and desktop), social networking, web browsers and e-publishing. I aim to use Apple's ecosystem as a starting point to study the impact of platform's governance of technology and market distribution on third-party developers.

The paper proceeds as follows. First, I review literature on field-level logic emergence and evolution, followed by literature on software platform governance. Then I describe the methodology used for the study. In the findings section, I first present a narrative on developers' institutional logics before and after the change in resource environment; then I show the changing pattern of the two logics from results of a content analysis; and lastly I illustrate the dynamics between the two logics. The study concludes with discussions and implications.

Theoretical background

Emergence of field-level institutional logics

Building on the seminal work by Friedland and Alford (1991), Thornton and Ocasio (1999) define institutional logic as "the socially constructed, historical patterns of cultural symbols and material practices, including assumptions, values, and beliefs, by which individuals and organizations provide meaning to their daily activity, organize time and space, and reproduce their lives and experiences." (P804). The meta-theory of institutional logic is that "to understand individual and organizational behavior, it must be located in a social and institutional context, and this institutional context both regularizes behavior and provides opportunity for agency and change" (Thornton and Ocasio 2008: 102). This meta-theory allows institutional logic to develop at multiple levels, just as

Friedland and Alford wrote in the original piece: "An adequate social theory must work at three levels of analysis – individuals competing and negotiating, organizations in conflict and coordination, and institutions in contradiction and interdependency (1991: 240).

Over the years, Thornton and her colleagues have developed institutional logic at the level of industry or field (e.g. Thornton et al. 1999, Thornton 2002, 2004). They argue for the effect of societal-level logic on the formation of field logic by positing that "field-level logics are both embedded in societal-level logics and subject to field-level processes that generate distinct forms of instantiation, variation, and combination of societal logics" (Thornton et al. 2012: p148). For instance, the fiduciary logic in public accounting is a hybridization of logic of the profession and religion; the aesthetic logic in architecture is a hybrid of professional and market logic, and the editorial logic in higher-education publishing is a variant of professional logic (Thornton, Jones and Kury 2005). These field-level logics are instantiations of societal-level logics, and the specific historical, cultural and material contingencies in the field lead to field-specific variations in practices (Thornton 2012: p149).

In a recent development of cultural emergence model on field-level logics, Thornton et al. (2012) delineate the mechanisms through which cross-level effects operate. Building on the premise that institutional logics are both symbolic and material (Friedland and Alford 1991), they expand it by taking a linguistic turn to explain the construction of field-level logics. Because institutional logics reflect cognitive, normative and material forces (Thornton et al. 2012), they are embodied in the vocabularies and communication of members of social groups (Loewenstein and Ocasio 2009). As narratives create new systems of categories that link category labels to field-level

organizing practices (Thornton et al. 2012:159), or change meanings of existing categories (Ruef 1999), distinct institutional logics emerge. Their model is included in Figure 1a in the appendix, and a brief summary of the constructs and the processes in the model is as follows. The authors argue that societal logics, or external logics, defined as "the institutional logics developed in other institutional fields", are building blocks for the formation of field-level institutional logics. Providing both opportunities and constraints for field-level practices, resource environment affects emergence of field-level institutional logics through material forces, as well as cognitive, cultural and political factors. Vocabularies of practice, defined as "systems of labeled categories used by members of a social collective to make sense of and construct organizing practices", provide a critical linchpin which brings together symbolic representations, in the form of theories, frames and narratives and practices in formulating field-level institutional logics (Thornton et al. 2012: 150-161). Relationships particularly examined in the current study are presented in Figure 1b.

Field level institutional logic change

In addition to logic emergence, institutional logic change at industry or field level has continued to be of interest to institutional logic scholars. According to the cultural emergence model by Thornton et al. (2012), evolution and change in institutional logics can result from exogenous changes in societal and external logics, changes in the resource environment, and internal contradictions between symbolic representations and material practices in institutional fields (p161-162). In examining financial intermediaries in the U.S., Lounsbury (2002) documented that the stable regulatory logic established in the 1930s was replaced by the market logic due to a deregulation act in 1980. He argued

that this industry-level logic shift may be better viewed as an outcome of the general cultural shift from a regulatory to a market logic that unfolded gradually over the period since midcentury (p257). This example shows the effect of changes in both societal level logic and resource environments on the shift of field-level logics. New government policies are often theorized as a form of changing resource environment and they not only affect field level logics through material changes in regulatory act, but also symbolic representation reflected in their underlying logics. For instance, researchers document responses from universities and museums to a new government policy of business and performance measure (Townley 1997, 2002), and physicians' responses to provincial government's structural change to the healthcare system in Alberta (Reay and Hinings 2005, 2009). These new policies challenge the incumbent logics with their distinct symbolic meanings and rationalities and bring forth new status in field-level logics. Changes in resource environments are also manifested in occurrence of critical events, which trigger field level logic change. Hoffman (1999) demonstrate that a series of disruptive events, such as the publication of *Silent Spring* of 1962, beginning of the Earth Day and formation of the EPA in 1970, and the discovery of Ozone hole in 1985, etc., led to the emergence of environmentalist logics in the U.S. chemical industry. Glynn and Lounsbury (2005) show that the 1996 Atlanta Symphony Orchestra musicians' strike resulted in an increased attention to the market logic, reflected in critics' reviews, in addition to the aesthetic logic, which previously had dominated the symphony practices. As well, Nigam et al. (2010) illustrate how environmental sensemaking of the event of President Clinton's healthcare reform initiative in 1993-1994 led to the emergence of a

new logic of managed care, which replaced previous logics of physician authority and managed competition model.

Besides exogenous shocks, internal contradiction is another source that catalysts field-level logic change. In a series of studies on the integration of a new organizational form in a mature institutional field – professional accounting, it was found that the big five accounting firms, which have privileged access to resources and practices, are able to initiate institutional logic change (a shift from a professional logic to a corporate logic) from the center of an institutional field, and the legitimacy was gained through contested arguments and languages, which expose the underlying contradictions inherent in professionalism (Suddaby and Greenwood 2005, Greenwood and Suddaby 2006).

Regarding the various forms of field-level institutional logic change, Thornton et al. (2012) categorize them into *transformational* change and *developmental* change based on the direction and extent of change. What is particularly relevant to the current study is logic contestation and co-existence of competing logics. Townley (1997, 2002) illustrate that universities and museums reject certain aspects of the business planning and performance measure because the rationalities and logics implied in the two sets of practices are in conflict. Marquis and Lounsbury (2007) document that the community banking and national banking logics were competing from the very beginning of the U.S. banking industry, and the community logic resisted to be engulfed by national banking logic when a regulatory policy allowed for national banks' acquisition of smaller, local banks. Besides contestation, competing logics can also peacefully co-exist. Purdy and Gray (2009) identify that diverse institutional practices co-exist as emerging field develops within a 22-year period through mechanisms such as transformation, grafting,

bridging and exit. Reay et al. (2009) demonstrate that physicians and government agency employees in the Alberta healthcare field are able to manage the conflict through the development of collaborative relationships. Dunn and Jones (2010) also suggest logic pluralism by revealing that in the field of medical education, logics of care and science are supported by distinct groups and interests and they co-exist and fluctuate over time.

Structure of software industry and third-party developers' resource environment

Because the focus of this study is institutional logic in the field of consumer software industry, here I review literature related to developers' resource environment. Third-party software developers' resource environment is highly tied to the structure of the software industry. A typical two-sided market, software platform aims to bring both sides: developers and users on board, partly using optimal pricing mechanisms, and grow the ecosystem through two-sided network externalities (Rochet and Tirole 2003, 2006, Cusumano et al. 2002, Iansiti and Levien 2004, Cusumano 2010, Gawer 2010). Developers are complementors to the platform – they are neither platform's employees nor their component suppliers through arms-length contracts. Developers depend on a platform's services such as Software Development Kit (SDK) and Application Programming Interface (APIs) in order to obtain access to the hardware and operating system and write complementary application or services (Evans and Schmalensee 2007, Ghazawneh and Henfridsson 2011). These services form the foundation for a platform's technology-based governance over third-party developers, a classic consideration of which is the level of intellectual property openness of the platform (West 2003, Boudreau 2010, Eisenmann et al. 2009). As a result, software platform constitutes the first component of third-party developers' resource environment. Aside from technological

reliance on the platform, third-party developers operate in a free market with typical economic conditions, where strategic behaviors are desired. Boudreau (2012) demonstrates that increased number of developers tend to reduce innovation incentives in the PDA market, hence an evidence of the crowding-out effect in a competitive market. Through a study of enterprise software industry, Huang, Ceccagnoli, Forman and Wu (2013) show that third-party developers with a greater stock of formal intellectual property rights (such as patents and copyrights), and those with stronger downstream capabilities (as measured by trademarks and consulting services) are more likely to protect themselves against the threat of platform expropriation. In studying relationship between first mover advantage and environmental characteristics in the iPhone app market, Srinivasan and Suarez (2009) discover that early entrants outperform late entrants, and entry timing is more important in higher growth rate genres and for incumbent rather than new developers. These findings suggest that there is a second component of third-party developers' resource environment, which is formed by developers' economic environment. Furthermore, these two parts of resource environment are interconnected – platform policies and its market performance such as installed base can influence third-party developers' market dynamics, competitor types, and customer base. Therefore, it would be more precise to say that developers operate in a semiautonomous economy. To the best of my knowledge, no studies have discussed the effect of platform governance and developers' own economy at the same time, thus identifying these would be a contribution of this research.

The platform-developer relationship warrants a little more discussion. On the one hand, developers are a platform's external innovation assets and are governed through

technological design and pricing mechanisms. On the other hand, because software platforms are private companies, they tend to engage in certain strategic actions and exert rules and regulations over developers, which are "distorted away from pure value creation in the ecosystem, and towards actions that lead to higher platform profits" (Boudreau and Hagiu 2009: 170). In other words, platforms' priorities are to protect their own interests, secure their competitive positions or protect interests of end users, and they can be in conflict with developers' interests and goals. First and foremost, platforms have incentives to enter developers' market (Gawer and Henderson 2007) or fold third-party innovation into the platform (Parker and Van Alstyne 2012). If the platform releases a similar application with a third-party offering and bundles it with the operating system, users can obtain it for free and thus do not have to purchase it from the third-party developer. Secondly, platforms restrict developers' access to the platform for quality assurance purpose. This approach is most commonly seen in the video game industry. Expansion of size on the developer side can result in congestion and crowding (Boudreau 2012), and consumers' search cost can increase due to information asymmetry between consumers and developers. In order to reduce consumers' search cost, platforms thus engage in centralized "quality certification" via prescreening developers (Boudreau et al. 2009, Gallagher and Park 2002, Evans et al. 2007). These screening policies usually only grant certain elite game development shops opportunities to write video console games. Besides developer prescreening, platforms also screen products and decide whether or not they are qualified for release. Apple's app review policy on the iOS App Store is the best example and its inconsistency and lack of transparency attracts much outcry from the developer community (Bergvall-Kåreborn, Howcroft and Chincholle 2010). Lastly,

platform's regulation is also manifested in restrictions on user-complementor interaction. Direct interactions between complementors and end users not through the platform can harm platform's economic interests (Rochet et al. 2003, 2006), or affect the effectiveness of the idiosyncratic activities the platform is designed to perform. This approach is adopted mostly by platforms who are market exchange owners; it is relevant for the discussion here because the platform governance change observed in this study is a software platform also taking on the role as a market exchange owner. Boudreau et al. (2009) show that TopCoder, a vendor for competition-based software outsourcing, prohibits interactions between developers and final customers to ensure that the software development in the contest be a sequential and planned process. It is worth noting that platforms' rules and regulations described above resemble behavior control mechanisms in the organizational control theory (Kirsch 1996, 1997), where behavior control means that "specific rules and procedures are articulated, which, if followed, will lead to desired outcomes" (Kirsch 1997: 217). However, as Tiwana et al. (2010) suggest, "the relationship between platform owners and third-party developers is not the classical principal-agent relationship (i.e., the platform owner does not hire developers to do a task specified by the former), as assumed in the control theory. It is plausible that the role of control mechanisms then is one of coordination rather than mitigating agency hazards, as control theorists widely assume" (p680). As will be shown later, platforms' strategic actions, especially their rules and regulations help explain the dynamics between incumbent and new logics for third-party developers.

Methodology

Research context

In order to examine questions regarding third-party developers' institutional logic and the impact of their resource environment on the logic, I chose to study independent developers of Apple's desktop platform: Mac indies during the 2001 to 2012 time span. 2001 marked the year when Mac OS X, Apple's new operating system[4] was introduced to the world after founder Steve Jobs returned to Apple. The time also coincided with the dot-com bust, which left many software developers unemployed, but also a relatively mature online payment infrastructure grown during the Internet bubble. The availability of a cool new technology, free development tools[5], and an online distribution channel sparked a wave of entrepreneurship where small or individual developers form business writing Mac apps and selling them through the Internet (Meeteren 2008). As I will show in detail in the findings section, over the years, these Mac indie developers formed an institutional logic based on their professional conduct, and relationship with the platform and the market. Up until March 2008, Apple had mainly governed third-party developers with its role as a technology platform. A change occurred on March 6, 2008. With the announcement of the iPhone SDK, Apple also announced the App Store, the exclusive distribution channel for the iPhone apps, and later the iPad apps (hence the iOS App Store). In October 2010, Apple announced the Mac App Store (MAS), a market place with identical design features as the iOS App Store, but for Mac applications. A major difference between policies regarding the two stores is that the MAS is not the exclusive

[4] Mac OS X is based on NeXTSTEP, the operating system from Steve Job's company: NeXT, founded in 1985 after he was forced out of Apple.
[5] Before OS X, becoming an Apple software developer required an investment of around 1,100 dollars on third party software like Codewarrier (Meeteren 2008:23)

distribution channel for developers as the iOS App Store. With Apple extending its governance terrain to the market place in addition to technologies, the incumbent Mac indie developers had to cope with many changes. Besides, a new group of Mac developers appeared, many of whom come from iOS development. Over time, a new logic emerged.

Data and analysis

Table 1 illustrates the data sources and analytical process. Following the argument that narrative and vocabularies reflect the underlying process for the emergence and change of institutional logic (Thornton et al. 2012), and based on the role of historical research in analyzing field-level institutional logics (e.g., Thornton and Ocasio 1999, Thornton 2002, Lounsbury 2002, 2007, Marquis and Lounsbury 2007), I rely on publicly available online data sources of the Mac indie community, iOS-turned Mac developers, sources on Apple's culture and policies, and existing studies on Mac and iOS developers for this research. While most data sources cover the period between 2004 and 2012, an ethnographic study on Mac indies by Meeteren (2008)[6] and an oral history project on Mac culture provide data between 2001 and 2004. I use qualitative approach with an interpretive philosophy (Klein and Myers 1999) to derive Mac indies' institutional logics. Because institutional logic is concerned with social actors' practices and meaning system, an interpretivism epistemology fits perfectly because it is to "understand how members of a social group, through their participation in social processes, enact their particular realities and endow them with meaning, and to show how these meanings, beliefs and intentions of the members help to constitute their social action" (Orlikowski and Baroudi 1991:13). Furthermore, I follow Nigam et al. (2010) by combining a narrative and

[6] Although this is an undergraduate thesis, this study is widely recognized among the Mac indie community.

content analysis in studying the research questions. Specifically, I derive ideal types of developers' institutional logics before and after platform's governance change, explicate characteristics of developers' resource environment, use content analysis to quantitatively show the logic shift pattern over time, and analyze the relationship between the societal-level logics to elucidate the dynamics between the two logics at the field level. In the following section, I detail the process of ideal type construction and content analysis. Dynamics between the two logics will be illustrated in the findings section.

Ideal types, established by Max Weber (1904), is a typological construct for theory building and modeling (Doty and Glick 1994). According to Weber (1904), ideal types are "formed by the one-sided accentuation of one or more points of view and by the synthesis of a great many diffuse, discrete, more or less present and occasionally absent concrete individual phenomena, which are arranged according to those one-sidedly emphasized viewpoints into a unified analytical construct" (Via Coser, 1977:223-224). Ideal types are the commonly used formal analytic models to compare empirical observations across institutional order; therefore, they are best developed at least in pairs, if not multiple characterizations (Thornton et al. 2008: 119). As mentioned in the theory section, the societies are organized by cultural subsystems or interinstitutional orders (Friedland et al. 1991), and each of the institutional orders or logics is composed of elemental categories, which represent the cultural symbols and material practices particular to that order. The elemental categories are established social-science concepts, some of which are derived from Weber (1922/1978). In explicating the cultural emergence model of field-level institutional logics, Thornton et al. (2012) maintain that the key constructs in the model, such as symbolic representations, practices, and

vocabularies of practice are all categorical elements of institutional logics (150). For the current study, I use ideal types to construct field-level logics, and incorporate the key constructs in the cultural emergence model through discussions of the elemental categories. Table 2a illustrates the interinstitutional system ideal types developed initially in Thornton and Ocasio (1999), then extended in Thornton (2004), Thornton et al. (2005), and Thornton et al. (2012). Table 2b shows an example of ideal types of field-level logic in architecture (Thornton et al. 2005: 144). It is worth noting that "the elemental categories on the vertical Y-axis are not exhaustive and can vary in terms of which ones are most salient to the researcher's questions and research context" (Thornton et al. 2012: 59).

I draw on data sources related to incumbent Mac developers, Mac culture and existing research on Mac developers (e.g. Meeteren 2008) to construct ideal types for the logic prior to platform's governance change. To construct ideal types for the new logic, I examine data sources related to incumbent Mac developers' changes, iOS-turned Mac developers, and existing studies on iOS developers (e.g., Qiu et al. 2012, Meeteren 2009, Bergvall-Kåreborn et al. 2010). The assumption is that Mac developers' new institutional logic is composed of iOS-turned Mac developers' practices and belief systems, as well as incumbent Mac developers' changes in practices and belief systems in response to both the iOS App Store and the Mac App Store (MAS). This assumption is grounded in the following observations. First and foremost, the two App Stores share design attributes from the platform and their impact on the customer base. Second, the iOS-turned Mac developers would inadvertently carry their practices from the iOS App Store to the MAS, and thus influence the market dynamics and other players, including the incumbent Mac

developers. Thirdly, incumbent Mac developers have obtained direct or indirect experiences with the iOS App Store, so they would anchor their perceptions about and actions on the MAS based on previous knowledge. Lastly, media or opinion leaders, who observe Apple as a whole, tend to formulate predictions of the new store based on the old, and their conjectures would influence developers' subsequent actions.

My first step was coding one of the data sources – discussions topics on the major Mac indie listserv: *MacSB* on Yahoo! Groups. I traced the discussion from its inception: 1/29/2004 to 12/31/2012, the end of my data collection, in a 9-year span. I divided listserv threads into 3 time periods in accord with Apple's governance change to examine the temporal shift in developers' attention and discussions. The first phase is from 1/29/2004 till 3/5/2008, the second phase is from 3/6/2008, when the iOS App Store was announced, till 10/19/2010; and the third-phase is between 10/20/2010, when the Mac App Store was announced, and 12/31/2012. I expect developers' logic change to appear in phase 2, and blossom in phase 3. Due to the high number of total messages on the listserv (close to 20,000)[7], I coded the first message in a given thread and used that to represent the subject discussed in that thread. This assumption is justified by the hidden profile theory, which suggests that members of a discussion session tend to be biased on the information initially shared in the dialogue (Stasser 1992). I also tried to mitigate the limitation of this assumption with the large number of threads coded. After removing off-topic threads such as occasional technical discussions and trolls, threads with no or only one reply or activity organizing threads, I coded in total 1,264 threads. This number is

[7] This number might not be big in terms of number of messages per month. However, it is worth noting that this is a business-oriented discussion list for Mac developers. Lists with technical focus have much more traffic in comparison. I also checked other non-technical discussion list, such as the UI-design group, and the traffic is a lot less than this business-oriented list.

comparable to that in existing literature which uses listserv as data source (c.f. Orlikowski and Yates, 1994, Kuk 2006, Kudaravalli and Faraj 2008)[8].

To map the comprehensiveness of the elemental categories in ideal types, and to explicate characteristics of the resource environment of Mac developers' institutional logic, I adopt the stakeholder perspective (Donaldson and Preston 1995, Jones and Wicks 1999, Agle, Mitchell and Sonnenfeld 1999) as a general coding framework to identify developers' practices and interpretations in relation to their stakeholders. A much cited definition of stakeholders is "those groups without whose support the organization would cease to exist" (Stanford Research Institute (SRI) 1963, quoted in Donaldson and Preston 1995: 72). My use of the stakeholder perspective is mainly at the descriptive and empirical level, which suits exploration of the new areas (Donaldson et al. 1995: 70-71), such as my case. I also went a step further to analyzing the focal organizational entity itself – developers' own identity and practices. Together, five categories emerged. Three of them are related to developers' stakeholders: the platform, customers and infrastructure service providers. The rest two are market competition and developers' entrepreneurial strategies, and developers' identity and routine tasks. Figure 1 shows the stakeholder coding framework.

I combined the deductive and inductive coding approach in analyzing the listserv threads. For the deductive part, I relied on the platform governance literature and ethnography about Mac indies by Meeteren (2008) to code developers' relationship with the platform and developers' identity and drew on studies on iOS app developers (Qiu et al. 2012, Meeteren 2009, Bergvall-Kåreborn et al. 2010) to guide my coding on the new logic. I also allowed themes to emerge during the coding process. To start off coding, I

[8] The average number of messages coded in these studies is around 1000.

sampled about 800 threads across three phases to construct a coding scheme. To explicate the emergence of new logic, I used themes generated in the 1st phase as a baseline, and then carefully compared the meaning of developers' messages in phase 2 and 3 with the existing themes. Following previous literature (e.g., Ruef 1999, Reay and Hinings 2005), which maintains that the new logic can reflect in the changing meaning of an existing theme or in a new theme, I formed a coding rule where any new meanings attached to the existing themes or brand new themes were classified as the new institutional logic attributed to the platform governance change. For instance, theme "platform's entry into developers' turf" means "market and product clash" before platform's change, while it means "distribution channel clash" afterwards. By the same token, theme "platform's rules and regulations" mainly indicates legal rules before the change, and it means administrative and technical rules afterwards. The original expectation was that the incumbent logic would consist of 100% of discussion topics in phase 1, and the new logic would start to appear in phase 2. However, while comparing the findings about the iOS app developers with Mac developers' practices in phase 1, I found that two forms of practices which belong to the new logic: "frequent app launch to gain visibility" and "aggregating sales from small apps" were already present in phase 1. This is because, as will be discussed in details in the findings section, one characteristic of the new logic is that the centralized distribution channel and platform's store design and policies incentivize developers to adopt a form of "hit"-oriented strategy. Prior to Apple's App Store, there are several third-party app aggregators, such as Version Tracker and MacUpdate where developers list their apps. While they are not run by the platform, they induce similar hit-oriented behaviors from developers. Therefore, for the two themes, and

only these two themes in the category "market competition and developers' entrepreneurial strategies": "app portfolio strategy" and "frequent releasing strategy", their meaning underwent change even prior to the platform governance change. These two themes mean "app diversification" and "marketing coordination or obtaining feedback" respectively in the incumbent logic. It is worth pointing out that not all themes change meaning after the platform change strikes, and not all App Store-related discussions fall under the new logic. In addition, the original logic continued to exist in phase 2 and 3. This is because developers were either discussing apps released in their traditional outlets, or they were still following the original logic even for App Store apps. An example would be the theme "platform choice" in the market and strategy category. Regardless of which distribution channel a developer adopts, s/he always needs to consider the issue of targeting just one platform or more. Based on the characteristics of the logics, the logic before the platform governance change was named "software ecosystem logic", and the one after the change was named "platform ecosystem logic".

Regarding the coding process, I followed Strauss and Corbin's approach (1998). I first used open coding to generate properties and dimensions of themes through constantly comparing the existing themes with information in the new threads. This was followed by an axial coding process to extract the sub-themes, namely the condition, interaction, cause and consequences of themes. Last step was selective coding, which includes integrating and refining themes, subthemes and their positions in each of the five categories. Extensive memos were taken and assisted the coding process. After the first round was completed, I discussed the coding scheme with the research team, clarified different interpretations and adjusted the scheme. Then, I recoded all threads one more

time. The coding scheme continued to evolve until all threads were coded. To validate the coding framework, inter-coder reliability test was performed. A research assistant coded 10% of the threads in each of the three phases. 15 threads were used for training in each phase and disagreement in the interpretation was discussed. Some coding differences were due to the specific knowledge about the Mac business or software development. Cohen's kappa was used to calculate level of agreement, because it takes into account the agreement occurring by chance (Viera and Garrett 2005). The final Cohen's kappa is 0.75, which indicates good agreement and above the threshold of 0.70 suggested for content analysis (Neuendorf 2002, Krippendorff 2004). A complete coding scheme is included in Table 11 in the Appendix.

To triangulate the listserv data, I also drew on supplemental online archival sources. Snowball and theoretical sampling were used to obtain these data (Miles and Huberman 1994, Strauss and Corbin 1998). I started with blog posts mentioned in the listserv discussions, blogs by established Mac or iOS developers in the community, and well-known industry press, and then expanded data sources from there. Theoretical sampling was also used. For instance, I selected blog posts from developers who differ in opinions, strategies and performances on the Mac App Store. In addition, I searched data based on critical issues about platform's policies which stirred heated debate among the developer community and the press, such as Apple's issuance of the App Store review guidelines, Apple's changing policies towards Adobe Flash, and Apple's policies of Sandboxing and Gatekeeper. Data searching process ended when no new information emerged. This search resulted in 77 pieces of text composed of developer blog posts, industry press articles from MacWorld, macstories, Engadget and The Verge; oral history narratives on

Mac culture from Folklore.org – a project dedicated to the development of the original Macintosh, Apple's official iOS and Mac App Store review guidelines, transcripts of developers' pod casts and presentations from the now deceased C4 conference for Mac indies. Most of these data are to make up for the listserv data in the relatively short time span in phase 3, among which ten pieces of texts are on iOS-turned Mac developers' strategies on the Mac App Store. In the meantime, nine pieces of texts are about Mac developers' reactions to Apple's iOS policies, hence for phase 2; and 13 pieces of texts are for phase 1. These texts were coded using the same coding scheme derived from the listserv discussion, and only one new theme emerged: "platform organizational learning", characterized by developers' reflections on the resemblance between Apple's App Stores with its music store: the iTunes. This led me to collect additional six pieces of texts on Apple's iTunes, including one academic study, one press article, two press interviews with Steve Jobs, and two YouTube videos of Steve Jobs' presentations about the iTunes. Finally, I mapped themes developed from the listserv discussion and supplemental texts, and existing studies on Mac and iOS developers to the elemental categories to construct the ideal types. Table 11 provides detailed explanation of the mapping process.

In addition to ideal type construction, I also used coding results from the listserv discussions to quantitatively demonstrate developers' temporal shift in logics. Due to data availability, while multiple sources about incumbent Mac developers, iOS-turned Mac developers and existing studies were used to construct the ideal types of the two logics, only listserv discussions were used to quantitatively capture Mac developers' logic shift over time. Therefore in the content analysis, the new logic (platform ecosystem logic) only reflects incumbent Mac developers' change, and does not include practices and

belief systems of iOS-turned Mac developers. As mentioned earlier, based on the stakeholder coding framework, I derived five broad categories of developers' stakeholder relationship and their identity and practices. I examine developers' logic shift in two aspects: change of distribution of two logics in each category over time, and change of percentage of each category over time. For each phase in each of the five categories, I calculated frequency of each theme through aggregating lower-level themes for the two logics. Then, I calculated the percentage score for the two logics in each phase. Then for instance if in phase 2 of any given category, the frequency for software ecosystem logic was 40, and that for platform ecosystem logic was 10, then software ecosystem logic constituted 80% of developers' discussion topics in phase 2 for this category, and platform ecosystem logic constituted 20%. Over time from phase 1 to phase 3, I thus observed the changing distribution between the two logics for any given category due to platform's governance change. I also calculated the percentage score of each category in each phase. For example, if 100 instances were coded for each of the 5 categories in phase 1, then each category equally constituted 20% of developers' discussion topics in phase 1. Over time, the percentage score revealed the changing pattern of developers' attention on their stakeholder relationship as well as their identity and strategies because of platform's governance change.

Findings

Findings are presented in three parts. First I delineate two ideal types of Mac indies' institutional logics before and after platform's governance change and their relationships with the resource environment. Secondly, I corroborate the logic change with results from

the content analysis. Thirdly, I analyze the status of the two logics by elucidating their relationship at the societal level.

Ideal types of Mac developers' institutional logics

Table 3 presents the ideal types of two logics. As noted in the methods section, Thornton et al. (2012) maintain that the elemental categories for ideal types of the institutional logic reflect the key constructs in the cultural emergence model. Among the elemental categories in the current study, except for "basis of attention", which entails mostly material practices, the other categories contain both material practices and symbolic meanings. Figure 3 shows the resource environment for the two logics and how the platform part of the resource environment influences developers' economy part of the resource environment. I integrate discussion of the resource environment with discussion of the institutional logic. In the end of the description of each logic, relationship between resource environment and institutional will be again summarized.

Individual developers started to write and disseminate their software almost three decades ago. Developers distributed their software as "shareware", initially through dial-up bulletin boards or via disks given away with computer magazines, and later via the Internet. Users can try a piece of software free of charge, and then send a check to developers to purchase a registration license of the full version of the software (The Economist 2004). However, making a living out of shareware was hardly attainable under a rudimentary distribution and payment system (Takeyama 1994). Despite that, writing commercial applications and selling directly to users have long been dreams for many individual developers. This dream was made possible at the turn of the 21st century, thanks to a maturing e-commerce and online payment infrastructure. Among the many

homebrew developers of various platforms, those writing applications for Mac, Apple's desktop computers, grew into a significant number.

Gradually from 2001 to 2008, a software ecosystem logic was formed for Mac indies. This logic is a hybridized mix of the professional and market logic, and is guided by the personal and market capitalism. The market logic here is characterized by indie developers' specialist position in the market through exploitation of periphery of the resource space. The resource partitioning theory posits that as the level of concentration in a market rises, generalist firms tend to become larger and more general and exploit resources available at the market center – the more generic consumer demand or mainstream taste preferences of consumers. This leaves resources located outside the generalist target areas for specialist firms. These firms, which tend to be small in scale, can exploit periphery of the resource space – the niche markets, without directly competing with the larger generalists (Carroll 1985, Carroll and Swaminathan 2000, Swaminathan 2001). Among third-party Mac software companies, Mac indies are the specialist firms. They are shy of marketing and distribution resources necessary to reach mass market users as a generalist firm does, but they produce niche software and exploit periphery of the customer base. In contrast, companies like Microsoft and Adobe, including platform Apple itself produce generalist software, which target center of the market. Specialist firms' scale limitation however is remedied by platform's governance change, as will be discussed later.

Under the software ecosystem logic, developers' identity is characterized by their awareness of being third-party complementors to a platform, and their value on independence as a business owner. A popular analogy among the Mac indie community is

that they are the "sharecropper" to Apple. Developers' app sales are highly reliant on platform's market share and installed base, and they face the risk of platform entering their product turf some day[9]. If product competition with the platform does occur, developers would produce a power user version of Apple's app in order to avoid market clash (Meeteren 2008). This essentially drives developers to further target periphery of the market, which tends to consist of hard-core Mac users, consumers with specialized needs or developers themselves. Mac indie's identity is at the same time strongly defined by the independence aspect. Developers cherish the autonomy of being their own boss, having a big say over the business, and resisting the bureaucracy of working for a "BigCo". They normally bootstrap the business without taking external funding, and command the freedom in choosing the kind of technology for writing apps. Sources of legitimacy come from quality of apps, reputation of developers and platform recognition. Mac indie's emphasis on quality is highly influenced by platform Apple's meticulous attention to detail on aesthetics and user interface design. Such artistic pursuit was passed down from the design philosophy of the original Macintosh. As Andy Hertzfeld, one of the original Macintosh team members recalled, *"The Macintosh was driven by artistic values, oblivious to competition, where the goal was to be transcendently brilliant and insanely great. We wanted the Macintosh to be a technical and artistic tour-de-force that pushed the state of the art in every conceivable dimension."* Customers are also an important force in shaping app quality. Developers convey that Mac users are more likely to search for and purchase third-party software from smaller developers than those of other platforms. Mac users, like developers themselves, tend to be immersed in the

[9] The Watson history was a famous anecdote among indie community. Watson was software which was built to complement a piece of Apple's software. Shortly after Watson won an Apple Design Award in 2002, Apple released a new version of Sherlock that incorporated many of Watson's features (Meeteren 2008: 60).

cultural meanings of Mac experience and have a sense of taste (Meeteren 2008). Their high bar for quality thus pushes developers to hone in the app experience even further. Moreover, Mac software-related media outlets endorse quality apps through ratings, reviews and awards. In the Mac community, famous developers are frequently quoted and their success stories are widely shared. Some earned their reputation from being former NeXT consultants and Apple employees. Others became popular because of their craftsmanship in apps and leadership in the community. Developers' tight-knit community and being consumers of each others' products enhance the reputation system. Legitimacy also comes from platform's recognition. Apple hands out the Apple Design Awards (ADA) at yearly developer conference: WWDC and winning ADA significantly boosts the amount of peer recognition in the community (Meeteren 2008). Additionally, Apple used to dedicate a webpage to listing third-party apps, and being featured there is deemed great honor, not to mention the associated sales bump. For Mac indies, the authority includes their skills and capabilities in producing apps with high engineering and design qualities, as well as the level of market acceptance towards an app. As a community of practice (Brown and Duguid 2001), developers form norms regarding how members should behave during the socializing process. Following the principle of reciprocity, a developer earns credit in the community by helping others that later can be "exchanged" if he or she needs help (Meeteren 2008). Mac indies pay particular attention to the etiquette in dealing with competitors in public. Developers have a general consensus that their products are differentiating and not competing with each other. If they need to list feature comparison with a competing product, they make sure to be considerate of the counterpart and act in good faith and good taste. Competition on price

incentives is not encouraged and direct product imitation is frowned upon (Meeteren 2008). When indies set out to start the business, their mission is to do something they love, keep the business sustainable and increase sales. To do so, they focus their attention on resolving entrepreneurial challenges, implementing infrastructure services best suited for the business and adapting to platform's system progress and technology change. Piracy is unavoidable in the software industry. Indies have mixed feelings towards their apps being cracked – they are upset and angry and yet feel flattered at the same time. Developers differ in dealing with piracy –some design better licensing schemes to reduce the likelihood of future piracy; others choose to stop the cat-and-mouse chasing and try to educate end-users and turn them into paying customers. Besides piracy, Mac indies face many additional entrepreneurial challenges. For example, developers need to make decisions on multi-platform or cross-platform development, revenue models, strategies on pricing, product portfolio and product releasing. Among others, marketing and PR is one of the most critical issue developers face. While "echo-chamber" marketing through developer and press endorsement create word of mouth effect among developer themselves or power users (Meeteren 2008), reaching a wider audience requires standard marketing techniques such as advertising, price promotions, branding and tools connecting with customers. While Internet is indie developers' major distribution channel, they also leverage other distribution options. For instance, developers distribute apps through physical CDs, offer site licenses or family packs, sell through retail stores, collaborate with resellers, and participate in magazine promotions or bundled sales promotions. Under the software ecosystem logic, not only do developers need to make decisions directly related to their business strategies, but they are also accountable for

their relationship with stakeholders, namely, customers, infrastructure providers and the platform. Small company size allows Mac indies to add a personal touch in tech support. Customers would receive a support email signed with developers' name rather than name of a company division. In the meantime, developers also try to balance the personal and professional side of the business. To make support more efficient, they streamline the process with various support tools and issue tracking systems. Indie business relies heavily on market service providers, such as app aggregators, e-commerce and payment providers, hosting services, update and publishing service providers. Under the software ecosystem logic, developers' attention towards Apple is mainly centered around the impact of platform's system change and upgrade on third-party development. For instance, developers need to decide between backward compatibility to support customers of old operating systems and embracing the new OS to leverage more platform features. The biggest technical change developers encountered during the observation period was Apple's switching from IBM PowerPC chips to Intel chips in 2005[10]. This caused huge uproar among the developer community: developers were concerned that Mac would lose its character and become just like any mass-consumed Wintel machines. A bigger impact however, was that developers had to convert their code to be compatible with the new system throughout the transition period. Under the software ecosystem logic, basis of strategy is an organic growth model. Developers write apps targeting serious customers for long-term use. They maintain an average price range of $20 to $50 per piece of software, because developers attach great importance on price as it represents the

[10] The official reason for Apple to switch to Intel was power consumption. PowerPC was not able to provide the level of performance per watt that Apple needed for its light systems, such as notebooks and small form factor desktops, but Intel was. (Steve Jobs' key note speech at 2005 WWDC: http://news.cnet.com/Apples-Intel-switch-Jobs-keynote-transcript---page-3/2100-1047_3-5748045-3.html?tag=st.next)

value of their work. After the first version of an app is released, developers continue to fix bugs and enhance features to both strengthen the existing customer base and cultivate new ones. Under software logic, platform-developer coordination is conducted through both informal and formal mechanisms. Developers convey that Apple's development environment is more stable and has created more productive programming environment for developers compared to Windows. Besides, Apple's original innovativeness culture is influential to third-party developers so that they love to be creative with the Mac. Again as Andy Hertzfeld recalled, *"the best thing about the Apple II was the spirit of its creation. It was not conceived or designed as a product in the usual sense; it was just Steve Wozniak trying to impress himself and his friends. Most of the early Apple employees were their own ideal customers… Its unique spirit was picked up and echoed back by third party developers, who sprung out of nowhere with innovative applications."* In addition to such cultural and affective properties, Apple connects employees and executives with developers through the evangelism team (Meeteren 2008). In the meantime, Apple also maintains a membership-based, formal developer relationship program (Apple Developer Connection, or ADC) accessible to everyone. And it regulates developers through legal tools, enforcing rules on use of platform trademark and administering NDA.

The above ideal-type attributes of software ecosystem logic demonstrate that both societal-level professional and market logic are in effect here. Mac indie developers need to solve engineering and design problems, and their reputation system and platform coordination mechanisms reward apps with high quality. In the meantime, developers target specialist market and need to resolve entrepreneurial challenges. Under software

ecosystem logic, Apple as resource environment provides the hardware, operating system, development tools and design guidelines to developers. It also influences developers with its innovative spirit and artistic pride in software design. Resource environment within developers' own economy comprises the Internet as a distribution channel, proliferation of infrastructure service providers and software technologies. Developers' customer base is mainly composed of those who have the need for and knowledge about specialized third-party software, such as power users, Mac enthusiasts or other developers. This resource environment not only facilitates Mac indies to run viable software business, it also helps developers to build a strong identity of independence. It is worth noting that developers' customer base is partially influenced by the potential threat of platform entering third-party developers' product market, which indicates that the platform component of the resource environment influences the developers' economy component of the resource environment. Under software ecosystem logic, platform's impact on third-party developers' market dynamics, such as market competition, and their market strategies is minimal, if not none.

A change in developers' resource environment occurred on March 6, 2008. Along with the announcement of the iPhone SDK (software development kit), Apple also introduced the App Store, the exclusive market place to distribute the iPhone and later the iPad apps. This platform's governance change marked the beginning of Apple as a technology platform entering the domain of software distribution, enacting the role as a market exchange owner. It also harbingered a different developers' logic – platform ecosystem logic to arise. On October 20, 2010, Apple announced the Mac App Store (MAS), a same market place as the iOS App Store, but for desktop applications, and the

MAS is not the exclusive channel. As noted in the methods section, the platform ecosystem logic is analyzed through an integration of developers' practices and symbolic constructions built around both the iOS App Store and the Mac App Store.

The platform ecosystem logic is a hybridization of corporate and market logic, and it is guided by the managerial and market capitalism. Here the corporate logic stems from platform's technical and administrative policies for developers. The market logic is characterized by indie developers' generalist position on the market, and this is strongly influenced by the design and attributes of a platform-controlled market place. Platform Apple has in three ways assisted Mac indie developers in taking on the role as a generalist without actually being one – indie developers are still of small-scale, but they are able to leverage center of the resource space as generalist firms do with the help of the platform. First, the App Store has legitimized and popularized software consumption through increasing average consumers' knowledge about and demand for software. Shopping for software is no longer an activity only for power users; it becomes a mass market phenomenon. An experienced developer shared his observation: *"I was just amazed by how popular it was, people just loved apps. I've been in software forever, and I've never seen, the kind of response to software, which is really weird."* And because these new users are mostly average users, they tend to have mainstream preferences for apps as opposed to geeky ones. Therefore they form center of the resource space which appeals to generalists. Secondly, the App Store creates a mass distribution channel for developers to reach a much larger user base that they were not able to in the past – the App Store essentially solves the scale issue critical for a firm to function as a generalist.

Thirdly, the App Store design, especially its top charts feature, incentivizes developers to write mass-appeal apps and target center of the market space.

Under this logic, developers' identity is that of subordinate third-party developer entrepreneur. Apple as an exchange place owner created administrative and technical rules that developers need to adhere to. From App Store developer enrollment status to app review process, Mac indies experienced unprecedented organizational bureaucracy in releasing an app to the market. Apple performs "centralized quality certification" (Hagiu 2009) through a review process. However, unlike game console companies, who filter out developers *before* a game is created, Apple rejects apps *after* they are created. And as Apple itself is learning, their review guidelines were long time missing, and the inconsistency in the review results is frequently reported. Even after the guidelines were published, they were subject to change. This led to much frustration among developers; they feel that they cannot trust the platform and are losing control over their apps and even their business. Furthermore, developers need to abide by Apple's technology requirement for producing apps, be it programming languages or APIs. Many Mac indie's existing applications cannot be sold through the App Store simply because they will violate the guidelines. Developers expressed their chagrin in listserv discussions about their increased dependency on the platform:

"For Indies this is really bad news, as we are forced into the Mac App Store and its dictatorship rules: content censorship, technology lock-in and revenue sharing."

"Is this worth it? Nimble development has always been something I've taken pride in, but the App Store process is the very antithesis of nimble. It's slow, bureaucratic, and opaque. And for what? Is the sales increase worth it?"

"So, the big question is, will you see RA applications in the Mac App Store? Right now, we don't know. With Apple's onerous guidelines, most of our applications would not be approved. Even if they would be, however, are the benefits good enough to give up being a truly independent software developer?"

Under platform logic, developers' legitimacy is derived from App Store chart ranking, platform recognition and end users' reviews and ratings. Design of the App Store layout makes various app charts and Apple's featuring sections particularly salient and accessible. Users tend to correlate higher app ranking with better app quality, and developers would also show respect to those who have apps with high rankings. Apple's featuring lists act as platform's official endorsement and significantly increase legitimacy of featured apps. User rating and reviews are a form of "decentralized quality certification" (Hagiu 2009), and higher user evaluation adds considerable credibility on an app for potential buyers. Sources of authority stem from platform's review and end user interests. On the App Store, every app goes through a review process before being published, and as alluded to earlier, Apple employs its power in deciding app qualifications. Developers convey a sense of fear about Apple's potential disapproval of apps, and they frequently exchange information with each other about the "suitability" of an app before submission. The other source for authority is end user interests. On the scale with consumers and developers on each side, Apple as a platform tilts the scale towards consumers. Many of Apple's policies are designed to protect user experience and security, but they do not always benefit developers. For instance, Apple's App Store review guidelines have detailed items on banning apps related to personal attacks, with objectionable content and pornography, and those offending religion, culture, and ethnicity. This is certainly commendable act from a user's standpoint, but Apple reviewers' subjectivity and inconsistency often results in unfairness, and thus hurts developers' interest. Apple recently implemented two policies to protect end user security. While the sandboxing policy has caused much turmoil among the developer community

because many have to relinquish certain app features or withdraw apps from the Mac App Store, the Gatekeeper policy keeps end users safe without upsetting developers. Under platform ecosystem logic, basis of norms is developers' self-interest and dependency on the platform. Developers tend to behave as if their goal is mainly to exploit profits from the App Store market, and Apple is seen more as an authoritative boss instead of a platform providing core technologies. Developers' mission is to build competitive position of apps on top charts, and write apps to be favored by the platform. Some developers deliberately update apps according to platform's OS upgrade cycle with a hope that their apps, which show off the latest platform technology, will be featured on the App Store. Under platform ecosystem logic, developers' basis of attention is standing out from competition, embracing the mass app users, and adapting to platform's changing technological and administrative policies. On the App Store, competition stays much closer to each other than on the web. Not only are apps organized according to categories, similar apps appear together after a user's search. Therefore developers need to design app graphics and descriptions to quickly grab user's attention. As noted earlier, the App Store has attracted a large number of users who have never made software purchase part of their lives, but it also brings new challenges to developers in tech support. Unlike power users who are familiar with third-party software experience, these new users exhibit certain "immature" behaviors. For instance, they would complain about features that developers do not claim to have or blame developers for issues they are not liable for, etc. Learning to interact with average users is thus a new task for developers. Besides close competition and average user support, platform's changing policies also keep developers preoccupied. The Mac App Store removed much flexibility that developers

had enjoyed in app transactions with users on their web stores: the App Store licensing scheme makes sharing review and testing copies less easy; a commonly used time-limited demo approach by Mac indies is forbidden on the App Store; paid upgrades are not allowed on the App Store; user information is hidden from developers, so reaching customers is not straightforward. To accommodate these changes, developers employ technical methods to address the licensing and app monetization challenges. They try to use in-app purchase through modular app design or new app release to solve the no-paid-upgrade issue, but they both have limitations. Developers also attempt to communicate with users through app description or sway them onto their web stores, although the latter bears the risk of app rejection. Besides, maintaining two customer bases and two app builds for the App Store and the web store adds burden to developers. Under the platform ecosystem logic, developers' basis of strategy is "hit" growth (Qiu et al. 2012). The "hit" here means striking app charts and constant new app releases. This is again mainly associated with design of the App Store and platform's review process. Because ranking on top charts gives apps legitimacy and increases visibility, and sales volume determines chart position, developers tend to lower app prices and engage in pre-release buzz marketing in order to climb onto the chart during the initial launch. Writing apps with mass-appeal and modeling after or even mimicking store-trending apps is frequently pursued on the App Store, because this increases the chance for apps to hit the charts. The "hit" growth strategy is also related to platform's review process. Since Apple evaluates apps *after* they are created, in order to minimize the risks of app rejection, developers release smaller-scale apps and save development cost. Consequently, developers diversify app portfolio with many small apps and accumulate profits from each of them. Having an

assortment of apps also taps into the new user group, who tend to be casual app consumers and view apps more like a form of entertainment rather than utility. Under the platform ecosystem logic, platform coordinates with developers mainly through formal mechanisms, namely, a pricing structure composed of membership fees and app royalty, and the technological, administrative and financial rules and regulations. Mac developers conveyed that the previously informal and personal exchange with Apple and Apple employees is disappearing. A seasoned Mac indie developer expressed his hope for a better communication on the App Store: *"I'm hoping to see the Mac App Store developer experience evolve to encompass the needs of established Mac developers when it comes to providing the best possible service to our customers; having official channels so that developers can contact empowered App Store staff for customer-service issue escalation and resolution would be a great step."*

The above ideal-type attributes of platform ecosystem logic exhibit the effect of societal-level corporate and market logics. The corporate logic is manifested in developers' increased reliance on the platform for technology and business needs. The market logic is reflected in developers' opportunities and incentives to function as a generalist company – in adoption of a "hit" strategy and embracing the mass users. Under platform ecosystem logic, Apple as the resource environment provides extensive rules and regulations on app qualification, licensing, monetization and versioning. These send a strong message of control and bureaucracy to Mac indie developers. Platform's entry into developers' distribution channel significantly alters the other part of developers' resource environment – the market economy in which developers operate. Through bringing in large number of new users and developers, and a store design which favors

frequent new app release, shorter development cycle and lower pricing, Apple creates a platform-controlled market environment, which denotes a very different kind of competition dynamics. These changes in the resource environment serve as a catalyst for incumbent Mac developers to redefine valuation of their software and labor, and their relationship with the platform.

Logic evolution of incumbent Mac indie developers: a stakeholder view

In the previous section, I illustrate and contrast the ideal types of software ecosystem logic and platform ecosystem logic along the elemental categories before and after change in the resource environment: namely, platform's governance change and the subsequent change in developers' economy. As noted earlier, Mac developers' platform ecosystem logic is composed of iOS-turned Mac developers' practices and belief systems, and incumbent Mac developers' changes in practices and belief systems in response to both the iOS App Store and the Mac App Store (MAS). In this section, I only focus on the incumbent Mac developers, and demonstrate how their logics shifted in a temporal structure using results of the content analysis based on Mac developers' listserv discussion. As explained in the methods section, I divided the listserv discussion into three phases. While instances of platform ecosystem logic are already present in phase 1, they began to take shape in phase 2 and flourish in phase 3. Recall that a stakeholder perspective was adopted in the listserv threads coding. This approach allows me to observe which part of software logic changed meaning and morphed into platform logic due to platform's governance change, and which part did not. In Table 4 through Table 8, themes are shown on developers' relationship to three stakeholders: customers, the platform, and infrastructure providers, as well as their own entrepreneurial strategies,

identity and routines. These themes are aggregated to top two levels from the original coding scheme in Table 11. Highlighted in yellow are the ones which changed meaning or whose sub-theme(s) changed meaning. It is worth noting that certain category and themes did not take on meanings of platform ecosystem logic. For instance, meanings remained the same for themes in the category "developers' relationship with infrastructure providers". This is because platform Apple acts as a competitor to these services – it takes over their functionality with the App Store, rather than changing the way these services work. In addition, theme "3rd-party developers as modular to a system" as part of developers' relationship with platform and "creation of an app other than coding" as part of developers' business operational routines were not affected by platform's governance change either. Note that the kind of governance change I examine is platform's entry into the marketplace domain. Hence, its impact is most salient in business-related themes, and less likely to occur on technology-related themes, albeit they also reflect developers' business decisions.

Table 9 shows the changing pattern of each of the five categories over time. With platform's governance change, developers' discussion topics related to the platform and customers have increased significantly. This is mainly because platform's new review process, manifested in the sub-theme "Platform's rules and regulations" under "Platform's coordination with developers", and platform's policies on apps' versioning, upgrade and licensing, reflected in the theme "Versioning and upgrade" and "Licensing" attracted much of developers' attention. In contrast, developers' discussions on infrastructure providers have reduced dramatically. This is not surprising, given most of the functionality is replaced with the App Store itself and developers' dependency on

them is decreasing. Developers' attention on entrepreneurial strategies, their identity and business routines have roughly remained the same. An interesting observation is that from phase 1 to phase 3, developers' discussions on market environment and entrepreneurial strategies, as well as their identity and routines consistently exceeded those on other categories. It indicates that how to compete on the market, how to appraise oneself and behave in a community, and how to operate the business has always been developers' top focus. The third-place changed from infrastructure service-related discussions in phase 1 to platform relationship in phase 3. It suggests that as the platform is involved in third-party applications' distribution, developers' attention towards the platform increased substantially.

Table 10 exhibits the changing distribution of software and platform logics over time in five categories. Except for relationship with infrastructure providers, which does not entail platform logic, other categories all witness a migration in discussion topics from software logic to platform logic due to platform's governance change. Interestingly, in categories on relationship with the platform and customers, platform logic-related discussion significantly surpassed that on software logic and became dominant in developers' discussion in phase 3. In contrast, software logic still prevailed in phase 3 for developers' discussion on market and strategies as well as their identity and business routines. To explain the latter phenomenon, let us recall that software logic in phase 2 and 3 is composed of two types of discussions: issues related with developers' traditional distribution channel: the web store, or same set of issues also applicable to App Store apps. Percentage distribution among themes in these two categories in phase 3 particularly suggests that "entrepreneurial decisions", together with developers'

operational routines, including “creation of apps other than coding” and “small business operations”, are what developers focus on the most regarding the web-store distribution issues. They are also less sensitive to platform governance change and are more likely to share in practices under two distinct distribution channels. The means across all categories shows that in phase 3, software ecosystem logic consists of close to 60% of developers’ entire listserv discussions, and platform ecosystem logic a little over than 40%. This suggests that the two logics co-exist with each other.

Dynamics between software ecosystem logic and platform ecosystem logic and platform’s organizational learning

In previous two sections, I contrast ideal types of software ecosystem logic and platform ecosystem logic, and show that they co-exist through an analysis of incumbent Mac developers’ listserv discussion. In this section, I argue that the two field-level logics are competing with each other by elucidating the conflicting points between their societal-level logic components. This analysis is based on the premise of the institutional logic theory that the interinstitutional logics in the society are interdependent and yet contradictory, each with differing belief systems and sources of rationality (Friedland and Alford 1991). Figure 4 provides a visual representation of the logic conflict. As discussed earlier, software ecosystem logic consists of professional logic and specialist market logic, and platform ecosystem logic comprises corporate logic and generalist market logic. I discuss their relationship according to the two dimensions of the platform ecosystem logic, as distinguished by the red and blue boxes in Figure 4. The first conflict between professional and corporate logic is manifested in developers’ autonomy in making engineering and design decisions vs. platform’s control in these aspects. Professional

logic implies that Mac developers enjoy the freedom in using the kind of technology in writing apps, deciding how their programs behave and interact with other technology after installation or upon running, and tweaking native user interface elements, to name just a few. Corporate logic limits much of this freedom in that Apple as the platform exerts stringent control on these fronts through App Store review guidelines (Frakes, 2010). What are business decisions to a platform is interpreted by developers as "politics". Developers, who always like to resolve technical problems and "get things to work", can accept that the platform has certain technical incompetence, but they are extremely frustrated by the administrative control or "politics" which they have no ways to fix.

The other conflict between professional and corporate logic resides in developers' professional orientation vs. platform's consumer orientation. Apple's sandboxing policy is the major trigger behind this conflict. Starting from June 1, 2012, when the operating system Mountain Lion was released, Apple mandated that new and significantly updated apps submitted to Apple's Mac App Store must implement sandboxing. Sandboxing refers to "compartmentalizing what data and features a specific app is granted access to; apps each can metaphorically play exclusively in their own sandbox, accessing only that data which Apple has granted that app entitlements to see" (Friedman 2012). Sandboxing is intended to enhance security, protecting Mac users from malware and poorly designed apps (Weatherhead 2012). This policy is consistent with Apple's increasingly emphasized position as a consumer product company. Late CEO Steve Jobs described the company identity at Apple's 4th-quarter earnings conference in 2010 by saying: "We're a very high-volume consumer-electronics manufacturer. Consumer-electronics companies sell stuff to consumers. Consumers, well, consume. Once they've chosen their platform, they

predictably follow its guidance" (Myslewski 2010). This stance and its related policies contrast sharply with many Mac developers' identity as a power user and doer. Andy Ihnatko (2011) from the *Macworld* was concerned that sandboxing risks eroding the Mac's identity. He strongly opined that "Mac users are doers, not consumers. The Mac must never, ever become a consumer product like the iPad, saddled with artificial limitations in the name of safety, reliability, and tidiness." Sandboxing restricts what a program can do with the system, thus also limits how much a user can do with their Mac. Developers' concern is that "the end result may be more safety for the average user, but at the cost of freedom – and advanced capabilities – for professional users" (Mogull 2012). Moreover, for power users or other developer customers, sandboxing policy deprives them with trust and confidence towards the Mac App Store as a market place to shop for apps. This is because certain apps they purchased before can no longer be updated, and hence caused them to lose software investments (Rentzsch and Pontious 2012).

Regarding the conflict between specialist market logic and corporate logic, it is reflected in developers' autonomy in determining customer-facing and market positioning strategies vs. Apple's control in them. As noted earlier, outside of the Mac App Store, developers are free to set up trial and demo for their apps, be it time-limited or feature-limited. They can charge for the upgrade to support continuous development, and design the licensing scheme for copy sharing, such as beta testing or app reviewing. The Mac App Store review guidelines explicitly prohibit all these flexibilities. In addition, while developers can acquire customer information for targeted marketing and troubleshooting with apps sold through their web stores, Apple makes user information hidden with apps sold on the App Store, and thus makes developer-user communication much more

difficult. Furthermore, outside of the App Store, developers have the autonomy in designing marketing messages and positioning itself in relation to competition, be it other developers or the platform itself. However, Apple removes these rights from developers by including the following clauses in the review guidelines among others: “apps with metadata that mentions the name of any other computer platform will be rejected”; “apps that look similar to Apple Products or apps bundled on the Mac, including the Finder, iChat, iTunes, and Dashboard will be rejected” (Frakes 2010). Apple also disallows developers to mention the web-store version of the app in the MAS app description (Rentzsch et al. 2012), which essentially reduces users’ chance of findings apps in alternative channels.

Now let us examine the second dimension of the platform ecosystem logic: generalist market logic. Its conflict with the professional logic is manifested in the different emphasis on app legitimacy. It is app quality as recognized by fellow developers, the platform and the users that the professional logic focuses on, whereas it is app ranking and users’ reviews and ratings that the market logic underscores. With regard to conflict between specialist market logic and generalist market logic, it is demonstrated in the distinct approach on growth strategy. Under a specialist market logic, developers favors organic growth, while under a generalist market logic, developers turn to hit growth. Compared with the level of conflict between generalist market logic and software ecosystem logic (the blue box in Figure 4), that between corporate logic and software ecosystem logic (the red box in Figure 4) is stronger for incumbent Mac developers and users. This is because the pressure from corporate logic is drastically different from the conventional way that Mac development is done, and it challenges developers’

independence identity and their identification with the Mac and Apple in the old days. People lament that "the Mac App Store, for all intents and purposes, will bring an end to the rogue, piratical culture that Jobs once championed[11]" (Myslewski 2010).

Developers respond to the change of Mac development and Apple in different ways. Some embrace the idea. It is almost unanimously agreed among developers that the Mac App Store brings better purchasing and maintenance experience for end users, not to mention the access to nearly all Mac users for indie developers. Developers report sales jump with the Mac App Store. Regarding platform's tightened control mechanisms, some developers surrender. For instance, in order to meet sandboxing requirement, developers have to re-architect their apps or castrate features to continue to publish on the MAS, although they state that doing so costs time and effort, sacrifices user experience and damages developer reputation (Haslam 2012). Others resist. Some developers maintain a relatively high price range for apps sold on the Mac App Store, refusing a "race to the bottom" pricing strategy, therefore challenging the generalist market logic. Some developers consider leaving the MAS and only releasing apps in the traditional channel: their websites. Yet others propose a "pro" version of the App Store or open a community-organized distribution channel as an alternative. Not only developers withdraw writing apps for the MAS, developer customers and power users abandon app consumption from there, too. Marco Arment, an influential iOS developer and a power user of Mac, warned that "the Mac App Store is in significant danger of becoming an irrelevant, low-traffic flea market where buyers rarely venture for serious purchases" (Arment 2012). Similarly, Neven Mrgan, designer at a well-known Mac indie shop: Panic, was also concerned about the future of the MAS: "the loss of casual users to iOS, and the loss of non-casual apps

[11] "Why join the navy if you can be a pirate?" – Steve Jobs 1983.

on the (Mac) App Store—and it starts to look like a problem" (Mrgan 2012). A collective sentiment among the incumbent Mac indie community towards policies such as sandboxing is that "it will create a lose-lose-lose situation for Apple, the developers and the users" (Haslam, 2012). Yet others, frustrated by the whole process, even decided to quit Mac development completely or transition to Linux. Developers shared the following sentiment on the listserv discussions:

"With the announcement of the Mac App Store, Apple has broken any lingering hope I had for one day succeeding at indie Mac development. Being treated as a responsible adult, innovating without restriction, connecting directly with customers, and being able to fix my mistakes quickly — the things I cherished most about my job at CT — are being gradually replaced by a "submission" process."

"I've been a Mac developer for many years now. Over the years I've seen countless technologies pass by. I've enjoyed seeing Apple in good times, and I've been with them in bad times. I've seen people at Apple come and I've seen them leave. And I've stayed because I love what I'm doing. But it's getting harder to love Apple. It's not only the constant NDA's. I'm tired of filing bug reports that don't get fixed for months or years for things that I could probably have fixed myself over a few days. I had moved completely to Cocoa at the time but I can understand how the Carbon developers felt after basically being left in the cold over a night. The slow response to the DigiNotar incident. HFS+ corruptions. WWDC. Section 3.3.1. App Store. I want out of it."

In addition to identifying the conflict between the software ecosystem logic and platform ecosystem logic, the data revealed that organizational learning (Levitt and March 1988, Huber 1991) of one of developers' resource environments: the platform exacerbates these conflicts for incumbent Mac developers. For platform Apple, this organizational learning comprises of two parts: continuous adoption of a centralized distribution channel and learning by doing. The former led to a misfit between platform's governance model and third-party developers' conventional practices. The latter caused great uncertainty among developers and users, and reduced their motivation in using the platform. If we were to trace the origin of the current Mac App Store, it is a carbon copy of the iOS App Store, and the iOS App Store is adapted from Apple's iTunes music

store[12]. Although the iOS App Store also attracts many complaints from developers due to Apple's rules and associated market dynamics, "its restrictions work for the most part, because the platform has grown around them. They mostly don't get in the way" (Marco Arment, in Friedman 2012). However, it became an issue when Apple was aiming to use the iOS App Store model, minimally changed, for the Mac App Store (Frakes 2010). Marco once again opined that "the App Store policies are being retrofitted into a well-established environment that they're fairly incompatible with" (in Friedman 2012). In addition to grafting a new model to an existing market, platform's learning by doing also deepens the logic conflict for Mac developers. Apple adjusts policies based on feedback from the market over time. While learning is common for any type of organization, the consequence of a platform's learning could be damaging for third-party complementors. Apple's learning by doing means that the timing of their many policies towards developers is *after the fact*, i.e., Apple pulls apps after they are published, or requires apps to be modified architecturally after they are completed. Such policy changes of Apple result in developers' waste in development and marketing effort, or users' loss in software investment, not to mention Apple is known for changing policies without informing its developers. All these have led to great uncertainty among developers and power users, and caused people to lose trust in the platform.

[12] Following Sprigman (2006), I illustrate the shared attributes between the iTunes music store and the iOS App Store as follows. First, in both cases, Apple dominates one side of the two-sided networks – the hardware sales market, and uses this to negotiate terms with content providers (in iTunes, it is the Big 4 music label companies, and in the iOS App Store, it is developers). Second, both stores create a "des-intermediation" model, which directly connects consumers with content. Third, Apple adopts the same 30/70 revenue split model with content providers. Fourth, the iOS App Store adopts the "top charts" and "hit" feature from the iTunes store. For one thing, the iTunes "re-empowers" singles (du Lac 2006), but it also reinforces the "hit" mentality for both the music industry and consumers. This is extended to the iOS App Store for software production and consumption. Developers refer to this as "pop software" model (Rentzsch et al. 2012).

Discussion and conclusion

This study set out to investigate the relationship between resource environments and field-level logic in the context of independent software developers of Apple's desktop platform (Mac indies). Following the cultural emergence model of field-level logics in Thornton et al. (2012), I attempt to answer three research questions: how do resource environments lead to the emergence of field-level logic of independent Mac developers? How do changes in resource environments impact third-party developers' institutional logic? And what are the dynamics between the incumbent logic and the new logic of third-party Mac developers and how do resource environments impact such dynamics? The findings reveal that two layers of resource environments are present for third-party developers given the structure of platform-oriented software industry: the platform governance pattern and developers' own economy. Both of them influence developers' logic through material practices and symbolic meanings. In addition, developers' economic environment is also affected by platform's governance. Exploiting a critical change of the resource environment – Apple extending the role from a technology platform to a market exchange owner, and through a narrative and content analysis, I show that a software ecosystem logic prevailed prior to the change, while a new logic: platform ecosystem logic emerged afterwards. Constructing the ideal types of the two logics via a stakeholder perspective, I demonstrate that these two field-level logics are both combinations of societal-level logics, and are also subject to the distinct processes of the software industry. Software ecosystem logic is a hybrid of professional and market logic, and platform ecosystem logic is a hybridization of corporate and market logic. As the platform expands its governance territory and tightens its rules and regulations over

developers, developers' practices towards and interpretations of platform's role and their own economic environment change as well. These changes are manifested in the elemental categories of the ideal types of the two logics. A content analysis of incumbent Mac indies' discussion also illustrates the temporal logic shift pattern evidenced in the changing field-level practices and symbolic meanings. Furthermore, a discussion of societal-level logics' conflict shows that Mac indies' incumbent and new logics are still in contestation with each other at this stage. Moreover, platform as part of the resource environments intensifies such contestation due to its organizational learning process.

This study contributes to the institutional logic theory mainly through a validation of the theoretical model of field level logics in Thornton et al. (2012). I specifically explicate the role of resource environments and societal level logics on the formation and dynamics of field-level logics. In accord with the dual view of institutional logic theory, I illustrate both the material and symbolic aspect of Mac indie's logic through analyzing the discourse of developers and trade press. This study also contributes to the literature on software platform governance by emphasizing the role of developers in the ecosystem. I position developers as institutional actors, who interpret and make meanings of platform's governance mechanisms and their change. This adds a fresh perspective on the extant platform ecosystem literature, which tends to focus on the platform itself, or merely treat developers as an ancillary component in the platform ecosystem competition. In this study I examine one platform and hence am able to integrate discussion of platform governance together with developers' reactions and interpretations. This study also contributes to the emerging literature on platform's non-pricing governance mechanisms (e.g. Boudreau and Hagiu 2009, Hagiu 2009), by systematically

demonstrating the rules and regulations of a technology platform-turned market exchange owner. For Apple, certain non-technology policies are to secure its role as a market exchange owner. In the meantime, it also issues technology-related policies not for technology's sake; rather, they are designed to protect the interests of one side of the two-sided networks – consumers', although these policies can hurt interests of developers, the other critical part in a two-sided network. This also reveals another trade-off for platforms – the balance between consumers' interests and developers' interests, in addition to the classic trade-off frequently discussed in the literature – that between value creation and value appropriation when considering platform's technology openness.

This study has several practical implications. It indicates that it is crucial for a platform to develop better understanding about third-party developers, especially about where they come from. For Apple, they face the risk of losing incumbent Mac indie developers on the Mac App Store because its technology policies inconvenience this group of developers. Apple can try to find a midpoint which can both protect users' interests and accommodate developers' needs. For a technology platform transitioning to becoming a market exchange owner, it is also critical to spend more energy in designing better store experience to facilitate buyer-seller transaction. For instance, lack of direct communication between developers and users has been complained about by developers for a long time. Apple could design mechanisms to enhance such communication experience while retaining its control as a platform. This study can also help further developers' understanding about their own practices and meaning system. Cultural anthropologists argue that "many of the cultural ideas and standards for interpretation which guide the ways in which people ascribe meaning are not well understood by the

people who use them…One of the great contributions of the cultural perspective has been to uncover the more tacit dimensions of human thought and to demonstrate how these hidden grammars of meaning help shape our lives" (Chambers 1985: p5). While this study is not ethnography per se, it does expose the underlying meaning system of developers, which might not be obvious to developers themselves.

This study has several limitations. First is regarding the causality inference of developers' discussion topic shift in the content analysis. I suggest that the shift in developers' practices and meaning system is due to platform's governance change; and yet the method used for the content analysis cannot rule out alternative explanations. For instance, discussion topics could be related to the type of people who post messages, the list guidelines enforced by the list moderator, the auto-correlation among threads topics, etc. However, the goal of the content analysis is not to argue for causality; rather, it is to demonstrate a general trend in the changing pattern of the two logics. Secondly, I only coded the first message in a thread and this could leave out certain dynamics within a given thread. This is less of an issue for the ideal type construction, because of the use of supplemental data sources. However, this could potentially influence the results of the content analysis, although I try to mitigate this limitation with the large number of threads coded.

This study suggests several directions for future research. First of all, given unique attributes of each platform and their governance mechanisms, it would be interesting to examine the resource environment in other software platforms or technology platforms which also incorporate a distribution channel in their governance model, and the impact of the resource environment on third-party developers' institutional logic. For instance,

Google and Microsoft, while both adopt a similar "app store" approach, have policies which differ significantly from each other and from Apple. The implications that these governance policies have on developers' economy, and consequently their practices and meaning systems would be interesting to study. Secondly, in this study we focus on small-scale independent third-party developers because of characteristics of Apple, for whom indie developers constitute a major component of the third-party developer base. For platforms such as Microsoft, their developers exhibit different firm size and scope compared with Mac indies. How these developers' institutional logic emerges and changes in relation to their resource environment would also be interesting to examine. Last but not the least, it is argued that "the conceptual scheme of the ideal types offers a guide for developing hypotheses about the effects of institutional change on the attributes likely to affect the dependent variables of interest" (Thornton 2004: 25). Thus, the ideal types derived in this study can potentially be developed into hypotheses about Mac indie developers and demonstrate that the effect on dependent variables of interest is institutionally contingent.

Figure 1a. Cultural emergence model of field-level institutional logics in Thornton, Ocasio and Lounsbury (2012: 151)

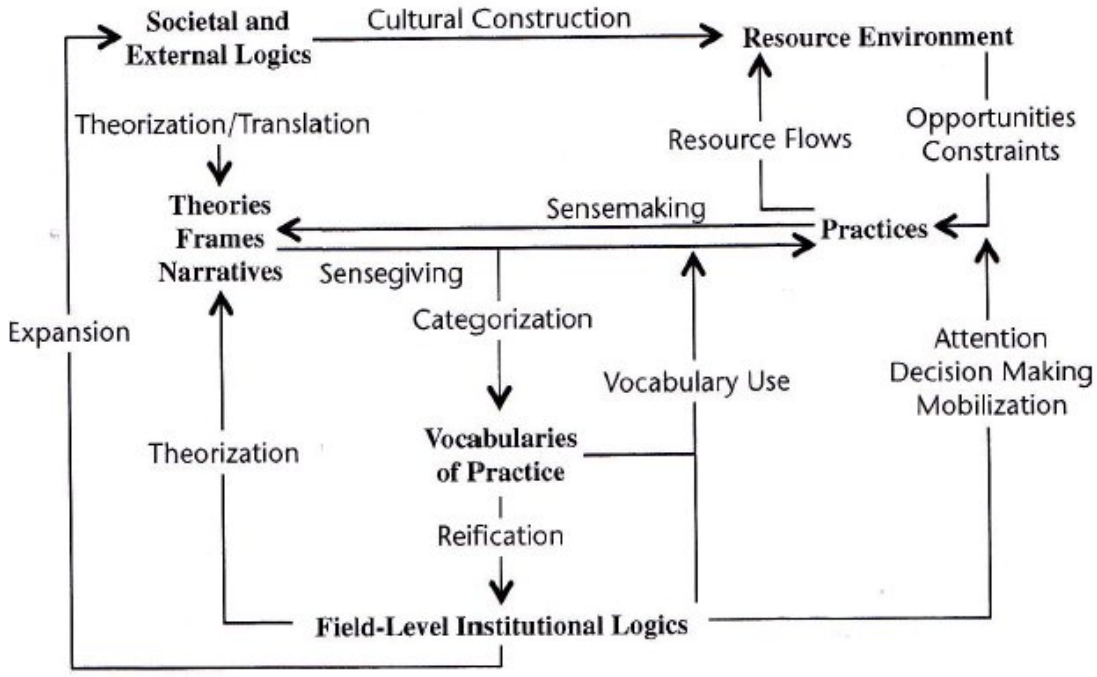


Figure 7.1. Cultural Emergence of Field-Level Institutional Logics

Figure 1b. Relationships examined in the current study

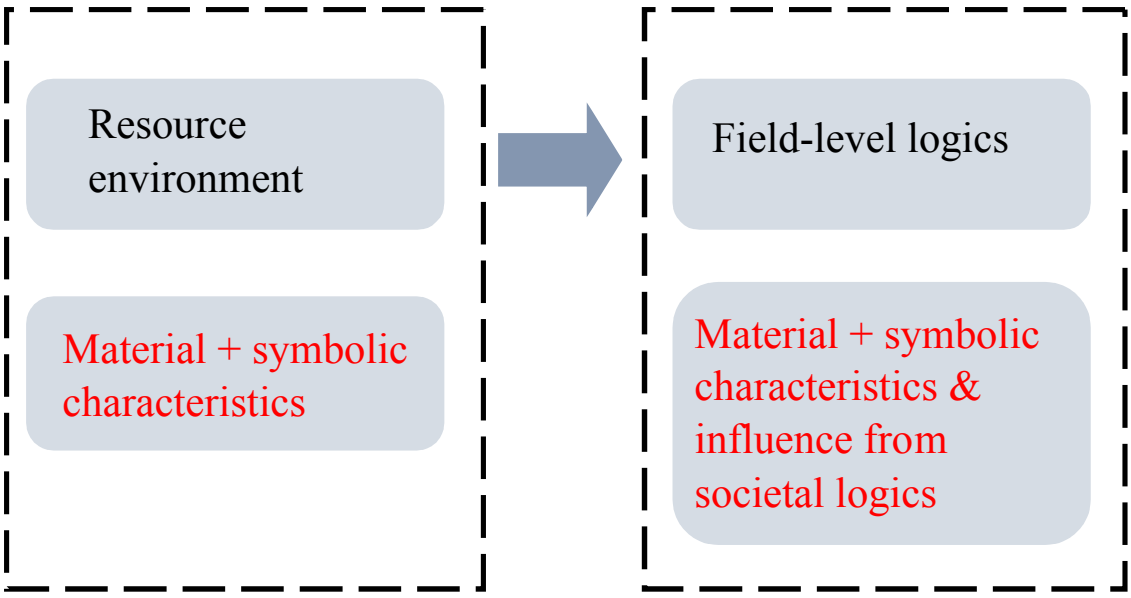


Table 1. Data sources and analytical process

Data sources and analytical process	Explanations
1. Time frame of interest on Mac indies' institutional logic	2001-2012
2. General data sources	Publicly available online data sources; existing studies on Mac indies and iOS developers. Most data sources cover 2004 to 2012; an ethnographic study on Mac indies by Meeteren (2008) and an oral history project on Mac culture provide data between 2001 and 2004
3. General analytical approach	Qualitative study with interpretive philosophy; combination of narrative (ideal types of institutional logic) and content analysis (shifting pattern of institutional logic); analyzing the relationship between the societal-level logics to elucidate the dynamics between the two logics at the field level
4. Ideal type construction of institutional logics and characteristics of resource environment	
4.1. Elemental categories of ideal types	From existing institutional logic studies
4.2. Data sources for the incumbent logic	Data sources on incumbent Mac developers, Mac culture and existing research on Mac indies (e.g. Meeteren 2008)
4.3. Data sources for the new logic	Data sources on incumbent Mac developers' changes, iOS-turned Mac developers, and existing studies on iOS developers
4.4. Qualitative coding: started with discussion topic of *MacSB* on Yahoo! Group	Divided listserv threads into 3 time periods in accord with Apple's governance change
4.4.1. General coding framework	Used stakeholder perspective to map the comprehensiveness of the elemental categories and to explicate characteristics of the resource environment; 5 categories emerged
4.4.2. Combination of deductive and inductive coding approach	Relevant literature: platform governance, Mac indies and iOS developers
4.4.3. Guiding principle of logic change coding	Used themes generated in the 1st phase as a baseline; any new meanings attached to the existing themes or brand new themes were classified as the new institutional logic attributed to the platform governance change. Two themes related to new logic also present in phase 1
4.4.4. Coding approach	Followed Strauss and Corbin's approach

	(1998): open coding, axial coding and selective coding. Extensive memos taken during coding process
4.4.5. Inter-coder reliability	Cohen's kappa is 0.75
4.4.6. Complete coding scheme generated	Table 11
4.5. Qualitative coding: continued with supplemental online archival sources	Used snowball and theoretical sampling to obtain 77 pieces of text
4.5.1. Coding of supplemental online archival sources	Used same coding scheme from the listserv discussion
4.5.2. New theme emerged: "platform organizational learning"	Obtained additional 6 pieces of text on Apple's iTunes
4.6. Mapping coding scheme and existing studies with elemental categories of ideal types	Elemental categories for a changed theme do not have to be the same
5. Content analysis	Used results from listserv discussion coding to quantitatively show incumbent Mac developers' temporal shift in logics
5.1. Change of distribution of two logics in each category over time	Percentage score calculated
5.2. Change of percentage of each category over time	Percentage score calculated

Table 2a. "Revised Interinstitutional System Ideal Types" in Thornton, Ocasio and Lounsbury (2012: 73)

Y-Axis:	**X-Axis: Institutional Orders**						
Categories	**Family 1**	**Community 2**	**Religion 3**	**State 4**	**Market 5**	**Profession 6**	**Corporation 7**
Root Metaphor 1	Family as firm	Common boundary	Temple as bank	State as redistributio n mechanism	Transaction	Profession as relational network	Corporation as hierarchy
Sources of Legitimacy 2	Unconditional loyalty	Unity of will Belief in trust & reciprocity	Importance of faith & sacredness in economy & society	Democratic participation	Share price	Personal expertise	Market position of firm
Sources of Authority 3	Patriarchal domination	Commitment to community values & ideology	Priesthood charisma	Bureaucratic domination	Shareholder activism	Professional association	Board of directors Top managem ent
Sources of Identity 4	Family reputation	Emotional connection Ego-satisfaction & reputation	Association with deities	Social & economic class	Faceless	Association with quality of craft Personal reputation	Bureaucratic roles
Basis of Norms 5	Membership in a household	Group membership	Membership in congregation	Citizenship in nation	Self-interest	Membership in guild & association	Employment in firm
Basis of Attention 6	Status in household	Personal investment in group	Relation to supernatural	Status of interest group	Status in market	Status in profession	Status in hierarchy
Basis of Strategy 7	Increase family honor	Increase status & honor of members & practices	Increase religious symbolism of natural events	Increase community good	Increase efficiency profit	Increase personal reputation	Increase size & diversifica tion of firm
Informal Control	Family politics	Visibility of	Worship of	Backroom	Industry	Celebrity	Organization

Mechanisms 8		actions	calling	politics	analysts	professionals	culture
Economic System 9	Family capitalism	Cooperative capitalism	Occidental capitalism	Welfare capitalism	Market capitalism	Personal capitalism	Managerial capitalism

Table 2b. "Ideal types of institutional logics in architecture" in Thornton, Jones and Kury (2005: 144)

Table 2. Ideal Types of Institutional Logics in Architecture.

Characteristic	Aesthetic Logic	Efficiency Logic
Economic system	Personal capitalism	Managerial capitalism
Sources of identity	Architect as artist–entrepreneur	Architect as engineer–manager
Sources of legitimacy	Reputation of architect Aesthetics of design	Scale and scope of firm Efficiency and economics of design
Sources of authority	Design prowess	Managing partner or supervisor
Basis of mission	Build personal reputation Build prestige of firm	Build multidisciplinary firm Build market position of firm
Basis of attention	Resolve design problems and entrepreneurial challenges	Resolve technological and organizational challenges
Basis of strategy	Increase prestige of patron or government sponsor Win design competitions	Increase number of corporate clients and engagement frequency Increase markets for services
Logic of investment	Build wealth and prestige of entrepreneurs	Build wealth of partners
Governance mechanism	Entrepreneurial firm (atelier) Profession	Partnership ownership Private global multidisciplinary corporation
Institutional entrepreneurs	H. H. Richardson, R. M. Hunt, R. R. Ware, Robert Venturi	Louis Sullivan, Wm Le Baron Jenney, Walter Gropius, Mies Van der Rohe

Figure 2. Stakeholder-view as a general coding framework. This figure depicts how resource environment affects developers' relationship with their stakeholders and their own identities and practices. In oval shapes are developers' relationships with stakeholders; in rectangles are developers' identities and practices.

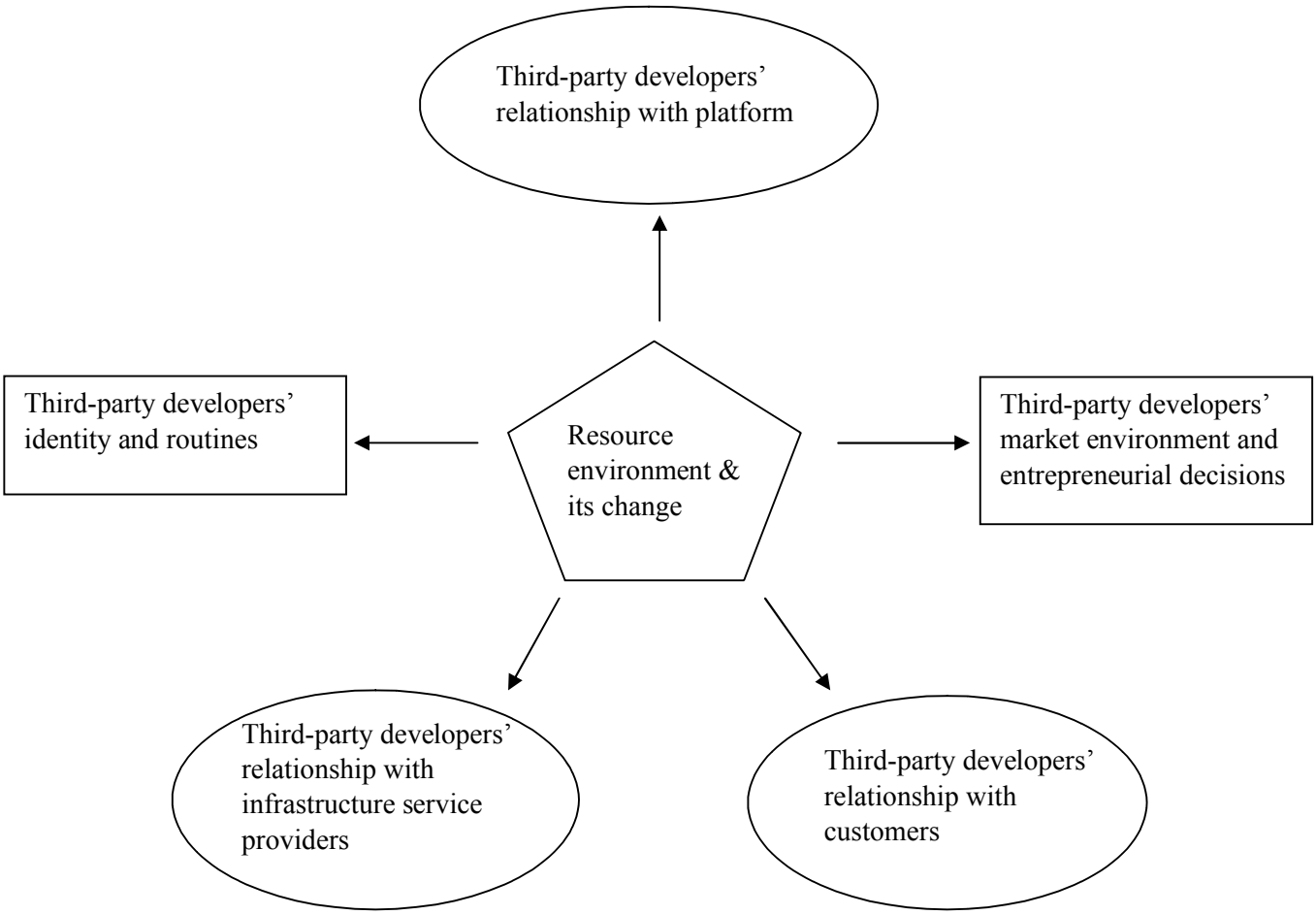


Table 3.

Two ideal types of Mac developers' institutional logics

Characteristics	Software ecosystem logic	Platform ecosystem logic
Societal-level logics	• Market (specialist) and Profession	• Market (generalist) and corporation
Economic system	• Market capitalism and personal capitalism	• Market capitalism and managerial capitalism
Sources of identity	• Independent third-party developer entrepreneur	• Subordinate third-party developer entrepreneur
Sources of legitimacy	• Quality of apps • Reputation of developer • Platform recognition via awards	• Chart ranking of apps • Platform recognition via featuring • End users' reviews and ratings
Sources of authority	• Software engineering and design prowess • Market acceptance	• Platform's review process • End user interests
Basis of norms	• Membership in community of practice	• Self-interest • Dependency on the platform
Basis of mission	• Build sustainable business • Increase sales	• Build competitive position of apps • Build apps to be favored by the platform
Basis of attention	• Resolve entrepreneurial challenges • Implement infrastructure best suited for the business • Adapt to platform's system progress and technology change	• Stand out in competition • Embrace the mass app users • Adapt to platform's changing technological and administrative policies
Basis of strategy	• Organic growth	• "Hit" growth
Platform-developer coordination	• Productive programming environment • Culture and affective attractiveness of the platform • Platform's formal and informal relationship with developers • Legal rules	• Pricing • Platform rules and regulations

Figure 3. The material and symbolic aspects of resource environment in two institutional logics of Mac developers. There are two layers of resource environment in each logic. Platform influences the customer base of developers' economy in software ecosystem logic, and it influences the entire developers' economy in the platform ecosystem logic.

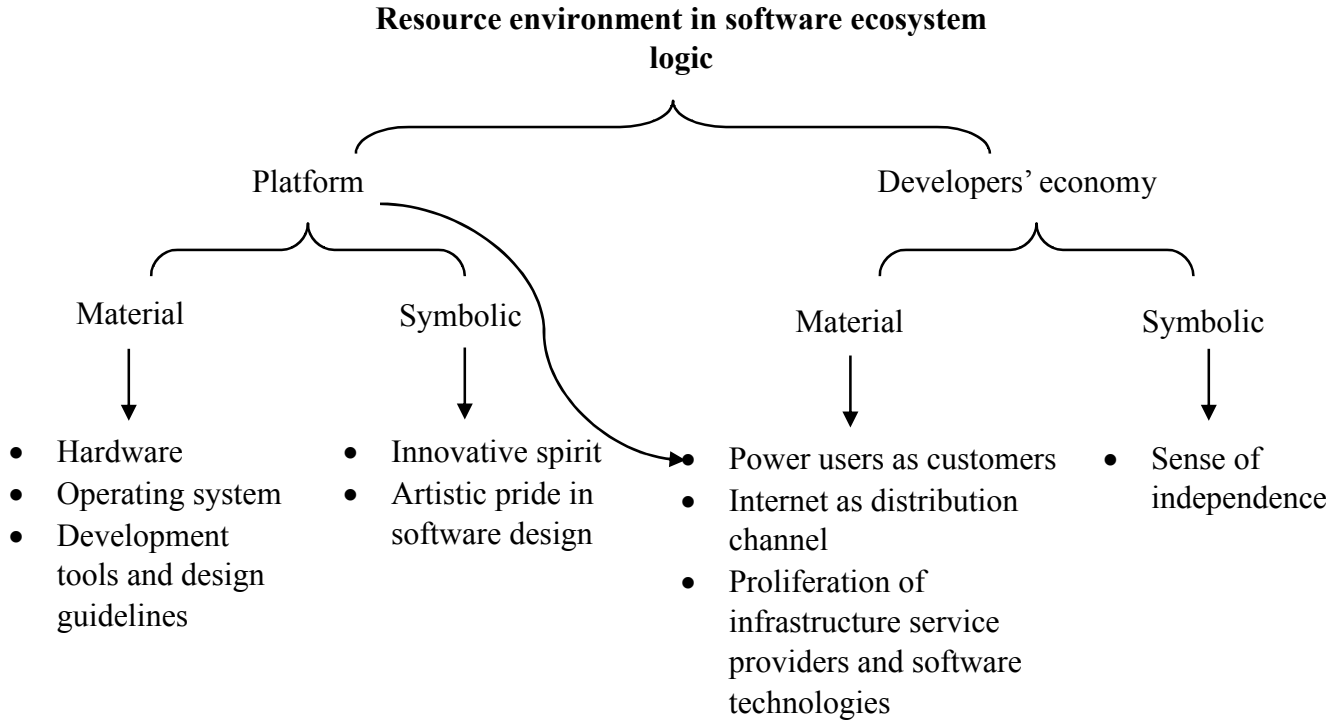


Figure 3. (Cont'd)

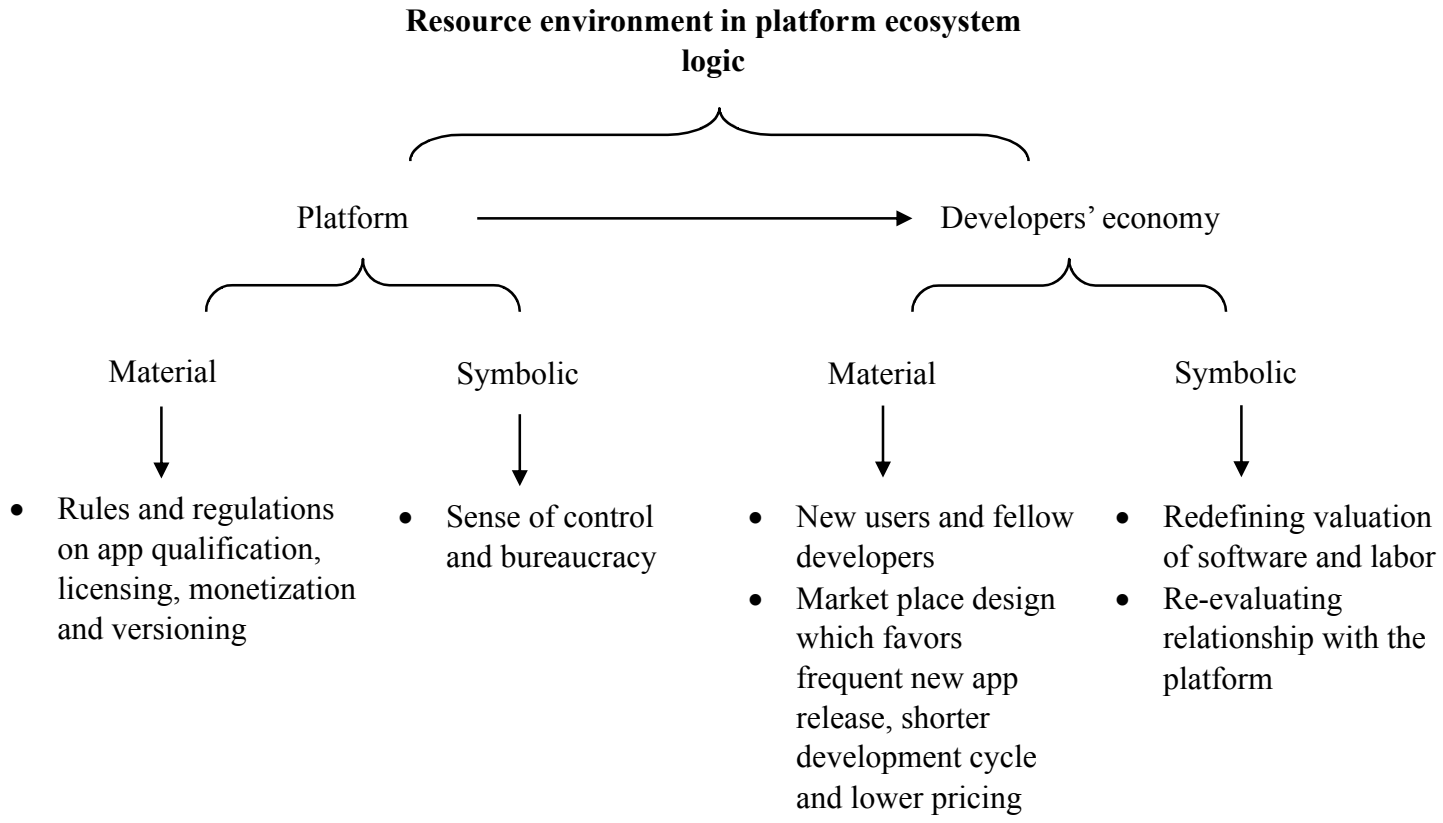


Table 4. Coding on "relationship with customers"

Percentage of customer incidents in each phase	**PHASE 1**[b]	11.35%[c]	**PHASE 2**	13.06%			**PHASE 3**	16.67%		
	Software logic		Software logic		Platform logic		Software logic		Platform logic	
	COUNT	%	COUNT	%	COUNT	%	COUNT	%	COUNT	%
Total coding incidents of customer per phase per logic	**88**[d]	**100.00%**	**57**	**85.07%**[g]	**10**	**14.93%**[g]	**6**	**18.75%**	**26**	**81.25%**
Themes										
Versioning and upgrade	**25**	**28.41%**[e]	**8**	**14.04%**	**3**	**30.00%**	**1**	**16.67%**	**18**	**69.23%**
Trial / Demo [a]	22	88.00%[f]	6	75.00%	1	33.33%	1	100.00%	6	33.33%
Upgrade	2	8.00%	1	12.50%	2	66.67%	0	0.00%	4	22.22%
Versioning	1	4.00%	1	12.50%			0	0.00%		
Multi-channel management					0	0.00%			8	44.44%[h]
User licensing	**28**	**31.82%**	**17**	**29.82%**	**3**	**30.00%**	**0**	**0.00%**	**2**	**7.69%**
Tech support	**35**	**39.77%**	**32**	**56.14%**	**4**	**40.00%**	**5**	**83.33%**	**6**	**23.08%**
Support challenges and approach	5	14.29%	9	28.13%	2	50.00%	4	80.00%	4	66.67%
Handling difficult customers	8	22.86%	3	9.38%	2	50.00%	1	20.00%	2	33.33%
Handling refund	6	17.14%	4	12.50%			0	0.00%		
Tools for support or feedback	16	45.71%	16	50.00%			0	0.00%		

[a] Highlighted in yellow are the themes whose meaning changed in the platform ecosystem logic compared to the software ecosystem logic.

[b] Phase 1: prior to announcement of iOS App Store; phase 2: after announcement of iOS App Store and prior to announcement of Mac App Store; phase 3: after announcement of Mac App Store

[c] In phase 1, 11.35% of developers' total discussions on the listserv is on relationship with customers.
[d] "COUNT" denotes number of incidents coded. In phase 1, 88 incidents were about relationship with customers
[e] Among all discussions on relationship with customers, 28.41% are about "versioning and upgrade"
[f] Among all "versioning and upgrade" codes, 88.00% are on "Trial / demo"
[g] In phase 2, 85.07% of all discussions on relationship with customers reflect software ecosystem logic, and 14.93% reflect platform ecosystem logic
[h] "Multi-channel management" is a brand new sub-theme for platform ecosystem logic; it consists of 44.44% of "versioning and upgrade" in phase 3

Table 5. Coding on "relationship with platform Apple"

Percentage of platform incidents in each phase	**PHASE 1 11.35%**		**PHASE 2 11.31%**				**PHASE 3 21.35%**			
	Software logic		Software logic		Platform logic		Software logic		Platform logic	
	COUNT	%	COUNT	%	COUNT	%	COUNT	%	COUNT	%
Total coding incidents of platform per phase per logic	**88**	**100.00%**	**43**	**74.14%**	**15**	**25.86%**	**10**	**24.39%**	**31**	**75.61%**
Themes										
3rd-party developers as modular to a system	**29**	**32.95%**	**12**	**27.91%**		0.00%	**6**	**60.00%**		0.00%
Developers' decisions on OS compatibility or choice of development machines	18	62.07%	7	58.33%		0.00%	6	100.00%		0.00%
Impact of platform technology transition on devs	11	37.93%	2	16.67%		0.00%	0	0.00%		0.00%
Impact of platform's clone machine on devs	0	0.00%	1	8.33%		0.00%	0	0.00%		0.00%
Impact of platform's UI or technology design on devs	0	0.00%	2	16.67%		0.00%	0	0.00%		0.00%
Strategic relationship between platform and 3rd-party developers	**27**	**30.68%**	**19**	**44.19%**	**7**	**46.67%**	**2**	**20.00%**	**5**	**16.13%**
Platform's entry into developers' turf	6	22.22%	1	5.26%	2	28.57%	0	0.00%	2	40.00%
Developers' reliance on platform's installed base	11	40.74%	5	26.32%	5	71.43%	0	0.00%	0	0.00%
Platform assisting developers' marketing	10	37.04%	13	68.42%	0	0.00%	2	100.00%	3	60.00%
Platform's coordination with developers	**32**	**36.36%**	**12**	**27.91%**	**8**	**53.33%**	**2**	**20.00%**	**26**	**83.87%**

Platform's formal developer relationship program	13	40.63%	4	33.33%		0.00%	0	0.00%		0.00%
Platform's rules and regulations	11	34.38%	5	41.67%	8	100.00%	0	0.00%	26	100.00%
Platform's cultural influence on developers	8	25.00%	3	25.00%		0.00%	2	100.00%		0.00%

Table 6. Coding on "relationship with infrastructure service providers"

Percentage of infrastructure incidents in each phase	**PHASE 1**	13.68%	**PHASE 2**	11.89%			**PHASE 3**	6.77%		
	Software logic		Software logic		Platform logic		Software logic		Platform logic	
	COUNT	%	COUNT	%	COUNT	%	COUNT	%	COUNT	%
Total coding incidents of infrastructure per phase per logic	**106**	**100.00%**	**60**	**100.00%**	**N/A**	**N/A**	**12**	**100.00%**	**N/A**	**N/A**
Themes										
App aggregators	**20**	**18.87%**	**10**	**16.67%**	**N/A**	**N/A**	**5**	**41.67%**	**N/A**	**N/A**
Payment or e-commerce services	**67**	**63.21%**	**39**	**65.00%**	**N/A**	**N/A**	**4**	**33.33%**	**N/A**	**N/A**
Hosting services	**13**	**12.26%**	**7**	**11.67%**	**N/A**	**N/A**	**3**	**25.00%**	**N/A**	**N/A**
Auto update services	**5**	**4.72%**	**4**	**6.67%**	**N/A**	**N/A**	**0**	**0.00%**	**N/A**	**N/A**
Software publishing services	**1**	**0.94%**	**0**	**0.00%**	**N/A**	**N/A**	**0**	**0.00%**	**N/A**	**N/A**

Table 7. Coding on "market competition / indie business strategies"

Percentage of market / strategy incidents in each phase	**PHASE 1** 34.45%				**PHASE 2** 37.04%				**PHASE 3** 30.73%			
	Software logic				Software logic		Platform logic		Software logic		Platform logic	
Total coding incidents of market / strategy per phase per logic	COUNT **262**	% **98.13%**	COUNT **5**	% **1.87%**	COUNT **183**	% **96.32%**	COUNT **7**	% **3.68%**	COUNT **43**	% **72.88%**	COUNT **16**	% **27.12%**
Themes												
External environment	**38**	**14.50%**			**17**	**9.29%**	**3**	**42.86%**	**2**	**4.65%**	**2**	**12.50%**
Dealing with piracy	26	68.42%	0	0.00%	14	82.35%	0	0.00%	2	100.00%	1	50.00%
Product, platform competition and competitors	9	23.68%	0	0.00%	3	17.65%	2	66.67%	0	0.00%	0	0.00%
Core customer base identification	3	7.89%	0	0.00%	0	0.00%	1	33.33%	0	0.00%	1	50.00%
Entrepreneurial decisions	**224**	**85.50%**	**5**	**100.00%**	**166**	**90.71%**	**4**	**57.14%**	**41**	**95.35%**	**14**	**87.50%**
Platform choice (multi-homing)	8	3.57%			6	3.61%			3	7.32%		
Revenue and licensing model choice	15	6.70%			0	0.00%			0	0.00%		
Pricing strategy	30	13.39%	0	0.00%	13	7.83%	3	75.00%	1	2.44%	8	57.14%
App product strategy	15	6.70%	3	60.00%	14	8.43%	1	25.00%	0	0.00%	1	7.14%
App releasing strategy	14	6.25%	2	40.00%	3	1.81%	0	0.00%	0	0.00%	1	7.14%
Marketing and PR strategy	117	52.23%	0	0.00%	102	61.45%	0	0.00%	33	80.49%	4	28.57%
Business expansion or discontinuance, alternative entrepreneurial options	25	11.16%			28	16.87%			4	9.76%		

Table 8. Coding on "developers' identity and business routines"

Percentage of identity / routine incidents in each phase	**PHASE 1**	29.16%	**PHASE 2**	26.90%			**PHASE 3**	25.00%		
	Software logic		Software logic		Platform logic		Software logic		Platform logic	
	COUNT	%	COUNT	%	COUNT	%	COUNT	%	COUNT	%
Total coding incidents of identity / routine per phase per logic	**226**	**100.00%**	**130**	**94.20%**	**8**	**5.80%**	**34**	**70.83%**	**14**	**29.17%**
Themes										
Identity of an indie app entrepreneur	**108**	**47.79%**	**43**	**33.08%**	**4**	**50.00%**	**7**	**20.59%**	**9**	**64.29%**
Individual-level identity	45	41.67%	23	53.49%	2	50.00%	2	28.57%	6	66.67%
Collective identity in a community of practice	63	58.33%	20	46.51%	2	50.00%	5	71.43%	3	33.33%
Creation of apps other than coding	**53**	**23.45%**	**36**	**27.69%**			**11**	**32.35%**		
Documentation or technical writing resources	10	18.87%	6	16.67%			0	0.00%		
App build choice	8	15.09%	2	5.56%			1	9.09%		
Beta testing resources	8	15.09%	7	19.44%			3	27.27%		
Aesthetic touch of the app or website	19	35.85%	16	44.44%			6	54.55%		
User experience consideration in app design	8	15.09%	5	13.89%			1	9.09%		
Small business operation	**65**	**28.76%**	**51**	**39.23%**	**4**	**50.00%**	**16**	**47.06%**	**5**	**35.71%**
Financial operations	21	32.31%	16	31.37%	2	50.00%	5	31.25%	1	20.00%
Legal issues	31	47.69%	24	47.06%	2	50.00%	9	56.25%	4	80.00%
Administrative process	13	20.00%	11	21.57%			2	12.50%		

Table 9. Changing pattern of each stakeholder category over time. Phase 1: prior to announcement of iOS App Store; Phase 2: after announcement of iOS App Store and prior to announcement of Mac App Store; Phase 3: after announcement of Mac App Store

Categories	PHASE 1	PHASE 2	PHASE 3
Percentage of platform incidents	11.35%	11.31%	21.35%
Percentage of customer incidents	11.35%	13.06%	16.67%
Percentage of infrastructure incidents	13.68%	11.70%	6.25%
Percentage of market / strategy incidents	34.45%	37.04%	30.73%
Percentage of identity / routine incidents	29.16%	26.90%	25.00%
Total	99.99%	100.01%	100.00%

Table 10. Changing distribution of software and platform logics in each stakeholder category over time

	PHASE 1				PHASE 2				PHASE 3			
	Software logic		Platform logic		Software logic		Platform logic		Software logic		Platform logic	
Categories	COUNT	%	COUNT	%	COUNT	%	COUNT	%	COUNT	%	COUNT	%
Developers' relationship with platform	88	100.00%	0	0.00%	43	74.14%	15	25.86%	10	24.39%	31	75.61%
Developers' relationship with customers	88	100.00%	0	0.00%	57	85.07%	10	14.93%	6	18.75%	26	81.25%
Developers' relationship with infrastructure providers	106	100.00%	0	0.00%	60	100.00%	0	0.00%	12	100.00%	0	0.00%
Developers' market environment and strategies	262	98.13%	5	1.87%	183	96.32%	7	3.68%	43	72.88%	16	27.12%
Developers' identity and routines	226	100.00%	0	0.00%	130	94.20%	8	3.16%	34	70.83%	14	29.17%
Mean		99.63%		0.37%		89.95%		9.53%		57.37%		42.63%

Figure 4. Conflict between software ecosystem logic and platform ecosystem logic for incumbent Mac developers and influence from platform organizational learning

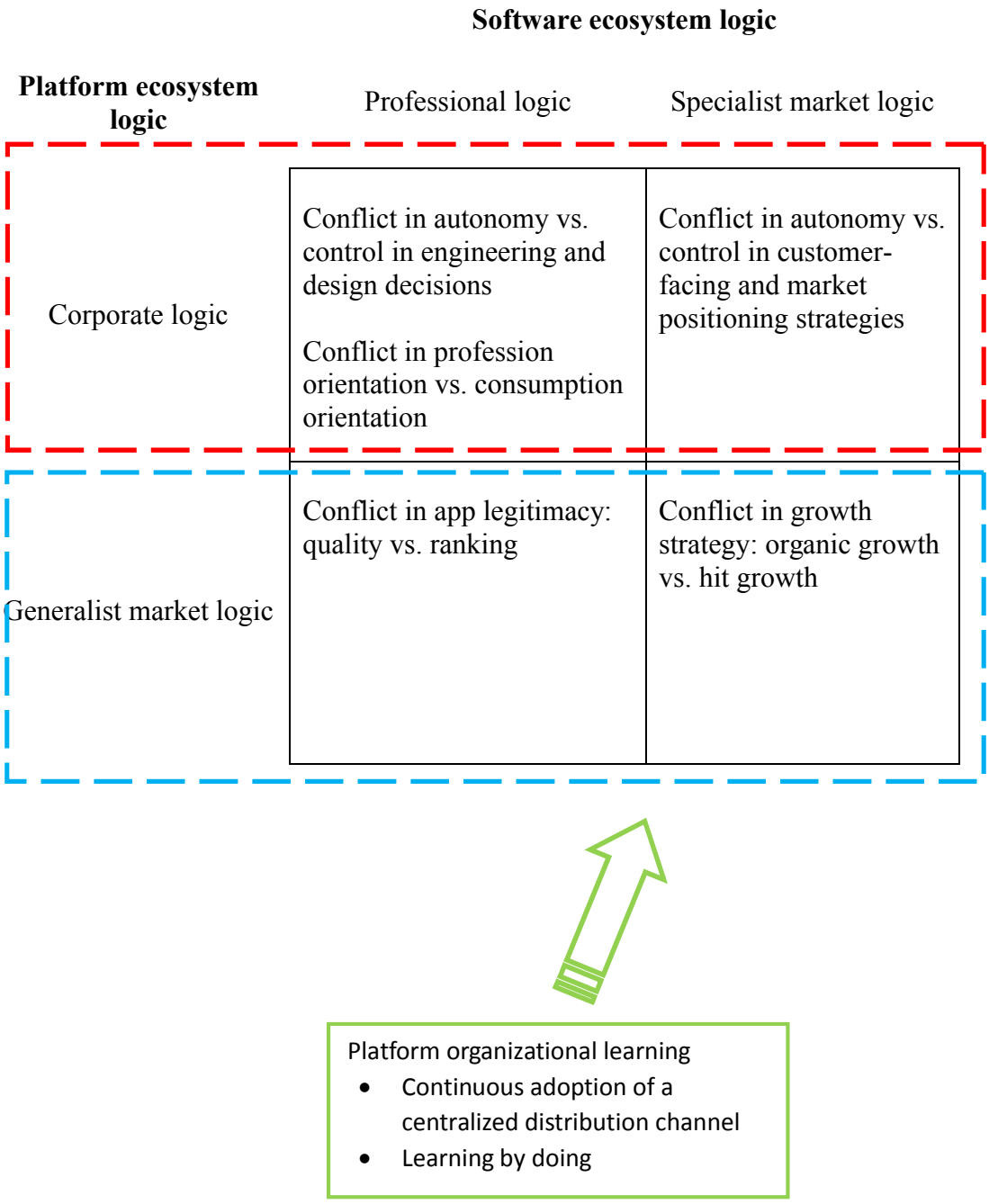


Table 11. Coding scheme developed from the *MacSB* listserv and applied in supplemental data sources to capture developers' institutional logics before and after platform's governance change. Highlighted in yellow are the themes whose meaning changed or whose sub-themes' meaning changed in the platform ecosystem logic compared to the software ecosystem logic. New meanings are included in the "platform ecosystem logic" column. Theme "Multi-channel management" only appeared in the platform ecosystem logic. Highlighted in blue are the elemental categories of the ideal types – this indicates the mapping between themes developed according to the stakeholder framework and elemental categories of the ideal types. It is worth noting that when a theme of software ecosystem logic changed meaning in the platform ecosystem logic, its corresponding elemental category of the ideal types could change, too. For instance, for the sub-theme "Technical requirement adherence" under "Platform's coordination with developers", it belongs to the elemental category "Platform-developer coordination" in the software ecosystem logic, and "Sources of authority" in the platform ecosystem logic.

Software ecosystem logic	**Platform ecosystem logic**
Relationship with platform Apple	
1. 3rd-party development as modular component to a system (Basis of attention)	
a. Developers' decisions on OS compatibility or choice of development machines	
b. Impact of platform technology transition on developers	
c. Impact of platform's clone machines on developers	
d. Impact of platform's UI or technology (OS) design on developers	
2. Strategic relationship between platform and 3rd-party devs	
a. Platform's entry into developers' turf: product market clash (Sources of identity)	Distribution channel clash (Sources of identity)
b. Developers' reliance on platform's installed base (relevant discussions on platform hardware market share and desktop platform competition) (Sources of identity)	Within-platform competition (traditional Mac platform competing for the up-and-coming mobile platform of the company) (Sources of identity)
c. Platform assisting developers' marketing: devs	Platform featuring on

listing apps on Apple download.com; Apple employee relationship marketing (Sources of legitimacy)	the App Store (Sources of legitimacy)
3. Platform's coordination with developers	
a. Platform's formal developer relationship program: ADC (Apple Developer Connection), WWDC and platform's privileged developer relationship program: ADA (Platform-developer coordination)	
b. Platform's rules and regulations (Platform-developer coordination)	
i. Legal aspect related to the platform: use of platform's graphic assets / Apple's trademark in developer's app; Apple's NDA	App Store membership enrollment and approval status (Platform-developer coordination)
ii. Technical requirement adherence	App Store review guidelines (Sources of authority)
c. Platform's cultural influence on developers: definition of a good product by Mac standards; impact of platform design philosophy on 3rd-party's design in icon, UI, and websites (Sources of legitimacy)	
Market or competition / indie business strategies	
1. External environment	
a. Piracy concerns / issue / overcome strategies (reaction: upset to happy; strategy: blocking to not bother) (Basis of attention)	Piracy falls under platform's responsibility (Platform-developer coordination)
b. Product, platform competition or competitor type (Basis of attention)	New devs tend to not aim for high quality in app; competitors cheating, writing negative reviews (Basis of strategy)
c. Core customer base identification (who are they: power users or average users) (Economic system)	Leisure users, viewing apps more like a form of entertainment rather than utility (Economic system; Basis of strategy)
2. Entrepreneurial decisions (Basis of attention)	

a. Platform choice (multi-homing): web vs. native debate; cross-platform development	
b. Revenue / licensing model choice (source code: open vs. closed source; payment options: commercial vs. free; voluntary pay (donation ware) vs. pay-after-trial (shareware))	
i. OSS and free vs. commercialized / shareware OSS	
ii. Closed source free vs. closed source commercial / shareware	
iii. Shareware vs. donationware vs. freeware	
c. Pricing strategy (Basis of strategy)	i. Low pricing related to top charts and store trend; (Basis of strategy) ii. Strong Price elasticity (Basis of strategy)
i. General question about determining price on an app	
ii. Pricing related to perceived customer reaction	
iii. Pricing related to support cost	
iv. Upgrade pricing	
v. Pricing based on competition	
vi. Price related to the macro-economy, i.e., inflation	
d. App product strategy (Basis of strategy)	
i. App market identification	
ii. App portfolio strategy (product diversification)	App portfolio strategy (aggregate sales of small apps) (Basis of strategy)
iii. Degree of synthesis	
1. Synthesis in ideation	
2. Synthesis in execution	
e. App releasing strategy (Basis of strategy)	
i. Optimal release timing	
ii. Releasing drama creation (buzz first or product first; secrecy or openness with the press)	
iii. Frequent releasing strategy (for marketing coordination or obtaining feedback)	Frequent releasing strategy (to obtain eye-balling and visibility on download sites)

	(Basis of strategy)
f. Marketing and PR strategy	i. Strategies on ranking, charting; attracting more and positive user reviews; competing for user attention (Basis of legitimacy)
i. General marketing approach or questions	
ii. Word of mouth marketing: PR services / PR release strategies & Review site / bloggers / "High-status dev"/ MUG (Mac user group) / user reviews	
iii. Online advertising	
iv. Relationship building with customers; social networking sites	
v. Affiliate marketing & advertising revenue	
vi. Price promotions / discounts	
vii. Branding / product or company image building	
1. General branding	
2. Product-centric or company-centric marketing	
3. Naming: naming the app, company, domain, or version	
viii. Marketing tools in connecting with customers	
1. Customer info management / CRM	
2. Virtual telephone service	
3. Teasing feature	
4. Blogging software	
5. Screencaster service	
6. Email marketing tools	
7. SEO & Web analytics	
ix. Distribution channel options (options contrasting to the App Store model; questions / pros and cons): i.e., physical CD; site licenses / family pack; retail store; MacWorld Expo; resellers; magazine; bundled sales (i.e., MacZot)	
g. Business expansion or discontinuance, alternative business options for indie	
i. Business acquisition	

ii. Localization and internationalization	
iii. Selling business or apps	
iv. Product or old version discontinue	
v. Indie contract work or consulting	
vi. Custom development for client	
Relationship with customers (Basis of attention)	
1. Shareware versioning	
a. Trial / Demo: time limited vs. feature limited; Pros and Cons about design issue	Trial mode change: no time-limited demo (Basis of attention)
b. Upgrade: design of upgrade policy	Upgrade process change: no paid upgrades (Basis of attention)
c. Software versioning	
	Multi-channel management (Basis of attention)
2. User licensing; licensing scheme design; market segment validation; product activation; distributing testing / reviewer copies	Licensing scheme changed: sharing testing / reviewer copies becomes less easy (Basis of attention)
3. Tech support	
a. Indie developers' support challenges and approach (support norm / philosophy; challenges including factors uncontrollable by devs)	The challenge became blocked direct communication with users, so developers had to come up with alternative ways for support (Basis of attention)
b. Handling difficult customers (dealing with negative user reviews)	"Immature" user behavior (Basis of attention)
c. Handling refund	
d. Tools for user support or feedback	
i. Tools in general	
ii. Bug / support / crash / feature request / report tracking system	
iii. Forum tools	
iv. Mailinglist management service	

Relationship with infrastructure service providers (Basis of attention)	
1. App aggregators (i.e., MacUpdate, Version Tracker, Apple download, CNET download; Bodega)	
2. Payment, e-commerce or licensing services providers (i.e., Kagi, eSellerate, Paypal; AquaticPrime, etc.)	
3. Hosting services (including email service / server)	
4. Auto update services: Sparkle	
5. Software publishing services	
Developers' identity and business routines	
1. Identity of an indie app entrepreneur	
a. Individual-level identity	
i. Independence: bootstrapping business instead of taking external funds; independence from working for others; independence from a large corporation or large player in the industry; freedom in choosing the kind of technology to use; regarding the term "sharecropper", it means the potential risk of "product clashing" with platform offerings (Sources of identity)	Decreased level of independence: welcoming the idea of App Store or resisting it; regarding the term "sharecropper", it means developers need to follow the technical and administrative rules and regulations from the platform (Sources of identity)
ii. Life-work balance / sustainability / day job-indie balance (Basis of mission)	
iii. Norms or ethics of being an indie or a software developer (Basis of norms)	
iv. Going indie (how to start out) / new product announcement / achievement announcement (Basis of mission)	
b. Collective identity (Basis of norms)	
i. Developer-collaboration / Co-op effort in marketing and product or business assessment	Organizing MAS alternative (Sources of identity)
ii. Knowledge resource or information sharing	
iii. Etiquette in treating competitors in the Mac business world	
iv. Sales pattern sharing / analysis to stimulate discussions on the list	App Store sales pattern analysis: comparing with the web store sales (Basis

	of attention)
2. Creation of the app itself other than the coding part (Sources of legitimacy)	
a. Documentation or technical writing resources	
b. App build choice: i.e., DiskImage or Zip	
c. Beta testing resources	
d. Aesthetic touch of the app / website	
i. Designer or design software choice	
ii. Audio engineer	
iii. Website building and management tools	
e. User experience design	
3. Small business operation (Basis of attention)	
a. Financial operations	i. Depending on platform for disbursement / payment (Platform-developer coordination)
i. Cost of ISV / financial matters	
ii. Insurance: health insurance / property insurance	
iii. Product tax-related or banking issues	
iv. Accounting software / accountant / payroll software	
b. Legal issues	i. Requirement from authority higher than the platform (i.e., U.S. government) (Platform-developer coordination) ii. iTunes Connect account (Platform-developer coordination)
i. General legal inquiry (copyright; incorporation)	
ii. Business structure / entity (incorporation / LLC / sole proprietorship)	
iii. Business liability	
1. Product liability insurance / merchant account	

2. Liability in promotion activities	
3. EULA / legalese / privacy policy	
4. Backup strategy / code escrow	
iv. Existing or potential legal dispute: Trademark / app (company) naming collision / copyright / clone; legal fight	
v. OSS licensing (using or licensing under OSS licenses)	
vi. Potential legal conflict between day job and moonlighting job	
c. Administrative process	
i. Formalizing business process (i.e., checklist, formal business plan, certification)	
ii. Personnel structure	
iii. Project management tools / time tracking tools / within-company communication tool	

References

Arment, Marco. 2012. The Mac App Store's future of irrelevance. July 26, 2012. http://www.marco.org/2012/07/26/mac-app-store-future

Baldwin, C. Y. and K. B. Clark, 2000, Design rules: The power of modularity. Cambridge, MA MIT Press.

Baldwin, C. Y. and C. J. Woodard, 2009, The architecture of platforms: A unified view. In A. Gawer (ed.) Platforms, markets and innovation. Cheltenham, UK and Northampton, MA, US: Edward Elgar.

Bergvall-Kåreborn, Birgitta; Howcroft, Debra; and Chincholle, Didier. 2010. Outsourcing creative work: a study of mobile application development. ICIS 2010 Proceedings. Paper 23.

Boudreau, K. 2012. Let a thousand flowers bloom? An early look at large numbers of software app developers and patterns of innovation. Organization Science, 23(5), pp.1409-1427.

Boudreau, K. 2010. Open platform strategies and innovation: Granting access vs. devolving control. *Management Sci*. 56(10) 1849-1872.

Carroll, G.R. 1985. Concentration and specialization: dynamics of niche width in populations of organizations. American Journal of Sociology, 90: 1262-1283

Carroll. G. and Swaminathan, A. 2000. "Why the Microbrewery Movement? Organizational Dynamics of Resource Partitioning in the US Brewing Industry." American Journal of Sociology, 106:715-762.

Chambers, E. 1985. Applied Anthropology: A Practical Guide. Prentice Hall, Inc., Englewood Cliffs, New Jersey. Reprinted in 1989 by Waveland Press, Prospect Heights, Illinois.

Coser, Lewis A. 1977. Masters of sociological thought: Ideas in historical and social context. Harcourt Brace Jovanovich (New York). 2nd edition

Cusumano, M. 2010. Technology strategy and management the evolution of platform thinking. Communications of the ACM 53 (1) 32-34.

DiMaggio, P., and W. Powell. 1983. The iron cage revisited: institutional isomorphism and collective rationality in organizational fields. American Sociological Review. 48, 147-160.

DiMaggio, P., and W. Powell. 1991. The iron cage revisited: institutional isomorphism and collective rationality in organizational fields in *The new institutionalism in organizational analysis*. W. Powell and P. DiMaggio (eds.), 63-82. Chicago: University of Chicago Press.

DiMaggio, Paul and Mullen, Ann L. 2000. Enacting community in progressive America: Civic rituals in national music week, 1924. Poetics (27), p135-162.

du Lac, J. Freedom. 2006. Downloads Make Singles a Hit Again. The Washington Post. February 8, 2006. http://www.washingtonpost.com/wp-dyn/content/article/2006/02/07/AR2006020702051_pf.html

Thomas R. Eisenmann, Geoffrey Parker, Marshall Van Alstyne 2009. Opening Platforms: How, When and Why? In Gawer, Annabelle (ed.), Platforms, Markets and Innovation, Edward Elgar: Cheltenham, UK.

Evans, David S., Richard Schmalensee. 2007. Industrial organization of markets with two-sided platforms. Competition Policy International 3(1) 151–179.

Frakes, Dan. 2010. The Mac App Store: The devil will be in the details. Oct 23, 2010. Macworld. http://www.macworld.com/article/1155120/mac_app_store_devil_in_the_details.html

Friedman, Lex. 2012. Sandboxing deadline arrives: What it means for Apple, developers, and you. Macworld.com. Jun 1, 2012. http://www.macworld.com/article/1167055/sandboxing_deadline_arrives_what_it_means_for_apple_developers_and_you.html

Gallaugher, J.M., Y. M. Wang. 2002. Understanding network effects in software markets: Evidence from Web server pricing. MIS Quart. 26(4), 303–327.

Garud, R., Jain, S., & Kumaraswamy, A. 2002. Institutional entrepreneurship in the sponsorship of common technological standards: The case of Sun Microsystems and Java. Academy of Management Journal, 45: 196 –214.

Gawer, Annabelle (ed.). 2009. Platforms, Markets and Innovation, Edward Elgar: Cheltenham, UK.

Cusumano, M.A. and Annabelle Gawer. 2002. The elements of platform leadership. Sloan management review. Spring, 51-58.

Gawer, A. 2010. Towards a general theory of technological platforms. DRUID conference presentation. Imperial College London Business School, June 16 - 18, 2010

Ghazawneh Ahmad & Henfridsson Ola. 2011. Micro-strategizing in platform ecosystems: a multiple case study. ICIS proceedings.

Ghazawneh Ahmad & Henfridsson Ola. 2010. Governing third-party development through platform boundary resources. ICIS proceedings.

Glynn, Mary Ann and Michael Lounsbury. 2005. 'From the Critics' Corner: Logic Blending, Discursive Change and Authencity in a Cultural Production System,' Journal of Management Studies 42 (5): 1031–1055.

Greenwood, Royston, and Roy Suddaby. 2006. 'Institutional Entrepreneurship in Mature Fields: The Big Five Accounting Firms,' Academy of Management Journal 49 (1): 27–48.

Hagiu, Andrei. 2007. "Merchant or Two-Sided Platform." Review of Network Economics 6, no. 2.

Hagiu, Andrei. 2009. Multi-Sided Platforms: From Microfoundations to Design and Expansion Strategies. Harvard Business School Working Paper, No. 09–115.

Hamburger, Ellis. 2012. on July 27, 2012. Sandbox of frustration: Apple's walled garden closes in on Mac developers. The Verge. http://www.theverge.com/2012/7/27/3186875/mac-app-store-sandboxing-frustration-mountain-lion

Haslam, Karen. 2012. Macworld UK. Sandboxing will "disadvantage Mac users," say developers. May 29, 2012.http://www.macworld.com/article/1166997/sandboxing_will_disadvantage_mac_users_say_developers.html

Haveman, Heather A., and Hayagreeva Rao. 1997. 'Structuring a Theory of Moral Sentiments: Institutional and Organizational Coevolution in the Early Thrift Industry,' American Journal of Sociology 102 (6): 1606–1651.

Hertzfeld, Andy. The Macintosh Spirit. Folklore.org. http://folklore.org/StoryView.py?project=Macintosh&story=The_Macintosh_Spirit.txt&topic=Apple%20Spirit&sortOrder=Sort%20by%20Date

Hertzfeld, Andy. The Apple Spirit.Folklore.org. http://folklore.org/StoryView.py?project=Macintosh&story=The_Apple_Spirit.txt&topic=Apple%20Spirit&sortOrder=Sort%20by%20Date&detail=medium

Hoffman, Andrew J. 1999. Institutional Evolution and Change: Environmentalism and the U.S. Chemical Industry. The Academy of Management Journal, 42(4), pp. 351-371

Huang, Peng; Ceccagnoli, Marco; Forman, Chris and Wu, D.J. 2013. Appropriability mechanisms and the platform partnership decision: evidence from enterprise software. Management Science, 59(1), pp. 102-121.

Huber, G. P. 1991. Organizational learning: the contributing processes and the literatures. Organ. Sci. 2(1) 88-115.

Iansiti , M. and R . Levien, 2004. The keystone advantage: What the new dynamics of business ecosystems mean for strategy, innovation, and sustainability. Boston, MA: Harvard University Press.

Ihnatko, Andy. Oct 2, 2011. App sandboxing risks eroding the Mac's identity. Macworld. http://www.macworld.com/article/1162504/app_sandboxing_risks_eroding_the_macs_identity.html

Katz, Michael L. & Shapiro, C. 1994. System competition and network effects. The Journal of Economic Perspectives, Vol. 8, No. 2, pp. 93-115

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.

Kitchener, Martin. 2002. 'Mobilizing the Logic of Managerialism in Professional Fields: The Case of Academic Health Centers Mergers,' Organization Studies 23 (3): 391–420.

Klein, Heinz K. and Michael D. Myers. 1999. A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. MIS Quarterly, 23 (1), 67-93.

Krippendorff, K. 2004.Content Analysis: An Introduction to Its Methodology. Sage. Thousand Oaks, CA.

Kudaravalli, Srinivas and Samer Faraj. 2008. The structure of collaboration in electronic network. Journal of the Association for Information Systems, 9(10/11), pp.706-726.

Kuk, George. 2006. Strategic interaction and knowledge sharing in the KDE developer mailing list. Management Science. 52(7), pp. 1031-1042.

Levitt, B., J. G. March. 1988. Organizational Learning. Annual Review of Sociology. (14) 319-340.

Lounsbury, M. 2007. A tale of two cities: competing logics and practice variation in the professionalizing of mutual funds. Academy of Management Journal, 50(2), p.289-307.

Lounsbury, Michael. 2002. 'Institutional Transformation and Status Mobility: The Professionalization of the Field of Finance,' Academy of Management Journal 45: 255–266.

Marquis, C. Lounsbury, M. 2007. Vive La Resistance: competing logics and the consolidation of U.S. community banking. Academy of Management Journal, 50(4), p. 799-820.

Meeteren, M. van. 2008. Indie fever: the genesis, culture and economy of a community of independent software developers on the Macintosh OS X platform. Bachelor thesis. Human geography, University of Amsterdam.

Meeteren, M. van. 2009. "Indie" desktop versus iPhone software development: A comparison of cultural value chains. Working paper.

Miles, M., A. M. Huberman. 1991. *Qualitative Data Analysis, 2nd edition*, Sage, Beverly Hills, CA.

Mogull, Rich. 2012. Answering Questions about Sandboxing, Gatekeeper, and the Mac App Store. TidBits. 25 Jun 2012. http://tidbits.com/article/13071

Mrgan, Neven. 2012. The Mac App Store's future. July 26 2012. http://mrgan.tumblr.com/post/28058883006/the-mac-app-stores-future

Myslewski. Rik. 2010. Apple posts $20bn+ quarter. 18th October 2010. The Register. http://www.theregister.co.uk/2010/10/18/apple_q4_2010/

Neuendorf, K. 2002.The Content Analysis Guide book. Sage, New York.

Orlikowski, Wanda J. and Baroudi, Jack J. 1991. "Studying Information Technology in Organizations: Research Approaches and Assumptions," Information Systems Research, 2(1), p1-28.

Parker G, Van Alstyne MW. 2012. Innovation, openness, and platform control. Working paper, Tulane University, New Orleans. http://ssrn.com/abstract=1079712.

Parker, Geoffrey G., Van Alstyne, Marshall W. 2005. Two-Sided Network Effects: A Theory of Information Product Design. Management Science, 51 (10), pp. 1494-1504.

Purdy and Gray. 2009. Conflicting logics, mechanism of diffusion, and multilevel dynamics in emerging institutional fields. Academy of Management Journal. 52(2), 355-380

Qiu, Y., Gopal, A. and Hann, I. 2012. On synthesizing professional and market logics in nascent entrepreneurship: a study of iOS app entrepreneurs. Working paper. R. H. Smith School of Business, University of Maryland, College Park.

Rao, Hayagreeva, Philippe Monin, and Rodolphe Durand. 2003. 'Institutional Change in Toque Ville: Nouvelle Cuisine as an Identity Movement in French Gastronomy,' American Journal of Sociology 108(4), 795–843.

Rao, Hayagreeva. 1998. "Caveat Emptor: The Construction of Nonprofit Consumer Watchdog Organizations." American Journal of Sociology 103:912–61.

Reay, Trish, and C. R. Hinings. 2005. The Recomposition of an Organizational Field: Health Care in Alberta, Organization Studies 26 (3): 351–384.

Reay, T., and Hinings, C. R. 2009. Managing the rivalry of competing institutional logics, Organization Studies (30:6), pp 629-652.

Rentzsch, Wolf and Pontious, Andrew. 2012. *Edge Cases*. Podcast, show #3. June 20, 2012. http://www.edgecasesshow.com/003-sandboxing-day.html

Rochet, J.C., J.Tirole. 2003. Platform competition in two-sided markets. J. Eur. Econom. Assoc. 1(4)990–1029.

Rochet, Jean-Charles, Jean Tirole. 2006. Two-sided markets: A progress report. The RAND Journal of Economics 37(3) 645– 667.

Ruef, Martin. 1999. Social ontology and the dynamics of organizational forms: creating market actors in the healthcare field, 1966-94. Social Forces, 77: 1405-34.

Rudmark, Daniel and Ghazawneh, Ahmad. 2011. Third-Party Development for Multi-Contextual Services: On the Mechanisms of Control. Proceedings of European Conference on Information Systems. Paper 162.

Sine, Wesley D., and David, Robert J. 2003. Environmental jolts, institutional change, and the creation of entrepreneurial opportunity in the US electric power industry. 32, pp.185-207.

Sprigman, Christopher. 2006. The 99¢ question. Journal on Telecommunication & High Technology Law, vol. 5, pp. 87-124.

Srinivasan, A., F. Suarez. 2010. First mover advantages in hyper-dynamic environments: a study of the iPhone ecosystem. *Presented at Acad. Management Conf.* August 6-11, Montreal, Canada

Strauss, Anselm and Corbin, Juliet, M. 1998. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. 2nd Ed. SAGE Publications. London.

Stasser, G. Information salience and the discovery of hidden profiles by decision making groups: a 'thought experiment', Organizational Behavior and Human Decision Processes 52 (1), 1992, pp. 156–181.

Suddaby, R. & Greenwood, R. 2005. Rhetorical strategies of legitimacy. Administrative Science Quarterly, 50: 35-67.

Swaminathan, A. 2001. Resource partitioning and the evolution of specialist organizations: The role of location and identity in the US wine Industry. Academy of Management Journal. 44: 1169-1185

Takeyama, Lisa. 1994. The shareware industry: some stylized facts and estimates of rates of return. Economics of innovation and new technology, 3(2), pp.161-174.

The Economist. 2004. Return of the homebrew coder. Mar 11th 2004 |From the print edition. http://www.economist.com/node/2476892

Thornton, P.H., W. Ocasio, M. Lounsbury. 2012. The Institutional Logics Perspective: a New Approach to Culture, Structure and Process, Oxford University Press, USA.

Thornton, P. H., W. Ocasio. 2008. Institutional Logics. Greenwood, R., C. Oliver, R. Suddaby, K. Sahlin-Andersson, eds. The Sage Handbook of Organizational Institutionalism. Sage Publications Ltd., 99-129

Thornton, P. H., C. Jones, K. Kury. 2005. Institutional logics and institutional change in organizations: transformation in accounting, architecture, and publishing. C. Jones, P. H. Thornton, eds. Transformation in Cultural Industries (Research in the Sociology of Organizations) Emerald Group Publishing Limited, 23 125-170

Thornton, P. H. 2004. Markets from Culture: Institutional Logics and Organizational Decisions in Higher Education Publishing. Stanford University Press, Stanford, CA.

Thornton, P. H. 2002. The rise of the corporation in a craft industry: conflict and conformity in institutional logics. Acad. Management J. 45(1) 81-101

Thornton, P. H., W. Ocasio. 1999. Institutional logics and the historical contingency of power in organizations: Executive succession in the higher education publishing industry, 1958–1990. American J. Sociol. 105(3) 801–843.

Thornton, P. H. 2001. Personal versus market logics of control: a historically contingent theory of the risk of acquisition. Organization Science. 12(3) 294-311.

Tiwana, A., Konsynski, B. and Bush, A.A. 2010. Platform evolution: coevolution of platform architecture, governance, and environmental dynamics. Information Systems Research, 21(4), PP. 675-687.

Townley, Barbara. 2002. The Role of Competing Rationalities in Institutional Change. Academy of Management Journal. 45(1), 163-179.

Townley, Barbara. 1997. The institutional logic of performance appraisal. Organization Studies, 18(2), pp. 261-285.

van Gestel, Nicolette and Hillebrand, Bas. 2011. Explaining Stability and Change: The Rise and Fall of Logics in Pluralistic Fields. Organization Studies, 32(2), 231-252.

Viera, Anthony J., and Garrett, Joanne M. 2005. Understanding interobserver agreement: the kappa statistic. Family Medicine, 37(5), pp. 360-363.

Wanda J. Orlikowski and JoAnne Yates. 1994. Genre repertoire: the structuring of communicative practices in organizations. Administrative Science Quarterly, 39(4), pp.541-574.

Weber, M. (1978). Economy and society: An outline of interpretive sociology. In: G. Roth & C. Witt ich (Eds). Berkeley, CA: University of California Press (Original work published 1922).

Weatherhead, Gabe. 2012. Litterboxing. June 21, 2012. http://www.macdrifter.com/2012/06/litter-boxing.html

West, J. 2003. How open is open enough? Melding proprietary and open source platform strategies. Res. Policy 32 (7) 1259–1285