Inf Syst E-Bus Manage (2018) 16:327–356
https://doi.org/10.1007/s10257-017-0356-9

ORIGINAL ARTICLE

Providing Software as a Service: a design decision(s) model

Abhijit Dutt[1] · **Hemant Jain**[2] · **Sanjeev Kumar**[3]

Received: 6 February 2017 / Revised: 24 May 2017 / Accepted: 7 June 2017 /
Published online: 19 June 2017

Abstract We examine how Software as a Service (SaaS) providers make different design decisions using a theoretical model. We consider two non-functional attributes: modularity of the software architecture and the architectural performance of the software. We model the relationship of these two attributes with factors such as user preferences, user demand, and the price of the service. In a significant departure from traditional models of IS product development, we considered marginal cost and maintenance cost of providing SaaS service to recognize that the SaaS service has characteristics of both a product and a service. We show how to find the optimal values of design attributes that maximize SaaS provider's profits, taking into account relevant factors such as user preferences, user demand and service price. Our research provides one of the first analytical models of optimal design decision making by SaaS providers. We use the model to further show how the SaaS providers should adjust the service design in response to changes in user preferences, associated costs and other relevant factors.

Keywords Cloud Computing · Software as a Service (SaaS) · Modularity · Architectural performance

✉ Abhijit Dutt
adutt@gmu.edu

Hemant Jain
Hemant-Jain@utc.edu

Sanjeev Kumar
sankum@umich.edu

[1] George Mason University, Fairfax, VA, USA

[2] University of Tennessee – Chattanooga, Chattanooga, TN, USA

[3] University of Michigan, Ann Arbor, MI, USA

1 Introduction

"Cloud Computing" has become so popular that it was included as a new word in the English language in 2012 with the following definition: *the practice of storing regularly used computer data on multiple servers that can be accessed through the Internet* (MerriamWebster 2012). A more technical definition is provided by NIST: *Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction* (Mell and Grance 2011). Cloud Computing refers to both the software applications delivered as services over the Internet and the hardware infrastructure such as datacenters that provide those services; offered as a service as well. The software applications that are offered as a service using a cloud environment are referred to as Software as a Service ("SaaS") (Armbrust et al. 2010). In this paper, we explore the role of the service provider in designing the service in the SaaS context. Specifically, we build a model of the two main design decisions—modularity and architectural performance, facing the SaaS provider, to discover insights into this rapidly evolving and very important component of Cloud Computing.

Cloud computing has seen explosive growth in recent years. According to IDC worldwide public cloud services spending in 2017 will reach $122 Billion and is expected to reach $203.4 billion by 2020. SaaS solutions will account for about 60% of this spending (IDC 2017). This rapid growth in SaaS is driven by innovative market offerings by SaaS providers such as Salesforce.com, SAP, Oracle and Microsoft (Kanaracus 2013). Marston et al. (2011) identify SaaS providers as a key stakeholder in Cloud Computing and detail their role to include owning and operating Cloud Computing systems to deliver services to third parties, performing maintenance and upgrades on the system, maintaining the software used on the cloud and the pricing of the cloud services.

For SaaS applications, the provider is responsible for designing, developing, deploying and operating the application. Recent research has focused on the SaaS provider. Ma and Seidmann (2008) looked at pricing strategy for a service provider in competition with a traditional software provider. Jiang and Seidmann (2014) looked at capacity planning for service facilities. Choudhary (2007a, b) showed that the SaaS model results in higher investment in product development and higher software quality compared to perpetual licensing. Marston et al. (2011) identify several issues related to SaaS provider such as pricing strategy, SaaS provider's economic value, risk transfer etc. as part of the suggested research agenda in Cloud Computing. However, little research attention has been devoted to understanding the decisions facing the SaaS provider in designing the service itself. In this paper we attempt to address this research need by exploring the two significant design decisions facing the SaaS providers: modularity and architectural performance of the service.

Software development has been well studied and there exists a significant body of research on software development organizations. Although current software development research provides a good foundation for studying SaaS providers, SaaS providers differ significantly from traditional software developers, mainly because SaaS applications differ significantly from traditional software. For example—traditional software is considered to have substantial development cost and negligible marginal cost of production (Krishnan and Zhu 2006). SaaS, on the other hand, may involve significant marginal cost of maintaining the infrastructure and providing the service. Similarly, while traditional software development focuses on requirements of a given client or a user, SaaS applications are usually designed to accommodate a broad range of users with different requirements. In this paper we have attempted to account for the different nature of SaaS and SaaS providers compared to their traditional counterparts by borrowing from Economics, Marketing and Services Science literature to augment our understanding from the extant Software Development literature.

Cloud Computing and SaaS specifically is a part of the growing "servitization" from a product service dichotomy to a product service continuum (Vandermerwe and Rada 1989). Transitioning from product manufacturer to service provider presents significant managerial challenges with only a sparse literature available to guide the transition (Oliva and Kallenberg 2003). Demirkan and Dolk (2013) discussed the need of understanding the service ecosystem. This represents an opportunity for novel research especially with focus on design of integrated product service offerings (Baines et al. 2009). Rai and Sambamurthy (2006) claimed that the growing interest in services management represents a research opportunity specifically for Information Systems ("IS") researchers in the context of design principles for digitally enabled services. Benlian et al. (2011) developed a measure for examining software quality in SaaS and asked for further research on design of SaaS services to achieve higher service quality. Other researchers have also emphasized the need for further research on problems facing service providers regarding service design and deployment (Spohrer and Maglio 2008; Demirkan et al. 2009; Zhang and Seidmann 2010). In this paper we follow this call for research and focus on the challenges faced by a service provider in the context of SaaS.

The rest of the paper is structured as follows. The next section reviews the relevant literature across several research disciplines. In Sect. 3 we build an analytical model of service provider's design and deployment decisions. Section 4 presents results from the model along with their managerial and research interpretation. We then conclude the paper in Sect. 5 by discussing results, their implications, limitations of the research and suggesting avenues for further research inquiry.

2 Literature review

The literature on Cloud Computing is continuing to evolve. In this section we will first review the relevant extant literature on Cloud Computing and then augment our understanding by exploring relevant literature in software development, services

science, marketing, economics and operations. We keep the focus of the literature review on the SaaS provider and the two non-traditional attributes of the SaaS service—modularity and architectural performance.

Cloud Computing is defined by Vaquero et al. (2008) as *a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs*. The above definition identifies the two main stakeholders: the provider of the cloud-based services and the customer of such services. This research focuses on the providers of services. The definition also indicates that the service provider can provide a range of services.

The services can be offered at different levels of abstraction depending on the type of specific access and control afforded to the customer. Cloud based services are typically classified into the following three levels of abstractions:

- *IaaS—Infrastructure as a Service* IaaS is the delivery of hardware and associated software as a service. It allows users to provision resources on demand. The IaaS provider has little responsibility except to keep the data center operational while the users deploy and manage the software services themselves (Bhardwaj et al. 2010). Typical examples of IaaS are Amazon's Elastic Compute Cloud (EC2) and Secure Storage Service (S3).
- *PaaS—Platform as a Service* PaaS provides an additional level of abstraction over IaaS. Along with the hardware infrastructure, PaaS also provides the software platform for the users to develop and run their own systems (Vaquero et al. 2008). Well known examples include Google's App Engine and Microsoft's Azure.
- *SaaS—Software as a Service* SaaS allows customers to use the provider's applications running on cloud infrastructure. The applications are accessible through interfaces such as the web browser (Mell and Grance 2011). SaaS enables a provider to offer IS application as a service to customers by hosting the IS application in their own IT infrastructure or infrastructure obtained as a service from an IaaS provider. In this case, customers only need minimal IT infrastructure for consuming the service. Customers only have limited user specific configuration capability and do not manage or control the underlying infrastructure and application platform (Höfer and Karagiannis 2011). Popular examples include G Suite, SalesForce CRM, Microsoft Office365.

Among the three categories, SaaS is at the highest level of abstraction as the customers are given access to a fully developed, self-sufficient, software service delivered over the cloud infrastructure. The SaaS provider carries the entire responsibility for design, development and delivery of the SaaS service. In this research, we focus on the design decisions facing the SaaS provider while designing the SaaS service.

There are a few published attempts at studying Cloud Computing, specifically decision making by providers, through the lens of analytical modeling. Niyato et al. (2011) studied cooperation between multiple cloud providers through a hierarchical cooperative game model to investigate decision making of cloud providers when cooperation can lead to higher profits. Toosi et al. (2011) studied the problem of increasing resource utilization for an IaaS provider in the context of a Cloud Federation. A Cloud Federation allows providers to trade their resources with each other to overcome resource limitations in their local infrastructure (Rochwerger et al. 2009). Carrying forward the idea, Goiri et al. (2010) developed a model of the operations of a Cloud Federation to examine how a provider can enhance its profits by exploiting the federation. A similar idea of virtual organizations was analyzed by Carroll and Grosu (2010) through a game theoretic framework to design a resource management system supporting the virtual organization. While the studies mentioned above provide a foundation for analytical models to study decision making of Cloud Computing Providers, to our knowledge no study has developed a model of design decisions of SaaS providers. Our research attempts to contribute to this gap.

SaaS architecture and service design has attracted significant research interest. We focus our literature review on the two aspects of SaaS service design that are key to this paper: Modularity and Architectural Performance.

Modularity in designing products, services and systems, including Information Systems have been well studied. Still, modularity remains an ill-defined construct. In the context of modularity in services, Hyötylainen and Moller (2007) argue that modularization aims at packaging individual functionalities so that the functionalities in one module have as much in common as possible and that the modules themselves are as reusable as possible. Booch (1993) defined modularity as property of a system that has been decomposed into a set of cohesive and loosely coupled modules. Coupling is defined as how inter dependent two modules are and cohesion is defined as how single minded a module is (Yourdon and Constantine 1979). In IS applications that use Service Oriented architecture (SOA) as its architectural model, the software services are accessed through its interface. Hence, if a customer builds a custom application using the service interface the service provider can modify elements within the modules that does not impact service interface without affecting the customer, thus making it easy to update or replace the modules (Janssen and Joha 2008).

In his seminal contribution, Parnas (1972) discussed modularization as a mechanism for improving the flexibility and comprehensibility of a system while allowing for shortening of its development time. Modularity in software architecture allows for information hiding or abstraction along with decomposition of the larger program into modules leading to simplification of the software development process (Parnas et al. 1985). Modular architecture has been shown to improve flexibility and reduce the cost and difficulty of adapting and coordinating (Sanchez and Mahoney 1996; Schilling 2000). Looking at service design, Pekkarinen and Ulkuniemi (2008) identified four dimensions of modularity: service, process, organizational and customer interface; and showed that modularity can be used to develop and deliver services cost efficiently and more flexibly. Hsu and Lin (2016) studied adoption of

cloud services in enterprises. Rule et al. (2008) discussed approach to measuring design modularity.

Modularity in software architecture has also been shown to affect maintenance costs of the software. Maintenance cost has been shown to be proportional to the size of the modules; in addition non-modular products are more expensive to maintain than modular products (Banker et al. 1993). Modularity in design has long been established as an effective mechanism to counter the negative effects of complexity in design (Baldwin and Clark 1997, 2000; Welch and Waxman 2003). Specifically, modularity allows for mass customization through assemble-to-order where a large variety of configurations can be built through a small number of modules (Da Cunha et al. 2007). Vitharana et al. (2004) show that business strategy and management goals can inform the design of modular, component based system to improve business-IT alignment. Degree of modularity has been shown to support both extensive customization and rapid new product development (Voss and Hsuan 2009). Modularity in architecture leads to ease of bundling different modules together and bundling has been shown to improve profits for the producer (Bakos and Brynjolfsson 1999). Chidamber and Kemerer (1994) provides a comprehensive metrics suite for object oriented design. It has six measures of software architecture design such as coupling between object classes, lack of cohesion in methods, response for a class which can be adopted for practical use. The design of metrics suite is based on strong theoretical base and was tested on two large size real industry projects.

Software with modular architecture has been shown to be, in general, better software. Modularity in SaaS services is especially important since it can have significant effect on how customers interact with the service. For example—modular architecture in SaaS was found to be important in achieving task disaggregation that led to easier management and governing through high powered incentives (Susarla et al. 2010). SalesForce.com, one of the most successful SaaS providers, has seen a steady increase in their sales and customer base that has been attributed to the flexibility of its Application Programming Interface (API) Force.com that allows users to quickly assemble their own customized applications. The Salesforce.com website notes: *Force.com comes with 60 predefined components that can be assembled with minimal coding in building-block fashion. Some of these components implement common Salesforce interface elements and others make new features available, such as AJAX-based partial page refreshes* (SalesForce.com 2014). Similarly, in case of another successful cloud computing provider, Amazon Web Services, the Vice President of Product Management and Developer Relations, notes: *in making its capabilities accessible to outside developers, Amazon broke its process into many modular services. This modularity has allowed Amazon to extend its business all the way to providing a complete online retailing environment* (PWC 2010). Thus, we can conclude that a modular architecture is desirable in SaaS and can lead to higher customer demand. However, SaaS customers may have differing levels of preference for modularity in SaaS service architecture. In this research, we specifically take customer's preference for modularity into account and build our model to allow the SaaS provider to design the service with the optimal level of modularity to maximize its profits. Voss and

Hsuan (2009) argue that the concept of modularity has been used to any significant extent in the design of services. We contribute to this research gap by explicitly considering the service provider's decision to build a level of modularity in the service to be provided.

The literature on architectural performance is not very extensive. Here we make an important distinction between architectural performance and operational performance. We consider architectural performance to be an attribute of software architecture—whether the architecture supports high level of performance. Thus, architecture performance, as conceptualized in this paper, can be considered as a design characteristic. Alternatively, a service can achieve higher performance through application of more resources (more bandwidth, more processing power, more storage etc. in case of SaaS) with the resulting higher costs. However, that is not the focus of our study since we focus only on the design stage of the service not the delivery. Making this distinction is necessary in the case of SaaS since it carries the characteristics of both products and services (Vandermerwe and Rada 1989). Architectural performance relates to the product part of SaaS while operational performance relates to the service part of SaaS.

Characteristics of software architecture that impact performance have been well studied in software engineering literature. Software architecture has been shown to have a significant effect on software performance (Williams and Smith 1998). Deciding on the optimal architecture is an important decision since the architecture is decided during the design phase and cannot be easily changed later in case of underperformance of the software (Williams and Smith 2002; Balsamo and Marzolla 2005). Architectural performance is especially important for SaaS providers since the SaaS architectures typically include aspects such as virtualization and resource time sharing that may have significant negative impact on service performance (Iosup et al. 2008). Further, performance of IS services have been shown to have significant impact on pricing and demand for the services. For example—higher performance of online services has been shown to allow vendors to charge higher prices (Jain and Kannan 2002). Hosanagar et al. (2005) studied the role of performance on the pricing of IS services and showed that it was possible for service providers to charge for a premium service even when a free service was available to customers.

Several researchers have argued for integrating performance analysis in the software development process, specifically during the software design phase (Balsamo et al. 2004). Traditional software development methods focus on matching the functional requirements and software correctness or bug-free development, leaving performance issues to be considered later in the development process. This approach does not take into account the fact that performance problems may require considerable changes in architecture of the software. In this research, we attempt to contribute to the research gap by introducing performance consideration at the service design phase and explicitly including architectural performance as a key element of the provider decision making. We start with the intuitive assumption that higher operational performance will lead to higher customer demand. As higher architectural performance leads to higher operational performance, we conclude that higher architectural performance will lead to higher

customer demand as well. However, as with modularity, we allow SaaS customers to differ in their sensitivity to architectural performance. Our model then allows for the provider to react to customer sensitivity to performance by choosing the architectural performance level that maximizes its profits.

Pricing of SaaS services is one of the main decisions made by the service providers. Harmon et al. (2009) argue for innovative value based pricing rather than the traditional cost based pricing for IT services (Rohitratana and Altmann 2012). Using an agent based simulation model, Rohitratana and Altmann (2012) show that a demand driven pricing scheme is the most effective method for SaaS services but it is also hard to implement since it requires perfect knowledge of market conditions. In this paper we continue the research forward by analyzing optimal pricing for a given context.

3 Model construction

We use theoretical modeling technique for our model formulation. A theoretical model could be of two different types—verbal or mathematical. We use mathematical modeling in this paper. A mathematical model consists of appropriate variables for the phenomenon under investigation as well as realistic assumptions on relationships among the variables along with optimization criteria. For example—in this paper we use profit maximization by a SaaS provider as the optimization criteria.

Once a mathematical model is built, then a researcher performs theoretical experiments by examining effects of change of some of the variables of the mathematical model. From the results of the above theoretical experiments, propositions are developed and managerial implications of the phenomenon are uncovered from the propositions (Moorthy 1993).

We start by modeling the demand function. We assume a monopolist vendor and we consider a fixed period (lifetime of the service) during which the SaaS is being offered. However, modeling a service is very different from modeling a product. Service could be defined as a relationship between a producer and a customer that creates and captures value and where the customer participates actively (Gadrey 2000; IBM n.d.). In other words, in the case of services, the customer could be considered as co-producers (Fitzsimmons and Fitzsimmons 2004). It requires that we model demand in an innovative way. When a customer subscribes to the service throughout its full lifetime we recognize that as unit demand. We recognize price as the total subscription charge for the service throughout its full lifetime. Hence, it is possible that a customer only purchases the service for a period less than its full time and pays only fraction of the total price. Unlike traditional product, it is possible that the demand of a SaaS is not an integer.

We model the effects of modularity and architectural performance on demand. Neither modularity nor architectural performance is a new construct in the IS area. However, we posit that it is necessary to look into those constructs using a service lens and to our knowledge that has not been done yet.

Kumar (2004) observed that modularity in product design enables a producer to offer mass customized product. Dewan et al. observed that using mass customization, it was possible for a producer to offer different variations of a product at different prices leading to increased profit by making the product attractive to a more diverse group of customers (Dewan et al. 2003). For services, the relationship between producers and customers continue throughout the lifetime of a service; using modularity, producers will be able to make changes to the service and customize appropriately for the customers. Hence, we assume that increase in modularity will give rise to more demand for the product.

Next we focus on the attribute Architectural Performance as it has some unique benefits for services. A service designed for higher architectural performance could be used by a provider to offer better service to a larger number of customers without making any changes in the infrastructure. A higher architectural performance will make the service attractive to a larger number of customers. Hence, we assume that higher architectural performance will lead to higher demand.

We assume a linear demand function and we include the sensitivity in demand from modularity and architectural performance to capture SaaS provider's response to customer's preferences regarding modularity and architectural performance. Although linear demand function has some limitations, linear demand function is widely used in the literature and considered adequate as a starting point (Barua et al. 1991; Choudhary 2007a, b).

$$d = \alpha - \beta p + \gamma m + \delta s \tag{1}$$

where p is price of the SaaS application amortized over its lifetime, m is the modularity level of the SaaS application, and s is the architectural performance level of the SaaS application.

α is primary demand due to functional attributes of the SaaS application and other non-functional attributes such as quality (except modularity and architectural performance), brand image, general economic fact that are outside the scope of this paper.

β represents price sensitivity of the demand, γ represents increase in demand from increase in modularity, and δ represents increase in demand from increase in architectural performance.

α, β, γ and δ are assumed to be greater than zero. All the terms used for building the models in this document are summarized in the Glossary of Terms presented in "Appendix 1".

As we discussed earlier, we also assume that both demand and price of the service is amortized over the lifetime of the service. If a customer subscribes to the service throughout its lifetime, the price is the total amount the customer will pay and correspondingly demand will be one unit.

Next we formulate the cost function. It is much more complex to model cost of services as unlike traditional IS, SaaS is not a development intensive product and a SaaS provider incurs three different types of costs—fixed and marginal. We will first work with fixed cost which involves the cost of developing the product as well as costs for setting up the necessary IT infrastructure so that the SaaS application could be offered to the customers.

We assume that development of modular software will require more production cost for vendors (i.e. higher upfront or fixed cost) (Bush et al. 2010). Similarly design of high performing architecture requires significant effort and time resulting in higher development cost. We assumed that the fixed cost arising from increased modularity and architectural performance to be a quadratic function of the modularity and performance respectively. This is in line with the standard practice in the IS literature. For example—fixed costs incurred to improve quality of a product is a convex function of the slope of product improvement curve (Choudhary 2007a, b).

We can express the fixed cost function as:

$$c_1 = A_1 + Cm^2 + Ds^2 \tag{2}$$

where A_1 is the fixed cost arising from factors other than modularity and performance, C is the parameter related to modularity during design and development of the application, and D is the parameter related to architectural performance during design and development of the application.

Previous research has indicated that a modular software is easier and cheaper to maintain (Banker et al. 1993). We treat maintenance cost (c_2) as amortized over the lifetime of the product as a fixed cost. Hence, c_2 can be expressed as:

$$c_2 = A_2 - Bm \tag{3}$$

where A_2 is the amortized maintenance cost over the lifetime of the product and B is the parameter related to modularity showing the saving in maintenance cost arising from modular design also amortized over the lifetime of the product.

Now we turn to marginal cost. The service provider incurs a marginal cost per service because in order to provide the service to a large number of customers, it is necessary to have an ongoing, well maintained IT infrastructure. This marginal cost consists of two parts. There could be a onetime cost of purchasing the infrastructure and a variable cost of maintaining the infrastructure and providing the service. Alternatively, the infrastructure could be obtained as a service from an IaaS provider and in that case the marginal cost will only consist of a variable component. Third, there will be a cost for maintaining the SaaS application itself. In our model, we include both maintenance cost and marginal cost per product as amortized over the lifetime of the product.

Marginal cost of providing software service is unique to service providers—traditional software vendor typically do not have significant marginal cost. This marginal cost will include both the cost of setting up infrastructure such as hardware, software as well as the amortized cost of providing the service. Thus, the marginal cost of providing service c_3 can be considered proportional to demand:

Hence, c_3 can be expressed as:

$$c_3 = Zd \tag{4}$$

where d is the amortized demand of the service, and Z is the marginal cost for servicing a unit demand amortized over the life of the service.

Adding (2), (3), and (4), our total cost function can be expressed as:

$$\theta = A - Bm + Cm^2 + Ds^2 + Zd \tag{5}$$

where $A = A_1 + A_2$.

We observe that in the above cost most of the terms are independent of demand. Hence, it shows that economy of scale plays an important role in SaaS.

Profit for a software producer (π) could be represented as:

$$\pi = dp - \theta$$

Using (1) and (5), the above profit can be rewritten as:

$$\pi = (\alpha - \beta p + \gamma m + \delta s)(p - Z) - (A - Bm + Cm^2 + Ds^2) \tag{6}$$

Our objectives are to find optimal values of price (p), modularity (m) and architectural performance (s) that will maximize the above profit function.

3.1 Boundary condition

We assume the boundary condition that Primary demand of a product α is greater than product of marginal cost Z and price sensitivity β or $\alpha > \beta Z$.

In any IS application, it is fair to assume that functional attributes are much more important than non-functional attributes. Although non-functional attributes are important in this case, it is unlikely that importance of non-functional attributes such as modularity and performance will be more than those of functional attributes and other non-functional attributes such as security. It is unlikely that a producer will produce a product based on only modularity and performance and without any regard to its functionality. We consider a case where both performance sensitivity (δ) and modularity sensitivity (γ) are close to zero or in other words, the customers do not care about the product's modularity and performance. Following Eq. (1), we can rewrite the demand function as: $d = \alpha - \beta p$.

In the above case, it is unlikely that a producer will produce product that does not have a projected positive demand. Hence, we assume that in this simplistic situation also demand should be positive. Hence, we obtain: $\alpha > \beta p$.

As Z is the unit marginal cost, it is fair to assume that p must always be greater than Z, since otherwise it does not make sense for a producer to produce any product. As we are modeling the situation during design and production phase, we exclude the possibility of a fire sale. Hence it follows that:

$$\alpha > Z\beta \tag{7}$$

We shall assume the above boundary condition in our model.

4 Results and analysis

We consider two cases. In the first case we assume no relationship among the two decision variables—modularity and architectural performance. In the second case, we assume that modularity and architectural performance are related to each other.

4.1 Case 1: Modularity and architectural performance are not related

In this case, we assume modularity in software architecture and the architectural performance are independent of each other. In order to find optimal values for our decision variables that will maximize the profit function, we differentiate profit (Eq. 6) partially with respect to p, m, and s, set them equal to zero, and express them as p^*, m^*, and s^*. After making some simplifications, we obtain the following expressions.

$$p^* = \frac{\alpha + Z\beta + \gamma m + \delta s}{2\beta} \tag{8}$$

$$m^* = \frac{B + \gamma(p - Z)}{2C} \tag{9}$$

$$s^* = \frac{\delta(p - Z)}{2D} \tag{10}$$

Solving above equations by substituting p^*, m^* and s^* for p, m and s respectively, we obtain the optimal values of decision variables as expressed below.

$$p^* = \frac{2CD\alpha + BD\gamma + Z(2CD\beta - D\gamma^2 - C\delta^2)}{4CD\beta - D\gamma^2 - C\delta^2} = \frac{2CD(\alpha - Z\beta) + BD\gamma}{4CD\beta - D\gamma^2 - C\delta^2} + Z \tag{11}$$

$$m^* = \frac{4BD\beta + 2D\alpha\gamma - 2DZ\beta\gamma - B\delta^2}{2(4CD\beta - D\gamma^2 - C\delta^2)} \tag{12}$$

$$s^* = \frac{(2C\alpha + B\gamma - 2CZ\beta)\delta}{2(4CD\beta - D\gamma^2 - C\delta^2)} \tag{13}$$

Substituting (8–10) to (1) and (6), optimal demand and profit can be expressed as:

$$d^* = \frac{D\beta(2C(\alpha - Z\beta) + B\gamma - 2C)}{(4CD\beta - D\gamma^2 - C\delta^2)} \tag{14}$$

$$\pi^* = \frac{4D(\alpha - Z\beta)(B\gamma + C(\alpha - Z\beta)) + B^2(4D\beta - \delta^2)}{4(4CD\beta - D\gamma^2 - C\delta^2)} - A \tag{15}$$

Based on the above optimal values of demand and profit, we first begin by conducting a sanity check on the model and checking that the model results in plausible solutions for optimal values.

Lemma 1 *All the decision variables as well as the optimal demand and profit have positive values when* $4CD\beta > (D\gamma^2 + C\delta^2)$.

Proof The above condition is derived from the Hessian matrix. The Hessian matrix is the second-order partial derivative of the profit function with respect to the variables p, m and s in the present case.

$$H = \begin{pmatrix} -2\beta & \gamma & \delta \\ \gamma & -2C & 0 \\ \delta & 0 & -2D \end{pmatrix}$$

To ensure that profit has a local maximum the determinants of the Hessian matrix need to be negative semi definite. A matrix is negative semi definite when its leading principal minors of different orders alternate in sign, starting with negative for the first leading principal minor leading principal minor (Winston 1993). Hence the principal minor of order 3 has to be negative. The only principal minor of order three is the determinant of the matrix itself which is $2(-4CD\beta + D\gamma^2 + C\delta^2)$, and that leads to the following condition.

$$4CD\beta > (D\gamma^2 + C\delta^2) \tag{16}$$

We note that first principal minor of second order is $(4C\beta - \gamma^2)$. However, it is positive when Eq. (16) is assumed. Using Eq. (16) as well as the boundary condition, the numerators and denominators in Eqs. (11–15) are all positives. □

Our model considers two customer parameters—customer sensitivity to modularity γ and customer sensitivity to performance δ. The model includes three vendor decision parameters—price, modularity and architectural performance. We would expect that changes in customer sensitivity to modularity will affect the level of modularity offered by the vendor and the price charged by the vendor for a given level of modularity which will impact demand. Similarly, we would expect that changes in customer sensitivity to performance will affect the level of performance offered by the vendor and the price charged by the vendor for a given level of performance which will impact demand. We explore these relationships in Propositions A1–A4 below. Proofs for all the propositions in this paper are presented in the "Appendix 2".

Proposition A1 *When customer demand is more sensitive to modularity, the vendor will offer a product that is more modular and will be able to charge a higher price. It will lead to higher demand and higher profit for the vendor.*

We show that if all other parameters remain the same, increase in customer preference for modularity will enable vendors to charge a higher price by providing a product with higher modularity. As the product now matches better with the customer preference, the vendor will experience a higher demand and will also be able to charge a higher price resulting in a higher profit for the vendor.

Proposition A1 provides an intuitive result. A vendor is better off by changing the product to better fit customer preferences. However, Proposition A1 does not take

into account the cost of introducing modularity in the product. We will study the impact of the cost of introducing modularity in Proposition A3.

Proposition A2 *When customer demand is more sensitive to performance, the vendor will offer a product with higher performance level and will be able to charge a higher price. It will lead to higher demand and higher profit for the vendor.*

Similar to Proposition A1 for modularity, Proposition A2 provides an avenue for the SaaS vendor to attract a larger demand, charge a higher price and consequently achieve a higher profit by adjusting the product design to better fit customer preferences.

Note that the customer only perceives the overall performance of the service. This performance can be achieved through better design (architectural performance) or through deploying better operational parameters such as faster connectivity, more processing power, more storage capacity etc. (operational performance). While both routes result in higher performance, their cost impact is very different. Architectural performance is achieved through higher development cost or fixed initial cost with no effect on ongoing marginal cost while increase in operational performance could be achieved by using better infrastructure (leading to higher fixed cost) and also by running and maintaining the infrastructure more efficiently (leading to higher marginal cost). We recognize that service providers may not choose to develop their own infrastructure and instead they could rent their infrastructure from another provider. However, the cost of better infrastructure will still be higher.

Now we consider the three environment parameters considered in the model—the fixed cost parameter related to modularity (C), the fixed cost parameter related to architectural performance (D) and the marginal cost of providing the service (Z). Proposition A3 explores the impact of fixed cost parameter related to modularity on the optimal price, optimal modularity level and optimal performance level. Proposition A4 explore the impact of fixed cost parameter related to performance on the optimal price, optimal performance level and the optimal modularity level. Finally, Proposition A5 explores the impact of the marginal cost on the optimal price.

Proposition A3 *As the cost of increasing modularity in the software increases, the optimal modularity level, optimal performance level and the optimal price will be lower, resulting in lower demand and profit for the vendor.*

Proposition A4 *As the cost of increasing architectural performance of the software increases, the optimal performance level, optimal modularity level and the optimal price will be lower, resulting in lower demand and profit for the vendor.*

Propositions A3 and A4 indicate that as the cost of increasing modularity and architectural performance increases, the price as well as the optimal modularity and architectural performance levels decrease. It is interesting that our model leads to the counter-intuitive result that an increase in a cost element will lead to lowering of price. However, this result is logically consistent. As the cost of increasing modularity becomes higher then, ceteris paribus, the SaaS provider will not benefit

by making the product more modular. Instead, the vendor will produce a product with lower modularity levels and would have to accept a lower price for that service.

Proposition A5 *As the marginal cost Z increases, the optimal price:*

- *increases if* $\beta > \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$
- *decreases if* $\beta < \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$
- *stays unchanged if* $\beta = \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$

Proposition A5 has important managerial implications. With an increase in marginal cost, all the decision variables except optimal price decrease. Depending on different conditions as indicated above regarding price sensitivities, fixed costs and other parameters, the optimal price may remain unchanged, increase or even decrease.

4.2 Case 2: Modularity (*m*) and performance (*s*) are related

A negative relationship between modularity and performance is well established in literature (Ulrich 1995). Lau Antonio et al. (2007) show that although modularity in product design is considered a key enabler of product success, modularity does not necessarily improve all the attributes of a product. The negative impact of modularity on performance was emphasized by the Internet Engineering Task Force (IETF) as: "modularity is one of the chief villains in attempting to obtain good performance, so that the designer is faced with a delicate and inevitable trade-off between good structure and good performance" (Clark 1982). Accordingly, we modify our model to include an inverse relationship between architectural performance and modularity such that an increase in modularity leads to decrease in architectural performance and vice versa.

We begin by assuming the following relationship between modularity and architectural performance:

$$s = X - Ym \tag{17}$$

where X and Y are parameters. X shows the maximum performance level and Y shows the ratio of change in s to change in m. It also follows that maximum value of m is X/Y (when s is zero). We also note that neither X nor Y is fixed. We recognize that measuring X and Y could be difficult and more research is needed for the purpose. However, we envisage that it is possible to estimate X and Y especially if we look into the area of software product lines (Bosch 2010). If a software project uses "the practice of continuous integration" (Ståhl et al. 2017) by measuring the well-known software metrics such as Depth of Inheritance Tree (DIT) and Number of children (NOC) over different versions, it is possible to study the changes in DIT or NOC in which impact architectural performance level over different versions. Using those data it would be possible to develop metrics that measure X and Y.

Using Eq. (1) and eliminating s using Eq. (17) we can rewrite our linear demand function as:

$$d = \alpha - \beta p + (\gamma - \delta Y)m + X\delta \tag{18}$$

Using (6) and (17), we can reformulate our profit function as

$$\begin{aligned}\pi &= (\alpha - \beta p + \gamma m + \delta s)(p - Z) - (A - Bm + Cm^2 + D(X - Ym)^2) \\ &= (\alpha - \beta p + (\gamma - \delta Y)m + X)(p - Z) \\ &\quad - (A - (B + 2DXY)m + (C + DY^2)m^2 + DX^2)\end{aligned} \tag{19}$$

We differentiate profit partially with respect to p and m, set them equal to zero, and solve for p and m. After making some simplifications, we obtain the following expressions for optimal price and modularity:

$$p^*_{ms} = \frac{2(\alpha + Z\beta)(C + DY^2) - Z(\gamma - Y\delta)^2 + 2X(C\delta + DY\gamma) + B(\gamma - Y\delta)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2}$$

$$p^*_{ms} = \frac{2(\alpha - Z\beta)(C + DY^2) + 2X(C\delta + DY\gamma) + B(\gamma - Y\delta)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} + Z \tag{20}$$

$$m^*_{ms} = \frac{2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} \tag{21}$$

Using Eqs. (17) and (21), we can express optimal performance as

$$s^*_{ms} = X - \frac{Y(2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta))}{4\beta(C + DY^2) - (\gamma - Y\delta)^2}$$

$$s^*_{ms} = \frac{4\beta XC - 2\beta YB - (\gamma - Y\delta)(Y(\alpha - Z\beta) + X\gamma)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} \tag{22}$$

Substituting (20–22) to (18) and (19), our optimal demand and profit can be expressed as:

$$d^*_{ms} = \frac{\beta(2(\alpha - Z\beta)(C + DY^2) + 2X(Y\gamma D + C) + B(\gamma - Y\delta))}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} \tag{23}$$

$$\begin{aligned}\pi^*_{ms} &= \frac{B^2\beta + A(\gamma - Y\delta)^2 + B(\alpha - Z\beta + X\delta)(\gamma - Y\delta) + D((Y(\alpha - Z\beta) + X\gamma)^2 - 4\beta Y(AY - XB)) + C((\alpha - Z\beta + X\delta)^2 - 4\beta(A + DX^2))}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} \\ &= \frac{B^2\beta + B(\alpha - Z\beta + X\delta)(\gamma - Y\delta) + D((Y(\alpha - Z\beta) + X\gamma)^2 + 4\beta XYB) + C((\alpha - Z\beta + X\delta)^2 - 4\beta DX^2)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} - A\end{aligned} \tag{24}$$

4.3 Boundary conditions

We make the following assumptions regarding this model:

As in case 1, we assume Eq. (7) or $\alpha > Z\beta$.

We observe from Eq. (18), that the demand function has $(\gamma - \delta Y)$ as the effective sensitivity for modularity. It is fair to assume that sensitivity for modularity is never negative, or:

$$(\gamma \geq \delta Y) \tag{25}$$

Lemma 2 *All the decision variables as well as the optimal demand and profit have positive values when* $4\beta(C + DY^2) > (\gamma - Y\delta)^2$.

Proof The Hessian matrix is

$$H = \begin{pmatrix} -2\beta & \gamma - Y\delta \\ \gamma - Y\delta & -2C - 2DY^2 \end{pmatrix}$$

To ensure that profit has a local maximum the Hessian matrix needs to be negative semi-definite. A matrix is negative semi-definite when its leading principal minors of different orders alternate in sign starting with negative (Winston 1993). Hence the leading principal minor of order 2 needs to be positive. Hence, we obtain

$$4\beta(C + DY^2) > (\gamma - Y\delta)^2 \tag{26}$$

Using Eq. (26) as well as above boundary conditions, the numerators and denominators in Eqs. (20), (21), (23), and (24) are all positives.

We also assumed that maximum value of m is X/Y.

Hence,

$$m^*_{ms} = \frac{2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} < \frac{X}{Y}$$

From the above inequality,

$$X(4\beta(C + DY^2) - (\gamma - Y\delta)^2) > Y(2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta))$$
$$4\beta XC > 2BY\beta + (Y(\alpha - Z\beta) + X\gamma)(\gamma - Y\delta)$$

The above inequality shows that the numerator of Eq. (22) is positive. Hence, s^*_{ms} is also positive. □

As modularity and performance are related, we need not consider their effects separately. In this case we will keep our focus on modularity. We will first consider the impact of customer sensitivity to modularity on three vendor parameters—price, modularity and performance. We explore these relationships in Propositions B1. Since the proof for B1 is similar to the ones already presented for Propositions A1

and A2, we are only providing the Propositions below for brevity. The proofs have been presented in the "Appendix 2" for completeness.

Proposition B1 *When customer demand is more sensitive to modularity, the vendor will offer a product that is more modular and will be able to charge a higher price. It will lead to higher demand and higher profit for the vendor.*

Now, as in Case 1, we consider the impact of the environment parameters considered in the model. As Propositions B2 below is similar to its Case A counterpart, we are only providing the Proposition here for brevity with the proof provided in the "Appendix 2" for completeness.

Proposition B2 *As the cost of introducing modularity in the product increases, the vendor will lower the optimal modularity level, optimal performance level and the optimal price resulting in lower demand and profit for the vendor.*

Proposition B3 *As the marginal cost Z increases the optimal price*:

- *increases if* $\beta > (\gamma - Y\delta)\text{^}2/2(C + D\ Y\text{^}2)$
- *decreases if* $\beta < (\gamma - Y\delta)\text{^}2/2(C + D\ Y\text{^}2)$
- *stays unchanged if* $\beta = (\gamma - Y\delta)\text{^}2/2(C + D\ Y\text{^}2)$

Similar to Case A, we reach a solution where the optimal response to an increase in marginal cost may include increasing the price, decreasing the price or keeping the price same; depending upon the price sensitivity and other factors.

Now we look at the proposition that is unique to Case B: maximizing profits when the relationship between modularity and architectural performance can be modified by the SaaS provider.

Proposition B4 *When modularity and architectural performance are related to each other, the producer's profit is maximum with respect to both X and Y when*

$$X = \frac{2(\alpha - Z\beta)(DY\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta))}{2(4CD\beta - D\gamma^2 - C\delta^2)}$$

In the above case, all the decision variables are equal to their values as in the case where modularity and performance are not related.

This proposition presents an interesting managerial implication that if the SaaS provider can freely change X and Y then the optimal values are same as in Case A. So it may not be necessary for the provider to consider Case B.

5 Discussion and conclusion

This research provides an economic basis for design decisions by the SaaS provider. Our model provides the optimal values for service modularity, architectural performance and service price for maximum provider profit. This is one of the first attempt to provide an analytical structure to the important role of the vendor in

cloud computing. We then used the model to provide proofs for several propositions that determine the impact of customer preferences and environment variables on design decisions made by the SaaS vendor.

Our results provide the optimal response by the SaaS providers regarding the level of modularity and architectural performance for the SaaS service in response to changes in cost parameters, user demand and user preferences. We showed that if the user preferences changed toward higher modularity or higher architectural performance, then the provider will respond by changing the designs to higher modularity or architectural performance respectively so as to continue to maximize profits. We further showed that an increase in the cost of providing modularity or architectural performance will lead the provider to reduce the level of modularity and architectural performance respectively. Next, we showed that as the marginal cost of providing the service increases, the SaaS provider can respond with different prices depending upon the price sensitivity of demand.

We considered two cases—first, when modularity and architectural performance are not related and second when they are related. Our most interesting result was that we found a condition where both cases lead to the same optimal values of design decisions. The optimal modularity and architectural performance levels are same in both cases if the vendor is able to adjust the parameters X and Y that control the relationship between modularity and architectural performance. So, a vendor can ignore the more complex Case-B and proceed with designing the service based on the simpler Case-A to reach the optimal solution. Only if it is not possible to attain the optimal solution of Case-A for some reason, then the vendor can attempt to reach the closest to best solution by calculating the optimal X and Y from Proposition B4 and designing the system to be as close to the optimal values as possible.

We believe this research makes substantial contributions to both research and practice. In the remaining section we provide details of our contributions to research and practice, present key limitations of the paper and directions for future research.

5.1 Research implications

This work makes several contributions to the research on Cloud Computing in general, and specifically on development of SaaS services. In contrast to traditional analytical model of software development that assumes the marginal cost of software to be negligible, we have specifically taken into account the "service" nature of SaaS by explicitly accounting for maintenance cost and marginal cost of providing the service for its entire life. Further, while much of the software development literature focuses on the functional attributes of the software being designed, we have focused on two non-functional attributes—modularity and architectural performance, that we believe are important characteristics to study. Through our model, we provide one of the first robust frameworks to study design decision making by SaaS providers in the context of changing costs, user demand and user preferences.

Our research further contributes by explicitly considering the cases when modularity and architectural performance are related and showing that for certain

conditions, the optimal decision parameters are the same as when the two attributes are not related. We borrow from existing service science literature, software engineering literature as well as marketing and operations literature to develop a robust model of SaaS provider's design decision making. Our model can work as the foundation to develop extensions that can take into account more real life contexts like vendor competition, complimentary products and user preference distributions to further develop the extant research on cloud computing.

5.2 Practice implications

Our results have significant implications for managers involved in design of SaaS services. To the SaaS vendor, our work provide an economic basis for making design decisions; specifically our model provide the optimal values of two non-functional design attributes—modularity and architectural performance; for maximizing provider profits. We further show the optimal response by SaaS providers to changes in user preferences and cost of introducing the modularity and architectural performance.

We also consider the more realistic case when modularity and performance are related and show how the optimal values of parameters change when we consider the situation when the two design attributes are related to each other. We show that for certain values of the parameters relating modularity and architectural performance the optimal values of design parameters are same as the case when modularity and architectural performance are not related to each other.

Our model serves as a foundation for building extended models that take more real life conditions such as multiple providers with multiple services into account, and that take operational decisions as well as design decisions into account. We expect that such extended models will be able to provide further managerial insights into decision making by SaaS providers.

5.3 Limitations and future research

We have focused on one aspect of decision making by the SaaS vendor: the design decisions related to modularity and architectural performance. While this is a good starting point to build the analytical structure to understand the role of vendor in SaaS ecosystem, various other decisions by vendors as well as decisions by customers, infrastructure providers and other stakeholders provide opportunities for extension of this research.

Further, our model limits itself to design considerations only and does not consider operational decisions by the SaaS vendor. We specifically make a distinction between architectural performance and operational performance, and focus on architectural performance. This research can be extended by explicitly accounting for operational factors, specifically operational performance.

Our model also assumes the context of one vendor and one product. The research can be extended to the context of multiple vendors with multiple products. A multiple vendor context will allow for explicit modeling of competition between vendors and customers choosing between competing options based on their

preferences. A multiple product scenarios will allow for extension of this research where design decisions for multiple products interact with each other to produce a more dynamic decision making model. For example—a SaaS vendor can provide a suite of services that are complimentary in nature from design point of view. Such a product offering will lead to very different decision parameters for design decisions than the ones calculated through our model.

Our model has employed several simplifications for achieving analytical tractability. For example—we employ a simplified pricing structure where we only consider the total amortized price of the SaaS service over the lifetime of the service. This abstracts away pricing innovations used by the SaaS industry such as fixed pricing, pay only for usage with no fixed fee, volume discounts etc.

Lastly, we used linear demand functions. Problems resulting from oversimplification of complex situations into a simpler functional form are well documented (Montgomery and Bradlow 1999). However, in the context of the simplified scenario considered in this paper with a single seller, one product and uniform customer preferences, we consider the linear form to be an adequate starting point. The model can be extended by using non-linear functions to model various parameters in future research. Further, we have assumed a static customer population with uniform characteristics such as preference for modularity and preference for performance. Our analysis can be extended by modeling a dynamic customer population that consists of individual customers distributed over a range of values for customer characteristics.

5.4 Conclusion

This research examines design decision making by SaaS providers. We considered two design decisions to be made by the SaaS provider: the optimal level of two non-functional attributes of the SaaS service: modularity of the software architecture and the architectural performance of the software. We modeled the relationship of these two attributes with factors such as user preferences, user demand, and the price of the service. In a major departure from traditional models of IS product development, we considered marginal cost and maintenance cost of providing SaaS service to recognize that the SaaS service has characteristics of both a product and a service.

We consider two cases for the relationship between the two design attributes considered in the study—modularity and architectural performance. Our results show the optimal values of design attributes that maximize vendor profit by taking into account relevant factors such as user preferences, user demand and service price. Our research provides one of the first analytical models of optimal decision making by SaaS providers. We show that SaaS providers adjust the service design appropriately in response to changes in user preferences, associated costs and other relevant factors.

The model developed in this paper is a good start that successfully models how SaaS providers can design the offered service in an optimal manner to maximize profits. We look forward to future research further developing the model for analyzing more real-life contexts such as multiple providers with multiple products to gain insights into how SaaS providers can achieve optimal design for their services.

Appendix 1: Glossary of terms

d	Demand of the service amortized over the lifetime of the service
p	Price of the service amortized over the lifetime of the service
π	Profit amortized over the lifetime of the service
m	Modularity level of the service
s	Architectural performance level of the service
β	Decrease in demand d from increase in price p (sensitivity to price)
γ	Increase in demand from increase in modularity (sensitivity to modularity)
δ	Increase in demand from increase in architectural performance (sensitivity to architectural performance)
A_1	Fixed cost for developing the product arising from factors other than modularity and architectural performance
A_2	Amortized maintenance cost over the lifetime of the product
$A = A_1 + A_2$	Total fixed cost part that includes amortized maintenance cost for providing the service
B	Parameter related to modularity showing the saving in amortized maintenance cost arising from modular design; it is also amortized over the lifetime of the service
Z	Marginal cost of providing service per unit amortized over the lifetime of the service
C	Parameter related to modularity during design and development of the service
D	Parameter related to performance during design and development of the service
θ	Total cost amortized over the lifetime of the product. $\theta = c_1 + c_2 + c_3$
$c_1 = A_1 + Cm^2 + Ds^2$	Fixed Cost for developing and setting up service
$c_2 = A_2 - Bm$	Maintenance cost amortized over the lifetime of the product
$c_3 = Zd$	Marginal cost amortized over the lifetime of the product
X	Maximum attainable architectural performance level of a service when the service is not at all modular
Y	Ratio of change in architectural performance level s to change in modularity level m of the service

Appendix 2: Proofs of propositions

Proof of Proposition A1 Taking the partial derivative of optimal p^* with respect to γ, we obtain

$$\frac{\partial p^*}{\partial \gamma} = \frac{D(B(4CD\beta + D\gamma^2 - C\delta^2) + 4CD(\alpha - Z\beta))}{(4CD\beta - D\gamma^2 - C\delta^2)^2}$$

The denominator of the above equation is always positive as it is a square of an expression. From Lemma 1 and boundary condition of Eq. (7), we find $4D\beta - \delta^2 > 0$ and $(\alpha - Z\beta) > 0$ respectively. Hence the numerator is always positive.

Therefore: $\frac{\partial p^*}{\partial \gamma} > 0$. □

Proof of Proposition A2 Taking the partial derivative of optimal m^* with respect to γ we obtain

$$\frac{\partial m^*}{\partial \gamma} = \frac{D((4D\beta - \delta^2)(C(\alpha - Z\beta) + B\gamma) + D\gamma^2(\alpha - Z\beta))}{(4CD\beta - D\gamma^2 - C\delta^2)^2}$$

The denominator of the above equation is always positive as it is a square of an expression. From Lemma 1, since $4D\beta - \delta^2 > \frac{D\gamma^2}{C}$, the numerator can be rewritten to

$$D\left(\left(\frac{D\gamma^2}{C}\right)(C(\alpha - Z\beta) + B\gamma) + D\gamma^2(\alpha - Z\beta)\right).$$

From Lemma 1 we find $(\alpha - Z\beta)$ is positive; hence the numerator is always positive.

Therefore, $\frac{\partial m^*}{\partial \gamma} > 0$. □

Proof of Proposition A3 Taking the partial derivative of optimal p^* with respect to C we obtain

$$\frac{\partial p^*}{\partial C} = -\frac{D\gamma(4BD\beta + 2D\alpha\gamma - B\delta^2 - 2DZ\beta\gamma)}{(4CD\beta - D\gamma^2 - C\delta^2)^2}$$

The denominator of the above equation is always positive as it is a square of an expression. Next, we can rewrite the numerator to:

$$\begin{aligned} &= -D\gamma(4BD\beta - B\delta^2 + 2D\alpha\gamma - 2DZ\beta\gamma) \\ &= -D\gamma(B(4D\beta - \delta^2) + 2D\gamma(\alpha - \beta Z)) \end{aligned}$$

Using Lemma 1 and the boundary condition, it can be shown that the numerator will be negative. Therefore, $\frac{\partial p^*}{\partial C} < 0$. □

Proof of Proposition A4 By taking the partial derivative of optimal m^* with respect to C we obtain

$$\frac{\partial m^*}{\partial C} = -\frac{(4D\beta - \delta^2)(2D(\alpha - Z\beta)\gamma + B(4D\beta - \delta^2))}{2(4CD\beta - D\gamma^2 - C\delta^2)^2} = -\frac{(4D\beta - \delta^2)m^*}{(4CD\beta - D\gamma^2 - C\delta^2)}$$

Using Lemma 1, we observe both numerator $(4D\beta - \delta^2)m^*$ and denominator $(4CD\beta - D\gamma^2 - C\delta^2)$ are positive. Hence, $\frac{\partial m^*}{\partial C}$ is negative of a positive quantity.

Therefore, $\frac{\partial m^*}{\partial C} < 0$.

By taking the partial derivative of optimal s^* with respect to C, we obtain

$$\frac{\partial s^*}{\partial C} = -\frac{\gamma\delta(2D(\alpha - Z\beta)\gamma + B(4D\beta - \delta^2))}{2(4CD\beta - D\gamma^2 - C\delta^2)^2} = -\frac{\gamma\delta m^*}{(4CD\beta - D\gamma^2 - C\delta^2)}$$

Using Lemma 1, we observe that numerator $\gamma\delta m^*$ and denominator $(4CD\beta - D\gamma^2 - C\delta^2)$ are positive. Hence, $\frac{\partial s^*}{\partial C}$ is negative of a positive quantity.

Therefore, $\frac{\partial s^*}{\partial C} < 0$. □

Proof of Proposition A5 By taking the partial derivative of optimal p^* with respect to Z, we obtain

$$\frac{\partial p^*}{\partial Z} = \frac{2CD\beta - D\gamma^2 - C\delta^2}{4CD\beta - D\gamma^2 - C\delta^2}$$

From Lemma 1, we conclude that denominator is always positive. Depending on the numerator, $\frac{\partial p^*}{\partial Z}$ becomes positive, negative or zero.

$$\text{Numerator} = 2CD\left(\beta - \frac{\gamma^2}{2C} - \frac{\delta^2}{2D}\right)$$

- If $\beta > \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$, then numerator is positive.
- If $\beta < \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$, then numerator is negative.
- If $\beta = \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$, then numerator is zero.

Proof of Proposition B1 In order to examine the variation in the optimal decision variables with respect to γ we differentiate $p^*_{ms}, m^*_{ms}, d^*_{ms}$ and π^*_{ms} with respect to γ. We obtain

$$\frac{\partial p^*_{ms}}{\partial \gamma} = \frac{4C(2DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta)) + B(4C\beta + 4DY^2\beta + (\gamma - Y\delta)^2) + 2DY(4DXY^2\beta + (\gamma - Y\delta)(X\gamma + Y(2\alpha - 2Z\beta + X\delta)))}{(4\beta(C + DY^2) - (\gamma - Y\delta)^2)^2}$$

$$\frac{\partial m^*_{ms}}{\partial \gamma} = \frac{(4C(2DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta)) + B(4C\beta + 4DY^2\beta + (\gamma - Y\delta)^2) + 2DY(4DXY^2\beta + (\gamma - Y\delta)(X\gamma + Y(2\alpha - 2Z\beta + X\delta))))}{(4\beta(C + DY^2) - (\gamma - Y\delta)^2)^2}$$

$$\frac{\partial d^*_{ms}}{\partial \gamma} = \frac{\beta(2(\gamma - Y\delta)(2DY(Y\alpha - YZ\beta + X\gamma) + 2C(\alpha - Z\beta + X\delta) + B(\gamma - Y\delta)) + (B + 2DXY)(4C\beta + 4DY^2\beta - (\gamma - Y\delta)^2))}{(4\beta(C + DY^2) - (\gamma - Y\delta)^2)^2}$$

$$\frac{\partial \pi^*_{ms}}{\partial \gamma} = \frac{(2DY(Y\alpha - YZ\beta + X\gamma) + 2C(\alpha - Z\beta + X\delta) + B(\gamma - Y\delta)) * m^*_{ms}}{4\beta(C + DY^2) - (\gamma - Y\delta)^2}$$

The denominator is always positive as it is a square of an expression. By carefully examining the numerators in all the above four cases, we find that the only way the numerator could be negative in each case if $\alpha < Z\beta$ or $\gamma < Y\delta$ or both. According to our boundary conditions, we have $\alpha > Z\beta$ and $\gamma > Y\delta$. Hence, we conclude $\frac{\partial p^*_{ms}}{\partial \gamma} > 0, \frac{\partial m^*_{ms}}{\partial \gamma} > 0, \frac{\partial d^*_{ms}}{\partial \gamma} > 0$, and $\frac{\partial \pi^*_{ms}}{\partial \gamma} > 0$. □

Proof of Proposition B2 By taking the partial derivative of optimal modularity level m^* with respect to C (cost parameter we obtain

$$\frac{\partial m^*}{\partial C} = -\frac{(4D\beta - \delta^2)(2D(\alpha - Z\beta)\gamma + B(4D\beta - \delta^2))}{2(4CD\beta - D\gamma^2 - C\delta^2)^2} = -\frac{(4D\beta - \delta^2)m^*}{(4CD\beta - D\gamma^2 - C\delta^2)}$$

Using Lemma 1, we observe both numerator $(4D\beta - \delta^2)m^*$ and denominator $(4CD\beta - D\gamma^2 - C\delta^2)$ are positive. Hence, $\frac{\partial m^*}{\partial C}$ is negative of a positive quantity.

Therefore, $\frac{\partial m^*}{\partial C} < 0$.

By taking the partial derivative of optimal s^* with respect to C, we obtain

$$\frac{\partial s^*}{\partial C} = -\frac{\gamma\delta(2D(\alpha - Z\beta)\gamma + B(4D\beta - \delta^2))}{2(4CD\beta - D\gamma^2 - C\delta^2)^2} = -\frac{\gamma\delta m^*}{(4CD\beta - D\gamma^2 - C\delta^2)}$$

Using Lemma 1, we observe that numerator $\gamma\delta m^*$ and denominator $(4CD\beta - D\gamma^2 - C\delta^2)$ are positive. Hence, $\frac{\partial s^*}{\partial C}$ is negative of a positive quantity.

Therefore,

$$\frac{\partial s^*}{\partial C} < 0.$$

Taking the partial derivative of optimal p^* with respect to C we obtain

$$\frac{\partial p^*}{\partial C} = -\frac{D\gamma(4BD\beta + 2D\alpha\gamma - B\delta^2 - 2DZ\beta\gamma)}{(4CD\beta - D\gamma^2 - C\delta^2)^2}$$

The denominator of the above equation is always positive as it is a square of an expression. Next, we can rewrite the numerator to:

$$= -D\gamma(4BD\beta - B\delta^2 + 2D\alpha\gamma - 2DZ\beta\gamma)$$

$$= -D\gamma(B(4D\beta - \delta^2) + 2D\gamma(\alpha - \beta Z))$$

Using Lemma 1 and the boundary condition, it can be shown that the numerator will be negative. Therefore, $\frac{\partial p^*}{\partial C} < 0$. □

Proof of Proposition B3 By taking the partial derivative of optimal p^* with respect to Z, we obtain

$$\frac{\partial p^*}{\partial Z} = \frac{2C\beta + DY^2\beta - (\gamma - Y\delta)^2}{4C\beta + 4DY^2\beta - (\gamma - Y\delta)^2}$$

From Lemma 2, we conclude that denominator is always positive. Depending on the numerator, $\frac{\partial p^*}{\partial Z}$ becomes positive, negative or zero.

$$\text{Numerator} = 2C\beta + DY^2\beta - (\gamma - Y\delta)^2$$

If $\beta > \frac{(\gamma - Y\delta)^2}{2(C+DY^2)}$, then numerator is positive.

If $\beta < \frac{(\gamma - Y\delta)^2}{2(C+DY^2)}$, then numerator is negative.

If $\beta = \frac{(\gamma - Y\delta)^2}{2(C+DY^2)}$, then numerator is zero. □

Proof of Proposition B4 By taking the partial derivative of optimal π^*_{ms} with respect to X, we obtain

$$\frac{\partial(\pi^*_{ms})}{\partial X} = \frac{2(\alpha - Z\beta)(YD\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta)) + 2X(C\delta^2 + D\gamma^2 - 4CD\beta)}{4C\beta + 4DY^2\beta - (\gamma - Y\delta)^2}$$

By taking the partial derivative of optimal π^*_{ms} with respect to Y, we obtain

$$\frac{\partial(\pi^*_{ms})}{\partial Y} = -\frac{(2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta))(2(\alpha - Z\beta)(YD\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta)) + 2X(C\delta^2 + D\gamma^2 - 4CD\beta))}{(4C\beta + 4DY^2\beta - (\gamma - Y\delta)^2)^2}$$

We observe,

$$\frac{\partial(\pi^*_{ms})}{\partial Y} = -(m^*_{ms})\left(\frac{\partial(\pi^*_{ms})}{\partial X}\right)$$

We note, as m^*_{ms} is always positive, hence both $\frac{\partial(\pi^*_{ms})}{\partial Y}$ and $\frac{\partial(\pi^*_{ms})}{\partial X}$ always have opposite signs and if one is zero then the other is zero too.

We observe using Lemma 2, denominator of $\frac{\partial(\pi^*_{\text{ms}})}{\partial X}$ is always positive. However, the numerator could be positive, negative or zero depending on the relationship between X and other parameters.

We observe, when

$$X < \frac{2(\alpha - Z\beta)(DY\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta))}{2(4CD\beta - D\gamma^2 - C\delta^2)}, \quad \frac{\partial(\pi^*_{ms})}{\partial X} > 0$$

That means π^*_{ms} will increase as X increases.

However, when

$$X > \frac{2(\alpha - Z\beta)(DY\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta))}{2(4CD\beta - D\gamma^2 - C\delta^2)}, \quad \frac{\partial(\pi^*_{ms})}{\partial X} < 0$$

which means π^*_{ms} will decrease as X increases.

Hence, π^*_{ms} will be maximum with respect to X and Y, when

$$X = \frac{2(\alpha - Z\beta)(DY\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta))}{2(4CD\beta - D\gamma^2 - C\delta^2)}$$

We also find from Eqs. 20–24, for the above value of X, the optimal decision variables become

$$p^*_{ms} = \frac{2CD(\alpha - Z\beta) + BD\gamma}{4CD\beta - D\gamma^2 - C\delta^2} - Z = p^*$$

$$m^*_{ms} = \frac{4BD\beta + 2D\alpha\gamma - 2DZ\beta\gamma - B\delta^2}{2(4CD\beta - D\gamma^2 - C\delta^2)} = m^*$$

$$s^*_{ms} = \frac{(2C\alpha + B\gamma - 2CZ\beta)\delta}{2(4CD\beta - D\gamma^2 - C\delta^2)} = s^*$$

$$d^*_{ms} = \frac{D\beta(2C(\alpha - Z\beta) + B\gamma - 2C)}{(4CD\beta - D\gamma^2 - C\delta^2)} = d^*$$

$$\pi^*_{ms} = \frac{4D(\alpha - Z\beta)(B\gamma + C(\alpha - Z\beta)) + B^2(4D\beta - \delta^2)}{4(4CD\beta - D\gamma^2 - C\delta^2)} - A = \pi^* - A$$

□

References

Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I et al (2010) A view of cloud computing. Commun ACM 53(4):50–58

Baines TS, Lightfoot HW, Benedettini O, Kay JM (2009) The servitization of manufacturing: a review of literature and reflection on future challenges. J Manuf Technol Manag 20(5):547–567

Bakos Y, Brynjolfsson E (1999) Bundling information goods: pricing, profits, and efficiency. Manag Sci 45(12):1613–1630

Baldwin C, Clark K (1997) Managing in an age of modularity. Harvard Bus Rev 75(5):84

Baldwin CY, Clark KB (2000) Design rules: the power of modularity. The MIT Press, Cambridge

Balsamo S, Marzolla M (2005) Performance evaluation of UML software architectures with multiclass queueing network models. In: Proceedings of the 5th international workshop on software and performance

Balsamo S, Di Marco A, Inverardi P, Simeoni M (2004) Model-based performance prediction in software development: a survey. IEEE Trans Softw Eng 30(5):295–310

Banker RD, Datar SM, Kemerer CF, Zweig D (1993) Software complexity and maintenance costs. Commun ACM 36(11):81–94

Barua A, Kriebel CH, Mukhopadhyay T (1991) An economic analysis of strategic information technology investments. MIS Q 15:313–331

Benlian A, Koufaris M, Hess T (2011) Service quality in software-as-a-service: developing the SaaS-Qual measure and examining its role in usage continuance. J Manag Inf Syst 28(3):85–126

Bhardwaj S, Jain L, Jain S (2010) Cloud computing: a study of infrastructure as a service (IAAS). Int J Eng Inf Technol 2(1):60–63

Booch G (1993) Object-oriented analysis and design with applications. Addison-Wesley, Reading

Bosch J (2010) Toward compositional software product lines. IEEE Softw 27:29–34

Bush AA, Tiwana A, Rai A (2010) Complementarities between product design modularity and IT infrastructure flexibility in IT-enabled supply chains. IEEE Trans Eng Manag 57(2):240–254

Carroll TE, Grosu D (2010) Formation of virtual organizations in grids: a game-theoretic approach. Concurr Comput Pract Exp 22(14):1972–1989

Chidamber SR, Kemerer CF (1994) A metrics suite for object oriented design. IEEE Trans on Softw Eng 20(6):476–493

Choudhary V (2007a) Comparison of software quality under perpetual licensing and software as a service. J Manag Inf Syst 24(2):141–165

Choudhary V (2007b) Software as a service: implications for investment in software development. In: 40th annual Hawaii international conference on system sciences, 2007, HICSS 2007

Clark DD (1982) Modularity and efficiency in protocol implementation. MIT Laboratory for Computer Science, Computer Systems and Communications Group, Request for Comments 817

Da Cunha C, Agard B, Kusiak A (2007) Design for cost: module-based mass customization. IEEE Trans Autom Sci Eng 4(3):350–359

Demirkan H, Dolk D (2013) Analytical, computational and conceptual modeling in service science and systems. IseB 11(1):1–11

Demirkan H, Kauffman RJ, Vayghan JA, Fill H-G, Karagiannis D, Maglio PP (2009) Service-oriented technology and management: perspectives on research and practice for the coming decade. Electron Commer Res Appl 7(4):356–376

Dewan R, Jing B, Seidmann A (2003) Product customization and price competition on the internet. Manage Sci 49(8):1055–1070

Fitzsimmons JA, Fitzsimmons MJ (2004) Service management operations strategy information technology, 4th edn. McGraw Hill, Irwin

Gadrey J (2000) The characterization of goods and services: an alternative approach. Rev Income Wealth 46(3):369–387

Goiri I, Guitart J, Torres J (2010) Characterizing cloud federation for enhancing providers' profit. In: 2010 IEEE 3rd international conference on cloud computing (CLOUD)

Harmon R, Demirkan H, Hefley B, Auseklis N (2009) Pricing strategies for information technology services: a value-based approach. In: 42nd Hawaii international conference on system sciences, 2009, HICSS'09

Höfer C, Karagiannis G (2011) Cloud computing services: taxonomy and comparison. J Internet Serv Appl 2(2):81–94

Hosanagar K, Krishnan R, Chuang J, Choudhary V (2005) Pricing and resource allocation in caching services with multiple levels of quality of service. Manag Sci 51(12):1844–1859

Hsu CL, Lin JCC (2016) Factors affecting the adoption of cloud services in enterprises. IseB 14(4):791–822

Hyötylainen M, Moller K (2007) Service packaging: key to successful provisioning of ICT business solutions. J Serv Mark 21(5):304–312

IBM (n.d.) http://www.research.ibm.com/ssme/services.shtml. Retrieved 9 May 2011

IDC (2017) Worldwide public cloud services spending forecast to reach $122.5 billion in 2017, according to IDC. http://www.idc.com/getdoc.jsp?containerId=prUS42321417

Iosup R, Ostermann S, Yigitbasi N, Prodan R, Fahringer T, Epema D (2008) An early performance analysis of cloud computing services for scientific computing. Technical report, TU Delft, Dec 2008

Jain S, Kannan P (2002) Pricing of information products on online servers: issues, models, and analysis. Manag Sci 48(9):1123–1142
Janssen M, Joha A (2008) Emerging shared service organizations and the service-oriented enterprise: critical management issues. Strateg Outsour Int J 1(1):35–49
Jiang Y, Seidmann A (2014) Capacity planning and performance contracting for service facilities. Decis Support Syst 58:31–42
Kanaracus C (2013) SaaS in 2013: companies and trends to watch. http://www.computerworld.com/s/article/9235279/SaaS_in_2013_Companies_and_trends_to_watch
Krishnan V, Zhu W (2006) Designing a family of development-intensive products. Manag Sci 52(6):813–825
Kumar A (2004) Mass customization: metrics and modularity. Int J Flex Manuf Syst 16:287–311
Lau Antonio K, Yam R, Tang E (2007) The impacts of product modularity on competitive capabilities and performance: an empirical study. Int J Prod Econ 105(1):1–20
Ma D, Seidmann A (2008) The pricing strategy analysis for the software-as-a-service business model. Grid economics and business models. Springer, Berlin, pp 103–112
Marston S, Li Z, Bandyopadhyay S, Zhang J, Ghalsasi A (2011) Cloud computing: the business perspective. Decis Support Syst 51(1):176–189
Mell P, Grance T (2011) The NIST definition of cloud computing (draft). NIST Spec Publ 800:145
MerriamWebster (2012) A sample of new dictionary words for 2012. http://www.merriam-webster.com/info/newwords12.htm
Montgomery AL, Bradlow ET (1999) Why analyst overconfidence about the functional form of demand models can lead to overpricing. Mark Sci 18(4):569–583
Moorthy KS (1993) Theoretical modeling in marketing. J Mark 57:92–106
Niyato D, Vasilakos AV, Kun Z (2011) Resource and revenue sharing with coalition formation of cloud providers. Game theoretic approach. In: 2011 11th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid)
Oliva R, Kallenberg R (2003) Managing the transition from products to services. Int J Serv Ind Manag 14(2):160–172
Parnas DL (1972) On the criteria to be used in decomposing systems into modules. Commun ACM 15(12):1053–1058
Parnas DL, Clements PC, Weiss DM (1985) The modular structure of complex systems. IEEE Trans Softw Eng 11(3):259–266
Pekkarinen S, Ulkuniemi P (2008) Modularity in developing business services by platform approach. Int J Logist Manag 19(1):84–103
PWC (2010) The cloud you don't know: an engine for new business growth. http://www.pwc.com/us/en/technology-forecast/2010/issue4/features/business-growth-pg1.jhtml
Rai A, Sambamurthy V (2006) Editorial notes: the growth of interest in services management: opportunities for information systems scholars. Inf Syst Res 17(4):327–331
Rochwerger B, Breitgand D, Levy E, Galis A, Nagin K, Llorente IM, Montero R, Wolfsthal Y, Elmroth E, Caceres J et al (2009) The reservoir model and architecture for open federated cloud computing. IBM J Res Dev 53(4):1–11
Rohitratana J, Altmann J (2012) Impact of pricing schemes on a market for software-as-a-service and perpetual software. Future Gener Comput Syst 28(8):1328–1339
Rule MDVAD, Cai SY, Sethi K, Khatchadourian R, Greenwood P, Rashid A, Valetto G, Guerra EM, Silva JO, Silveira FF, Fernandes CT (2008) Measuring software design modularity. In: Contemporary modularization techniques (ACoM. 08), p 1
SalesForce.com (2014) Salesforce1 platform cloud computing—programmable user interface. http://www.salesforce.com/au/platform/cloud-platform/programmable-ui.jsp
Sanchez R, Mahoney JT (1996) Modularity, flexibility, and knowledge management in product and organization design. Strateg Manag J 17:63–76
Schilling MA (2000) Toward a general modular systems theory and its application to interfirm product modularity. Acad Manag Rev 25(2):312–334
Spohrer J, Maglio PP (2008) The emergence of service science: toward systematic service innovations to accelerate co-creation of value. Prod Oper Manag 17(3):238–246
Ståhl D, Mårtensson T, Bosch J (2017) The continuity of continuous integration: correlations and consequences. J Syst Softw 127:150–167
Susarla A, Barua A, Whinston AB (2010) Multitask agency, modular architecture, and task disaggregation in SaaS. J Manag Inf Syst 26(4):87–118

Toosi AN, Calheiros RN, Thulasiram RK, Buyya R (2011) Resource provisioning policies to increase IaaS provider's profit in a federated cloud environment. In: 2011 IEEE 13th international conference on high performance computing and communications (HPCC)
Ulrich K (1995) The role of product architecture in the manufacturing firm. Res Policy 24(3):419–440
Vandermerwe S, Rada J (1989) Servitization of business: adding value by adding services. Eur Manag J 6(4):314–324
Vaquero LM, Rodero-Merino L, Caceres J, Lindner M (2008) A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput Commun Rev 39(1):50–55
Vitharana P, Jain H, Zahedi F (2004) Strategy-based design of reusable business components. IEEE Trans Syst Man Cybern Part C Appl Rev 34(4):460–474
Voss CA, Hsuan J (2009) Service architecture and modularity. Decis Sci 40(3):541–569
Welch JJ, Waxman D (2003) Modularity and the cost of complexity. Evolution 57(8):1723–1734
Williams LG, Smith CU (1998) Performance evaluation of software architectures. In: Proceedings of the 1st international workshop on software and performance
Williams LG, Smith CU (2002) PASA SM: a method for the performance assessment of software architectures. In: Proceedings of the 3rd international workshop on software and performance
Winston WL (1993) Operations research: applications and algorithms. Duxbury Press, Belmont
Yourdon E, Constantine LL (1979) Structured design: fundamentals of a discipline of computer program and systems design. Prentice-Hall, Englewood Cliffs
Zhang J, Seidmann A (2010) Perpetual versus subscription licensing under quality uncertainty and network externality effects. J Manag Inf Syst 27(1):39–68

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com