A Conceptual Model to Evaluate Decisions for Service Profitability

Eng Lieh Ouh
Institute of Systems Science
National University of Singapore
25 Heng Mui Keng Terrace, Singapore
e-mail: englieh@nus.edu.sg

Stan Jarzabek
Department of Computer Science, School of Computing
National University of Singapore
Computing 1, 13 Computing Link, Singapore
Faculty of Computer Science,
Bialystok University of Technology
e-mail: stanjarzabek@gmail.com

***Abstract*— Service profitability depends on the cost of engineering a service for a given base of tenants, on service provisioning cost, and on the revenue gained from selling the service to that tenant base. The tenant base depends on the range of service variability, i.e., on Service Provider's ability to vary service requirements to meet tenant expectations. These various factors that have to do with service profitability form a complex web of information that makes it difficult to analyze and see the exact impact of decisions regarding the choice of service architecture or the use of service adaptation techniques. To make this analysis easier for Service Providers, we built a conceptual model that helps Service Providers identity factors affecting service profitability and the interplay among them. Based on that model, Service Providers can answer questions regarding how choices of the service architecture or tenant base affect service profitability.**

Keywords-service provider; service profittability; service architecture; service variability; tenant base; service engineering; service provisioning.

I. BACKGROUND AND MOTIVATION

Service Providers maximize service profits by looking into ways to best reduce their engineering and provisioning costs while selling the service to possibly a large number of satisfied tenants. The choice of service architecture plays a critical role in balancing the way these three forces affect service profitability. However, the most scalable and cheapest for service provisioning shared service architectures tend to restrict service adaptability. It is our goal in this paper to analyze the interplay among conflicting forces that affect service profitability, and identify the detailed factors behind these forces. The conceptual model of service profitability presented in this paper is to help Service Providers better see 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. This conceptual model can be further extended with provisions for quantitative analysis of service profitability, which is the subject of our on-going work.

In our previous work [1], we described the decisions that Service Providers typically make during service engineering regarding the choice of service architecture, service packaging, service hosting and the use of static or dynamic binding techniques to adapt services to needs of various tenants. Service components can be encapsulated at a service or tenant specific level packaging. For service hosting, a service can be hosted on a dedicated or shared process instance. Service architectures differ in how the service code is managed during service engineering, service execution and service hosting. As compared to [1], this paper further elaborates on and formalizes earlier findings to build a conceptual model.

Section II introduces Service Variability and Service Architectures. We give our proposed Service Profitability Model in Section III, followed by an Analysis of Service Profitability in Section IV. Related work is presented in Section V. Our conclusion is in Section VI.

II. SERVICE VARIABILITY AND SERVICE ARCHITECTURES

The range of service variability is the extent to which a Service can be adapted to varying service requirements of different tenants. The larger the range the service can accommodate, the higher the number of tenants and the higher the revenue for the Service Provider. Service architecture is composed of a set of components and the relationships among them to implement a service. A service supports a set of common features [15], shared by all the tenants**,** and a set of variant features (F_{RSV}), that are in a range of service variability, i.e., features that are of interest to some but not all tenants. Each feature $f \in F_{RSV}$ corresponds to a set of variation points in service architecture components. The purpose of variation points is to enable customization of components whenever f is required by a tenant. A selected variant to be bind to a variation point is composed of service components of one or more modules as shown in Figure 1.

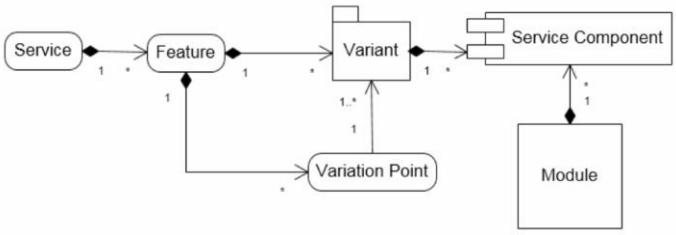

Figure 1. Service, Features and Variants

Fully-Shared (SA_{FS}) is a service architecture based on a shared process instance with service components being shared by tenants during service execution. Dynamic binding techniques are used to bind the variants to the variation points of the feature for service running in SA_{FS}. The range of service variability that is supported by SA_{FS} is as follows:

 ISBN: 978-1-61208-387-2

- At least one new variant or variation point needs to be designed and deployed onto existing process instance to support the varying requirements.

- Existing features can be configured on existing process instance to support the varying requirements

Partially-Shared (SA_{PS}) is a service architecture based on a shared process instance but, as opposed to SA_{FS}, the service components in SA_{PS} can be tenant level or service level packaged. In other words, the service components packaged in the module can be servicing a set of tenants or a particular tenant. Both static and dynamic techniques can be used to bind the variants to the variation points of the features for Service running in SA_{PS}. The range of service variability that is supported by SA_{PS} includes the variability supported by SA_{FS} and the following:

- For at least one new variant, variation points need to be designed and deployed onto a dedicated process instance to support the varying requirements.

- Existing features can be configured, but some features need to be deployed onto a dedicated process instance to support the varying requirements.

Non-Shared (SA_{NS}) is based on each tenant having its own, dedicated process instance and the service components being tenant level packaged. Both static and dynamic techniques can be used to bind the variants to the variation points of the features for Service running in SA_{NS}. The range of service variability that is supported by SA_{NS} includes the variability supported by SA_{PS} and the following:

- All required features need to be deployed onto a dedicated process instance to support the varying requirements.

The service architecture can be hybrid, comprising of a combination of existing service architectures. The SA_{FS+PS} is the hybrid service architecture comprising of SA_{FS} and SA_{PS}. $SA_{FS+PS+NS}$ is the hybrid service architecture comprising of $SA_{FS,}$ SA_{PS} and $SA_{NS.}$ The deployment diagrams of the three basic service architectures $SA_{FS,}$ SA_{PS} and SA_{NS} are shown in Figure 2. The white portions indicate components of the architecture that are not shared among tenants and the dark portions indicate components that are shared among tenants.

Service architectures that share runtime resources to minimize provisioning cost potentially limit the extent to which services can be adapted to varying requirements of tenants. For example, a tenant who requires service to be processed and data isolated due to security regulations cannot be onboard together with other tenants (that do not have such requirements) with a service architecture that does not have clear separation of runtime resources between tenants. Having clear separation of runtime resources among tenants incurs higher provisioning cost for the Service Provider, which would be likely passed on to the tenant as higher service price. On the other hand, there are also tenants who are price-sensitive with minimum variations of requirements. In this case, Service Providers can best minimize cost by engineering the service on a service architecture that shares resources. To the Service Provider, what are the factors and how they interplay can greatly impact their service profitability. We formalized these factors into a Service Profitability Model to help Service Providers analyze a complicated web of interrelated factors affecting service profitability.

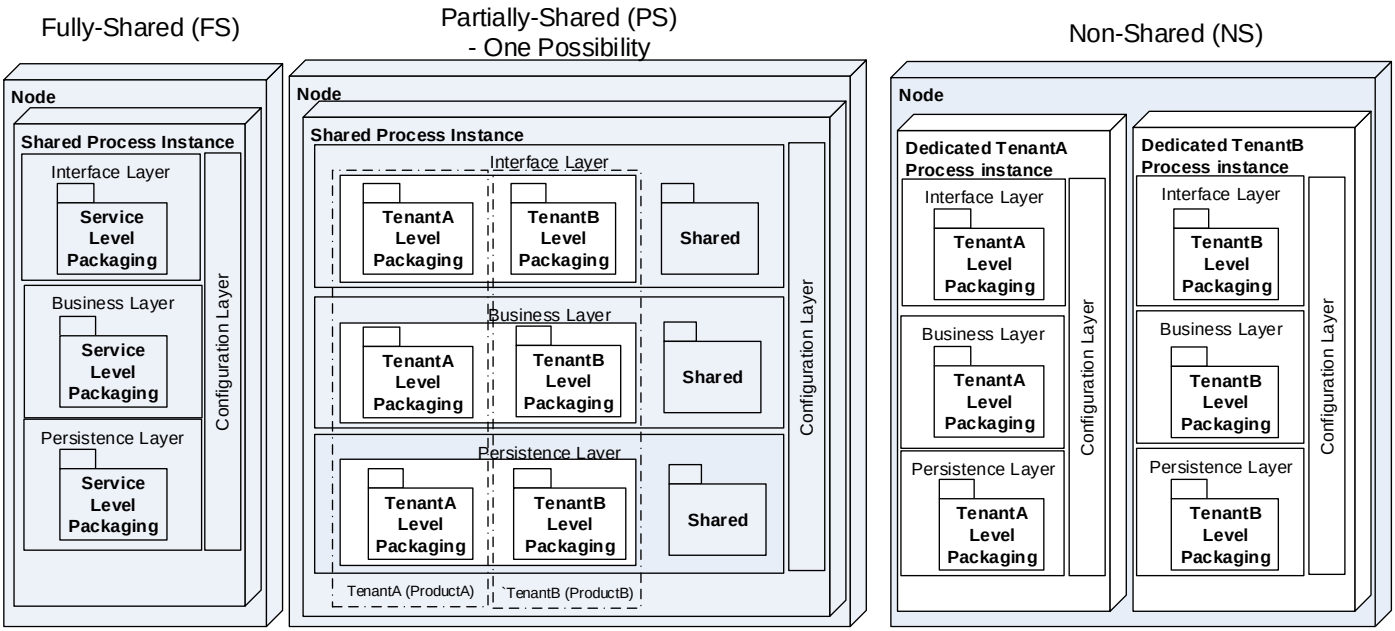

Figure 2. Service Architectures

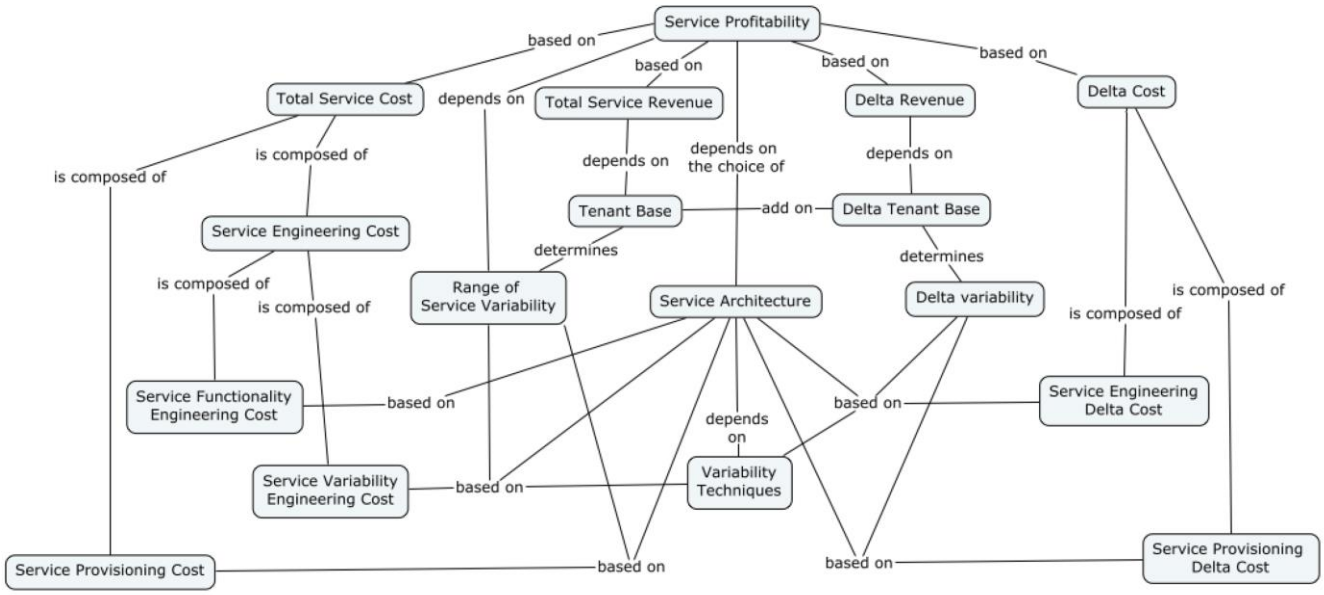

Figure 3. A conceptual model of Service Profitability

III. CONCEPTUAL MODEL OF SERVICE PROFITABILITY

Service Providers seek to maximize their profitability in the long term by minimizing their costs while maximizing their revenue. A Service Provider can re-engineer an existing application into a service or develop a service from scratch. This is the cost incurred in engineering the functionality of the service. The engineering cost varies depending on the choice of service architecture, but is independent of the number of tenants. Besides engineering service functionality, a Service Provider needs engineer variability into a service to accommodate variations in requirements among tenants. For business strategic reasons, a Service Provider may have some target tenants in mind and engineer the variability to meet the varying requirements of the target tenants. The extent of the service variability can vary with each variability technique and the selected service architecture which in turn impact the cost to engineer for variability. Service engineering costs can be estimated with software cost estimation tools (e.g., COCOMOII [2]) typically in terms of function points or lines of code. The other factor to the cost on top of service engineering cost is the provisioning cost. This cost is incurred to host the service in a hardware and network environment to serve the tenant's requests. The provisioning cost varies with the selected service architectures and the extent of service variability. The provisioning can be hosted internally or externally, virtualized or non-virtualized. To minimize provisioning cost, a virtualized environment is usually adopted. One key factor to whether the hosting is internal or external depends on each organization's security policy. Provisioning cost of virtualized environment can be estimated on the respective cloud hosting sites (e.g., Amazon Web Services [3]). During the lifecycle of the service, new tenants might be interested to subscribe to the same service. For business reasons, a Service Provider may also wish to onboard new tenants to maximize their revenue. However, the current extent of service variability might not exactly fit the requirements of the new tenants. The additional engineering and provisioning costs vary with the selected service architectures, adopted variability technique and the extent of variability of the tenant's requirements. The costs to engineer and provision for delta variability can be estimated similarly as described earlier. Service revenue will vary with the number of initial set and new set of tenants. Service profitability for providing the service in the long term can be measured by the net present value of the revenue gained minus the costs incurred over a pre-determined investment horizon, taking into account the value of money over time. For the rest of this section, we define the terms and use them in the conceptual model of service profitability. A conceptual model of Service Profitability is found in Figure 3.

A. Explanation of the terms in conceptual model

The Range of Service Variability (RSV) is the extent to which a Service can be adapted to varying service requirements of different tenants. The larger the range the service can accommodate, the higher the number of tenants and the higher the revenue for the Service Provider. The Tenant Base (TB) is composed of the users of a given Service, or onboarded tenants. To onboard 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. Service Providers dream to engineer a service where RSV fulfills the varying requirements of the TB, maximizing service profits. Service Profits are determined by the Total Service Cost (TSC) incurred and Total Service Revenue (TSR) gained when providing the service. Costs 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). For this, we collectively use the term Service Engineering Costs (SEC) to denote the both the Service Functionality Engineering Cost (SFEC) and Service Variability Engineering Cost (SVEC). SVEC is based on the selected Variability Techniques (VT) to support a given RSV

 ISBN: 978-1-61208-387-2

on a given SA. There are also costs involved to provide the hardware and infrastructure resources to support a given TB on a given SA. For this, we use the term Service Provisioning Costs (SPC). TSR is the total revenue from selling the Service on a given service SA.

Delta Variability (DV) is the change to existing Service requirements required to onboard new tenant(s). We denote these newly onboarded tenants as Delta Tenant Base (DTB). Delta Cost (DC) is the cost to implement DV for a given DTB on a given SA. Similarly to TSC, DC is composed of Service Engineering Delta Cost (SEDC) and Service Provisioning Delta Cost (SPDC).

IV. An Analysis of Service Profitability

In this section, we show use cases for the conceptual model presented as ten questions illustrating specific profitability-related dilemma of a Service Provider during service planning.

A. Service Profitability Model and Service Archiectures

For $SA_{FS,}$ the service engineering cost to support the given range of variability is composed of implementing the dynamic binding techniques to bind the variants to the variation points identified for each feature of the Service. The service provisioning costs to support the given range of service variability is composed of provisioning the hardware and infrastructure resources to support the tenant base. Among the basic architectures, SA_{FS} have the lowest provisioning costs due to sharing of resources. Delta variability that is the changes to existing Service requirements by new tenants have to be supported by the service architecture to onboard the tenant. If the delta variability is outside the given range of service variability of SA_{FS} then there are two possible scenarios. One scenario is that the Service Provider incurs the service variability delta costs to engineer the delta variability into the existing $SA_{FS.}$ (e.g., implementing new variants or variation points that can be shared among tenants). The second scenario is that the tenant cannot be onboard as the delta variability cannot be supported on existing SA_{FS} (e.g., tenant require total isolation of software, process and data).

For $SA_{PS,}$ the service variability engineering costs to support the given range of service variability is higher than SA_{FS} as the Service Provider needs to implement both the dynamic and the static binding techniques. The service provisioning costs are also higher than SA_{FS} due to the tenant level packaging leading to the need to provision more resources to support more software components. If the delta variability of new tenants is outside the given range of service variability of $SA_{PS,}$ a Service Provider can onboard the tenants by incurring service variability delta costs to engineer the delta variability into the existing $SA_{PS.}$ (e.g., implementing new variants or variation points that can be shared among tenants or supporting isolation of software or data). As SA_{PS} is running on shared process instance, the Service Provider is unable to onboard the tenant if the tenant has requirements that require a dedicated process instance (e.g., isolation of processes).

For $SA_{NS,}$ the service variability engineering costs to support the given range of service variability is lower than SA_{PS} as it needs to implement only based on static binding techniques. The service provisioning costs are the highest among SA_{NS} and SA_{PS} as dedicated resources are provided for each tenant. If the delta variability of new tenants is outside the given range of service variability of SA_{NS}, SA_{NS} can support delta variability of new tenants (even for isolation of process) due to dedicated process instances.

For $SA_{FS+PS,}$ the service variability engineering costs to support the given range of service variability involves the implementation of both dynamic and static binding techniques and the need to support two basic service architectures. It is higher than any of the basic architectures $SA_{FS,}$ SA_{PS} or $SA_{NS.}$ Depending on whether the service is provisioned on SA_{FS} and/or $SA_{PS,}$ the provisioning costs for SA_{FS+PS} can be calculated from the respective provisioning costs of SA_{FS} and/or SA_{PS}. The support of delta variability is similar to the scenarios in basic architectures of SA_{FS} or $SA_{PS.}$

For SA_{FS+PS} and $SA_{FS+PS+NS,}$ the service variability engineering costs to support the given range of service variability involves the implementation of both dynamic and static binding techniques and the need to support all three basic service architectures. It is the higher than the basic service architectures. Depending on whether the service is provisioned on SA_{FS}, SA_{PS} and/or $SA_{NS,}$ the provisioning costs for SA_{FS+PS} can be calculated from the respective provisioning costs of SA_{FS}, SA_{PS} and/or SA_{NS}.

B. Key Questions to Service Profitability

Questions 1 to 3 and 4 to 6 are related to the revenue and cost of the Service Profitability Model. Questions 7 to 9 are related to service architectures, while the last question is related to the overall service profitability.

1) How many tenants will have to be onboarded so that the profit from service outweighs the cost of building it?

The answer to this question will help the Service Provider to estimate the breakeven point for service cost and revenue. For a given service architecture, the Service Provider can simulate the number of tenants against the expected cost incurred to build the service for the Service Provider to have a better insight on the breakeven point for the initial total service costs.

2) How many new delta-tenants will have to be onboarded to outweigh the cost of implementing changes (delta-requirements) required for new tenants?

To Service Providers, this is another question on the breakeven of their investment. It differs from question 7 in terms of the delta cost against the delta revenue gained from the delta tenant base. For a given service architecture, the Service Provider can simulate the number of delta tenants against the expected delta cost incurred to build the service, the Service Provider has a better insight on the breakeven point for the delta costs.

3) Which pricing stratgey should a Service Provider adopt for the service?

The right pricing strategy positively impacts the total service revenue and delta revenue to be gained. The pricing strategy can be pay as per use, subscription-based, tiered based, transaction based or mixture of the above. The pricing strategy can also vary across time. For a given service architecture, the

 ISBN: 978-1-61208-387-2

Service Provider can simulate the expected distribution of tenants for each of the pricing strategies across the investment horizon and evaluate for overall service profitability.

4) Is it better to re-engineer from existing code or develop services from scratch?

The decision to this question impacts the service functionality engineering cost, which is independent of the number of tenants. Based on simulation of the expected distribution of tenants, this question can be addressed to see if the impact of the additional costs to develop from scratch versus the savings by re-engineering from existing code.

5) Should a Service Provider provision the service internally or externally?

The impact of hosting choices on service profitability can be inferred from service provisioning and service provisioning delta costs. The cost of provisioning the service internally typically incurs higher upfront setup and maintenance cost than external provisioning. However, it provides higher degree of security in terms of privacy and isolation of data and processes. Provisioning the service externally can decrease upfront setup and maintenance costs, but the level of control is also reduced. Based on the above, simulations can be run with these costs and expected distribution of tenants to analyze overall service profitability.

6) Which variability technique should a Service Provider apply to address the required range of service variability?

A Service Provider can apply static variability techniques to address higher degree of variability, but can also be more costly as compared to only applying dynamic variability techniques to address runtime variability. The adoption of variability technique impacts the service variability engineering cost and service engineering delta cost. Additionally, due to the variability technique, new tenants with requirements that does not fit the service variability will either be unable to onboard or the service architecture needs to evolve. Assuming the service architecture remains the same, the Service Provider can simulate the expected distribution of tenants for each of the adopted variability techniques and evaluate for overall service profitability.

7) To what degree does the selection of a service architecture impact service profitability?

This is a typical profitability-related question a Service Provider tries to answer. Based on the conceptual model, a Service Provider can estimate the total service costs incurred for each choice of the service architecture. In addition, a Service Provider may simulate a set of expected delta tenants that can be onboarded within the pre-determined specified investment horizon and measure the impact to total service costs, total service revenue and overall service profits.

8) Should a Service Provider onboard new tenant if this requires the change of service architecture?

Onboarding new tenants present new business opportunities, but also incurs costs to implement extra delta variability into the Service. This cost grows further if onboarding new tenants requires the change of service architecture, e.g., from shared to dedicated. In this case, the Service Provider needs to simulate different distributions of tenants and evaluate the overall profitability. These distributions differ in their degree of varying requirements and evaluate the service profitability impact on each of the service architectures. From the results, a Service Provider can have a better insight whether to evolve their architecture or not.

9) Should a Service Provider adopt a specific service architecture or a hybrid of service architectures?

Besides deciding on the type variability techniques, adopting a hybrid of service architectures requires additional cost incurred to manage the variability across service architectures. The Service Provider can simulate the expected distribution of tenants for each service architecture including different hybrids and evaluate for overall service profitability.

10) If a Service Provider has an objective to achieve a certain level of service profitability within a certain investment horizon, what costs and revenue the Service Provider needs to incur and gain?

This question can be considered an optimization problem in terms of maximizing the total service revenue and delta revenue while minimizing the total service costs and delta cost for a given level of service profitability. From the conceptual model, factors that affect the total service cost and total service revenue can be defined as equations to be evaluated by an optimization solver. More than one optimal set of values is possible in this case. The Service Provider can evaluate and design towards one of these values in this optimal set.

Each of these questions can be addressed at the lower level of abstraction. For example, in the first question, the assumption of a given service architecture can be further decomposed to evaluate against a set of service architectures for the Service Provider to be able to evaluate more scenarios.

V. RELATED WORK

There is much literature on how to minimize service engineering and provisioning costs to improve service profitability. Existing methods focus on variability techniques to enable late binding of the service variants, customization of business process execution language (BPEL) process with variability descriptors [8] and variability modeling techniques to manage the variability in service applications [9][10]. Kwok et al. [11] propose to lower the provisioning cost through resource calculations with constraints to decide the best server to onboard a new tenant. A resource consumption estimation model to optimally place onboarding tenants is also studied in [12].

A single-instance multi-tenant service application enables a Service Provider to achieve economies of scale through runtime sharing. However, runtime sharing can make tenant-specific variations difficult to achieve in such an application as it needs to realize the variability across different tenants in the single-instance application [13]. For the software as service paradigm to truly meet its potential, Sengupta et.al. [14] propose that vendors will need to move away from building rigid “one-size-fits-all” systems, or those that offer a fixed set of available customization options from which tenants must select. Service paradigm essentially is an economic model for software consumption; hence, many of these activities would

 ISBN: 978-1-61208-387-2

have to be grounded on the basis of financial reasoning that can benefit the vendor as well as the tenants. In contrast with other works that addresses costs or benefits independently, this paper takes a holistic view of service adoption in terms of service profitability. To the best of our knowledge, there are no such economics-driven service adoption evaluation methods; therefore service adoption is frequently made without holistically evaluating whether it is economically worthwhile to invest for the long term. In this study, we seek to propose an economic model of service profitability based on high-level conceptual model of service profitability (original contribution) and existing value-added software engineering metrics and economics-driven models used in other areas. We believe that the reasoning based on service profitability is able to address the service paradigm with a more balanced view than before.

VI. CONCLUSION

Our proposed Service Profitability Model formalizes the interplay of multiple factors that influence service profitability. The model addresses decisions related to the tenant base, required range of service variability, service architecture and the use of variability techniques. Our model shows how these decisions affect service cost and revenue. We illustrated the usage of our profitability model with an analysis of service architectures and service profitability scenarios. We believe the model will help Service Providers maximize service profitability. For future work, we intend to extend our Service Profitability Model with quantitative methods and tools that can help Service Providers examine factors that affect service profitability.

REFERENCES

[1] E. L. Ouh and S. Jarzabek, “Understanding Service Variability for Profitable Software as a Service: Service Providers’ Perspective,” in 26th International Conference on Advanced Information Systems Engineering (CAiSE), 2014, pp. 9-16.

[2] “COCOMO II” http://csse.usc.edu/csse/research/COCOMOII [retrieved: Jan, 2015].

[3] “Amazon Web Services Pricing” http://aws.amazon.com/pricing/ [retrieved: Jan, 2015].

[4] A. Mili, S. F. Chmiel, R. Gottumukkala, and L. Zhang, "An integrated cost model for software reuse," in Proceedings of the 22nd international conference on Software engineering (ICSE), 2000, pp. 157-166.

[5] "The Apache Open for Business Project (OfBiz)" http://ofbiz.apache.org/ [retrieved: Jan, 2015].

[6] Stanislaw Jarzabek and Dan Daniel, "Adaptive Reuse Technique" http://art.comp.nus.edu.sg/ [retrieved: Jan, 2015].

[7] "FeatureIDE" http://wwwiti.cs.uni-magdeburg.de/iti_db/research

[8] R. Mietzner and F. Leymann, "Generation of BPEL Customization Processes for SaaS Applications from Variability Descriptors," in IEEE International Conference of Services Computing, 2008, pp. 359-366,.

[9] R. Mietzner, A. Metzger, F. Leymann, and K. Pohl, “Variability modeling to support customization and deployment of multi-tenant-aware Software as a Service applications,” in Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems, 2009, pp. 18-25.

[10] B. Morin, O. Barais, and J. M. Jzquel, "Weaving Aspect Configurations for Managing System Variability," in Proceedings of VaMoS, 2008, pp. 53-62,.

[11] T. Kwok and A. Mohindra, “Resource calculations with constraints, and placement of tenants and instances for multi-tenant SaaS applications,” in Service-Oriented Computing (ICSOC), 2008, pp. 633-648.

[12] Y. Zhang, Z, Wang, B. Gao, C. Guo, W. Sun, and X. Li, “An effective heuristic for on-line tenant placement problem in SaaS,” in IEEE International Conference on Web services (ICWS), 2010, pp. 425-432.

[13] I. Kumara, J. Han, A. Colman, T. Nguyen, and M. Kapuruge, "Sharing with a Difference: Realizing Service-based SaaS Applications with Runtime Sharing and Variation in Dynamic Software Product Lines," in IEEE International Conference on Services Computing (SCC), 2013, pp. 567-574.

[14] B. Sengupta and A. Roychoudhury, “Engineering multi-tenant software-as-a-service systems,” in 3rd International Workshop on Principles of Engineering Service-Oriented Systems, 2011, pp. 15-21.

[15] J. Lei, B. Sengupta and A. Roychoudhury, "Tenant Onboarding in Evolving Multi-tenant Software-as-a-Service Systems," in IEEE International Conference on Web Services (ICWS), 2012, pp. 415-422.

[16] M. Svahnberg, J. Van Gurp, and J. Bosch, ”A taxonomy of variability realization techniques. Software: Practice and Experience,” 2005, pp. 705-754.

[17] Y. Zhang, Z. Wang, B. Gao, C. Guo, W. Sun, and Xu Li, “An effective heuristic for on-line tenant placement problem in SaaS,” in International Conference on Web services (ICWS), 2010, pp. 425-432.

[18] M. Ma, and J. Huang, ”The pricing model of cloud computing services,” in Proceedings of the 14th Annual International Conference on Electronic Commerce, 2012, pp. 263-269.

[19] A. Mukhija, D. S. Rosenblum, H. Foster, and S. Uchitel, “Runtime support for dynamic and adaptive service composition. in Rigorous software engineering for service-oriented systems,” 2011, pp. 585-603.

[20] H. Jegadeesan and S. Balasubramaniam, ”A method to support variability of enterprise services on the cloud,” in Cloud Computing, 2009, pp. 117-124.

[21] D. Ma, ”The business model of" software-as-a-service". in IEEE International Conference on Services ComputingC(SCC), 2007, pp. 701-702.

[22] W. Frakes and C. Terry, "Software reuse: metrics and models," ACM Computing Surveys (CSUR), vol. 28, no. 2, pp. 415-435, 1996.

[23] V. Choudhary, "Software as a Service: Implications for Investment in Software Development," in 40th Annual Hawaii International Conference on System Sciences (HICSS), 2007, pp. 209a.

[24] B. Boehm, "Software Engineering Economics," in IEEE Transactions on Software Engineering, 1984, pp. 4-21.

[25] C. P. Bezemer, A. Zaidman, B. Platzbeecker, T. Hurkmans, and A. t Hart, “Enabling multi-tenancy: An industrial experience report,” in 2010 IEEE International Conference on Software Maintenance (ICSM), 2010, pp. 1-8.

[26] H. Wang and Z. Zheng, “Software architecture driven configurability of multi-tenant SaaS application,” in Web Information Systems and Mining, 2010, pp. 418-424.

[27] R. Mietzner, F. Leymann, and M.P. Papazoglou, “Defining composite configurable SaaS application packages using SCA, variability descriptors and multi-tenancy patterns,” in Third International Conference on Internet and Web Applications and Services (ICIW), 2008, pp. 156-161.

[28] T. Kwok, T. Nguyen, and L. Lam, “A software as a service with multi-tenancy support for an electronic contract management application,” in IEEE International Conference on Services Computing (SCC), 2008, pp. 179-186.

[29] Y. Zhang, S. Liu, and X Meng, “Towards high level SaaS maturity model: methods and case study,” in IEEE Asia-Pacific Services Computing Conference (APSCC), 2009, pp. 273-278

 ISBN: 978-1-61208-387-2