Platform Evolution:
Coevolution of Platform Architecture, Governance, and Environmental Dynamics

Amrit Tiwana
University of Georgia

Benn Konsynski
Emory University

Ashley A. Bush
Florida State University

Forthcoming in *Information Systems Research*

Abstract

The emergence of software-based platforms is shifting competition towards platform-centric ecosystems, although this phenomenon has not received much attention in information systems research. Our premise is that the coevolution of the design, governance, and environmental dynamics of such ecosystems influences how they evolve. We present a framework for understanding platform-based ecosystems and discuss five broad research questions that present significant research opportunities for contributing homegrown theory about their evolutionary dynamics to the IS discipline, and distinctive IT-artifact-centric contributions to the strategy, economics, and software engineering reference disciplines.

Keywords: Platforms, ecosystem, architecture, modularity, environment, evolutionary dynamics, coevolution, governance

1. INTRODUCTION

Platform-based software ecosystems such as the Firefox browser and its 8,000 add-on "extensions" or Apple's iphone operating system (iOS) and its 140,000 "apps" are emerging as a dominant model for software development and software-based services. Unlike traditional software development, they leverage the expertise of a diverse developer community—with skills and an appreciation of user needs that platforms owners might not possess—to creatively develop new capabilities unforeseeable by the platform's original designers. The notion of platforms refers to disparate things in marketing (product lines), software engineering (software families), economics (products and services that bring together groups of users in two-sided networks (Eisenmann, Parker and van Alstyne 2006)), information systems (infrastructural investments (Fichman 2004)), and industrial organization (forming systems (Katz and Shapiro 1994)). Building on the Baldwin and Woodard's (2009) synthesis of the commonalities across these conceptualizations, we define a software-based *platform* as the extensible codebase of a software-

based system that provides core functionality shared by the modules that interoperate with it, and the interfaces through which they interoperate (e.g., Apple's iOS and Mozilla's Firefox browser). We define a *module* as an add-on software subsystem that connects to the platform to add functionality to it (e.g., iphone apps and Firefox extensions). We refer to the collection of the platform and the modules specific to that platform as that platform's *ecosystem* (Cusumano and Gawer 2002). (Figure 1 illustrates these distinctions, with the corresponding definitions in Table 1.) This combination of the platform and modules idiosyncratic to that platform can create formidable competitive barriers for rival platforms.

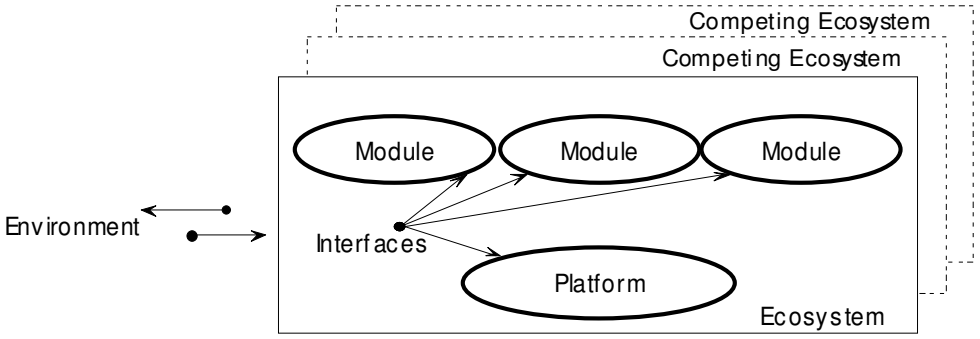

Figure 1: Elements of platform-centric ecosystems.

Software platforms present a significant challenge and opportunity for IS research for six reasons. First, competition is increasingly shifting towards competing platform-centric ecosystems (Katz and Shapiro 1994), unlike standalone systems that have been the mainstay in the IS systems development literature. This trend is pervasive in browsers (e.g., Firefox, Chrome, and Opera), smartphone operating systems (iphone, Android), Web services (Google Payments, Amazon Elastic Cloud), social media (Facebook, Apple's Ping), marketplaces (SABRE, eBay), and gaming consoles (Xbox, Apple's ipod Touch, Sony Playstation). Second, the conventional notion of firm boundaries is expanding to harness outside expertise and ingenuity on an unprecedented scale (e.g., over 100,000 developers for the iphone OS). In contrast, the IT-line function interface within the firm has historically been the locus of IS innovation in IS research. Prior IS development research cannot readily be applied here because conventional coordination mechanisms are not scalable to this magnitude. Third, technical architectures and organizing principles of these platforms *jointly* determine their evolutionary trajectories, which in

turn influence platform differentiation. Systems markets such as ecosystems are also inherently dynamic (Katz and Shapiro 1994). However, the IS literature focuses on explaining conventional notions of performance emphasizing predictability rather than explaining how systems evolve over time (Orlikowski and Iacono 2001), and the dynamics of their evolutionary trajectories also remain understudied in management research (Schilling 2000). Even though the importance of architecture is acknowledged in practice, very little attention has been paid to incorporating it in IS theory development. Although platform ecosystems function like markets in terms of involving module developers, they are rarely spot transaction oriented and require substantial coordination (Katz and Shapiro 1994). When or how their architecture facilitates such coordination has received limited research attention. Fourth, governing platforms requires a delicate balance of control by a platform owner and autonomy among independent developers. Neither the IS controls literature nor the IT governance literature addresses this tension. Fifth, platforms do not exist in vacuum and how well or how poorly they respond to the dynamics of their environment can be influenced by platform designers' technical choices. Finally, the IT artifact has historically tended to disappear from view, treated as a monolithic blackbox, or become the *omitted* variable (emphasis in original) (Orlikowski and Iacono 2001). Platforms offer the IS discipline an unusual opportunity to bring the IT artifact into the core of theory development about how platforms evolve and contribute unique insights distinctive from strategy, economics, and software engineering. These issues are particularly germane to IS because understanding how platforms evolve without considering their technical design attributes and relying solely on non-IS perspectives can mislead one into overlooking the important interactions of the IT artifact with its internal and external environment.

The overarching objective of this commentary is to identify underexplored questions about how the *coevolution* of endogenous design and governance choices by platform owners and the dynamics of their exogenous environment influences their evolutionary dynamics. Although platforms compete for both users and developers, in this commentary we focus exclusively on the supply (developer) side rather than demand (consumer) side of platforms; software-based platforms (excluding hardware-based ones); and on those that are two-sided markets connecting platform-specific module developers to end consumers

(thereby excluding IT platforms that firms build primarily for their own use, intermediaries, and one-sided markets).

The remainder of this commentary proceeds as follows. In §2 we identify five broad research questions on how the coevolution of the endogenous choices by platform owners and their exogenous environment influences the evolutionary dynamics of ecosystems and modules over the different temporal horizons described in §3. We discuss four theoretical lenses for constructing causal linkages from platform architecture, governance, environmental context to its evolutionary dynamics at both ecosystem and module levels in §4. In §5, we describe the potential theoretical contributions from tackling these research problems.

2. PLATFORM DESIGN, GOVERNANCE, AND THEIR ENVIRONMENT

We develop the overarching idea that the evolutionary dynamics of platform-based ecosystems and their modules is influenced by the coevolution of the platforms owners' choices endogenous to the ecosystem (e.g., platform architecture and governance) and the environmental dynamics exogenous to the ecosystem. We identify five broad research questions stemming from this tripartite coevolution perspective. Each research question can be studied either at the ecosystem or the module/ platform level of analysis, with different causal explanations and potential contributions.

Figure 2 presents a research framework that provides a roadmap for the subsequent discussion. The overarching research problem illustrated by the framework is about how the coevolution of platform owners' choices endogenous to an ecosystem and its exogenous environmental dynamics influences the evolutionary dynamics of ecosystems and modules. The framework is not intended to be exhaustive, but representative. We first describe elements of platform architecture, governance, environmental dynamics, and their coevolution. These constitute the core elements of the research problems corresponding to our five research questions discussed here. We then describe different facets of evolutionary dynamics spanning different timeframes that can be studied at either the ecosystem or module level, each potentially leading to different but complementary contributions. We then discuss four theoretical lenses that we believe can provide the starting point for theoretically developing middle-range explanations for how

architecture, governance, and environment—and their fits and misfits—influence their evolutionary dynamics. We illustrate with representative exemplars how each perspective can be invoked in theory construction using selective combinations of elements from the left and right side of Figure 2.

Table 1: Definitions of core concepts underlying software platform-centric ecosystems

Concept	Definition	Representative References
Platform	The extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it, and the interfaces through which they interoperate.	(Baldwin and Woodard 2009; Eisenmann et al. 2006)
Module	An add-on software subsystem that connects to the platform to add functionality to the platform.	(Baldwin and Clark 2000; Sanchez and Mahoney 1996)
Ecosystem	The collection of the platform and the modules specific to it.	(Cusumano and Gawer 2002)
Interfaces	Specifications and design rules that describe how the platform and modules interact and exchange information.	(Katz and Shapiro 1994)
Architecture	A conceptual blueprint that describes how the ecosystem is partitioned into a relatively stable platform and a complementary set of modules that are encouraged to vary, and the design rules binding on both.	(Baldwin and Woodard 2009; Katz and Shapiro 1994; Sanchez and Mahoney 1996; Ulrich 1995)

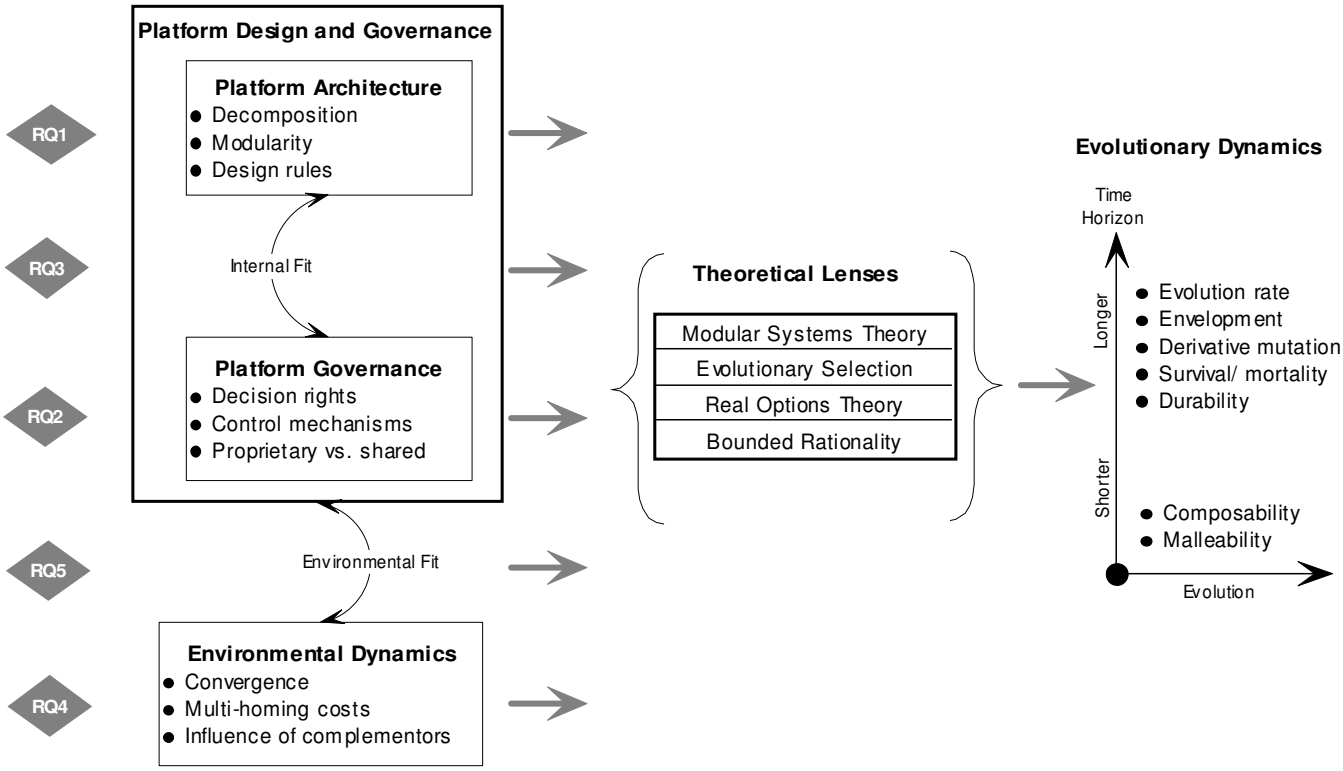

Figure 2: A framework for studying platform evolution.

2.1. Platform Architecture

We define *platform architecture* as a conceptual blueprint that describes how the ecosystem is partitioned into a relatively stable platform and a complementary set of modules that are encouraged to vary, and the

design rules binding on both (see Fig 1) (Baldwin and Woodard 2009; Katz and Shapiro 1994; Sanchez and Mahoney 1996; Ulrich 1995). A platform's architecture therefore partitions the ecosystem into the platform codebase that ideally exhibits low variety and high reusability and modules that exhibit high variety and low reusability within the ecosystem (Baldwin and Woodard 2009). The first set of fertile research opportunities exist around the research question about *how platform architecture influences the evolutionary dynamics of ecosystems and modules in platform settings?*

The challenge is that platform architectural choices—often irreversible—must accommodate changes unforeseen at the time that it was created (Baldwin and Woodard 2009). They must permit changes to individual modules without compromising their ability to function together again; we therefore label this problem as the Humpty Dumpty problem. The business ramifications of architectural choices are long-lasting as a platform owner can be locked into them for a substantial period of time (Pil and Cohen 2006). An ideal architecture should support variety in the present and evolvability over time (Baldwin and Woodard 2009). The properties of platform architecture can be studied from three distinctive perspectives: (a) decomposition, (b) modularity, and (c) design rules.

a. Decomposition refers to how the form and function of a platform's ecosystem is broken down into constituent atomic subsystems (Simon 1962). It defines which subsystems and functionality are part of the platform codebase and which ones reside outside it, and their separability (Schilling 2000). A platform ecosystem can hierarchically be decomposed into smaller subsystems until further decomposition no longer aids description and comprehension (this "atomic" level is subjective). The number of subsystems into which a platform or module can be partitioned represents its *span* (Simon 1962). Decomposition minimizes interdependence among the evolution processes within components of the ecosystem, supporting change and variation and also helps cope with complexity. But it comes at an upfront design cost and can also irreversibly constrain or overly expand the scope and span of an ecosystem's individual components. These tensions have evolutionary consequences as we subsequently discuss.

b. Modularity refers to the degree to which changes within a subsystem do not create a ripple effect in the behavior of other parts of the ecosystem (a behavior known as encapsulation). In contrast, low

modularity can result in wide-ranging, unpredictable ramifications of *any* change in the ecosystem (Baldwin and Woodard 2009). Architectures can range anywhere along a continuum of perfectly modular to perfectly integral (Ulrich 1995). Modularity can be achieved by increasing decoupling between modules and through platform-module interface standardization. Decoupling describes a behavior when internal changes in a module or the platform do not affect the behavior of other parts of the ecosystem. Decoupling therefore hides intra-module decisions from the rest of the ecosystem, emphasizing only its externally-visible properties. Interface standardization represents the degree to which a module interacts with the platform using stable, well-documented, and predefined standards (e.g., by use of application programming interfaces (APIs)). Standardization therefore decreases asset specificity of modules (Schilling 2000). Although the modularity literature asserts that modularity decreases coordination costs and transaction costs across the module boundary (Baldwin 2008), this premise is yet untested. On one hand increasing modularity can decrease coordination costs among constituents of a platform's ecosystem, frees up cognitive resources of developers to focus on more challenging problems, makes interfirm ignorance valuable (Tiwana 2008a), and encourages even greater specialization that drives development of capabilities among ecosystem constituents. On the other hand, it can also facilitate imitation, progressively erode the distinctiveness of modules and ecosystems (Pil and Cohen 2006), narrow the scope of learning by platform owners, and cause the loss of the synergistic specificity (Schilling 2000). (Synergistic specificity refers to the degree to which a platform achieves increased functionality by its modules being specific to each other.) A related issue that is beyond the scope of this commentary but nevertheless important is interdependence among *ecosystems* (e.g., a module in one ecosystem might be the platform of another).

c. Design rules refer to the rules that platform owners expect module developers to obey to ensure interoperability with the rest of the ecosystem (Baldwin and Clark 2000; 2006). Two properties of design rules are critical—how stable they remain over time relative to the platform and their versatility (Baldwin and Woodard 2009). Stability of design rules ensures that module developers who join the platform at different times can make the same assumptions about other parts of the ecosystem without needing to

verify those assumptions. However, such stability also means that the properties of design rules cannot adapt over time (e.g., PC VGA standards have persisted for 30 years), which requires that they also be versatile. Versatility means that they must not overly constrain modules in ways that decrease variety and flexibility of the ecosystem as a whole. Therefore, platform owners face a challenge in how to make design rules stable enough to sufficiently constrain developers' yet versatile enough to not overly constrain them.

2.2. Platform Governance

We define platform governance as who makes what decisions about a platform. The second set of research opportunities exist around the research question about *how platform governance influences the evolutionary dynamics of ecosystems and modules in platform settings?* A central governance challenge is that a platform owner must retain sufficient control to ensure the integrity of the platform while relinquishing enough control to encourage innovation by the platform's module developers. As a platform can have too much, too little, or the theoretically-elusive "just right" level of governance, we label this the Goldilocks Governance Problem. A platform's governance design can be studied from three distinctive perspectives: (a) decision rights partitioning, (b) control, and (c) proprietary versus shared ownership. These perspectives theoretically translate to governance by sharing responsibilities and authority, governance by aligning incentives, and governance by sharing stakes.

a. Decision rights partitioning refers to how decision making authority is divvied between the platform owner and module developers. Decision rights simply refer to who has the authority and responsibility for making specific decisions. Distinguishing among three broad classes of decision rights would be instrumental in future theory development: (a) *What* a subsystem should do (e.g., features and functionality), (b) *how* it should do it (e.g., design, concept implementation, and user interface), and (c) who controls the ecosystem's internal interfaces.[1] Interfaces can be longer-lasting and more stable than

[1] "What" decision rights have been labeled as decision control rights, specification, and strategic decision rights. "How" decision rights similarly have been ascribed labels such as decision management rights, implementation, and execution decision rights. A finer grained classification of decision rights is to build on the four types of decisions in Fama and Jensen (1983): (1) ratification (approval authority for modules' functionality), (2) monitoring (authority to specify and implement performance measurement criteria used in a platform), (3) initiation (authority to make decision about utilization of resources (such as network bandwidth,

the platform itself, and they *define* the boundaries of modules (Baldwin and Woodard 2009). Control over interfaces amounts to control over the platform and its evolution (Baldwin and Woodard 2009), and is particularly germane to understanding decision rights beyond how and why decision rights. Platform governance then entails how the authority and responsibility for *each* class of decisions is divvied up among the platform owner and module developers (i.e., degree of decentralization). Just as it is incorrect to assume that decision rights pertaining to the platform must reside solely with its owner (Baldwin and Woodard 2009), decision rights pertaining to modules do not necessarily need to reside solely with module developers. A more important question is about how they should be shared and when. Decision rights partitioning poses a tension for balancing autonomy among developers and coordination across the ecosystem; specialized skills and knowledge pertaining to modules is fragmented among module developers but their outputs need to be integrated with the rest of the ecosystem. Depending on the unit of analysis, the focal subsystem could be decision rights pertaining to individual modules or the platform. Either is a fruitful avenue that would yield potentially different types of insights.

b. Control refers to the formal and informal mechanisms implemented by a platform owner to encourage desirable behaviors by module developers, *and vice versa*. Formal control can take two forms: (a) *output* control (the platform owner prespecifies the criteria by which module developers' outputs are evaluated, rewarded, or penalized) and (b) *process* control (the platform owner prescribes methods and procedures to module developers). Informal control is via *clan* control (fostering common values, shared beliefs, and norms to guide module developers' behaviors). Three subtleties provide an unorthodox twist to control in platform settings. First, the premise of control is the existence divergent interests among parties. Paradoxically, the relationship between platform owners and module developers is not the *classical* principal-agent relationship (i.e., the platform owner does not hire module developers to do a task specified by the former), as assumed in control theory (Kirsch 1997). Yet, a variety of control mechanisms are widely observed in platform settings. It is plausible that the role of control mechanisms

---

pixels, memory) within a platform ecosystem), and (4) implementation (decisions about implementing approved modules or executing ratified functionality). Ratification and monitoring together comprise "how" decision control rights, and initiation and implementation represent "what" decision management rights (Tiwana 2009).

then is one of coordination rather than mitigating agency hazards, as control theorists widely assume. Second, they do not necessarily have divergent or zero-sum type interests. Further, input control (Cardinal 2001) (e.g., Apple tightly controlling what applications are distributed on its itunes platform) has largely been overlooked in the IS literature. Open versus closed platform architectures therefore simply represent differences in input control exercised by a platform owner. Third, control can be *bidirectional* where module developers also simultaneously exercise control on the platform owner. Platform governance using the control perspective can therefore be conceptualized as the directionality, variety, and degree of use of various control mechanisms.

c. Proprietary versus shared ownership. A final governance attribute is whether a platform itself is proprietary to a single firm or shared by multiple owners (Eisenmann et al. 2006). This property is conceptually distinct from and should not be confounded with the open versus closed architecture or the open-versus closed-source distinction (for a detailed discussion see (Gawer 2009)) (e.g., Google Chrome is proprietary but open-source; GNU Linux is open-source and shared by multiple developers; Apple's iOS is proprietary and closed-source). This property can influence platform evolution because it determines how widely dispersed or concentrated stakes in and ownership of the platform is.

2.3. Internal Fit - Interactions of Platform Architecture with Platform Governance

The third set of research opportunities exist around the research question about *how internal fit—between platform architecture and platform governance—influences the evolutionary dynamics of ecosystems and modules in platform settings?* The overarching lens for theory development about such interactions is the notion of complementarities (substitution) i.e., where having more of one thing makes having more of the other more (less) valuable. The key notion worthy of conceptual development is that a platform's architecture can reinforce (a positive interaction) or diminish (a negative interaction) the influence of platform governance on evolutionary dynamics. Furthermore, as architecture changes over time, coevolution of governance can influence evolutionary outcomes desirable by the constituents of the ecosystem. (We subsequently illustrate this idea with an example in §4.)

2.4. Environmental Dynamics

The fourth set of promising research opportunities reside around the question of *how environmental dynamics exogenous to ecosystems influence the evolutionary dynamics of ecosystems and modules in platform settings*. Four environmental dynamics in particular deserve theoretical attention. First, the technological trajectories—the rapidity, unevenness, scope, and unpredictability with which complementary and substitutive technologies are emerging—can affect their evolution. The morphing of technologies in application domains adjacent to the ecosystems due to integration of data, video, voice, and hardware can affect the trajectories of an ecosystem both positively and negatively. Such technological *convergence* can offer opportunities for a platform to expand into the domain of adjacent but unrelated platforms and simultaneously allow unrelated platforms to offer the focal platform's functionality as part of a multi-product bundle (Eisenmann et al. 2006). Convergence is therefore laden with envelopment opportunities, particularly since adjacent platforms often have overlapping user and developer bases (Eisenmann et al. 2006). For example, digital music players such as the ipod have expanded into adjacent application domains of movie players, email and Web functionality of personal computers, payment devices, and navigation systems. Second, *multihoming costs* (Armstrong and Wright 2007)—or the developers' costs of associating with more than one platform—can affect the focal platform's evolution. Homing costs represent the aggregate of adoption, operating, and opportunity costs incurred by a developer to maintain affiliation with a platform. When multihoming costs are high, a module developer needs a good reason to affiliate with multiple platforms (Eisenmann et al. 2006). What role do development toolkits provided by rival platform owner play in decreasing multihoming costs incurred by developers and how do they influence evolution at the ecosystem level? What role do "adapters" described by Katz and Shapiro (1994) play in overcoming incompatibility among rival platforms? As competing platforms decrease such costs of multihoming and switching by offering software development kits (SDKs), adapters, or compatible interfaces to module developers, they can create what Katz and Shapiro (1994) describe as "tipping" wherein rival platforms begin to pull developers away from the focal ecosystem. A third environmental dynamic is the power or *influence*

exerted by complementors that directly or indirectly provide services to one or more platform but are not part of the module developer community. Examples include service suppliers (e.g., AT&T supplies network bandwidth to Apple's iphone OS; Warner Brothers supplies movie content to Netflix) and regulatory agencies (e.g., Federal Trade Commission and Federal Communications Commission). The inherent tensions between potentially divergent interests of platform owners and complementors (see Baldwin and Woodard 2009) can influence evolutionary dynamics in ecosystems.

2.5. Environmental Fit – Interactions of Environmental Dynamics with Endogenous Platform Attributes

The fifth set of promising research opportunities exist around the question of *how environmental fit—between the endogenous attributes of an ecosystem (architecture, governance) and the dynamics of its exogenous environment—influences the evolutionary dynamics of ecosystems and modules in platform settings.* Environmental dynamics plausibly affect and are affected by the decisions endogenous to a platform ecosystem. For example, architectural and governance choices that are appropriate for one environmental context might be inappropriate for another; we therefore label this the "hammer-and-nail" problem. Endogenous choices of platform owners influence expectations, facilitate coordination, and achieve compatibility outside the ecosystem (Katz and Shapiro 1994). Misfits between the platform owner's choices and the context in which is exists might accelerate mortality or provide the conditions for it to thrive. For example, misfits in platform governance with environmental dynamics might cause a platform owner to be slow to recognize envelopment opportunities, and misfits with architecture might make the platform owner slow to mobilize resources to exploit them. In contrast, an architecture that embeds a variety of real options in the ecosystem can make possible proactive envelopment attacks. When technologies advance rapidly or at an uneven pace, architectural choices can allow or obstruct new inputs from proliferating the ecosystem as they become available. This question also focuses attention on when—rather than whether—certain endogenous choices are appropriate. For example, platform modularization also involves an upfront initial cost, leaving open questions of how much modularity is appropriate in an ecosystem and when. Theory development around these issues would likely focus on

studying interactions of platform design/ governance with environmental dynamics in Figure 2, and their coevolution. Furthermore, as the environmental dynamics exogenous to a platform change, coevolving the architecture and governance of the platform can steer the ecosystem towards a more desirable evolutionary trajectory. Subsequent to sufficient progress in theory development on internal fit and environmental fit, we believe that it would be promising to study meta fit between the two.

3. CONCEPTUALIZING EVOLUTIONARY DYNAMICS IN PLATFORM ECOSYSTEMS

To fully explore these five sets of research opportunities, we must precisely define what aspect of evolutionary dynamics is being explained. This requires supplementing the pervasive, classical notions of performance (e.g., efficiency and effectiveness in systems development) with dynamic, temporally-spread dependent variables that are underexplored in the IS literature. Temporality is inherently subjective and relative to the lifecycle of the population of comparable platforms; for example, personal computer operating systems historically have had about a 5-10 year lifespan, mainframes about 30-50 years, and smart phone applications about 6 months to two years. Nevertheless the shorter-longer temporal distinction in Figure 2 can be a useful starting point for bringing in the time dimension into theory development either using the platform's ecosystem or the module as the unit of analysis. Each would entail different theorizing and lead to potentially different insights.

The longer term evolutionary dynamics of ecosystems encompass five promising criterion variables: (1) evolution rate, (2) envelopment, (3) derivative mutation, (4) survival/ mortality, and (5) durability. *Evolution rate* refers to the rate or intensity at which the platform, an ecosystem, or an individual module in an ecosystem evolves over time. A widely-held premise that awaits testing is whether and when modules or platforms that evolve at a faster rate outperform those that evolve at a slower pace. Metaphorically speaking, does a rolling stone really gather no moss? *Envelopment* refers to the phenomenon where a platform swallows a platform in an adjacent market by offering its functionality as a multi-product bundle. Unlike other evolutionary dynamics that apply at multiple levels of analysis, envelopment is theoretically meaningful primarily at the platform level.) For example, Apple's ipod

progressively enveloped gaming console, Web browser, email system, phone, camera, and video player functionality from adjacent markets. Similarly, Netflix enveloped rent-on-demand services offered by cable companies. Such opportunities often arise from convergence of disparate technologies, and endogenous choices by platform owners can constrain or enable exploiting them. *Derivative mutation* refers to the unanticipated, serendipitous creation of a spin-off platform or module that inherits some properties of the parent but with completely different function from its parent. This new input is simply a byproduct of the adaptation of the original system (Schilling 2000). Unlike envelopment where the scope of the platform expands, mutation creates a distinct derivative platform or module.[a] Another related evolutionary dynamic that lends itself particularly well to archival historical analysis is module and platform *mortality* and survival. Finally, the persistence of a module or ecosystem's market advantages and distinctiveness—which Pil and Cohen (2006) call *durability*—can be shaped by its internal fit, environmental fit, and meta fit. This property is particularly important in platforms (Eisenmann et al. 2006).

In the shorter-term, evolutionary dynamics can focus on two promising criterion variables: (1) composability and (2) malleability. *Composability* refers to the ease with which functionality-extending changes can be made to a module or platform without compromising its integration with or functionality of of the ecosystem (Messerschmitt and Szyperski 2003: 63). Although IS research has historically conceptualized integration of systems as a one-shot task, in ecosystems, it is an ongoing process where changes in the platform codebase or any module can introduce integration problems. We therefore view composability as two-dimensional comprising module-to-platform composability and cross-module composability. *Malleability* refers to the ease with which the platform or module can be reconfigured to refine or extend their behavior to adapt to evolving user needs or to exploit technological advances. For example, a module might substitute its internal implementation of a specific function (e.g., copy-and-paste) with equivalent functionality that might have subsequently been introduced into the platform codebase. The relationships among various evolutionary dynamics that are not discussed in this commentary also await theoretical development.

4. LENSES FOR THEORY CONSTRUCTION

Four theoretical perspectives not widely used in IS can provide useful lenses for developing causal middle-range explanations linking platform architecture, governance, and environmental dynamics to evolutionary dynamics at the module or ecosystem level: (1) modular systems theory, (2) evolutionary selection, (3) real options theory, and (4) bounded rationality. We briefly illustrate the key ideas in these perspectives for theorizing about evolutionary dynamics of ecosystems and their modules. We then present one *illustrative* example of how each theoretical lens can be invoked to help explain why a specific set of representative variables (subset from Figure 2, left) can influence a specific type of evolutionary dynamic (see Fig 2, right) at a particular level of analysis (e.g., module or ecosystem).

Modular Systems Theory. Complex systems such as platform ecosystems are composed of interacting subsystems that are always to some degree interdependent (Schilling 2000). The premise of modular systems theory is that a complex system composed of smaller subsystems that interact exclusively using predefined, stable interfaces are more amenable to change than those that are monolithic. Modularization can increase cross-module independence and core-module independence within ecosystems. This allows individual subsystems in a platform ecosystem to independently evolve, unconstrained by having to coordinate or having to know internal details of other subsystems. Internal changes within a subsystem are less likely to disrupt other parts of the ecosystem, and simple compliance with interface standards ensures interoperability. Four ideas from modular systems theory are especially pertinent to understanding platform evolution: modularity (a) decreases overt coordination costs between module developers and platform owners by providing an embedded coordination mechanism (Sanchez and Mahoney 1996), (b) can decrease the effort required by a module developer to manage dependencies with the rest of the ecosystem, decreasing cross-module and module-to-platform systems integration costs (Schilling 2000), (c) can substitute for formal process control (Tiwana 2008b), thereby increasing module developers' autonomy, and (d) can decrease the need for knowledge outside module developers' task boundaries, engendering deeper specialization.

> *Illustrative exemplar.* Consider an illustration of how modular systems theory can help explain the influence of internal fit—between modularity and formal control—on composability at the *module* level of analysis. Modularity between a platform and a module decreases their interdependence and isolates ripple effects on the ecosystem from internal changes to a module. Therefore it decreases the need for coordination by a module developer with the platform owner, reducing module-to-platform integration effort. This increases the ease with which a module developer can make functionality-enhancing changes to a module (composability). Formal outcome control on the other hand simply governs outputs of a module developer without regard to the development process, strengthening the aforementioned link to composability. In contrast, formal process control attempts to govern the process, which is redundant with the integration-facilitating role already played by modularization. Therefore, increasing modularity then requires some output control in conjunction with minimal process control to enhance composability at the module level.

Evolutionary Selection. The premise of the theory of evolutionary selection is that complex systems that evolve at a faster rate and with greater diversity are more likely to evolve to achieve better fit with their environment than those that do not posses these traits. Simon's (2002) premise is that more decomposable complex systems evolve faster because they require less time to evolve by recombination, and will undergo more diverse evolutionary experiments. Therefore, they will increase in fitness to the environment faster than less decomposable ones. However, greater crowding of an ecosystem with proliferating modules (akin to natural environments) can create snowballing negative "indirect (Katz and Shapiro 1994)" or "same-side (Eisenmann et al. 2006)" network effects wherein additional developers find it increasingly unattractive to join the platform; and appropriate governance practices might be able to flip the sign of these same-side network effects. This perspective can help understand both the evolution of modules and the survival of competing ecosystems.

> *Illustrative exemplar.* Consider an illustration of how evolutionary selection theory can jointly help explain how architectural decomposition and the influence of complementors influences survival and mortality at the *ecosystem* level. The more decomposable a platform's ecosystem is, the greater is the speed and diversity of the adaptive experiments that its constituents engage in (Simon 2002). This diversity is more likely to lead to some evolutionary variant that has greater fitness to the evolving environment. This is likely to increase the chances of the platform's survival over time. At the same time, a greater influence of complementors in the environment is likely to contrain and even cancel out some of the adaptive benefits of decomposition, increasing the likelihood of the platform's mortality (suggesting a negative interaction).

Real Options Theory. A real option refers to the right without the obligation to do something. It entails future flexibility, with its value increasing with uncertainty and a longer time frame over which it can be exercised. Different strategic and operational real options can be embedded in a platform at an upfront

option acquisition cost, and each must also be exercised at an opportune time to realize its potential value. Therefore, benefiting from real options requires both consciously generating (embedding through design) and suavely exercising them. Platform architecture attributes can embed real options, and platform governance attributes can aid discovery of opportunities for exploiting them. Thus these two interrelated notions of embedding and exercising real options can provide conceptual building blocks to explain evolutionary dynamics. Similarly, the six modular operators—splitting, substitution, augmentation, exclusion, inversion, and porting—identified by Baldwin and Clark (2000: 346) embed different forms of options-like flexibility (see Gamba and Fusari 2009), each of which can be of greater value when the exogenous dynamics pose greater uncertainty about an ecosystem's environment.

> *Illustrative exemplar.* Consider an illustrative example of how real options theory can help explain how environmental fit—between platform modularity and convergence—influences envelopment at the *ecosystem* level. Convergence of unrelated technologies provides opportunities for envelopment of an adjacent ecosystem by a focal ecosystem (Eisenmann et al. 2006). However, a platform owner can be slow to exploit such opportunities if the architecture of the platform acts as a constraint. Modularization of the platform's architecture can embed various real options in the platform (e.g., splitting, substitution, augmenting, inversion/ refactoring, and porting) (Gamba and Fusari 2009), creating different forms of future flexibility to create unplanned variants of the platform. Modularization and convergence therefore correspond to the notions of options embedding and uncertainty in options theory. Thus convergence provides the *opportunity* for envelopment and modularity provides the *ability* to act on such opportunities. The two therefore mutually reinforce each other (i.e., a positive interaction).

Bounded Rationality. Bounded rationality interpreted in our context refers to the cognitive limits of individual developers in their ability to process and interpret a large volume of potentially pertinent information in their development work. Bounded rationality encompasses two concepts: search and satisficing (Simon 1979). Search in our context refers to how extensively a developer searches for information to guide development decisions. Search scope is capped by a heuristic-driven aspiration level that defines a "good enough" solution at the outset of the search process. Therefore terminating information search is central to how rapidly developers can act on their ideas, e.g., about improving a module. As the complexity of an ecosystem grows, the number and complexity of interdependencies with the platform and other modules can grow exponentially (Mihm, Loch and Huchzermeier 2003). Design attributes of a platform that intentionally narrow the scope of information that a developer must consider

about the rest of the ecosystem (i.e., partition tasks), assumptions that she can safely make about them (e.g., stable design rules), and the containment of ripple effects from changes internal to that module (e.g., modularization) can mitigate bounded rationality constraints of individual developers that can otherwise impede platform evolution.

> *Illustrative exemplar.* Consider an illustration of how the bounded rationality perspective can help explain how internal fit—between architecture modularity and decision rights decentralization—influences evolution rate at the *module* level. Greater modularity decreases the need for module developers to be aware of internal details of other modules or the platform, decreasing the *need* for overt coordination or parallel changes as a module developer makes internal changes to her module. This allows the module to evolve autonomously and without having to consider how those changes affect the rest of the ecosystem, increasing evolution rate. Similarly, decentralization of module-specific decision rights gives more autonomy to module developers who are more likely than other members of the ecosystem to possess specialized module-specific knowledge. Therefore modularity strengthens the benefits of decentralizing governance in accelerating platform evolution by decreasing systems integration costs incurred by a module developer. Therefore, these attributes independently and jointly enhance module evolution rate by reducing individual developers' search scope.

4.1. Recommendations for Designing Field Studies

We have five recommendations as scholars engage in thought experiments for theory construction and subsequent field studies. First, resist the temptation to stop at linear relationships and instead also explicitly consider the possibility of nonlinear and threshold effects (e.g., modularity might speed evolution until a threshold and then impede it, in a curvilinear pattern). Second, substitute a process description for an initial state description of a relationship to identify plausible mediating constructs in developing middle-range theories. Third, be conscious in the choice of—and avoid confounding—the ecosystem, modules, or the platform as the unit of analysis (see Figure 1). The same relationship at a different plausible unit of analysis can potentially generate completely different types of insights. Rich longitudinal data can be accessed at the module level in open-source repositories such as Mozilla and Sourceforge. Fourth, recognize the value of using archival data on modules and platforms that suffered from mortality (e.g., PalmOS, Web TV, Minitel.fr, IBM's OS/2, demised browser extensions) as they might yield rich insights into platform evolution just as dead species do for biological evolution. For example, understanding how and why Palm which pioneered the handheld digital assistant market suffered from market exit in 2010 could lend new insights into how platform-centric ecosystems evolve.

Finally, recognize that addressing these research questions will require new ways to measure concepts and theorizing should reflect that. The data will invariably be a mix of longitudinal objective data (e.g., from source code repositories) and primary data, offering unusually rich opportunities for testing causation and causal ordering. Although much of the field work might be in specific platform contexts, ensuring that concepts are consistently and generically defined in a platform-independent manner will facilitate cumulative knowledge accumulation.

5. POTENTIAL CONTRIBUTIONS

Each of these problems offers unprecedented opportunities for the IS discipline to make distinctive theoretical contributions to the IS literature as well as the reference disciplines of strategy, economics, and software engineering.

Architecture-to-evolutionary dynamics link. Although IS scholars have repeatedly called for the need to focus on the IT artifact, we have fallen short on doing so (Orlikowski and Iacono 2001). Incorporating platform architectures in explaining the evolution of platform-centric ecosystems offers a unique opportunity to bring the IT artifact into the nucleus of theory development. Particularly, new IS theory development should focus on the coordinative role of architecture where conventional overt coordination mechanisms do not scale. The unique contribution to strategy includes a more nuanced way of thinking about technological architecture and its role as an embedded, mass-coordination mechanism, the microprocesses through which architecture shapes technological evolution, and addressing the open question about the link between modularity and sustainable market advantages (Pil and Cohen 2006). Modular system's theory (Schilling 2000) predicts adoption of modular designs, not its evolutionary consequences; such studies would therefore complement the management literature. Such work can also contribute unique insights to the burgeoning stream on two-sided platforms in economics by explaining how the design of complex systems influences evolution in markets. Finally, it can contribute new insights to software engineering into how software architecture shapes its evolutionary dynamics.

Governance-to-evolutionary dynamics link. Developing this link contributes an interfirm perspective using a broader variety of governance notions to the IT governance literature that has simplistically

viewed governance as the degree of centralization/ decentralization of IT activities *within* the firm (Sambamurthy and Zmud 1999). It can also extend the exclusively-project-level IS controls literature by studying situations where control can be bidirectional. Complex alliance networks such as platform ecosystems can provide a new twist to the classical notion of interfirm alliances in the strategy literature. Such work can complement the burgeoning exclusively-macro, two-sided markets literature in economics in which governance and control are completely overlooked (Lerner, Pathak and Tirole 2006). Insights into who should control what aspects of the design and development process, how the distribution of project responsibilities influences software evolution, and how these choices interact with environmental pressures can also contribute new knowledge to the software engineering literature.

Internal fit between platform architecture and platform governance. The IS discipline provides a unique vantage point at the *intersection* of technical design (the software engineering view) and governance (the management view). Developing the overarching idea that the benefits of technical architectural choices can be reinforced or diminished by how a platform is governed represents an opportunity to use an IS perspective to contribute distinctive insights absent in strategy, economics, and software engineering, in addition to new theory entirely unique to the IS discipline.

Environmental fit-to-evolutionary dynamics link. Developing explanations for how endogenous choices by platform owners can be reinforced or diminished by the dynamics of its exogenous environment provides a fertile opportunity for contributing how the architecture and governance choices can help explain the evolutionary trajectories of platforms that are replete with the pressures of convergence of technologies, coexistence of multiple rival platforms, survival and durability of technologically ill-engineered platforms (the Windows paradox), and the varying influence of complementors and regulatory pressures. This would address the understudied issue of when—rather than whether—specific architectural choices pay off. By gradually unblackboxing the IT artifact, this could contribute unique insights to IS and address observed phenomenon that theories in our reference disciplines do not yet fully explain.

6. CONCLUSION

The focus of this commentary was to draw attention to the neglected problem of how the coevolution of endogenous choices by platform owners and the dynamics of an ecosystem's exogenous environment influences its evolutionary dynamics. The emergence of software-based platforms presents a significant opportunity for research contributions around IT artifacts with the potential to contribute homegrown theory to the IS discipline. It also presents opportunities for IS researchers to use an IS perspective to make distinctive contributions to our reference disciplines. In this commentary, we developed a framework for understanding platform-centric ecosystems and identify five broad questions around the proposed coevolution perspective. The competitive shift towards such ecosystems, the technology-mediated expansion of the very meaning of a firm, the role of architecture as a coordination device, and the need for autonomy without losing control raise interesting theoretical problems for IS researchers. Our objective in this commentary was merely to provide a starting point for a research conversation and deeper questioning around software-based platforms among IS scholars.

REFERENCES

Armstrong, M., and Wright, J. 2007. Two-Sided Markets, Competitive Bottlenecks and Exclusive Contracts. *Economic Theory*. **32**(2) 353-380.

Baldwin, C. 2008. Where Do Transactions Come From? Modularity, Transactions, and the Boundaries of Firms. *Industrial and Corporate Change*. **17**(1) 155-195.

Baldwin, C., and Clark, K. 2000. *Design Rules: The Power of Modularity*. MIT Press, Cambridge.

Baldwin, C., and Clark, K. 2006. The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? *Management Science*. **52**(7) 1116-1127.

Baldwin, C., and Woodard, J. 2009. *The Architecture of Platforms: A Unified View*. in: Platforms, Markets and Innovation, A. Gawer (ed.), Edward Elgar, Cheltenham, UK, 19-44.

Cardinal, L.B. 2001. Technological Innovation in the Pharmaceutical Industry: The Use of Organizational Control in Managing Research and Development. *Organization Science*. **12**(1) 19-36.

Cusumano, M., and Gawer, A. 2002. The Elements of Platform Leadership. *Sloan Management Review*. **43**(3) 51-58.

Eisenmann, T., Parker, G., and van Alstyne, M. 2006. Strategies for Two-Sided Markets. *Harvard Business Review*. **84**(10) 1-10.

Fama, E., and Jensen, M. 1983. Separation of Agency and Control. *The Journal of Law & Economics*. **26**(June) 301-326.

Fichman, R. 2004. Real Options and IT Platform Adoption: Implications for Theory and Practice. *Information Systems Research*. **15**(2) 132-154.

Gamba, A., and Fusari, N. 2009. Valuing Modularity as a Real Option. *Management Science*. **55**(11) 1877-1896.

Gawer, A. 2009. *Platforms, Markets and Innovation*. Edward Elgar, Cheltenham, UK.

Katz, M., and Shapiro, C. 1994. Systems Competition and Network Effects. *Journal of Economic Perspectives*. **8**(2) 93-115.
Kirsch, L.J. 1997. Portfolios of Control Modes and IS Project Management. *Information Systems Research*. **8**(3) 215-239.
Lerner, J., Pathak, P., and Tirole, J. 2006. The Dynamics of Open-Source Contributors. *American Economic Review*. **96**(2) 114-118.
Messerschmitt, D., and Szyperski, C. 2003. *Software Ecosystem*. MIT Press, Cambridge, MA.
Mihm, J., Loch, C., and Huchzermeier, A. 2003. Problem -Solving Oscillations in Complex Projects. *Management Science*. **49**(6) 733-750.
Orlikowski, W., and Iacono, S. 2001. Desperately Seeking The "IT" in IT Research—a Call to Theorizing the IT Artifact. *Information Systems Research*. **12**(2) 121-134.
Pil, F., and Cohen, C. 2006. Modularity: Implications for Imitation, Innovation, and Sustained Advantage. *Academy of Management Review*. **31**(4) 995-1011.
Sambamurthy, V., and Zmud, R. 1999. Arrangements for Technology Governance: A Theory of Multiple Contingencies. *MIS Quarterly*. **23**(2) 261-290.
Sanchez, R., and Mahoney, J. 1996. Modularity, Flexibility, and Knowledge Management in Product Organization and Design. *Strategic Management Journal*. **17**(1) 63-76.
Schilling, M. 2000. Toward a General Modular Systems Theory and Its Application to Interfirm Product Modularity. *Academy of Management Review*. **25**(2) 312-334.
Simon, H. 1962. The Architecture of Complexity. *Proceedings of the American Philosophical Society*. **106**(6) 467-482.
Simon, H. 1979. Rational Decision Making in Business Organizations. *American Economic Review*. **69**(4) 493-513.
Simon, H. 2002. Near Decomposability and the Speed of Evolution. *Industrial and Corporate Change*. **11**(3) 587-599.
Tiwana, A. 2008a. Does Interfirm Modularity Complement Ignorance? A Field Study of Software Outsourcing Alliances. *Strategic Management Journal*. **29**(11) 1241-1252.
Tiwana, A. 2008b. Does Technological Modularity Substitute for Control? A Study of Alliance Performance in Software Outsourcing. *Strategic Management Journal*. **29**(7) 769-780.
Tiwana, A. 2009. Governance-Knowledge Fit in Systems Development Projects. *Information Systems Research*. **20**(2) 180-197.
Ulrich, K. 1995. The Role of Product Architecture in the Manufacturing Firm. *Research Policy*. **24** 419-440.

AMRIT TIWANA is an associate professor at The University of Georgia. Prior appointments were at Emory University and Union Pacific Professorship at Iowa State University. He received his Ph.D. from Georgia State University. He received ISR's outstanding Associate Editor award in 2009 and serves on various editorial boards. His work has appeared in *Information Systems Research, Strategic Management Journal, ACM and IEEE Transactions, Decision Sciences, Journal of Management Information Systems, California Management Review*, and others.

BENN R.KONSYNSKI is George S. Craft Professor of Business Administration for Information Systems at the Goizueta Business School, Emory University. Prior appointments were at Harvard University and the University of Arizona, where he co-founded the group decision support laboratory. He holds a Ph.D. in Computer Science from Purdue University and specializes in digital commerce and interorganizational systems. He has published in diverse journals and serves on boards and advisory roles in public and private organizations.

ASHLEY A. BUSH is an associate professor at The Florida State University. She received her Ph.D. from Georgia State University. Her research has appeared in *Journal of Management Information Systems*, *IEEE Transactions on Engineering Management*, *Communications of the ACM*, *Information and Organization*, and others.

[a] For example, the creation of the Apple ipod Touch and ipad platforms from the iphone operating system are instances of derivative mutation of the iOS platform. Similarly, a Windows peer-to-peer client (Kazaa) evolving into a voice-over-Internet application (Skype) is an example of derivative mutation of a module.