An Adaptability-Driven Model and Tool for Analysis of Service Profitability

Ouh Eng Lieh[1(✉)] and Stan Jarzabek[2]

[1] Institute of Systems Science, National University of Singapore,
25, Heng Mui Keng Terrace, Singapore 119615, Singapore
englieh@nus.edu.sg
[2] Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland
s.jarzabek@pb.edu.pl

Abstract. Profitability of adopting Software-as-a-Service (SaaS) solutions for existing applications is currently analyzed mostly in informal way. Informal analysis is unreliable because of the many conflicting factors that affect costs and benefits of offering applications on the cloud. We propose a quantitative economic model for evaluating profitability of migrating to SaaS that enables potential service providers to evaluate costs and benefits of various migration strategies and choices of target service architectures. In previous work, we presented a rudimentary conceptual SaaS economic model enumerating factors that have to do with service profitability, and defining qualitative relations among them. A quantitative economic model presented in this paper extends the conceptual model with equations that quantify these relations, enabling more precise reasoning about profitability of various SaaS implementation strategies, helping potential service providers to select the most suitable strategy for their business situation.

Keywords: Service provider · Service profitability · Service architecture · Service variability · Service engineering

1 Introduction

Cloud computing paradigm promises service providers to reach large customer base, selling software at cheaper price (which benefits customers). Still, cloud computing, and Software-as-a-Service (SaaS) in particular is not for all businesses, nor for all software applications. The question whether or not a service solution will be profitable is not easy to answer. The following sample illustrates why informal analysis of SaaS profitability is difficult:

Service profitability depends on the cost of engineering a service for a given customer base, on service provisioning cost, and on the revenue gained from selling the service to that customer base. The customer base depends on the level of service adaptability, i.e., on Service Provider's ability to vary service requirements to meet requirements of various customers. The cost of engineering a service for adaptability depends on selected service architecture. A service architecture that minimizes provisioning cost at the same

S. Nurcan et al. (Eds.): CAiSE 2016, LNCS 9694, pp. 393–408, 2016.
DOI: 10.1007/978-3-319-39696-5_24

time limits service adaptability, which in turn decreases the customer base and profit. The cost of engineering a service further depends on the required level of service adaptability, and whether we build service from scratch or by modernizing the legacy code. In the latter case, the cost depends on availability of modernization methods that aid and automate the migration process.

SaaS profitability is determined by a complex web of inter-related and often conflicting forces that make manual analysis unreliable. Service profitability has been widely discussed primarily in the context of novel service business and pricing models, engineering and provisioning methods (covered in Sect. 6) however no formal treatment of the subject taking into account the many factors has been proposed. These existing works take into account only a small subset of these factors, giving a potentially incomplete analysis of service profitability especially without considering the trade-offs among these factors. In this paper, we fill this gap by proposing a quantitative economic model of service profitability.

In our previous works [1, 2], we set a rudimentary formal ground for analysis of service profitability. We identified factors that affect service profitability, and built a qualitative conceptual model formalizing dependencies among those factors. Our conceptual model shows how decisions regarding the choices of service architecture, dynamic (at service runtime) versus static (at the service construction-time) service adaptation techniques, or the size of the tenant base affect service profitability. The conceptual model built so far allows us to do qualitative analysis of service profitability. It helps a Service Provider to spot the decisions regarding service architecture and provisioning that are relevant in her situation, but it does not allow her to reason about the impact of those decisions on profitability in any rigorous way.

The contribution of this paper is a quantitative economic model of service profitability that we built on top of the conceptual model. This extended model assigns weights to various decisions and captures the impact of those decisions as formulas, in quantitative way. In particular, our economic model allows Service Providers to reason about profitability under various assumptions and answer questions such as: "To what degree does the selection of service architecture impact service profitability?"

Our economic model can help Service Providers to evaluate the impact of various decisions and SaaS strategies in a more systematic way than the conceptual model alone, but is difficult to use manually. Therefore, we also implemented a tool that automates model calculations and guides Service Provider in exploring migration strategies to SaaS and their expected profits. Section 2 documents our proposed Service Profitability Model. We introduce a process method and tool to evaluate for service profitability in Sects. 3 and 4. Experiments and analysis are in Sect. 5, followed by related work in Sect. 6. Conclusions and future works are in Sect. 7.

2 Service Profitability Model

The concept map shown in Fig. 1 summarizes our model concepts. In our earlier paper, we described how this concept map can be applied to address key service profitability questions that are of interest to Service Providers. In this paper, we extends the

conceptual model with definitions and equations that quantify the relationships among the factors in the concept map.

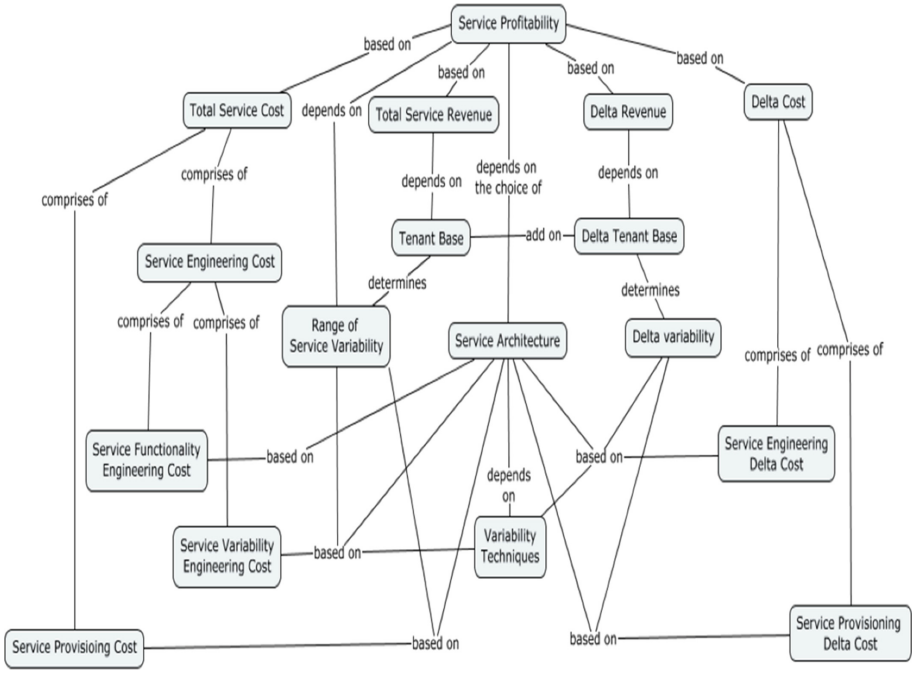

Fig. 1. Concept map of the service profitability

2.1 Preliminary Concepts and Definitions

Definition 1 (Tenant Base). The initial Tenant Base (*TB*) comprises of the initial tenants of a given service. Delta Tenant Base (*DTB*) comprises of the tenants a Service Provider expects to on-board in a specific time frame in the future.
Service Provider needs to address two sets of tenants' requirements for a given service: the initial tenant base (*TB*) and the delta tenant base (*DTB*). A Service Provider may already have the need to provide the service to a set of initial tenants. On the other hand during service period, new tenants may request to use the service and these tenants form the delta tenant base.

Definition 2 (Range of Service Variability). The Range of Service Variability (*RSV*) is the extent to which a Service can be adapted to varying service requirements of different tenants.

The bigger range the service can accommodate, more tenants can be on-boarded and higher revenue for the Service Provider. To on-board a tenant, the Service must be able to meet the requirements of that tenant. As *RSV* reflects the Service Provider's ability to customize the Service, *RSV* determines the *TB* that can be supported. On the other

hand for business strategic reasons, the Service Provider may have some target tenants in mind and engineer the *RSV* to meet the varying requirements of the target *TB*. Service Provider's dream is always to engineer a service where *RSV* fulfills the varying requirements of the *TB*, maximizing service profits.

Definition 3 (Service Engineering Costs). Service Engineering Costs (*SEC*) are incurred to engineer the functionality of the Service and to engineer the Service to support a given *RSV* on a given Service Architecture (*SA*). We termed this as Service Functionality Engineering Cost (*SFEC*) and Service Variability Engineering Cost (*SVEC*) respectively, collectively termed as the Service Engineering Costs (*SEC*).

Service Variability Engineering Cost (*SVEC*) is the cost of engineering Service based on the selected Variability Techniques (*VT*) to support a given *RSV* on a given *SA*, defined as a function of < *SA*, *RSV*, *VT*>. The relevant variability techniques to support a given *RSV* are discussed in [1]. Service Functionality Engineering Cost (*SFEC*) is the cost of engineering Service functionality to support a given *RSV* on a given *SA*. Service Engineering Cost (*SEC*) is the cost of engineering the Service for a given *SA* and *RSV* defined as a function of < *SA*, *RSV*, *VT*>.

$$SEC < SA, RSV, VT > = SFEC < SA > + SVEC < SA, RSV, VT >$$

Definition 4 (Service Provisioning Costs). Service Provisioning Cost (*SPC*) is the cost to provide hardware and infrastructure resources to provision a Service to support a given *RSV* on a given *SA*. *SPC* is defined as a function of < *SA*, *RSV*>. *SPC* can be incurred upfront independent of number of tenants termed as $SPC_{Upfront}$ or during operation based on the number of tenants termed as SPC_{Op}.

Definition 5 (Total Service Costs). Total Service Costs (*TSC*) is the total service costs of engineering (*SEC*) and provisioning (*SPC*) the Service on a given service *SA* to a given *TB*.

$$TSC < SA, RSV, VT > = SEC < SA, RSV, VT > + SPC < SA, RSV >$$

Definition 6 (Total Service Revenue). Total Service Revenue (*TSR*) is the total revenue from selling the Service on a given service *SA* to a given *TB*.

Definition 7 (Delta Variability). Delta Variability (*DV*) is the changes to existing Service requirements required to on-board a given *DTB*. The degree of engineering changes due to *DV* is termed as $DV_{engineering}$ and the degree of provisioning changes due to *DV* is termed as $DV_{provisioning}$.

Definition 8 (Delta Cost). Delta Cost (*DC*) is the engineering and provisioning costs to implement *DV* for a given *DTB* on a given *SA*.

Service Engineering Delta Cost (*SEDC*) is the cost to engineer the *DV* of a given *DTB* for Service on a given *SA* defined as a function of < *SA*, *DV*, *VT* >. Service Provisioning Delta Cost (*SPDC*) is the cost to provide hardware and infrastructure resources to support the *DV* of a given *DTB* on a given *SA*, defined as a function of < *SA*, *DV*>.

DC comprises of Service Engineering Delta Cost (*SEDC*) and Service Provisioning Delta Cost (*SPDC*) and is defined as a function of $< SA, DV, VT >$.

$$DC < SA, DV, VT > = SEDC < SA, DV, VT > + SPDC < SA, DV >$$

Definition 9 (Delta Revenue). Delta Revenue (*DR*) is the revenue from selling a Service to a given *DTB*.

Definition 10 (Service Profits). Service Profits (*SP*) is the profits from selling a Service to a given *TB* and *DTB* defined as a function of $< TSC, TSR, DC, DR >$.

2.2 Service Profitability

Service Profits (*SP*) are the monetary benefits from selling the service, taking into account the cost and revenue of both the tenant base and the delta tenant base: Providing services for multi-tenants is typically a multi-year project for the Service Providers. As such, we take into account the present value of money over time for economic analysis and use the net present value, expanding on existing literature studies [3, 4] for the analysis of service profitability. The net present value takes into account the engineering and provisioning costs and the revenue gained for both the tenant and delta tenants over an investment cycle. Given an investment cycle Y referring to the expected duration of the investment measured in number of years and discount rate d referring to the time value of money, typically range between 0.1 and 0.2, *SP* can be measured as given by:

$$SP = -(SEC + SPC_{Upfront}) + \sum_{y \in Y}^{Y} \frac{TSR_{(y)} - SPC_{Op(y)} + DR_{(y)} - DC_{(y)}}{(1+d)^y} \tag{1}$$

Intuitively, the overall profits are based on the service costs incurred (fixed costs) regardless of the number of on-boarded tenants and the yearly profits (taking into account the value of money over time) with the tenant and delta tenant base. The expected return on investment (*ROI*) takes into account service profits over the total service and delta costs for a specified number of investment years and can be measured by:

$$ROI = \frac{SP}{\left(SEC + SPC_{Upfront}\right)} \tag{2}$$

Dividing the above *ROI* value by the number of investment years would yield the annualized *ROI*.

3 Analysis of Service Profitability

In this section, we introduce a process to analyze service profitability. The steps of the process are primarily composed of tenant management, service cost management, service delta cost management and service profitability shown in Fig. 2. In each step, the Service Provider enters estimations to characterize her situation.

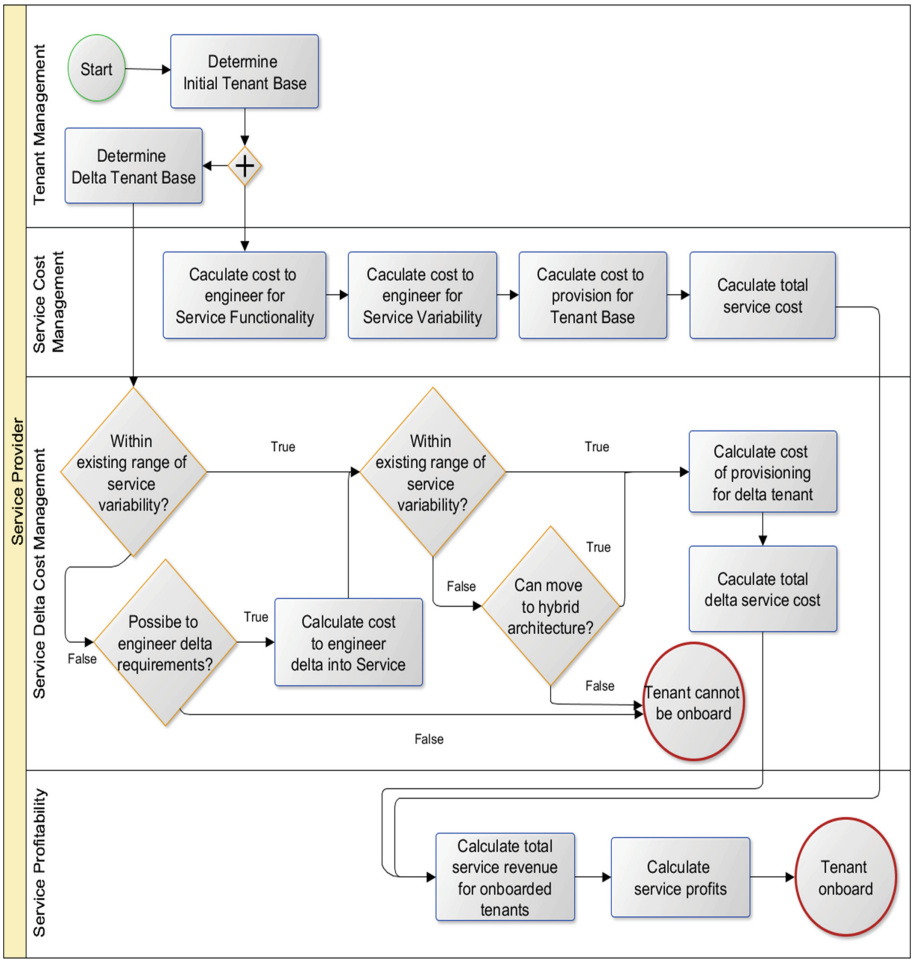

Fig. 2. Process to analyze service profitability

3.1 Tenant Management

This process step allows the Service Provider to specify or make their assumptions on the initial and delta set of tenants' requirements. These requirements can be in terms of the number of tenants, number of users per tenant and their requirements. The Service Provider should be able to specify or assume (e.g. based on variability distributions) the tenant's requirements. These inputs and assumptions is essential for the subsequent computations of the costs, revenue and profits. The Service Provider can modify these inputs to model different possible set of initial and delta tenants.

3.2 Service Cost Management

In our previous study [2], we introduced five types of service architectures. Fully-Shared (SA_{FS}), Partially-Shared (SA_{PS}), Non-Shared (SA_{NS}) and Hybrids (SA_{FSPS} and SA_{FSPSNS}). To manage the variability of tenant's requirements for SA_{FS}, we can design using service oriented architecture (SOA) technique for dynamic binding of the service components.

For SA_{NS}, we can design using product-line variability management technique [5] for static binding of the service components. Similar techniques can be used to manage variability for SA_{PS}, SA_{FSPS} and SA_{FSPSNS}. For analysis of service profitability, Service Provider estimates the costs to engineer both the service functionality and service variability of a given service for the initial tenant base. In order to engineer the service for varying requirements of the tenants, the Service Provider needs to adopt variability techniques of which the costs can differ substantially. These service costs can also differ if the Service Provider decides to migrate from existing application or develop the service from scratch. Cost estimation for both service engineering and service variability engineering can be calculated using existing effort cost estimation models such as COCOMO II. Based on a given service architecture, Service Provider also has to incur cost to provision the service for the tenants. These provisioning costs can differ if the Service Provider decides to lease from external cloud providers, hosting as private cloud or adopt hybrid cloud. Cost estimation for the service provisioning on the cloud can be calculated by using available data from existing cloud providers (e.g. Amazon EC2).

3.3 Service Delta Cost Management

Besides calculating the cost for the initial tenant base, it is also important for the Service Provider to analysis the potential delta tenant base for a more complete service profitability analysis. For each of the delta tenant, a Service Provider need to estimate the effort and cost to engineer the service and service variability for the varying requirements of the tenant. Based on the range of service variability supported by a given service architecture, the Service Provider may need to decide whether to incur additional engineering to address the delta variability as required by the new tenant. The Service Provider may choose to incur the additional cost to engineer the requirements or choose not to on-board the new tenant (e.g. due to shortage of resources).

The Service Provider also has to provision the service for the delta tenant based on the given service architecture. If the given service architecture cannot support the delta tenant's provisioning requirements (e.g. high isolation requirements on a given fully-shared service architecture), the Service Provider can consider to move to hybrid service architecture or choose not to on-board the new tenant. Cost estimation for the delta service engineering and service provisioning can be calculated using the same models as in service cost management.

3.4 Service Profitability

The services costs, service delta costs and revenue gained from on-boarded tenant collectively determined the service profits. Service revenue from the tenant can vary

based on the type of pricing models such as pay-per-use, subscription-based. Together with service costs, service delta costs and revenue, service profits and *ROI* can be calculated over a pre-determined service period. The *ROI* can be further annualized based on the number of investment years. The Service Provider can evaluate the outcomes of the service profitability based on different inputs and decisions made.

4 Tooling for Service Profitability Analysis

It is a Service Provider's challenge to design a service for profitability considering multiple factors affecting costs and gains. When designing and provisioning a service, Service Provider may also have various goals for profitability analysis or constraints regarding tenant base or service costs. We implemented a Service Profitability Analyzer (SPA) tool to allow Service Provider to analyze service profitability under various strategies for service design and deployment. SPA implements the concepts and equations defined in the Profitability Model to carry out the analysis under required assumptions. The user interface of SPA is shown in Fig. 3.

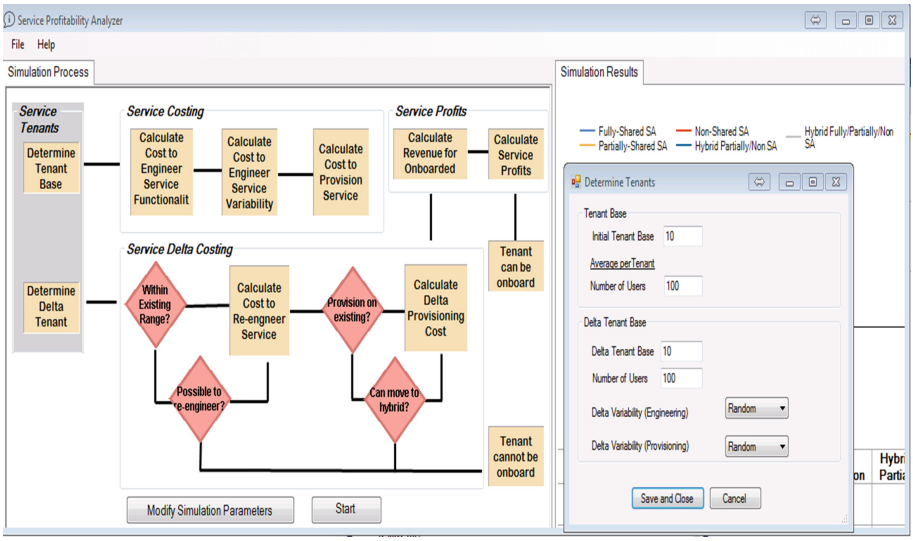

Fig. 3. Service profitability analyzer

A typical scenario is that Service Provider wants to know which service architecture to adopt to maximize profits. To analyze that, Service Provider inputs Service Costing values for the service to SPA. Based on COCOMO II, Service Provider can compute the service functionality and service variability engineering costs. The effort estimation of COCOMO II is based on the following formula where EAF is the effort adjustment Factor derived from the software cost drivers E is an exponent derived from the five software scale drivers.

$$Effort\ Estimation = 2.94 * EAF * (KSLOC)^{E} \quad (3)$$

One model to calculate provisioning costs is based on Amazon Web Services (AWS). The tool allows for cost inputs for the key web services provided by Amazon; Amazon Elastic Compute Cloud (EC2), Simple Storage Service (S3), Relational Database Service (RDS), DynamoDB, Route 53 and CloudFront. Amazon EC2 provides resizable compute capacity while Amazon S3 provides the fully redundant data storage infrastructure for storing and retrieving any amount of data. Amazon RDS makes it easy to set up, operate, and scale a relational database (e.g. MySQL, Oracle, SQL Server) in the cloud while Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. In terms of networking and content delivery, Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service and Amazon CloudFront provides an easy way to distribute content to end users with low latency and high data transfer speeds. The calculations of the Amazon services are based on the publicly accessible pricing calculator [6].

To capture the demand parameters of the service, the tool enables Service Provider to input the service tenant's requirements in terms of the number of users and tenants for the initial tenant base. In addition, Service Provider can also provide the expected number of delta tenants and the estimated variability ranges of these delta tenants for both the engineering and provisioning requirements as shown in Fig. 3. The possible delta variability values are low, high and random. A low delta variability value indicates a high proportion of the delta tenants require minimum changes required to existing service and a high delta variability value indicates a high proportion of the delta tenants require substantial changes to existing service. A random delta variability value allows the tool to simulate based on a random distribution of the tenants' delta variability.

With these demand inputs, the tool simulates the number of tenants of a given delta variability values for a specified number of investment years. Each tenant along with their delta variability (*DV*) values are used to calculate the delta engineering (*SEDC*) and provisioning cost (*SPDC*) for that tenant as follows.

$$SEDC = DV_{engineering} * SEC * \alpha_{SA,VT} \quad (4)$$

$$SPDC = DV_{provisioning} * SPC * \beta_{SA} \quad (5)$$

α and β are weighted cost coefficients, denoting the impact to the engineering and provisioning costs with a given service architecture and applied variability technique. The higher the value of these cost coefficients, the higher the cost incurred. With the Eqs. 1–5, the overall profits and *ROI* can be calculated.

We implemented SPA using Visual Basic .Net to compute the equations defined in the Service Profitability Model and allow for dynamic user interactivity. The design of the tool is modular and allow Service Provider to plug-in her own model to calculate estimates of the engineering and provisioning costs. These plug-ins are implemented as dynamic link libraries (DLLs) using .Net. The tool uses .Net reflection to dynamically load the DLLs during runtime. A Service Provider may have more cost factors. For

example, branding or marketing costs which impact service profits. The SPA tool can be further adapted with plugins to include such factors.

5 Experiments and Analysis

5.1 Experiments Overview

We conducted experiments using the SPA tool to analyze service profitability of adopting five service architectures for Apache OfBiz (OfBiz) [7]. OfBiz is an open source Java Enterprise Edition (J2EE) package used for enterprise resource planning. In our study we focus on four OfBiz existing services (eCommerceStoreService, OrderService, CatalogService and PartyService) to be migrated to support multiple tenants. These services comprises of 136 Java classes with close to 48 K lines of code, 61 Groovy files, 127 Freemarker templates and 24 XML Widgets.

To capture the demand parameters of the service, we set the 8 initial tenant base and 30 delta tenants over a 5 year investment period. For the service costs calculation of each service architecture, we first capture the engineering provisioning costs for SA_{FS} and then extrapolate for the rest of the architectures. We estimate the engineering costs by measuring the five major components; external inputs, external outputs, external inquiries, internal logical files and external interface files to obtain 394 unadjusted function points. We use these unadjusted function points value to calculate the effort of engineering the functionality of the service (*SFEC*) based on the COCOMO II model. The COCOMO parameters "required software reliability", "architecture/risk resolution" and "platform" are set to high and the "parameter developed for reusability" is set to very high, the rest of the parameters are set to nominal. We extrapolate service variability engineering costs (*SVEC*) for SA_{PS} by 20 % more than $SA_{FS,}$ based on an earlier study by Poulin and Himler [8] showing that building components for an SOA requires an approximate of 20 % additional cost over development for one-time use. For SA_{NS} using static binding variability technique based on our experiences with the adaptive reuse technique and the study of cost estimation in Software Product Lines [9], we assumed an additional 30 % more in *SVEC* than SA_{FS} to account for more extensive reuse effort to adopt product line techniques and development the product line assets. For SA_{PS}, SA_{PSNS} and SA_{FSPSNS}, we estimated another additional 10 % over SA_{NS} in engineering cost to manage variability using both SOA and product-line techniques.

For service engineering delta costs α, α_{NS} is lower than α_{FS} to account for greater flexibility in static binding [9] used in SA_{NS} over dynamic binding used in SA_{FS}. α_{PS} of SA_{PS} is higher than α_{NS} or α_{FS} based on the reasoning that additional effort is required to maintain both static and dynamic bindings. The values of α_{PS+NS}, $\alpha_{FS+PS+NS}$ are also set higher to account for the effort to maintain the hybrid service architectures. Based on the above reasoning, we make the assumption of $\alpha_{NS} < \alpha_{FS} < \alpha_{PS} < \alpha_{PSNS}$, α_{FSPSNS}. The weighted cost coefficient α values used in our experiments are as follows. These α values are configurable on the tool and sensitivity of these values are also discussed in Sect. 5.3.

$$\alpha_{NS}(SA_{NS}) = 0.2, \alpha_{FS}(SA_{FS}) = 0.3, \alpha_{PS}(SA_{PS}) = 0.4$$

$$\alpha_{PSNS}(SA_{PSNS}) = 0.5, \alpha_{FSPSNS}(SA_{FSPSNS}) = 0.5$$

For service provisioning cost (*SPC*), we based our estimates on Amazon AWS pricing calculator [6], using the recommended deployment architecture of a large web application (all instances on demand). The estimated costs to provision for the initial tenant base of SA_{FS} is based on the configuration of 10 m1.medium, 10 m1.large and 10 m1.large Amazon EC2 instances for web, application and database Servers respectively, shared by all tenants. For $SA_{NS,}$ SA_{PS}, SA_{PSNS}, dedicated instances of 1 m1.medium, 1 m1.large and 1 m1.large Amazon EC2 instances for web, application and database Servers respectively are allocated to each of the 10 initial tenants. For $SA_{FSPSNS,}$ we start with the same provisioning configuration as SA_{FS}.

For service provisioning delta costs β, β_{FS} is set lower to account for the sharing of resources while β_{NS} is set to 1 to denote the additional isolated resources required. β_{PS} is set between β_{FS} and β_{NS} to denote partial sharing of resources. The cost coefficient $\beta_{PSNS,}$ β_{FSPSNS} are set to either 0.2, 0.5 or 1 depending on the adopted service architecture for that tenant. To capture the service revenue earned by the Service Provider, we use the subscription-based strategy. Each tenant pays a price depending on her requirements. Based on the above reasoning, we assumed that $\beta_{FS} < \beta_{PS} < \beta_{NS.}$ The weighted cost coefficient β values used in our experiments are as follows. These β values are configurable on the tool and sensitivity of these values are also discussed in Sect. 5.3.

$$\beta_{FS}(SA_{FS}) = 0.2, \beta_{PS}(SA_{PS}) = 0.5, \beta_{NS}(SA_{NS}) = 1$$

$$\beta_{PSNS}(SA_{PSNS}) = 0.5 \text{ or } 1, \beta_{FSPSNS}(SA_{FSPSNS}) = 0.2,\ 0.5 \text{ or } 1$$

In summary, the assumptions made in the conduct of our experiments are (i) (*SVEC*) for SA_{PS} < (*SVEC*) for SA_{NS} < (*SVEC*) for (SA_{PS}, SA_{PSNS} and SA_{FSPSNS}) (ii) $\alpha_{NS} < \alpha_{FS} < \alpha_{PS} < \alpha_{PSNS}$, α_{FSPSNS} and (iii) $\alpha_{FS} < \beta_{PS} < \beta_{NS.}$ The actual values of *SVEC*, α and β are configurable on the tool user interface. In Sect. 5.3, we further evaluate the threat to validity due to the possible sensitivities of the actual values used for α, β and the extrapolation of *SVEC*.

In the initial case when fixed costs are incurred for engineering functionality and variability, assuming no upfront provisioning costs and no on-boarded tenants, the model returns the following values as shown in Fig. 4. These values are also manually calculated for verification using the functionality engineering cost and functionality variability engineering costs for each type of architecture. Note that the negative profits are the same for partially-shared and hybrids due to our same assumptions of calculating the service variability engineering cost (*SVEC*) for these architectures.

SA	Fully Shared	Partially Shared	Non Shared	Hybrid Partially/Non	Hybrid Fully/ Partially/Non
Service Profits	($257,998)	($283,798)	($279,498)	($283,798)	($283,798)

Fig. 4. Service profits (no tenant)

We further conduct three experiments with delta variability low, high and random. We executed each simulation 100 times recording the *ROI* and annualized *ROI* of the service profits for all the five service architectures.

5.2 Analysis

For the case of low delta variability as shown in Fig. 5, service profits of SA_{FS} ($1,112k) and annualized *ROI* of 86 % is the highest among the service architectures as expected. The maximum sharing of service components reduces the overall cost, while still being able to fulfill the tenant's low variation of requirements maximize the service profits.

Number of Simulations : 100 Number of Investment Years : 5
Delta Variability (Engineering) : Low Delta Variability (Provisioning) : Low

SA	Fully Shared	Partially Shared	Non Shared	Hybrid Partially/Non	Hybrid Fully/ Partially/Non
Service Profits	$1,112,070	$835,913	$858,450	$840,052	$926,574
Expected ROI	431%	295%	307%	296%	326%
Annualized ROI	86%	59%	61%	59%	65%

Fig. 5. Low delta variability

However in the case of high proportion of new tenants with high delta variability as shown in Fig. 6 adopting SA_{FS} becomes the unwise decision in terms of service profits and *ROI* as some tenants cannot be on-board without significant architecture changes, resulting in reduced revenue to cover the initial engineering costs incurred. Adopting SA_{NS} is the best decision in this case that maximizes the annualized *ROI* (61 %) and service profits ($858k). Service Provider needs to think carefully if the assumption of supporting tenants with high variability of requirements holds or not as the profitability is entirely opposite if not. This insight also highlights the importance of Service Provider to evaluate more factors (e.g. tenant's variability) for service profitability in addition to reducing cost by sharing resources.

Number of Simulations : 100 Number of Investment Years : 5
Delta Variability (Engineering) : High Delta Variability (Provisioning) : High

SA	Fully Shared	Partially Shared	Non Shared	Hybrid Partially/Non	Hybrid Fully/ Partially/Non
Service Profits	$301,832	$674,543	$858,561	$762,837	$781,171
Expected ROI	117%	238%	307%	269%	275%
Annualized ROI	23%	48%	61%	54%	55%

Fig. 6. High delta variability

SPA output also contains results of tenants of random delta variability as shown in Fig. 7. In this case, adopting either SA_{NS} with service profits ($857k) and annualized *ROI* (61 %) or the hybrids achieved good results as adopting these service architectures allow for more tenants to be on-boarded easily though there are higher initial engineering costs. Service Provider if unsure about the variability of tenant base should base their decision on this analysis instead. In this case, Service Provider is likely to achieve better *ROI* and service profits with using SA_{NS} or SA_{FSPSNS}. This also relates well with the fact that although managing variability incurs early high cost, the benefits is substantial over the long run.

Number of Simulations : 100 Number of Investment Years : 5
Delta Variability (Engineering) : Random Delta Variability (Provisioning) : Random

SA	Fully Shared	Partially Shared	Non Shared	Hybrid Partially/Non	Hybrid Fully/ Partially/Non
Service Profits	$645,602	$743,446	$857,870	$793,718	$842,720
Expected ROI	250%	262%	307%	280%	297%
Annualized ROI	50%	52%	61%	56%	59%

Fig. 7. Random delta variability

5.3 Threats to Validity

The sensitivity of the parameters potentially impacts the simulation results. We did further simulations varying *SVEC*, α and β values, while keeping to the assumptions of (i) (*SVEC*) for SA_{PS} < (*SVEC*) for SA_{NS} < (*SVEC*) for SA_{PS}, SA_{PSNS} and $SA_{FSPSNS,}$ (ii) $\alpha_{NS} < \alpha_{FS} < \alpha_{PS} < \alpha_{PSNS}$, α_{FSPSNS} and (iii) $\beta_{FS} < \beta_{PS} < \beta_{NS}$. For each assumption, we generate 100 sets of random values within the assumption constraints and conduct the

experiments separately again. We observe similar trends as per our analysis and observations with the original values of these parameters. However, we noted that the possible ranges of *SVEC*, α and β values are extensive and complete validation of these values is considered as part of our future validation work.

These experiments are based on service costs of one case study of an open source package and the use of existing cost estimation models such as COCOMO II also have its own level of confidence. To further validate the model and tool, we acknowledge the need for more case studies with actual values should be compared with the simulated values.

6 Related Works

Analysis of service profitability is an area that attracts much interest. Service profitability is one key economic benefit for Service Providers and can be analyzed from multiple perspectives.

One perspective of analyzing service profitability are the area of new/novel business and pricing models that maximize revenue. Ma [10] proposes an analytical SaaS business model which analyzes user's fit and exit costs that help Service Providers to increase SaaS competitive ability. Ma and Seidmann [11] propose a pricing strategy analysis for SaaS business model to study the competition between the SaaS and the traditional COTS (Commercial off-the shelf) software. Gabriella and Arto [12] analyzes the relationship between architectural practices different pricing models to maximize revenue. Xu and Li [13] analyzes service profitability in terms of dynamic pricing mechanisms and formulate revenue maximization problem with dynamic pricing as a stochastic dynamic program.

Another perspective of analyzing service profitability focuses on effective service adaptation of multiple tenants, placement of tenants and resource allocation to minimize costs and maximize service profitability. Bikram and Abhik [14] discuss engineering issues that can impact service profitability and propose for a more tenant-driven evolution of a SaaS where a vendor can accommodate changes to a SaaS to meet tenant needs, within reasonable limits. Ju [15] proposes a formal model as a bi-objective optimization problem that attempts to maximize vendor profit and tenant functional commonality. Mietzner et al. [16, 17] propose to adopt variability techniques to enable more flexible late binding of service variants, customization of BPEL process with variability descriptors and Morin et al. [18] propose to use aspect techniques to weave aspects for variability management. Kwok and Ajay [19] proposes a method for optimal placement of tenants and instances based on their proposed multi-tenant placement model without violating any SLA requirements of all tenants in a set of servers.

For this study, our analysis of service profitability takes into account existing works in service pricing, engineering and provisioning and other factors to provide a more complete perspective and tradeoffs in quantitative analysis of service profitability.

7 Conclusions and Future Works

Our proposed economic model of service profitability formalizes the interplay of multiple factors that influence service profitability. We augmented a conceptual model of service profitability with impact and effort formulas to conduct both qualitative and quantitative analysis of how multiple service implementation scenarios affect service profits. A tool interprets the model helping the Service Provider to explore a space of decisions that affect service profitability. Our economic model accounts for factors such as the tenant base, delta tenant base to be on-boarded in the future, tenant's variability, cost estimations models, service architecture and the use of variability techniques. Our model shows how these factors affect service cost, revenue, profits and *ROI*. We illustrated the usage of our profitability model and supporting tool with experiments conducted on an open source package. The evaluation results provide quantitative insight to the benefit of incurring initial cost to address variability for higher long-term profitability. We believe this work is useful to Service Providers to make more informed decision and help in building a business case that maximize service profitability.

In our future work, we will explore application of scientifically proven negotiating decision and optimization models to provide a formal ground for economic models of service profitability, suitable for evaluating in quantitative terms the impact of decisions involved in planning SaaS adoption strategies.

References

1. Ouh, E.L., Jarzabek, S.: Understanding service variability for profitable software as a service - service provider's perspective. In: 26th International Conference on Advanced Information Systems Engineering (CAiSE) (2014)
2. Ouh, E.L., Jarzabek, S.: A conceptual model to evaluate decisions for service profitability. In: 7th International Conferences on Advanced Service Computing (2015)
3. Mili, A., Chmiel, S.F.o., Gottumukkala, R., Zhang, L.: An integrated cost model for software reuse. In: 22nd International Conference on Software Engineering (ICSE) (2000)
4. Frakes, W., Terry, C.: Software reuse - metrics and models. J. ACM Comput. Surv. (CSUR) **28**(2), 415–435 (1996)
5. Jarzabek, S., Daniel, D.: Adaptive reuse technique. http://art.comp.nus.edu.sg
6. Amazon Web Services, 3-Tier Auto-scalable Web Application Solution. http://calculator.s3.amazonaws.com/index.html#key=calc-LargeWebApp-140323. Accessed Aug 2015
7. Apache, Apache OFBiz. https://ofbiz.apache.org/. Accessed Aug 2015
8. Poulin, J., Himler, A.: The ROI of SOA based on traditional component reuse, 2006. http://semanticommunity.info/@api/deki/files/2729/=ROI_of_SOA.pdf
9. Nolan, A.J., Abrahão, S.: Dealing with cost estimation in software product lines: experiences and future directions. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 121–135. Springer, Heidelberg (2010)
10. Ma, D.: The business model of "Software-as-a-Service". In: IEEE International Conference on Services Computing (SCC) (2007)
11. Ma, D., Seidmann, A.: The pricing strategy analysis for the "Software-as-a-Service" business model. In: Altmann, J., Neumann, D., Fahringer, T. (eds.) GECON 2008. LNCS, vol. 5206, pp. 103–112. Springer, Heidelberg (2008)

12. Gabriella, L., Ojala, A.: SaaS architecture and pricing models. In: IEEE International Conference on Services Computing (SCC) (2014)
13. Xu, H., Li, B.: Dynamic cloud pricing for revenue maximization. In: IEEE Transactions on Cloud Computing (2013)
14. Sengupta, B. Roychoudhury, A.: Engineering multi-tenant software-as-a-service systems. In: 3rd International Workshop on Principles of Engineering Service-Oriented Systems. ACM (2011)
15. Ju, L., Sengupta, B.: Tenant Onboarding in Evolving Multi-tenant Software-as-a-Service Systems. In: 19th International Conference on Web Services (ICWS) (2012)
16. Mietzner, R., Metzger, A., Leymann, F., Pohl, K.: Variability modeling to support customization and deployment of multi-tenant-aware software as a service applications. In: ICSE Workshop on Principles of Engineering Service Oriented Systems (PESOS) (2009)
17. Mietzner, R., Leymann, F.: Generation of BPEL customization processes for SaaS applications from variability descriptors. In: International Conference of Services Computing (SCC) IEEE (2008)
18. Morin, B., Barais, O., Jézéquel, J.-M.: Weaving aspect configurations for managing system variability. In: 2nd International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS) (2008)
19. Kwok, T., Mohindra, A.: Resource calculations with constraints, and placement of tenants and instances for multi-tenant SaaS applications. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 633–648. Springer, Heidelberg (2008)
20. Zhang, Y., Wang, Z., Bo, G.: An effective heuristic for on-line tenant placement problem in SaaS. In: International Conference on Web Services (ICWS) IEEE (2010)