CONFIGURABILITY, MATURITY, AND VALUE CO-CREATION IN SaaS: AN EXPLORATORY CASE STUDY

Completed Research Paper

Eruani Zainuddin  
Queen’s Business School  
Queen’s University  
Kingston, Canada  
ezainuddin@business.queensu.ca

Paola Gonzalez  
Queen’s Business School  
Queen’s University  
Kingston, Canada  
pgonzalez@business.queensu.ca

Abstract

This study answers the research question, “How do value co-creation components – value, offering, value networks, user involvement, and interaction process – change over time as Software-as-a-Service (SaaS) configurability moves toward maturity?” We conducted a case study of GlobalSchool, a SaaS company providing administrative software to small-sized schools. We refined the SaaS maturity model by integrating the concept of self-service. We further assessed configurability (along with SaaS maturity) from the co-creation of value perspective. Our findings show that value co-creation components are dynamic, changing at different maturity levels. We also identified two drivers for change – knowledge and volume of clients. Our study contributed toward the SaaS and value co-creation literature. The managerial implications include the need for SaaS vendors to balance between providing support and self-service, solicit feedback from long-standing clients, and slowly transition clients to the self-service concept.

Keywords: Cloud computing, SaaS, software maturity, service science, value co-creation
Introduction

“The highest quality [SaaS] solutions will also offer extensive, configurable capabilities that enable you to modify key aspects of the application’s appearance and to better suit your individual needs.” – Dan Carmel (2009), CEO of SpringCM, on how to evaluate potential Software-as-a-Service document management solutions.

Software-as-a-Service (SaaS) refers to the selling of software that is owned and managed by the vendor, and delivered as a service over the Internet. The SaaS application is based on a single set of common code and data definitions, and distributed in a one-to-many manner to all clients (Xin and Levina 2008). This one-to-many distribution manner is referred to as multi-tenancy. The Gartner Group estimated that the worldwide SaaS revenue surpassed the projected forecast of $9.2 billion in 2010, up almost 16% from the 2009 (DaRold and Ridder 2011). Another study conducted by the American IDC (2009) research group projected that 50% of organizations will use SaaS for strategic business functions. This is a major transition since SaaS is currently known to support mostly non-critical business applications (Gartner Group 2006). These statistics imply that the SaaS market is expanding, and SaaS will have more significant impact on individual organizations.

Due to its relative novelty, there is a paucity of SaaS research especially from the Information Systems (IS) perspective. Recent call-for-papers for SaaS-related studies (e.g., cloud computing and service science) in peer-reviewed IS journals, along with specific suggestions from researchers (e.g., Candan et al. 2009), show that SaaS is slowly gaining traction among IS researchers. Thus, more research is needed in this area.

One of the defining characteristics of SaaS is configurability. Configurability provides the mechanisms for clients to adapt the software to fit their individual requirements (Nitu 2009). In other words, configurability grants a certain degree of flexibility to software, enabling multi-tenancy (Nitu 2009). Research has shown that SaaS vendors could offer configurability in two ways – by allowing clients to configure the software themselves (client-enabled) or by configuring the software on behalf of the clients (vendor-supported) (Zainuddin and Staples 2011). Configurability is an important characteristic in SaaS such that it is used to determine SaaS maturity (Hudli et al. 2009). More mature SaaS solutions have more configurability options.

Previous studies have examined SaaS configurability and maturity from the technical perspective (Arya et al. 2010; Kwok et al. 2008; Nitu 2009; Sun et al. 2008; Wang and Zheng 2010). Alter (2008) proposed that service systems such as SaaS can be examined from the co-creation of value perspective. We followed Alter’s (2008) suggestion and examined SaaS configurability through this framework. We identified five components of value co-creation – value, offering, value networks, user involvement, and interaction process. Andersson et al. (2007) found that components of value co-creation are dynamic, exhibiting change over time. Therefore, we framed our research question as follows,

How do value co-creation components – value, offering, value networks, user involvement, and interaction process – change over time as SaaS configurability moves toward maturity?

To answer the above research question, we conducted a case study of a SaaS provider. GlobalSchool\(^1\) provides administrative software solution to small-sized schools. We refined the SaaS maturity model by incorporating the element of self-service (i.e., vendor-supported and client-enabled configurations). Our findings show that components of value co-creation change over time. We also found that knowledge and volume of clients are drivers for change. Our work contributed toward the SaaS and value co-creation literatures.

We structure this paper as follows: First, we provide the background on SaaS and value co-creation framework. We argue that configurability can be viewed from the value co-creation framework. Second, we describe the research method that was utilized. Third, we present our findings – the SaaS maturity model and SaaS maturity from the co-creation of value framework. And last, we discuss the theoretical and managerial contributions of our study, limitations, and potential avenues for future research.

\(^1\) We use pseudonyms for all the names in this study.
Literature Review and Conceptual Framework

Our study draws on the SaaS and value co-creation literatures. In this section, we first provide an overview of SaaS. We focus our discussion on the concept of configurability, an essential mechanism in SaaS. Next, we review the co-creation of value framework and outline its integral components. These components – value, offering, value networks, user involvement, and interaction process – act as the conceptual foundation to our study. Finally, we argue how SaaS configurability can be viewed from the co-creation of value framework.

SaaS Background

SaaS is a business model where vendors maintain ownership and control of their software. The typical SaaS vendors built their software from the ground-up, and continue hosting and maintaining their software in centralized locations (Santy 2010). In addition, SaaS solutions are usually web-based and accessible via Internet browsers (Santy 2010). Users do not have to install the software on their personal computers. From an architectural standpoint, SaaS is based on the multi-tenancy model. Multi-tenancy refers to the concept of a single software instance (i.e., a set of common code and data definitions) serving multiple clients (Mäkilä et al. 2010). At present, successful SaaS solutions are characterized by horizontal applications with common processes such as business intelligence, human resources, and customer relationship management (Mertz et al. 2009). However, a recent forecast by the Gartner Group estimated that 50% of organizations are planning to adopt SaaS solutions for specialized and more complex applications (Da Rold and Ridder 2011).

Due to its relative novelty, research in SaaS is still at the nascent stage. We categorize the available studies into three streams. First, studies that examine SaaS adoption (Benlian et al. 2009; Xin and Levina 2008) and continual usage (Benlian et al. 2010). These studies view SaaS adoption and continual usage as a sourcing decision and utilize organizational level theories such as transaction cost, resource-based view, and institutional theory. Second, studies that compare SaaS with other business models such as packaged software and application service providers (Choudhary 2007; Fan et al. 2009). These studies utilize econometric techniques to model the differences between SaaS and other business models. And third, studies that focus on implementing the appropriate architecture and data models for SaaS. The main objective of this study is to offer guidance to address security, scalability, and multi-tenancy concerns in SaaS (Cusumano 2010; Hudli et al. 2009; Kwok et al. 2008; Nitu 2009; Sun et al. 2008; Wang and Zheng 2010).

The advantages of SaaS include low upfront costs, faster implementation, and flexible subscriptions (i.e., clients can subscribe or unsubscribe at any point in time) (Plummer 2011). In addition, SaaS vendors are expected to constantly and continuously release new features and make them freely available to clients (Gordon 2010). Thus, clients have access to the latest innovation without additional costs. Lastly, SaaS is more environmentally sustainable due to its centralized servers and data processing facilities. The centralization of servers and data processing facilities reduces electronic waste as well as usage of power supply (Schaeffer 2007).

There are however, a few disadvantages to SaaS. These disadvantages include data security concerns, lost of control for clients (i.e., clients are unable to determine the future growth of the software), and the one-size-fits-all model that may not work well for complex and atypical organizational processes (Plummer 2011). Nevertheless, SaaS vendors have incorporated several strategies to minimize the negative aspects of SaaS. These strategies include developing the technologies to address data security concerns, building the capabilities to gather client feedback and integrate the feedback into the product roadmaps, and implementing configurability in the software to accommodate the needs of a diverse set of clients (Song et al. 2010).

Configurability in SaaS

Multi-tenancy, which refers to the concept of serving multiple clients with a common source code, is at the heart of SaaS (Hudli et al. 2009). Multi-tenancy implies that the requirements of various clients are fulfilled by a single instance of the software application. The key to achieving multi-tenancy in SaaS is configurability (Arya et al. 2010; Kwok et al. 2008; Nitu 2009; Sun et al. 2008). Configurability provides
the mechanisms for clients to tailor the software to fit their unique requirements, and therefore, confers
the software with a certain degree of flexibility (Nitu 2009). There are five configurable aspects in SaaS
(Arya et al. 2010), which include the following:

- **User interface** – The ability to change the look and feel of the user interface features to reflect the
  clients’ preferences. For example, enabling the clients to change the icons, fonts, and background colors
to reflect corporate branding.

- **Workflow** – The ability to change the behavior of the software in terms of the activities, user roles, and
  rules. For example, enabling the clients to add their own business rules to reflect specific organizational
  workflows.

- **Data** – The ability to store specific data requirements in the database. For example, enabling the clients
to add and define their own columns to tables, or to create their own tables. This is the most critical
  aspect in SaaS configurability because data drives the SaaS application.

- **Access control** - Each client using SaaS will have multiple individuals using the software. SaaS access
  control configurability includes the ability to create individual accounts for end users, and determine
  which resources and functions each user should be allowed to access.

- **Miscellaneous configurability options** – These options extend the software by enabling it to
  accommodate clients in different domains. For example, adding language options (e.g., Spanish,
  Chinese, Japanese, etc.) to serve clients from different regions.

SaaS maturity is determined based on its configurability level (Hudli et al. 2009). Hudli et al. (2009)
described SaaS maturity as:

- **Level 1 maturity** implies that the software application is customized for individual clients, and does
  not offer any configuration option.

- **Level 2 maturity** implies that the software application offers minimal configurability options. Thus,
  limited instances of software applications are available to clients.

- **Level 3 maturity** implies that the software application offers extended configurability options for
  clients. At this level, the software fully supports multi-tenancy, and only a single instance of the
  software application is available to all clients.

- **Level 4 maturity** implies that apart from full multi-tenancy, the software application is hosted in a
  multi-tiered architecture. Thus, the software application is both highly configurable and scalable.

A study by Zainuddin and Staples (2011) indicated that during the early stages of SaaS development, the
configuration capabilities of the software tend to be vendor-supported as opposed to client-enabled.
Vendor-supported configurations require vendors to set up the software on behalf of their clients, while
client-enabled configurations allow clients to set up the software themselves (i.e., self-service or do-it-
yourself). As SaaS vendors’ understanding of the client domain (i.e., clients’ goals, workflows, rules, and
constraints) increase by means of client feedback, vendors will implement more and more client-enabled
configurations in their enhancement releases. Therefore, SaaS software evolves from having a high degree
of vendor-supported configurations to having a high degree of client-enabled configurations.

There are two important points that we need to make regarding SaaS configurability. First, both the
maturity levels and the client-enabled configuration options that SaaS vendors should try to achieve
varies. Sun et al. (2008) argued that each SaaS vendor should first define their strategies based on various
factors such as targeted customer segments and software scope. If the strategy is well defined, SaaS
vendors can succeed even if their configurability options are minimal (i.e., SaaS vendors remaining at the
early levels of maturity rather than progressing to higher levels of maturity) (Sun et al. 2008) and/or
offering a mixture of vendor-supported and client-enabled configurations (Zainuddin and Staples 2011).

Second, despite the fact that configurability is a defining characteristic of SaaS, configurability in itself is
not a new concept for software applications. Earlier business models such as off-the-shelves and
Application Service Provider (ASP) also incorporate configurability options into software applications.
There are both similarities and differences between SaaS and these other (or non-SaaS) solutions with
regards to the configurability characteristics. We highlight these similarities and differences in Table 1.
Table 1. Similarities and Differences between SaaS and Non-SaaS Applications Based on Configurability Characteristics

<table>
<thead>
<tr>
<th>Configurability Characteristics</th>
<th>Similarities</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial design</strong></td>
<td>-</td>
<td>SaaS is designed from the ground-up with configurability in mind – i.e., initial design of SaaS takes into account that the software will need to be modified in the future to include various configurability options. Non-SaaS applications are not necessarily designed with configurability in mind.</td>
</tr>
<tr>
<td><strong>Configurable aspects</strong></td>
<td>Both SaaS and non-SaaS solutions offer configurability capabilities to their clients. Various aspects of both types of software applications can be made configurable – i.e., configurations for user interfaces, workflow, access control, etc.</td>
<td>Because SaaS is designed with configurability in mind, it is often easier to add various configurability options later on. Non-SaaS applications are not designed with configurability in mind, and thus, it is harder to add configurability options later on. Most non-SaaS vendors will include configurability aspects that are easier to integrate (e.g., user interfaces), but leave out the more difficult configurability aspects (e.g., workflow and data) for customization instead.</td>
</tr>
<tr>
<td><strong>Client requirements</strong></td>
<td>The purpose of software configurability is to cater to the specific needs of specific clients. More mature SaaS and non-SaaS solutions will offer more configurability capabilities – i.e., there are more parts of the software that clients can configure.</td>
<td>Despite more configurability capabilities, clients who are dealing with more mature SaaS do not necessarily find that the software will meet all of their requirements. Clients are limited because in most cases they need to work within the confines of the options provided by SaaS vendors; customization (i.e., changing the source code of the software) is rarely an acceptable approach especially in more mature SaaS. In most cases, clients who are dealing with non-SaaS solutions will have the option to customize the software applications to meet their specific requirements.</td>
</tr>
<tr>
<td><strong>Vendor business value</strong></td>
<td>-</td>
<td>Configurability adds to SaaS vendors’ business value. It is a low cost and viable alternative to cater to the needs of a high volume of clients. The other alternative – customization – is a costly endeavor, and maintaining multiple software versions is often too difficult and costly in the long run. Both configurability and customization adds value for non-SaaS vendors. Vendors are often paid additional sums to customize on behalf of their clients and/or vendors do not have to maintain software applications for a large volume of clients.</td>
</tr>
</tbody>
</table>
Based on the above discussion, the configurability dimension in SaaS is a single-line spectrum with vendor-supported and client-enabled configurations at the opposite ends. SaaS software evolves not only from being customized to being highly configurable (Hudli et al. 2009), but also from having a high degree of vendor-supported configurations to having a high degree of client-enabled configurations. It is important for SaaS vendors to decide the configurability levels in which they would like to offer in their software. Furthermore, configurability is a defining characteristic of SaaS that can be analyzed from the co-creation of value framework.

Co-creation of Value

Normann and Ramirez (1993) introduced the idea of co-creation of value as a new business strategy. Instead of being passive consumers of value, clients are given the alternative to create value for themselves. Vendors provide clients with the necessary tools and options to create value. These tools and options must be simple and intuitive, and contain the elements that can be combined to reflect clients’ preferences. By allowing clients to combine different elements, vendors are able to provide personalized products to a wide set of clients. Prahalad and Ramaswamy (2004) extended the original idea of co-creation of value by emphasizing client-vendor interactions. Clients transmit their preferences to vendors through multiple channels including public online postings, personal interactions with vendor representatives, and/or remarks to individuals in their personal networks. Vendors need to capture, analyze, and integrate clients’ preferences to create better services and/or products (i.e., improving the tools and options for clients to create value). The co-creation of value views clients as being engaged from the very beginning of the product lifecycle – from development to consumption. In the end, the clients and vendors “converge toward a unique co-creation experience, or an experience of one” (Prahalad and Ramaswamy 2004, p.6).

IKEA, the world’s largest retailer of home furnishings, provides a good example of the co-creation of value business strategy. IKEA integrates a business system that creates value by matching the capabilities of its participants (e.g., clients, suppliers) more efficiently. The company promises to deliver well-designed products to its clients at significantly lower prices than other furniture stores. The clients on the other hand, must be willing to transport and assemble the products themselves. IKEA’s strategic intent is to “create a business system that allows them [the clients] to do it [assemble furniture and home furnishings] better” (Normann and Ramirez 1993, p. 3). Therefore, IKEA stores will carry easy to assemble furniture and mix-and-match furnishings. IKEA also offers guidance for home furnishings through its quarterly magazine – IKEA Family Live. Self-assembled furniture and mix-and-match furnishings allow IKEA clients to personalize their home decor to fit their own styles. To achieve its strategic intent, IKEA relies on knowledge and relationship (Normann and Ramirez 1993). Knowledge refers to understanding of clients’ preferences, which will then be integrated into the products. Meanwhile, relationship refers to access to clients’ preferences at every stage of the product lifecycle. The relationship facilitates IKEA in obtaining the required knowledge.

There are five components in the value co-creation framework (Andersson et al. 2007). We describe these components below.

Value

The traditional understanding of value focuses on value-in-exchange. Value is defined as how clients derive benefits. As a concrete example of value-in-exchange, clients derive benefits from usage of well-designed and affordable IKEA furniture (i.e., benefits from utility). Value is static and easy to measure; typically through the monetary amounts that vendors receive from the exchange activities (Vargo et al. 2008). Meanwhile, an exchange is perceived as the transaction between two parties – the client and vendor (Kotler and Levy 1969; Kotler 1972). Most researchers suggest that clients can gain value (or derive benefits) through consumption. Woodruff and Gardial (1996) however, suggested that clients can gain value through ownership.

From the co-creation of value framework, value is seen from the context of value-in-use. Value is again defined as how clients derive benefits. Benefits can be in the form of physical product, service and infrastructure, as well as relationship (Normann and Ramirez 1993). However, value is dynamic and not easy to measure. Value changes during the different stages of the co-creation process (Andersson et al. 2007).
Clients do not gain value only at the very end of the process, through consumption and ownership. Clients also gain value during their engagements (i.e., while interacting with vendors to co-create value). Based on the concept of value-in-use, clients derive benefits from usage of well-designed and affordable IKEA furniture, the entertainment that they receive while shopping, and the ability to mix-and-match furniture to suit their individual styles (i.e., personalization).

**Offering**

The traditional understanding of value in products and services is derived by the exchange-activities between clients and vendors. These activities are regarded as one-off transactions or frozen activities. In the co-creation of value framework, the exchange-activities are on-going transactions wherein the traditional roles of clients and vendors are blurred (Normann and Ramirez 1993). These activities or interactions are called offerings, occur throughout the co-creation of value process, and are perceived as dynamic and complex (Normann and Ramirez 1993). Being dynamic implies that exchanges occur and evolve over time (Prahalad and Ramaswamy 2004). Meanwhile, being complex refers to the fact that overall, there is no clear cut answer as to who is the designer of the product (i.e., the vendor or the client). In other words, it is unclear as to whether vendors provide products or services to their clients.

For example, one of IKEA’s offerings includes the point at which clients look through IKEA’s quarterly magazine. Clients are interacting with IKEA through the magazine as the interface. Clients can derive value by getting home improvement ideas or engaging in a leisure activity, and may also start to take on the role of designers by thinking of ways to combine the items in the magazine to fit their styles. The next step in IKEA’s offering includes the point at which clients visit IKEA’s on-premise stores. Parents can leave their children at the play areas, and engage in shopping activities. Here, clients can derive value by engaging in a leisure activity (kids-free), and take on a more active designer role by selecting the different parts of IKEA’s products for themselves. In this case, it is not clear if the exchange is a product or a service (e.g., home improvement as opposed to leisure). Furthermore, clients become designers of their own furniture.

**Value networks**

The delivery of services over the internet has changed the value system concept from a chain to that of a network (Böhm et al. 2010). Peppard and Rylander (2006) argued that the process of creating value in a cloud computing environment is determined by the interdependencies between actors in the market. These actors are not limited to clients and vendors, but extend to external entities such as hardware providers, network providers, and professional groups (Andersson et al. 2007). These actors are autonomous, and operate together in a framework of common principles and service level agreements. Value for clients and vendors are created by the interactions among actors, wherein each actor contributes incrementally towards co-producing value (Bovet and Martha 2000). Normann and Ramirez (1993) suggested that the strategic analysis of value creation should not be the organization or industry, but rather the value-creating network itself. Therefore, analysis of co-creation of value should include all of the actors involved or the value networks.

**User involvement**

Client organizations, or more specifically individual users, are involved in the co-creation of value process. In this study, we define user involvement from the co-creation of value perspective (as opposed to user

---

2 Normann and Ramirez (1993) used the term value constellation in their seminal work. Value constellation reflects the interaction process in which clients are at the center, and other actors surround and interact with clients. Later authors such as Böhm et al. (2010) and Peppard and Rylander (2006) used the term value networks instead, to reflect the online environment. Interactions are more free-flowing within value networks, without clients having to be at the center.

3 The term user involvement and user participation are used interchangeably in the co-creation of value literature.
involvement in the IS development literature. User involvement focuses on the user experience when he/she interacts with vendors (Normann and Ramirez 1993). There are six characteristics to evaluate user involvement in value co-creation (Alam 2002, 2006; Magnusson 2003). These characteristics are:

- **Purpose** – What are the objectives and goals of the users?
- **Stages** – At which stages of the value co-creation process are the users involved in?
- **Intensity** – How does the frequency of user involvement varies across the different stages of the value co-creation process?
- **Quality** – Are the users communicating openly, honestly, and constructively with vendors?
- **Mode** – What are the means through which input and information are obtained from the users?
- **Type of user** – What are the different types of users involved in the value co-creation process?

**Interaction process**

A distinctive feature separating goods from services rests on the concept of value (Grönroos 1998). In the manufacturing industry, the prevailing view of value for clients is embedded in final products. Yet, in services, the value is placed on the service offerings during the co-creation of value process (Normann and Ramirez 1993). According to the later, value is not created by the provider but rather derived during two types of interactions. First, value is derived during interactions between clients and vendors. And second, value is derived in the interactions between clients/vendors with other actors in the value networks.

In exploring innovations in service industries, Alam (2002, 2006) argued that interactions are not only important on new service performance but also on the continuous improvement of on-going services. He emphasized that both intensity and quality of user involvement during all stages of new service development are key aspects of the interactions. Intensity refers to the frequency of client feedback during the developing process, which in turn will be dependent on the willingness of clients to provide the feedback. Meanwhile, quality refers to the thoroughness of the feedback (i.e., honest, open, and constructive). These aspects are influenced not only by the maturity of the users but also by the type user. Specifically, Von Hippel (1978) highlighted the usefulness of lead users for new product development. Lead users can be organizations, groups or individuals who are always looking for the latest innovations in the markets (i.e., usually early adopters of a technology). They tend to experience market needs ahead from other users. Furthermore, user involvement can change over time due to the maturity of the offerings and because lead users may display different participation behaviors during the value co-creation process. As a result, the interaction process during co-creation of value may also change over time (Andersson et al. 2007; Christensen 1997).

We would also like to highlight the differences between offerings, user involvement, and interaction process. In the case of offerings versus interaction process, an offering refers to a particular interaction between a client and a vendor at a single point in time. However, an interaction process is more holistic, giving an overall view of the relationship. An interaction process also contains multiple offerings and interactions among all the actors in the value networks. Meanwhile, in the case of user involvement versus interaction process, user involvement focuses on user experience; emphasizing client (or user) activities. An interaction process on the other hand, emphasizes how all of the actors in the value networks act against one another.

**SaaS Configurability from the Co-creation of Value Perspective**

We argue that SaaS vendors emulate IKEA's co-creation of value strategy through client-enabled configurations (or self-service configurability options). With the wide range of clients’ software needs, SaaS vendors need to create the means to capture and translate different needs into more flexible software services. Such a service will allow vendors to maintain a single application for all clients (i.e., multi-tenancy), and at the same time, provide as much personalization as possible. Moreover, this personalization incorporates a simple and intuitive design that enables clients to configure efficiently and satisfactorily. Client experience while configuring a SaaS application is an important aspect of success/acceptance of any SaaS software (Nitu 2009). During the early stages, SaaS vendors tend to
implement vendor-supported configurations (Zainuddin and Staples 2011). As vendors' understandings of their clients increase by means of client feedback, more and more client-enabled configurations will be implemented. The feedback also enables SaaS vendors to integrate clients’ best practices into their software.

In summary, we draw three parallels between client-enabled configurability in SaaS to the co-creation of value process (see Co-creation of Value section). These parallels are: (1) giving clients the tools and options to configure the software to fit their requirements (i.e., allowing clients to co-create value for themselves); (2) ensuring that the tools and options are simple and intuitive to facilitate clients in co-creating value; and (3) capturing and integrating clients’ needs by means of client feedback, as well as incorporating best practices. We therefore conclude that the process of providing client-enabled configurability options in SaaS applications can be viewed as the co-creation of value process. The dynamic aspect of this process led us to our qualitative research.

**Research Method**

Our main objective is to examine how value co-creation components change over time as SaaS configurability moves toward maturity. Research on SaaS from the IS perspective is still novel. Furthermore, to the best of our knowledge, the value of co-creation framework has not been applied to study the SaaS maturity model. Thus, we regard our work as exploratory and prefer to examine this topic in detail within its natural setting (or context). The case study is the most appropriate method to answer research questions framed as "how" questions that require explanatory answers (Benbasat et al. 1987; Dubé and Pare 2003; Yin 1994). Data from a case study will enable us to create “operational links that can be traced over time, rather than mere frequencies or incidence” (Yin 1994, p. 9). A case study is also the more suitable approach when the phenomenon of interest has not been fully understood (Yin 1994). Lastly, a case study approach seemed the most appropriate because of the exploratory nature of the research question and the need for the contextual background.

Our unit of analysis is a SaaS solution. Specifically, we analyzed the SaaS solution offered by GlobalSchool. GlobalSchool provides administrative software for small-sized schools (i.e., schools with less than 400 students). The company started in 2002, and was originally a technology consulting and services company. GlobalSchool transitioned into the SaaS business model in late 2008. About 80% of its clients are located in North America, while the rests are located in other parts of the world. GlobalSchool runs a highly distributed operation. Its employees are located in three different countries – Malaysia, United States, and Canada. To support this highly distributed operation, employees stay connected through various communication media such as emails, online chats, and teleconferences. Meanwhile, clients are supported through online chats, emails, and phone calls.

We utilized the case study research strategy, using a single case study design. We view our single case as a revelatory case. According to Yin (1994, p. 42), a single case study design is appropriate when the situation exists such that “an investigator has an opportunity to observe and analyze a phenomenon previously inaccessible to scientific investigations.” In this particular situation, we were given access to GlobalSchool’s communication server (i.e., we have access to emails, client-vendor chat logs, etc.) and engineering repository. In the words of Eisenheurdt’s (1989), we fulfill the criterion of a particular and salient situation where the process of interest is “transparently observable”. Furthermore, although multiple case studies allow for cross-case comparisons, the strength of a single case study is in its in-depth analysis and rich descriptions (Creswell 2007). Thus, by focusing on a single unit of analysis, we were in a better position to meet our objective. Lastly, we followed the guidelines and standards established for case study research in the IS field (Benbasat et al. 1987; Dubé and Pare 2003).

**Data Sources and Collection**

In this study, we analyzed co-creation of value from both clients and vendors’ perspectives. There were three data sources: internal documents, engineering repository, and semi-structured interviews.

The first author had the opportunity to examine the internal documents from GlobalSchool. These internal documents include email messages, client-vendor chat logs, and meeting notes. The internal documents contain information on GlobalSchool’s business jargons and internal working processes. The
prior knowledge of GlobalSchool enabled us to create the interview questions and facilitated our data analysis.

The engineering repository contains information on engineering releases (i.e., a release can either be a functional enhancement or a correction to an error). These releases include those from September 2009 (i.e., official launch date for GlobalSchool's SaaS solution worldwide) to current. The repository has information on release dates, release descriptions, programmers in charge, and sources of the suggestion/bug report. We were able to use the engineering repository to trace the maturity of GlobalSchool's SaaS solution.

We conducted semi-structured interviews with two key informants – GlobalSchool's Director of Account Management ("John") and a representative from one of GlobalSchool's clients ("Tina"). John's position as the "interface" between GlobalSchool's engineering and their clients, enable him to answer both the technical and client–related questions (i.e., questions on configurability and co-creation components). Meanwhile, Tina's school ("New Academy") is one of GlobalSchool’s early clients. We regard New Academy as a lead user (see the sub-section on Interaction process). Tina is New Academy's Information Technology administrator and contact person to GlobalSchool. This allows her to answer questions on co-creation components, especially those related to the interaction process. Both interviews lasted for slightly over thirty minutes.

We also conducted a fifteen-minute follow-up interview with John. The follow-up interview was to clarify certain answers and to gather more data that were missing from the earlier interview.

**Data Analysis**

Data analysis followed a five-step procedure, and these steps are:

Step 1: Analyzing the engineering repository and tracing GlobalSchool’s SaaS maturity. We chronologically ordered all releases related to the configurability. This includes past and planned releases, new configurations, as well as changes to existing ones. We identified the different maturity levels that emerged.

Step 2: Identifying and extracting information from the interviews. The interviews were transcribed. The second author then coded the interviews based on a coding guide. The guide was developed based on the co-creation of value framework. Both authors then reviewed the coded interviews to reach a mutual agreement on the interpretations.

Step 3: Analyzing the interviews to come up with an overall model of our findings.

Step 4: Strengthening our analysis through data triangulation. Here, we reviewed the internal documents and interview transcripts to strengthen the SaaS maturity model (see Step 1). We also reviewed the internal documents to strengthen our overall model (see Step 3).

Step 5: Submitting a copy of our findings to the interview participants. We have only received feedback from the vendor regarding the SaaS maturity model, and incorporated the feedback accordingly. We will incorporate more feedback as we receive them.

We uploaded the collected data and analysis of findings onto a secured online data storage facility. Both authors have password-access to this storage facility. This procedure is in line with the suggestion made by Yin (1994), which is to maintain a case study database to increase its reliability.

**The SaaS Maturity Model**

We explain our findings in this section. We first offer a refined version of the SaaS maturity model (Table 2). Next, we discuss the SaaS maturity model from the co-creation of value perspective. We also provide a model that summarizes our findings (Figure 1).

**Integrating the Concept of Vendor-supported and Client-enabled Configurations**

We offer a refined version of the SaaS maturity model. Our maturity model contains five levels of maturity as opposed to four (as in the earlier SaaS maturity model). We added the transformation from
providing vendor-supported to client-enabled configurability options. Our model is derived from data in the engineering repository. We also used information found in the internal documents and interview transcripts to complete and strengthen our model. Table 2 displays the maturity characteristics for each level along with the evidence from GlobalSchool.

<table>
<thead>
<tr>
<th>Maturity level</th>
<th>SaaS maturity characteristics</th>
<th>Evidence of Maturity in GlobalSchool.com</th>
</tr>
</thead>
</table>
| Level 1        | The software application is customized for individual clients, and does not offer any configuration option. At this point, the software application does not support multi-tenancy. | Before September 2009  
GlobalSchool was serving four clients. Each school had their own version of the software. Three of the schools were located in Malaysia, and the fourth was located in the United States. [Source: Internal documents] |
| Level 2        | The software application offers minimal configurability options. These configurability options are mostly vendor-supported. There are limited instances of software applications available and vendors provide extensive configuration services to clients. Thus, the software application does not support multi-tenancy, and is not self-serviced. | September 2009 until June 2010  
September 2009 is official worldwide launch date for GlobalSchool’s solution.  
GlobalSchool was maintaining two software instances. New clients and previous Malaysian clients started using a single software instance.  
Ten modules were launched. Two modules (i.e., report cards and attendance) and initial data upload required vendor-supported configurations. Software did not allow clients to create their own data fields (i.e., no data configurability), and only allowed clients to change background colors (i.e., very limited user interface configurability). [Source: Internal documents]  
A few configuration-related suggestions were logged into the engineering repository. However, feedback and actual releases were mostly related to corrections of errors. [Source: Engineering repository]  
Note: We observed two periods of gaps that accumulated to two months of no release. |
| Level 3        | The software application offers extended configurability options for clients. The configurability options are a combination of vendor-supported and client-enabled. The software fully supports multi-tenancy, and only a single instance of the software application is available to all clients. However, vendors provide some configuration services to clients. The software application is not fully self-serviced. | July 2010 to current  
GlobalSchool maintains only a single instance of the software for all clients (i.e., full multi-tenancy).  
Client-enabled configurations were released for the report cards and attendance modules. Other client-enabled configurations released include data and interface configurability options (i.e., clients can create their own data fields, and change many parts of the user interface – the whole “look and feel” of the software).  
Six new modules were released in stages. Five modules support full self-serviced, and one module needs vendor-supported configurations. [Source: Engineering repository] |
Table 2. The SaaS Maturity Model with the Concept of Vendor-supported and Client-enabled Configurations

<table>
<thead>
<tr>
<th>Maturity level</th>
<th>SaaS maturity characteristics</th>
<th>Evidence of Maturity in GlobalSchool.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 4</td>
<td>The software application offers extended, client-enabled configurability options for clients. Thus, the software application supports multi-tenancy and full self-service.</td>
<td>GlobalSchool’s software has not reached Level 4 and Level 5 maturity. However, the company is aiming towards the Level 4 maturity. Logs in the engineering repository show plans for client-enabled configurability options. Furthermore, John stated during the interview that, “the [desirable] goal of our services is that the schools can configure everything on their own.” [Source: Engineering repository and interview]</td>
</tr>
<tr>
<td>Level 5</td>
<td>Apart from supporting extended client-enabled configurations, the software application is hosted in a multi-tiered architecture. The software is highly configurable, self-serviced, and scalable.</td>
<td></td>
</tr>
</tbody>
</table>

In GlobalSchool, the configurability-related releases were made on a continuous basis. We did observe two extended periods of gaps. The first gap involved three weeks of no release, and was caused by an internal re-structuring of the development team. Meanwhile, the second gap involved four weeks of no release, and was caused by a move to a new office and server locations.

GlobalSchool aims to reach a Level 5 maturity level – i.e., GlobalSchool would like offer a highly configurable, self-serviced, and scalable SaaS solution. Nevertheless, this does not imply that the company wishes to make their software “a hundred percent configurable.” Instead, the target is to offer “between sixty to seventy percent configurability options, with all options being self-serviced.”

From the Co-creation of Value Perspective

We analyze the SaaS maturity model from the co-creation of value perspective. We focus our discussion on components of value co-creation that influence SaaS configurability and those that are dynamic. We center our discussions on Level 2 until 4 maturity presented in Table 2. Level 2 to 4 underline the transformation to client-enabled configurability options (i.e., the co-creation of value process). Although co-creation of value has traditionally been analyzed from the clients’ perspective, we also took into account GlobalSchool’s point of view.

Level 2

At this level, clients see value in vendor-supported configurability. Tina from New Academy stated during her interview that, “it was so much information, that for me to [configure]... I would personally prefer [the configuration] by GlobalSchool.” She further clarified that “if you are brand new, you wouldn’t know that ins and outs of [the software].” This implies that in the beginning, clients lack the understanding of the software to configure the software for themselves. Meanwhile, from the vendors’ perspective, the configurability options at this point are not simple and intuitive enough for the clients. Providing vendor-supported configurations eases the clients’ adoption process. John from GlobalSchool remarked that “the configuration screens were too complicated” and “we were forced to configure everything at this point, otherwise we would have lost our customers.” John further explained that the configurations were complicated because GlobalSchool “did not really understand who and what to cater to” such that the configurations catered to almost every single possibility. According to John, the initial configuration options are too complicated – e.g., “too many layers,” “too many clicks needed,” and “screens contain too many elements such as buttons, paragraphs, and instructions.” The need to get clients coupled with lack of knowledge pushed GlobalSchool to provide vendor-supported configurations. At this maturity stage, both clients and vendors derive similar benefit from vendor-supported configurations. Vendor-supported configurations enable clients to adopt the software.
The offerings, or interaction points between vendors and clients, include vendors configuring the software for clients. To configure for clients, vendors need to understand client requirements. Tina stated that to configure: “[GlobalSchool] needed to figure out what we needed.” This led to frequent communications through online chats, emails, and phone calls. In terms of user involvement, although communications were frequent, clients were unable to give specific feedback on configurability options. The primary reason is still lack of understanding. As Tina asserted, “our [configuration related] feedback was not precise at this point because we weren’t used to having a program like this.”

Frequent communications facilitate close relationships between clients and vendors. As John informed us, “the service was high...originally we do everything for you... [For example] send us the file we will upload it for you. We had lots of meetings.” John also admitted that the small volume of clients at this stage enables GlobalSchool to cultivate close relationships with most of their clients.

**Level 3**

At this level, vendors derive value by transitioning from vendor-supported to client-enabled configurations. Client-enabled configurations allow vendors to maintain low support costs. Vendors can accommodate an increasing volume of clients, without increasing their number of staff. This is reflected in John’s statement that,

“The more support we have to give the more money we, the company, have to spend to hire staff, to train staff. But, if everything can be done by the clients on their own, by them to [configure] on their own, [that] is an objective that we want to meet to bring costs down.”

This statement was made when we asked John to comment on why GlobalSchool’s is implementing more client-enabled configurations.

The value networks expanded slightly. During Level 2, only clients and vendors were involved in the value co-creation process. In Level 3 however, vendors may integrate their software with other SaaS providers. This integration may require vendor-supported configurations. GlobalSchool for example, offered their clients the ability to integrate their GlobalSchool accounts to an external accounting software solution. This ability requires vendor-supported configurations. Here, GlobalSchool provides vendor-supported configurations because they do not fully understand the technical (“accounting software is from outside and we don’t know how it behaves”), user behavior (“don’t know how users go about doing this”), and organizational/business (“what tables and columns do they need”) aspects.

Vendors start to provide more and more client-enabled configurations. As a result, vendors’ offerings for configuration support start to decrease. This also affects user involvement. The communication frequencies between vendors and older clients start to decrease. John stated that at present: “We [GlobalSchool] don’t hear much from users because they are able to configure it on their own, and figure it out for themselves. They’ll ask us questions every now and then.” Tina also voiced the same scenario, in which communication only occurs once in a while. She attributed this to “understanding the software better” such that “[configurability] becomes easy.” Although the communication frequency decreases, older clients are able to give more precise configuration suggestions because they now understand how the software works. Tina felt that, “We [understand] the program’s ins and outs better now. We can see something that is lacking but would be good.”

The interaction process at this stage is mixed with both distant and close relationships. GlobalSchool is able to maintain close relationships with older clients (e.g., New Academy) and develop close relationship with some new ones. The latter schools are the ones that asked for some form of vendor support. John’s statement reflected this situation,

“Some would interact with us and some don’t. That is probably the way we categorize them... because when they communicate with us, we have a feel of how they are. We see how things are going and how they react.”

John further explained that, “For schools that don’t communicate with us it is hard or difficult for us to understand what they are going through... what they need.” John’s explanation highlighted an important fact. At this maturity stage, vendors derive benefit from vendor-supported configurations because they
encourage communication with clients. And communication in turn, will enable vendors to understand their clients better.

**Level 4**

At this level, clients derive value from full client-enabled configurations. Clients will benefit from having more control of their software. They will also be able to make faster changes to the software, when required. Tina stated that, “I think if things should change over time within the school itself, it will be easier to do it ourselves... if we could manage it on our own.” Older clients such as Tina will have the advantage of “knowing a lot more [about the software]” such that they feel more comfortable with the full self-service concept. Vendors also offer the same opinion. John used Facebook as an analogy, “When you look at some of the software out there, when you look at Facebook, do you want to have to call Facebook whenever you have to change something? You want it to be as easy to use, if you have to call someone to use it, they probably wouldn’t use it. So that is one of the benefits of going self-service.”

To be able to offer client-enabled configurations, GlobalSchool must ensure that the configuration options are simple and intuitive. This implies that GlobalSchool does not cater for every single possibility (as stated in Level 2 discussion, catering for every single possibility complicates the configurability options). Instead, according to John, GlobalSchool only needs to offer options that are “common to a lot of schools.” For this to occur, GlobalSchool must accumulate knowledge on what these common options are.

Due to full client-enabled configurations being implemented, vendors provide very minimal configuration support. In other words, vendor-supported configuration offerings will be minimal. In the case of GlobalSchool, we predict that offerings will occur when “[users] don’t know where [he/she] can make the changes, [he/she] had to ask...” As a result, communication frequencies are low.

Value networks will experience growth. John believes that Application Programming Interface [API] is “a sign of mature SaaS.” He informed us that GlobalSchool has plans to provide APIs so that they “will be able to provide [services] that GlobalSchool would not normally provide on its own.” Thus, we believe that value networks will increase at this stage.

The interaction process at this stage consists of mostly distant relationships. As client volume increases, vendors are not able to cultivate close relationships. John commented that, “So with the way the schools of growing we feel [that] we’re within our current parameters. I don’t doubt that once we have reached a certain number of schools we have to change the way we work.” When asked to explain what changes GlobalSchool needs to make to accommodate more clients, John was unable to give a definitive answer.

**Drivers of Dynamic Behavior**

Our findings highlight that the co-creation of value components are dynamic, as SaaS moves toward full self-service configurability (or higher maturity). All of the components in the co-creation of value framework – value, offering, value networks, user involvement, and interaction process – display different attributes during different levels of maturity. For example, the interaction process between clients and SaaS vendors can be marked as close at Level 2 maturity, mixture of close and distant at Level 3 maturity, and mostly distant at Level 4 maturity.

We observe two factors that drive the dynamic behavior. The first factor is the level of knowledge held by vendors and clients. For example, client communication behavior (i.e., user involvement) changes with increase understanding of software. When understanding is low, clients communicated more frequently but with less precision. As client understanding increases, communication becomes less frequent but with more precision. This particular finding echoes Prahalad and Ramaswamy’s (2004) view that knowledge is at the “heart” of the value co-creation framework.

The second factor is the volume of clients. This factor affects the dynamic behavior in vendors. For example, client volumes determine the types of relationships (i.e., interaction process). At Level 2, when

---

4 GlobalSchool has not reached this maturity level. Therefore, we asked the interview participants for their opinions on what would happen during Level 4 maturity.
number of clients is small, vendors are able to cultivate close relationships with their clients. Meanwhile, at Level 3 and 4, vendors’ relationships with their clients change. Vendors become less able to cultivate close relationships as their number of clients increase. As a result, Level 3 has a mix of close and distant relationships, and Level 4 has mostly distant relationships. We summarize our findings into Figure 1 (p. 17).

In the following sections, we continue our discussion by presenting the contributions of this study. We also provide the managerial implications, limitations and suggestions for future research.

**Discussion**

Our objective in this study is to address the research question, “How do value co-creation components – value, offering, value networks, user involvement, and interaction process – change over time as SaaS configurability moves toward maturity?” Our findings offer important theoretical and managerial implications. In this section, we outline the implications, assess the limitations, and consider avenues for future research.

**Theoretical and Managerial Implications**

First, we contributed to the growing SaaS literature. Specifically, to the SaaS literature related to software maturity. Available studies have approached SaaS maturity from the technical view (Arya et al. 2010; Kwok et al. 2008; Nitu 2009; Sun et al. 2008; Wang and Zheng 2010). These studies examine the architectures of SaaS providers, and how different types of architectures support different configurability levels. Our study incorporated socio-technical elements into SaaS maturity. We investigated the roles of clients and vendors in moving towards client-enabled configurability from the value co-creation framework. We show that through configurability options, SaaS vendors attempt to follow the marketing principle of mass customization wherein the offering consists of two production concepts that at first glance seem to be opposite – mass production and customization. Mass production attempts to offer standard products to a mass market at low costs whereas customization strives for satisfying individual customers’ needs with a comparable efficiency (Da Silveira et al. 2001). The main objective is that nearly everyone finds exactly what they want (Pine and Davis 1993) at an affordable price. Furthermore, we integrated the concept of self-service into the SaaS maturity model. We portrayed SaaS maturity as a single-line spectrum that goes from vendor-supported to client-enabled configuration. This is the central premise of the SaaS business model which has been missing in current research.

Second, one of the criticisms in SaaS is the control and ownership that vendors have over the software. Clients do not own the software and do not govern the future growth of the software (Plummer 2011). However, in our findings, we found that client-enabled configurability options gave clients a sense of ownership. Demirkan et al. (2011) argued that in a service system, ownership is actually accountability. Extending Demirkan et al.’s (2011) argument to the concept of self-service in SaaS, we argue that clients feel ownership when they are responsible for configuring the software themselves.

Third, we shed new insights into the concept of client-enabled and vendor-supported configurations. In our study, we found that for a particular configuration option, SaaS vendors do not have to cater to all possibilities. Trying to cater to all possibilities only complicate the configurability option further, and thus, pushes SaaS vendors to provide vendor-supported configurations. To offer client-enabled configurations, SaaS vendors only need to offer most common options to their clients. Another interesting insight relates to the concept of simple and intuitive configuration tools. We found in our literature review that tools and options need to be simple and intuitive to facilitate co-creation of value. However, in our case study, we also found that even complex configuration tools can indirectly aid in co-creation of value. Specifically, when configuration tools are complex, both clients and SaaS vendors resort to vendor-supported configuration options which require more communications between the two parties. The feedback during these interactions enables vendors to enhance their software applications and provide client-enabled configurations later on.

And fourth, we contributed to the co-creation of value framework. Our findings show that knowledge acts as a driver for change in the value co-creation framework. This is not surprising because knowledge is regarded as an important ingredient in value co-creation (Payne and Holt 2001; Pralahad and...
Ramaswamy 2004; Vargo et al. 2008). Other than knowledge, we identified volume of clients as a driver for change on the part of vendors. To the best of our knowledge, this has not been identified in the extant literature.

We also outline several managerial implications of this study. These managerial implications include:

First, SaaS vendors need to balance between providing support and self-service. We found that with the increase of self-service configurations, communications between clients and vendors become less frequent. This has negative implications to building relationships and obtaining quality feedback. We suggest SaaS vendors, especially the more matured ones, find alternate ways to initiate contact with clients. Vendors could provide feedback forms or surveys, and establish protocols to contact clients.

Second, feedback from long-standing clients should not be dismissed as it may provide more value. We urge SaaS vendors to be proactive in seeking feedback from these clients. As found in our case study, older clients such as New Academy develop an understanding of the “ins and outs of the system.” Thus, they are able to give better or more precise feedback.

And last, clients need to develop an understanding of the software before they could appreciate self-service. Therefore, vendors need to provide some support to new clients. This is true, even for vendors who have reached a high maturity level. Perhaps, vendors should provide support at the beginning of the software adoption, before slowly transitioning the clients to full self-service (i.e., still provide support even though full self-service configurations are available).

**Limitations and Future Research Directions**

Having a single organization as the source of data is a potential limitation of our study. However, studying a single site is acceptable as the mode of generalization appropriate for a qualitative study such as ours is analytic generalization – generalization to theory rather than to population (Yin 1994). The single site does make this study vulnerable to the idiosyncrasies of the said vendor. Therefore, it is important to articulate the idiosyncrasies in GlobalSchool that may influence our findings. GlobalSchool provides administrative software for schools. This implies that GlobalSchool’s software supports various types of organizational workflows. We believe that GlobalSchool may require more configurability options, or configurations options may be more important. Future research studying different types of SaaS vendors is needed to determine if our findings can be generalized to other types of SaaS vendors.

Another limitation of this study is in having only one client to be interviewed. Our immediate plan is to expand this case study to include interviews with multiple clients. More specifically, we would like to interview clients who adopted GlobalSchool’s solution at different points in times. Findings from the related literature suggest that initial clients derive value by getting access to an on-demand customized software application that satisfies their needs. As the software maturity increases, late clients derive value by accessing a mass-customized software application that carries out the best practices of the market sector (Squire et al. 2004). We also acknowledge that the evidence to support analysis at Level 4 and 5 is not as strong because the evidence is “projected” as opposed to “actual.” However, we emphasize that there is an opportunity for a longitudinal approach that follows SaaS maturity from early to late maturity levels.

This study shows that SaaS maturity is a dynamic process; the changes go beyond changes to the software architecture. Value co-creation components change at different maturity levels. An interesting avenue for future study is to consider factors that would impede this dynamic process, which in some cases are not the direct opposite of enabling factors. Greater attention must be paid to communication structures and facilities, vendors’ business strategies, and institutional pressures (e.g., laws and regulations).

**Conclusion**

In conclusion, we hope to have advanced research on SaaS by drawing attention to configurability and maturity. Our study highlights self-service as an important element in SaaS configurability. We also show that SaaS maturity can be examined through the value co-creation framework. We found that the components of the value co-creation vary across different maturity levels. Knowledge and volume of clients act as drivers for change. We hope that the ideas and results in this study become sound building blocks in increasing understanding of SaaS maturity.
Figure 1. An Integrated Framework for Configurability, Maturity, and Co-creation of Value in SaaS
References