Software Development Outsourcing, Asset Specificity, and Vendor Lock-in

Emergent Research Forum (ERF)

Mazen Shawosh
University of Georgia
m.shawosh@uga.edu

Nicholas Berente
University of Notre Dame
nberente@nd.edu

Abstract:

In software development vendor contexts, asset specificity is commonly thought to lead to vendor lock-in. But asset specificity in the context of software development is not one thing. It could involve specific functional assets, specific knowledge assets, and specific technical assets. Based on Transaction cost economics (TCE) and IT outsourcing literature, we theorize about the ways that different forms of software development asset specificity interact to influence lock-in in software vendor relationships. We develop three hypotheses about (1) the moderating effect of knowledge asset specificity on the relationship between functional asset specificity and lock-in; (2) the moderating effect of technical asset specificity on the relationship between functional asset specificity and lock-in; and (3) the moderating effect of software popularity on the relationship between knowledge asset specificity and lock-in.

Keywords

Asset specificity, vendor lock-in, software development outsourcing.

Introduction

Companies that outsource software development projects generally do not want to be locked into a relationship with a particular vendor, because they want to maintain bargaining power and keep their options open. It is in a software development vendor’s best interest, however, to find ways to make itself indispensable to its clients (Levina and Ross 2003). If the vendor can develop customer-specific capabilities, applications, or other assets, the vendor can increase the customer’s switching costs and essentially lock the customer into working with that vendor. Thus, a fundamental concern for software development organizations and their clients involves the impact of asset specificity on vendor lock-in.

Software development outsourcing is a form of IT outsourcing (Chen and Bharadwaj 2009; Smuts et al. 2010; Tiwana and Kim 2015), and asset specificity has been dealt with at length in the IT outsourcing literature. This literature typically takes the customer’s view and identifies lock-in risks that are associated with outsourcing decisions (Bahli and Rivard 2003; Lonsdale 2001). It is well established that asset specificity drives lock-in (Aubert et al. 2004), but what is asset specificity in the context of software development outsourcing? Existing research emphasizes different elements of asset specificity, for example, technical resources (Kim and Young-Soo 2003), human assets (Kim and Young-Soo 2003; Poppo and Zenger 2002), client-specific IS activity (Nam et al. 1996), and proprietary software (Oh et al. 2006). The extant literature does not distinguish between these different types in software development contexts. Further, the relationship between asset specificity and lock-in is implied by the research — it is assumed to be the mechanism that ties asset specificity to outcomes. Thus, existing theorizing on IT outsourcing essentially assumes that asset specificity drives lock-in, without explicitly investigating how different forms of asset specificity might result in differing levels of lock-in. To this end, in this study we will attempt to answer the following research question: how do the different types of software asset specificity influence the levels of vendor lock-in?

Using Transaction Cost Economics (TCE) as our theoretical lens and relying on the IT outsourcing literature, we propose that different types of software development-related assets will lead to different lock-in outcomes. We theorize how different forms of specific assets (functional, knowledge, and technical asset
specification) interact with one another to drive lock-in. To test our hypotheses, we will conduct a field study and collect data using the matched-pair survey approach.

**Hypotheses**

Winkler and Benlian (2012) divide software asset specificity into three dimensions: functional specificity, knowledge asset specificity (human asset specificity), and technical specificity. *Software functional specificity* is related to the adaptation and customization of software functionalities to a specific firm. Thus, the higher the functional specificity is, the greater the lock-in. *Software technical specificity* is the degree to which components are tightly integrated together in a particular implementation (Williamson 1979; Winkler and Benlian 2012). Software with tightly integrated components will have higher asset specificity for a particular application. Figure 1 depicts our research model.

**Figure 1. Research Model Depicting the Moderated Relationships Between the Different Types of Asset Specificity and Vendor Lock-in**

**Hypothesis 1: Moderating Influence of Knowledge Asset Specificity**

From the TCE perspective, knowledge possessed by humans can be thought of as an asset that can be redeployed to alternative uses and by alternative users without sacrificing productive value (Williamson 1996). If the degree of knowledge acquired by the individuals working on a particular IT project is very high, then that resource becomes indispensable for the organization. Tiwana (2003) classifies the knowledge needed in software development as application problem domain knowledge (on the client side) and technical knowledge (on the vendor side) (Dibbern et al. 2008; Tiwana 2003). Application problem domain knowledge is high “when the development or maintenance work required knowledge about unique software applications and/or business processes of the client” (Dibbern et al. 2005, 2008, p. 347), which leads to extra costs to clients (Dibbern et al. 2008).

We argue that knowledge asset specificity will moderate the effect of functional asset specificity on vendor lock-in. Custom, proprietary software has high asset specificity and will drive vendor lock-in. The source code of the custom, proprietary software is hidden from the client. This secrecy means that organizations that outsource this software to a vendor automatically increase their transactional risks, by creating opportunities for lock-in. In situations where there is substantial knowledge specificity, knowledge and experience of the client will amplify the effect of the custom, proprietary software on vendor lock-in. The combined effect of both custom software and intimate client knowledge will make for extreme switching costs. Conversely, even if software is highly customized, absence of the complementary knowledge and understanding of the client will reduce the positive effect that software asset specificity has on lock-in. Hence:

**H1:** The impact of functional asset specificity on lock-in will be strengthened (weakened) by high (low) knowledge asset specificity.
Hypothesis 2: Moderating Influence of Technical Asset Specificity

Beyond the specific knowledge assets, there is also the degree to which open standard interfaces, such as application programming interfaces (APIs), web services, and other standards are utilized between elements of the code – what Winkler and Benlian (2012) refer to as technical asset specificity in a software context. The example of Bell Atlantic demonstrates the power of non-standard interfaces in driving vendor lock-in (Shapiro and Varian 1998). Whenever Bell Atlantic required an upgrade or new features, they had to rely on the technical and functional expertise of AT&T while being charged huge sums of money. To lock Bell Atlantic out from any chance to obtain compatible equipment from other vendors, AT&T used proprietary interfaces among components. Thus, technical asset specificity clearly drives vendor lock-in. However, there is also a moderating effect of technical asset specificity on the relationship between the functional asset specificity of software on vendor lock-in. Just as with knowledge, the use of non-standard, tightly integrated interfaces – even if the code is indeed open source – will reduce the degrees of freedom and optionality of the software code. Modular software code enables freedom and mixing and matching of components (Baldwin and Clark 2000). Recent advancements in software modularity enable alternative arrangements that reduce the risk of lock-in, including multi-sourcing arrangements (Aubert et al. 2016). The more custom the software code is and the more combined it is with tightly integrated components, the higher the increase in switching costs from that vendor. Similarly, standard interfaces will reduce the impact of even highly customized code on vendor lock-in, as the client has greater optionality through the standard interfaces. Standards are a classic way to avoid many risks (Lyttinen et al. 1998), including lock-in risk. This characteristic leads to the second hypothesis:

**H2**: The impact of functional asset specificity on lock-in will be strengthened (weakened) by high (low) technical asset specificity.

Hypothesis 3: Moderating Influence of Software Popularity

Crowston et al. (2006) and Stewart and Ammeter (2002) utilize the concept of software popularity as one of the measurements of software use to indicate IS success. Software popularity in the context of software development is concerned with the number of potential or actual users of the software (Crowston et al. 2006; Stewart and Ammeter 2002). The concept of software popularity can be applied to the context of IT outsourcing. A software, either in the form of an application (e.g., SAP) or technology (e.g., programming languages), can enjoy high or low spread in the market. Low popularity can result from old software or a situation in which other similar software has dominated the market or from software that is new and has not gained momentum. Another source of low popularity is a vendor’s use of proprietary or highly specialized software that is not available elsewhere. An example is the case of Bell Atlantic and AT&T digital switches (Shapiro and Varian 1998).

An example of technology would be the kind of programming language used in application and software development. A client, for instance, uses an application written in COBOL, and the vendor’s IS staff who work on this application now become highly specific assets for this client. This specificity is because COBOL is infrequently used in system development these days (i.e., low in technology popularity1) in comparison to other mainstream programming languages. This relative rarity makes it a highly valuable software asset. An attempt to change to another vendor may result in the client suffering from the burden of switching costs. It may also lead to incurring the added cost of searching and finding another vendor that has developers with COBOL programming skills. Vendors are more accustomed to popular software programming languages such as Java, Python, C, C++, and C#, and these languages have taken precedence and now dominate the environment. This intuition leads to our third hypothesis:

**H3**: The impact of knowledge asset specificity on lock-in will be weakened (strengthened) by high (low) high software popularity.

Proposed Research Design

We will empirically test the hypotheses in a field study of software development outsourcing relationships. Our unit of analysis is the vendor-client dyad. We will collect the data using matched-pair surveys with key informants from both client and vendor sides. Several IT outsourcing studies have adopted this approach (e.g., Ko et al. 2005; Rustagi et al. 2008; Wiener et al. 2015). The client is a giant petroleum and chemicals

---

1 TIOBE Index for June 2016 [http://www.tiobe.com/tiobe_index](http://www.tiobe.com/tiobe_index)
company in the Middle East implementing more than 100 software development projects. This company is considered the largest producer of crude oil in the world. From the client side, key decision makers in the outsourcing relationship will complete the survey, and from the vendor side, managers with deep knowledge of the project will complete the survey. To increase the response rate, client managers will identify the appropriate vendors and help secure their participation. First, we will administer surveys to vendor firms. Based on responses, we will administer surveys to key managers in charge of those relationships from the client side. If a particular client manager deals with more than one vendor, that manager will be asked to fill out multiple surveys. The main advantage of the matched-pair survey approach is that it reduces common method bias and the single-informant problem (Podsakoff et al. 2003).

We will measure the study’s theoretical constructs using multi-item scales. We adapted constructs and their measures from the extant literature (Ang and Cummings 1997; Bahli and Rivard 2013; Benlian 2009; Bush et al. 2010; Ibrahim et al. 2012; Winkler and Benlian 2012). These constructs show evidence of validity and reliability. Control variables will be included to account for rival explanations. These controls include trust (Rai et al. 2012; Rustagi et al. 2008), uncertainty (Rustagi et al. 2008), loyalty (Cahill et al. 2010; Tian et al. 2008), and contract-related information (Susarla et al. 2010).

Conclusion

We anticipate that this paper will have both practical and theoretical implications. Clients planning to outsource software development should be aware of the vendor lock-in risks. To mitigate the risks, clients may request their vendors to use popular technologies that other competing vendors offer and strict adherence to standards. They might also consider keeping key knowledge assets within the organization as a hedge against lock-in risk. Future research should consider the impact of different types of assets on lock-in outcomes in their theorizing. In addition, empirical tests of these relationships can lend further insight into the relationship between IT asset specificity and IT outsourcing outcomes.

References


