CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

RESEARCH OF A FINANCIALLY VIABLE PROCESS VARIABILITY MODEL IN SOFTWARE AS A SERVICE SOLUTIONS

A graduate thesis submitted in partial fulfillment of the requirements
For the degree of Master of Science in Software Engineering

By

Jelugbo Babatunde

December 2014

The graduate project of Babatunde Jelugbo is approved by:

_____	_____
Taehyung Wang, Ph.D.	Date
_____	_____
Richard Covington, Ph.D.	Date
_____	_____
Adam Kaplan, Ph.D. Chair	Date

California State University, Northridge

DEDICATION

This paper is dedicated to all my classmates with whom I have done group projects or presentations and my professional colleagues who helped me in the course of my study to improve my knowledge.

Also to all my friends and family who have encouraged me to continue when this was more difficult than I thought it should have to be.

ACKNOWLEDGEMENT

Glory be to God and Special thanks to Professor Adam Kaplan who helped me through this accomplishment. Without his unrelenting help and support, it would be impossible for me to complete this thesis.

Having the opportunity to discuss and rub minds with me assisted me in analyzing things with a better perspective especially from the investment perspective. I really appreciate all his efforts to help my dream become true. Thank you.

I also want to thank my friends at the International Students Inc. and Shepherd of the hills Northridge for their support, without them, achieving this might have remained a dream rather than reality, I appreciate every effort towards my success.

Last but not least, thanks a lot to my family who has never let me down with love, emotional and financial support.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT
RESEARCH OF A FINANCIALLY VIABLE PROCESS VARIABILITY MODEL

IN SOFTWARE AS A SERVICE SOLUTIONS

By

Jelugbo Babatunde

Master of Science in Software Engineering

The purpose of this study is to design a model for examining the possibility of process level variation for perceived Software as a Service (SaaS) solutions and predicting its corresponding financial exposure.

SaaS has drawn various concerns especially in *Enterprise Resource Planning (ERP) solutions*. ERP are a variety of business management software usually packaged as a suite of integrated applications that can be used by a company to store and manage data from every phase of business.

Enterprise-level organizations have been hesitant to adopt SaaS-based architecture with the opinion that hosted or on-demand applications will not be able to handle their complex processes, provide vertical functionality and handle the ever changing landscape of their business model. Hence, adoption of SaaS ERP solutions have been limited to user-centric applications rather than enterprise-centric solutions.

Even if there was a perceived model that could possibly solve this problem, one major consideration exists for any organization willing to commit to such an endeavor. The question that must be answered is: will it be financially viable to support such a software as a service solution compared to an on-premise solution considering the Software as a Service "pay as you use" model?

In this study, we consider limitations of existing SaaS *multi-tenancy architecture,* an architecture where a single instance of the software runs on a server, serving multiple clients. We consider possible solutions and models as well as their financial

implications.

CHAPTER 1

1.1 Introduction

SaaS is a software delivery model in which software and associated data are hosted in a central location on the cloud. Clients access SaaS solutions using a thin client usually a web browser. It is important to note that a major selling point of SaaS vendors is the potential to reduce IT support costs by outsourcing infrastructure, software maintenance and support to the SaaS vendor.

SaaS solutions are usually offered as one-size-fits-all solutions i.e. all clients use the same software.

This is achieved via a *multi-tenancy architecture*, which permits clients to share the same code. Therefore any feature or functionality added to the software due to a customer's change request or other requirement change becomes available to all customers. This approach enables SaaS vendors the economies of scale needed to offer their software cost-effectively, while easing the task of upgrading their customer’s software to newer versions with added functionality.

On the other hand, consider an enterprise level software such as that used by the financial industry. An on-premise risk management software that calculates market risk can be used by a Financial Bank and Securities Trading company albeit with careful customization. The software, for example, needs to recognize the changing market price of securities as the risk parameter and security asset as exposures of a securities trading company as opposed to fluctuating interest rate as risk parameter and cash at bank as exposures in a Financial bank. Also the software needs to recognize how to extract the proper data, and subsequently transform and load it in order to generate an accurate result.

In the context of SaaS, the complexity of implementing such business logic nullifies

the "one size fits all" multi-tenancy approach of SaaS. Other things need to be considered during the requirements specifications stage such as:

- Various use case scenarios
- Architecture that supports modifiability
- Technology requirement
- Compliance requirements which might vary across industries etc.
- Cost, etc.

1.2 Statement of the Problem

SaaS users appreciate the flexibility of services and the concept of “pay as you use” , yet SaaS vendors have decided to stick with *vertical solutions* i.e. software that are aimed at addressing the needs of any given business within a specific industry or market., focusing on a product line with a set of routine functionalities.

However, one question that remains to be answered is the possibility of establishing a model to implement SaaS solutions that can service a *horizontal market*, software that can be useful in a wide range of industries i.e. its scope of usefulness is not limited to few industries.

A practical case of this can be seen in SAP Business ByDesign, an attempt by SAP to deploy its ERP solution via cloud which was eventually halted and broken down into a number of different applications that could be purchased independently, as needed. The vision of the project was described as “beautiful but too big” by Lars Dalgaard the founder of SuccessFactors, a company acquired by SAP.

Many SaaS vendors understand the advantages of SaaS but do not understand the financial aspects of cloud computing. We will consider the economic viability of the model stated above using a *financial decision support framework*. Financial decision

support frameworks are predictive models which exploit patterns found in historical and transactional data to identify financial risks and opportunities.

This thesis looks not only at the possibility of horizontal SaaS solutions but also encompasses the design of a decision-support framework for calculating financial metrics which will help understand the financial aspects of SaaS and learn how individual type of related costs and benefits can positively or negatively affect the total metrics.

1.3 Purpose

The purpose of this research is to:

1. Establish the possibility of developing a SaaS solution capable of handling varying business process.
2. Develop a financial decision-making tool for calculating financial metrics of SaaS solution development.

1.4 Related Work

There has been limited work in the area of establishing a variability process model for SaaS solutions. One of such study carried out by Mietzner and Leymann [1] describes the concept of using a variability descriptor to define variability points for the process layer and related artifacts of process-based, service-oriented SaaS applications. Their work embraces the concept of business process web services model which can be used to guide a customer in the customization of the SaaS Solution.

Another study by Salih et al [2] proposed variable service process for customizing multi-tenancy at runtime which will help appreciate the full benefits of variability concept for SaaS application. They also proposed a feature meta-model for implementing a graphical editor that can be used to define all rules and linkages

among elements of the service process.

Moreover, there have been prior studies on financial metrics for SaaS solutions. The two common metrics described in the literature are the Total Cost of Ownership (TCO) and Return on Investment (ROI) metrics.

TCO was created by Kirwin [5] and the concept was to understand the total life cycle cost, i.e. the cost of acquiring, using, managing and disposing of an asset over its useful life time. TCO determines all *direct cost*, cost that relates to tangible assets e.g. servers, peripherals, network etc. and *indirect costs,* cost due to time or productivity losses such as downtime in technology of an ERP solution and simulates various implementation scenarios to see which yields the best TCO. According to Kirwin [5], indirect cost greatly affects the TCO value due to process and people issues.

ROI metric [8] evaluates the total gain against total investment as a ratio thereby measuring the profitability of an investment. This indicates how well money is utilized and also help decide if it is wise to invest or not.

ROI can be used to provide a rationale for future investments and acquisition decisions, project justification, evaluation of existing systems and project post-implementation assessment.

ROI has been used employed in more IT project decision in the past than TCO because TCO projects cost only while ROI on the other hand draws a comparison between cost and expected benefits.

CHAPTER 2-LITERATURE REVIEW

2.1 Introduction

SaaS stakeholders often have contradicting requirements and interests. From a user's perspective, the quality and non-functional properties of an application have to be exhaustive. On the other hand, a SaaS vendor is interested in reducing operating costs while he maximizes profit, however, subscribers are interested in offering well-tailored and specific functionalities to their users.

In order to achieve an optimal compromise for all stakeholder's objectives, multiple levels of variability have to be supported by reference architectures for these SaaS applications.

This chapter takes a detailed and critical view of the existing research studies relating on variability process in SaaS and prevailing financial considerations.

2.2 Horizontal vs. Vertical SaaS

The concepts of variability process and its financial implications have been a reoccurring term and consideration in software as a service.

Enterprise software are commonly faced with the issue of trade-off between standardisation across industries and software customization for specific businesses.

Most software vendors starts vertically focusing on a particular business line or specialising in an industry, however, with growth expansion becomes a necessity and they achieve this by adding functionalities suitable for other industries or, acquire other software companies to speed up this process [9].

Wertz [10] in Techcrunch.com highlighted the advantages of vertical SAAS, stating that due to their narrow focus, vertical SaaS vendors enjoy lower customer acquisition costs along with lower capital requirements, however, the vertical markets can be limited in size and in order to scale, companies will need to enter into parallel verticals once they own their first which is a more painstaking venture.

David [11] also buttressed the points made by Wertz [11], highlighting lower Sales & Marketing Expense, and better product for the customer through deep industry expertise and focus, ability to rapidly capture substantial market share and Sector specific data advantages as driving factors towards more vertical solution.

The summary of these two authors are based on a narrow focus. David [11] believes that having a more refined market allows Vertical SaaS providers reach customers more quickly and that deeper industry knowledge drives more effective marketing while Horizontal software providers will need to continually replenish Sales & Marketing to sell across multiple verticals, and marketing messages can be too vague and broad-based for the end user.

Wertz [10] as well supports a narrow focus ideology, highlighting that this will help in better understanding the user's need and can help streamline user experience.

While their points are well articulated, it however seems to be provider-centric, focusing on low budget, and time to market alone which are important but not the only factors that are worthy of consideration.

One interesting point, stated by both authors is that the unique infrastructure of the cloud model itself makes it easier to deliver industry specific updates rapidly across multiple organizations. This point buttresses the fact that an architecture for variability process is worth considering and probably a successful horizontal SaaS Model can help SaaS vendors expand their scope

2.3 Economics of SaaS

Ritobaan [9] stated that SaaS is a costly business and due to economics, SaaS vendors are loathe to customise their applications for a customer.

Vendors can achieve competitive advantage by choosing one of two generic business strategies; lowest cost or differentiation, and a variation on both of these themes by focusing on the needs of a specific market segment [12].

In lowest cost strategy, customers are subscribed to a shared vertically integrated infrastructure i.e., hardware, software, maintenance, etc. over the internet thereby

achieving a lower economies-of-scale. The cost saved is then passed on to the customer resulting in a lower total cost of ownership (TCO) of SaaS.

The differentiation strategy involves re-engineering and automating high-value customer processes thereby creating a perceived unique value to the customer, Joel [12] recognise that some markets have customer's whose needs are so unique and applications that are so complex that they are intractably fragmented.

Joel adopted this concept from the book Competitive Strategy by Michael E. Porter [22].

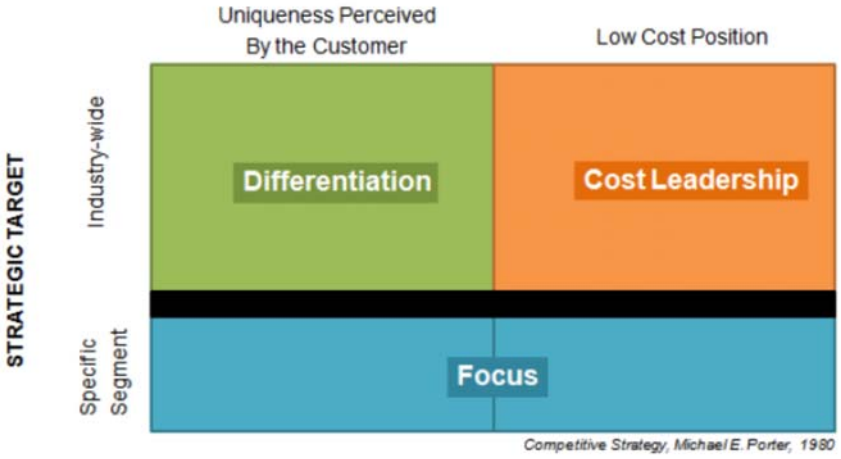

Figure 1: Competitive Strategy Model (Michael E. Porter 1980)

SaaS strategy is built on a premise of lower cost, Successful vertical SaaS has employed focusing on the needs of a specific market segment as a strategy, however, successful horizontal SaaS framework should leverage on differentiation, by providing variability framework that will remove the limitation of vertical SaaS.

Economically speaking, with the success of Software-as-a-Service (SaaS) , it is evident that customers prefer the features of on-premise software and the advantages of the cloud which include pay-as-you-go billing, ease of updates, elasticity and low capital expenditure (CAPEX, cost used in acquiring physical assets).

However, SaaS companies invest a lot of money to deliver these features. According to Rod Drury, CEO of Xero "It takes $100-200 million of investment

to build truly horizontal, global accounting engine". A study by venture capital firm Siemer in 2013 shows that the difference in five-year compound annual growth rate (CAGR, The year-over-year growth rate of an investment over a specified period of time) between SaaS and on-premise software companies tends to be as much as 26%, as a result, SaaS companies need more capital investment to break-even and bear higher operating expenses than traditional on-premise software [9].

According to Ritobaan [9], this implies that SaaS vendors will be very reluctant to customise their software for customer and will rather try to make an application that is rich enough in functionality to address core requirements of a business, hence, from a SaaS vendor's perspective, the overall goal would be to make a product that is as standardised as possible across the widest possible section of customers. This supports Horizontal SaaS over Vertical SaaS economically, a contradiction to David's [11] and Wertz [10] points in support of vertical SaaS over Horizontal SaaS.

From this, we have an idea that generally speaking, horizontal SaaS might be a more profiting venture compared to Vertical SaaS from a financial point of view.

2.4 Variability Process Models

In considering horizontal SaaS, variability modelling is important for managing variability in SaaS product families. Generally speaking, there are existing variability modelling techniques each with its own approach to model the variability provided by a software product.

Also, in developing components for a software product, varying properties such as reusability, architecture, requirements and test-cases have to be considered alongside the prevailing produce differences.

Variability management handles these differences by introducing use, and evolution of variability, i.e., the ability of a software system or artifact to adapt through extension, change, customization or configuration for use in a particular context [13].

A software context is the set of software products and artifacts that are created during software engineering and the specific environments these products are used in.

Managing variability is a complicated task faced with a number of challenges which originate from the various number of choices software engineers can make during product derivation, and the complexity of the constraints between them [13].

2.4.1 Variable service process by feature meta-model [2]:

In the variable process model, the writer dwell more on the service model stating that variability affects both the service interface and the service provider implementation and in implementing a horizontal service model, proper care must be taken to ensure the correctness of the entire process family and all of its configuration.

In his approach, Salih et al [2] considered three factors:

a. Service Life Cycle: Salih et al [2], while acknowledging that the processes do not get executed during the Service Life Cycle phase, emphasized the importance of a service strategy, the Service Life Cycle provide the necessary guidance to achieve success through a five layers strategy as depicted below:

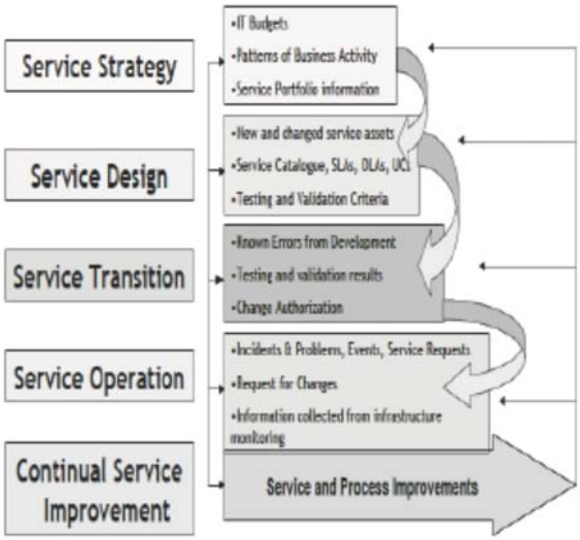

Figure 2: Service Model (Salih et al 2012)

The main objective of the strategy includes:

- Design, develop and implement service management as a strategic asset and assisting growth of the organization.
- Develop the IT organization's capability to manage the costs and risks associated with their service portfolios.
- Define the strategic objectives of the IT organization.

b. Service Process: Salih et al [2] believes that SaaS can optimize key service operating and achieve superior customer satisfaction by:
    - Building better quality product via process
    - Better implementation and customer support.
    - Different operational processes structure for management

    Considering that changes are inevitable, Salih et al [2] believes that proper Service Operation will help achieve stability in light of these changes and as a result Service Operation staff must ensure that changes are absorbed without hostile impact on the stability of the IT services. Change in services might arise due to various reasons such as:

    a. Software or Hardware upgrade
    b. Changes to meet dynamic business requirements.
    c. Enhancement in process.

c. Service Flexibility: Another important consideration by Salih et al [2] is the service flexibility which is important in implementing a scalable and adaptive service. Likened to the concept of dynamic service routing where routing path is not fixed, and can be changed by selecting service from the a service set

dynamically to meet the consumers need.

It is worthy to state here that such implementation needs to consider communication between the various process of the service, how it scales up or down on resources based on processes included in each service subscription and keeping track of resource usage, data backup etc. for individual subscribers.

The approach by Salih et al [2] was in two phases, designing a variable service and implementing a feature meta-model.

2.4.1.1 Variable Service:

Leveraging on the continual service improvement model Salih et al [2] designed a service that can be variable or continuous in improvement at runtime for SaaS, with this the SaaS model can be adaptive when there is a change in requirement or need to satisfy a subscriber's service level as shown in figure 2.

Using the model, the service state is monitored and the service level management is responsible for determining the level service requirement and realized service level agreement. Metrics such as service level achievement, cost of services and number of request from customer are taken and changes are dynamically applied to the service as necessary.

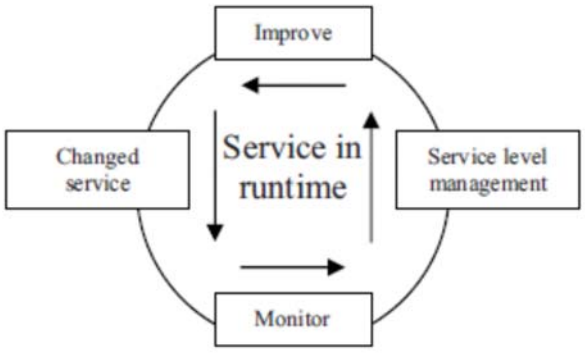

Figure 3: Continuous Service Improvement Model

2.4.1.2 Feature Meta Model:

In a variable service model, there is need for descriptors for describing the variable part of service provided. In achieving this, Salih et al [2] employed the use of feature meta model which provides a technical and code level solution for SaaS providers by helping them resolving their application variations while still shielding them from the implementation details.

The feature meta data use rules and constraints to determine the need to enable or disable feature for a subscriber which helps SaaS providers realize variation at run time for service process by applying rules and constraints that can be changed automatically in source code.

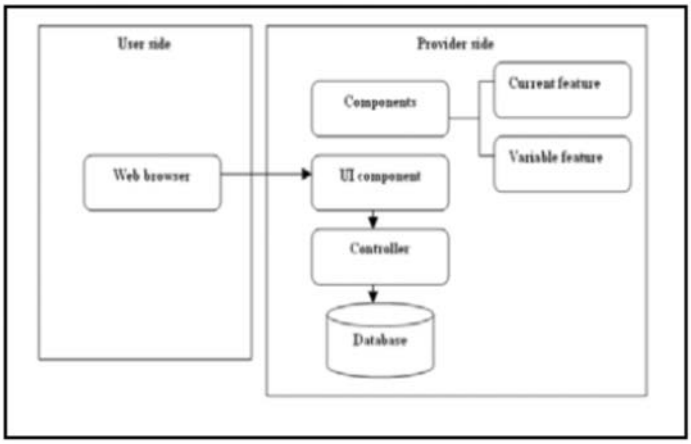

Figure 4: Meta Data Model (Salih et al. 2006)

2.4.2 Generation of BPEL Customization Processes [1]:

In this approach Mietzner et al [1] considered automatically generating process-based customization tools out of variability points in a SaaS application in order to support customers during the customization of an application.

Variability points are parts of an application that are unspecified or defaulted and can

be customized by a subscriber to suit their specific requirements. These variation points allow the specification of points in a software design that can be customized in order to create new product line members.

At subscription, the customer needs to customize the application by specifying concrete values for the variability points of the chosen application template, different values might be permitted at various variability points. Also some variability points can depend on other variability points. The end result is a solution tailored to a user's requirement.

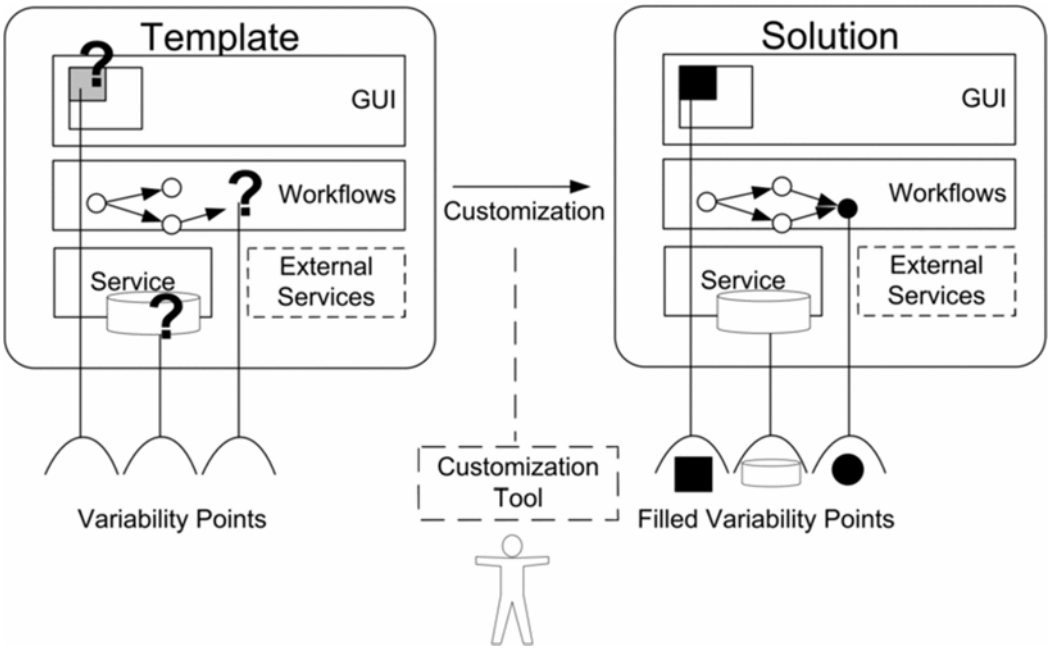

Figure 5: BPEL Customization Model (Mietzner et al. 2008)

Mietzner et al [1] model leverage on Web Services Business Process Execution Language (WS-BPEL), a standard executable language for specifying actions within business processes with web services but with more concentration on the implementation level.

Mietzner et al [1] approach was to consider a middleware for implementing product family lines which can then be deployed to the back-end. The Business Process

Execution Language (BPEL) seems to be a right approach as processes in BPEL export and import information by using web service interfaces exclusively [14]. BPEL uses the concept of opaque tokens which are variability points left open by explicitly marking them as opaque or implicitly by omitting them, however these tokens need to be defined and the abstract processes refined into executable processes in a procedure referred to as executable completion before the process becomes executable and can be deployed in a BPEL engine.

The BPEL specification defines two profiles for abstract processes, the first profile for observable behavior whose function is to describe the communication of a service with the outside and the second profile which describes a template process with variation points.

BPEL Opaque lack fundamental capabilities needed in a SaaS scenario such as ability to specify dependencies between opaque tokens or alternative values for an opaque token which is an important mechanism to specify complex configuration possibilities in a product line such as horizontal SaaS.

Mietzner et al [1] extended the BPEL approach by introducing a framework where variability descriptors can be plugged in, define a concrete format for these variability descriptors and integrate with Reusable Asset Specification (RAS) which allows the specification of concrete user-defined variability descriptors to be plugged-in into the general variability-point description mechanism.

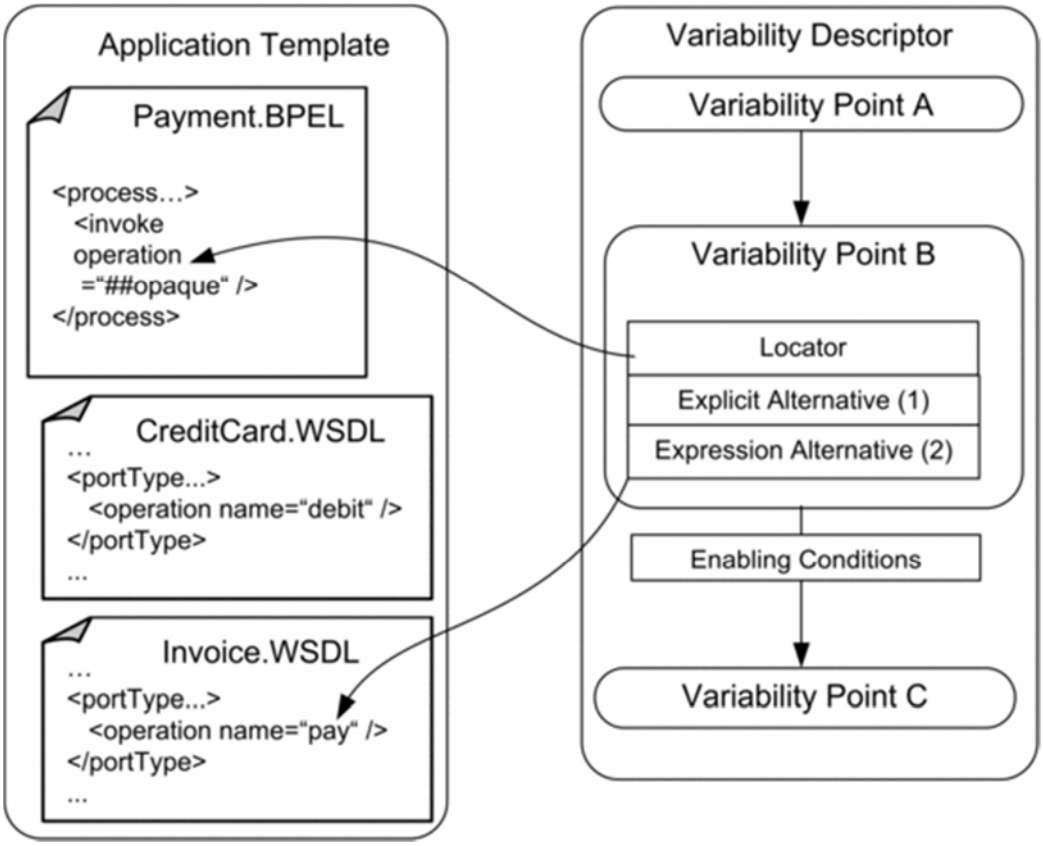

Figure 6: Mietzner Reusable Asset Specification Model (Mietzner et al 2008)

2.5 Proposed Research Study / Framework

In order to achieve the purpose of this study, we will leverage an existing study on variability process by Salih et al [2]. However we will leverage these results from an engineering point of view by taking the theoretical framework and focusing on its implementation if possible, or further refine the framework for possible implementation.

We will consider possible software architectural frameworks and design patterns that can be employed to achieve its implementation such as the combination of Model View Controller (MVC) and Template design pattern. MVC can help create room for extending software features without affecting existing features. Template design pattern is a behavioral design pattern that is employed by defining the program skeleton of a procedure or algorithm in a method, permitting developers to change the

flow of an algorithm without changing its structure by redefining the steps of such an algorithm. One or more algorithm steps can be overridden by subclasses to allow varying behaviors while ensuring that the master algorithm is still followed.

On the financial side we will consider metrics to justify the viability of such a project by adding a risk factor metric. We are currently considering the Value at Risk (VaR) and Capital Asset Pricing Model (CAPM) metrics.

VaR is based on the common-sense fact of the odds of losing money which is a distress factor for all investors. VaR answers the question, "How much can I possibly lose in a worst-case scenario?" [6]

VaR statistic has three components which are time period, confidence level and a loss amount or percentage, hence, Value at Risk calculates the maximum loss expected on an investment, over a given time period and given a specified degree of confidence.

CAPM is a model for estimating the cost of a particular asset in a portfolio, in this context, an asset could be developers, tools, and infrastructures etc. which will help us understand the performance of an asset on a risk-adjusted basis. The basic concept of CAPM is to illustrate that the only reason an investor should earn more, on average, by investing in one asset rather than another is that one asset is riskier [7].

Finally, we can use our variability process model to determine cost requirements based on selected architecture and design. We can estimate cost for infrastructure, developers, tools etc. which will serve as input into our financial model to calculate the viability of our SaaS project by determining our return rate and risk factor.

CHAPTER 3 - THE PROCESS VARIABILITY MODEL

The model will enable customization at various levels of the software application. In order to achieve this, the layered architectural pattern as the highest level of SaaS maturity will be an appropriate approach. The layered architecture is a pattern which enables us break down complicated system into adequate level of abstraction.

SaaS application maturity can be expressed using a model with four distinct levels, at the highest level of maturity, the vendor deploys multiple subscribers on a load balanced farm of identical instances, with each subscriber's data kept separate and configurable metadata providing a unique user experience and feature set for each subscriber [15].

The system becomes scalable to large number of subscribers, resources on the back end can be increased or decreased as necessary in response to demands without need for additional re-architecting of the application.

In order for SaaS vendors to support subscribers with a host of options and satisfy subscriber’s specific requirements such that it is possible for each subscriber to have a unique software configuration without extra development or operation cost, variability techniques can then be employed at the various levels of abstraction.

With such situation where functional requirements vary for different subscribers, we want to strike a balance and avoid duplication of efforts by maintaining cohesion as best as possible and specificity where applicable.

The solution will be to develop a customization mechanism that enable subscribers express their requirements. Such mechanism must be easy to use with default configurations provided via template objects. Specified configuration are then deployed to specific layers in our architecture.

To maintain cohesion, domain knowledge can then be applied to help during the

customization process, also knowledge mining from previous subscriber's configuration can be used to assist new subscribers in their customization process.

A typical SaaS layered configuration consist of four layers, the presentation layer, application layer, business logic layer and the data layer.

The presentation layer is the user interface through which the user interacts with the system. The application layer also referred to as the service layer or controller provides interface to functionalities or business logic within the application. The business logic layer function includes application of business rules, workflow and policies; data retrieval, processing, transformation, and management; and ensuring data consistency and validity. The data layer is concerned with the data structure and how data are persisted into the database.

Although all layers play major role in achieving a variability framework, our focus will be on the service layer and the business logic layer which are both concerned with the operation of the SaaS application and how business objects interact to achieve varying complex task.

All SaaS applications have a multi-tenant architecture at the data layer level so we will assume the database has been partitioned to support the proposed framework.

3.1 The Service Layer:

The service layer provides functional capabilities fulfilling system requirements which could be basic or atomic i.e. providing basic operations or complex i.e. invoking a series or related atomic services.

Variability in the service layer will be implemented using descriptor file that describes service configuration for a specific subscriber profile. In differentiating the service layer, we need to understand the various parts of the service in the description file.

3.1.1 The Integration Service:

This service implements methods for defining communication between processes internally and externally, data and workflow integration.

3.1.2. The Repository Service:

This service persist, analyze business information, and communicate with relevant services to fulfill business requirements. All process component are data-driven and hence have a repository which store template objects, for workflow, services etc.

3.1.3 The Metadata service:

This service facilitates definition, exchange and sharing of information relevant to the business between the various services.

For illustration purpose, we will consider using a sample Risk Management Software that can be used by different line of business such as, Investment Bank, Real Estate, and Insurance Sector etc. to manage their risk exposure to financial loss.

A typical complex service will be calculating possible financial loss of a portfolio which could be a collection of securities in the investment bank context where historical aggregation of fluctuation in stock prices could be used in evaluating potential loss, financial loan in the commercial bank context where historical aggregation of fluctuation in interest rate could be used in evaluating potential loss , real estate in the real estate management context where historical aggregation of fluctuation in periodic return on investment could be used in evaluating potential loss, and in insurance context where a portfolio can be a combination of any of real estate, securities trading and financial loan , hence, a combination of the three could be used in evaluating potential loss depending on the composition of the portfolio.

For simplicity purpose, we will consider the service at an atomic level, the application

of the domain knowledge of risk management is shown below where a service could be finding the right parameter or adequate interval of periods between parameter fluctuations. These tasks can be automated and customized, selection of appropriate service from a collection of service can be automated.

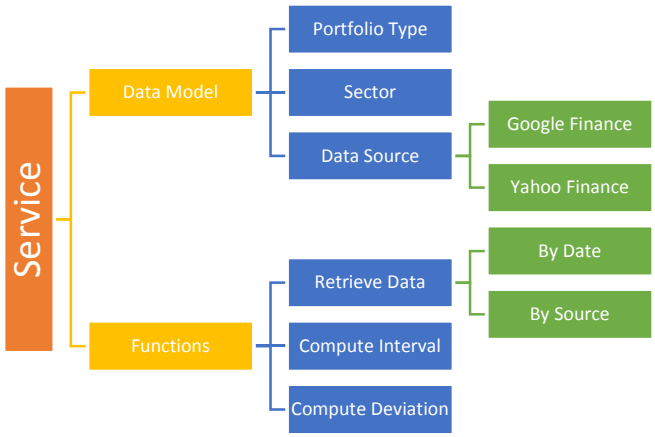

Figure 7: Sample Data and Functional Model

Figure 7 represents a sample risk management domain knowledge structure, functional requirements are accomplished via the functions flow. A subscriber who needs to retrieve historical price of stock price and calculate deviations can do so by calling the desire service in order based on value specified in the descriptor

3.2 The Business Logic Layer:

The business layer implements a workflow, organizing services from the service layer to achieve more complex business processes or workflow, hence, it is a collection of set of activities that represent business processes.

A subscriber customizes the business logic layer by specifying process in workflow using a template object. The workflow default template object are based on the business domain knowledge, common areas are statically defined in the template while description points can be specified for editable part of the templates.

For instance, calculation of Value at Risk is a series of process from end to end

beginning with sourcing data for a specific date range, a specific interval between the retrieved data is then carefully chosen which could be daily, weekly, monthly etc. depending on the frequency of change in data. The deviation between selected intervals is then calculated which in turn serves as input into the VaR model for final calculation of value at risk value.

Figure 8 is a sample workflow template implementation of a subscriber who requires to evaluate his financial risk exposure base on existing portfolios, once the portfolio is set up, service template will be generated based on classification of portfolio assets which could be, stock trading, real estate etc. or a mix of asset. Variable service objects will include data retrieval, frequency estimation while deviation estimation and VaR algorithm are fixed points.

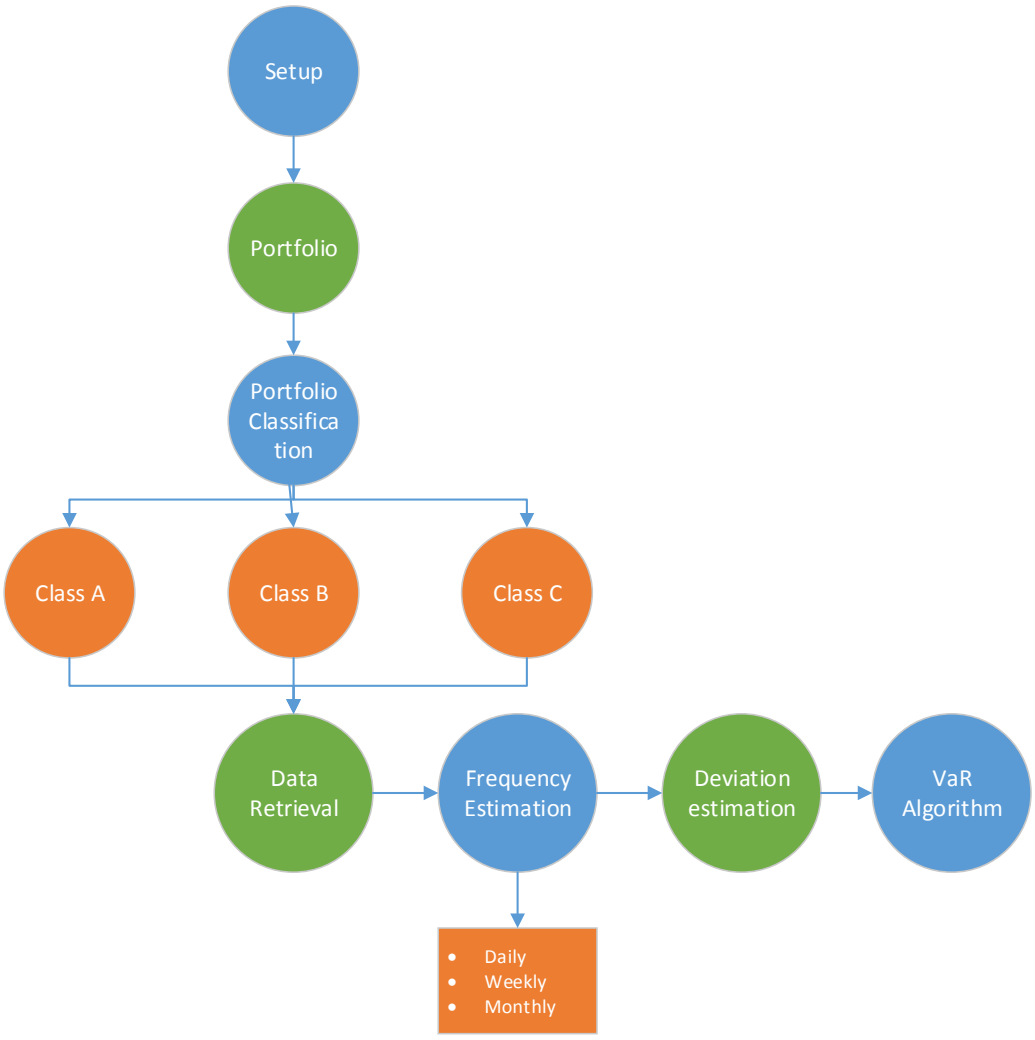

Figure 8: sample workflow template implementation

3.3 The Model

The model consist of three parts controlled by a role-based user access control allowing users access to the critical services necessary for the application configuration / usage. The access control is used to determine which functions and resources users should be allowed to access as defined by the subscriber. Users can be grouped into different roles depending on the organizational structure access rights control can be constituted accordingly.

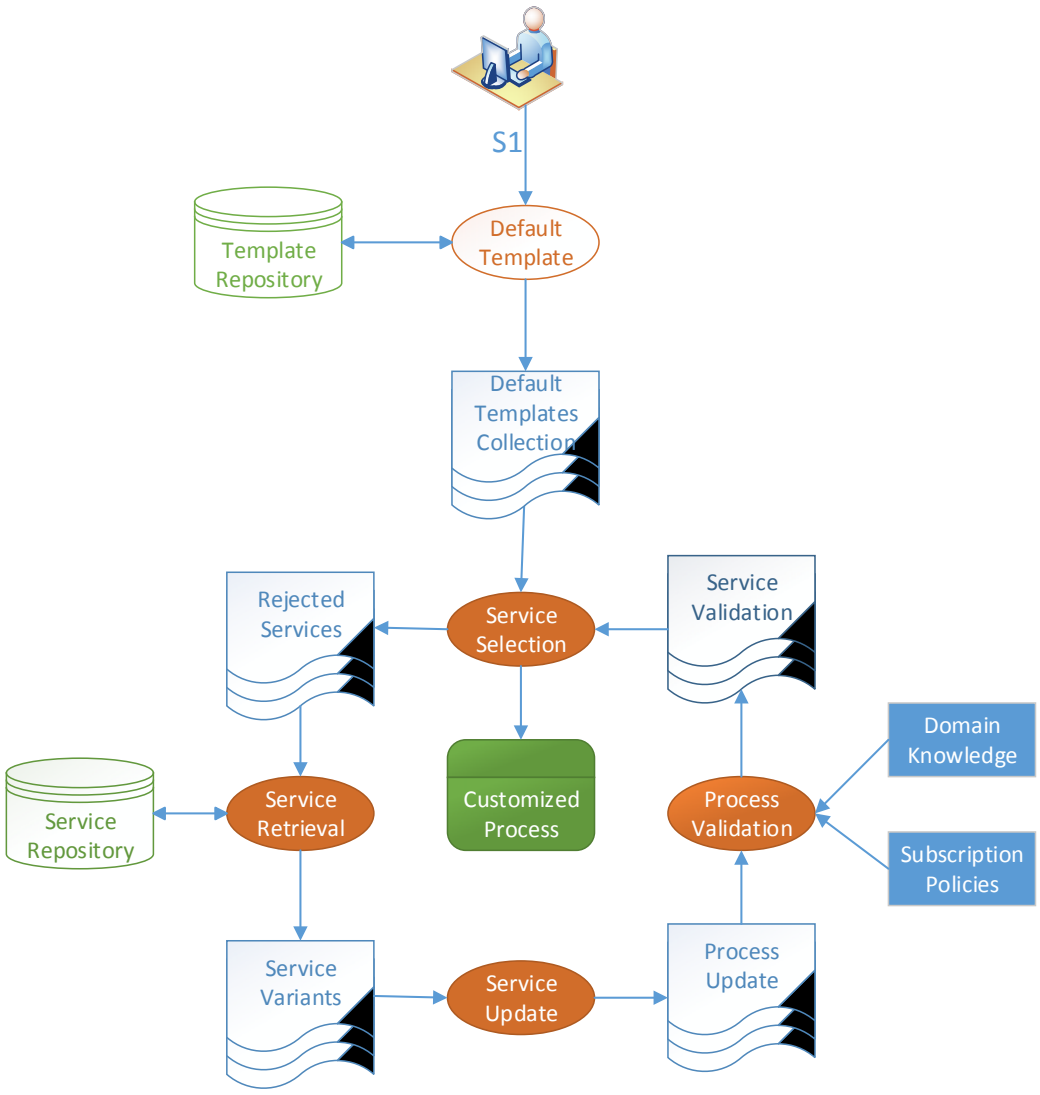

Figure 9: Proposed Variability Model

3.3.1 Customization Process:

Customizing a process via the model will involve definition or values for process variability points by an administrator. A template is provided by default, and customization values for business processes are validated and persisted via the metadata service.

Also, service components are selected and integrated based on the defined business process flow for use at runtime.

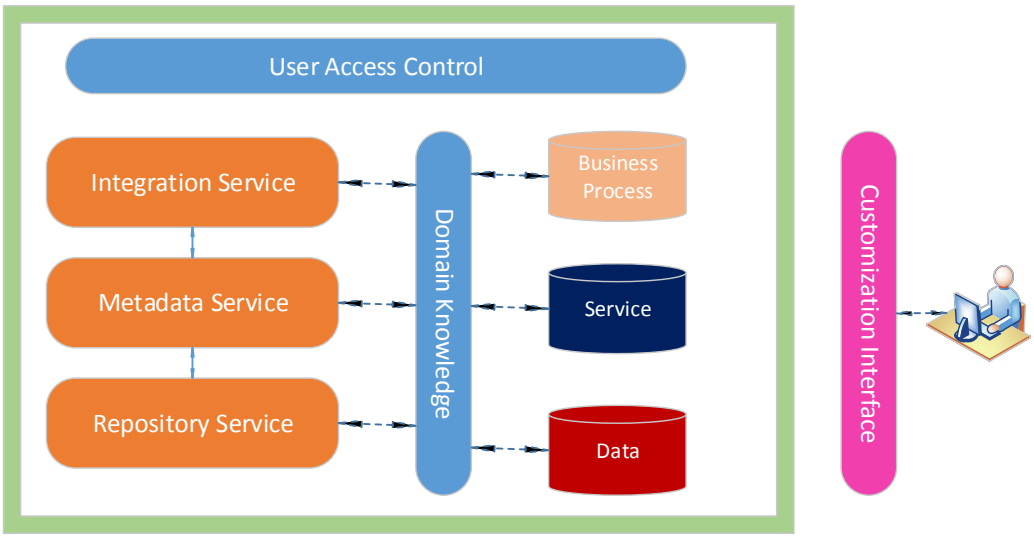

Figure 10: Logical Representation of Variability Model

Subscriber S1, a securities trading company, owns stock portfolio trading on the NYSE decides to manage financial exposure of her portfolio using our service and desire no special requirements as the current default settings supports her business process.

Subscriber S2, a securities trading firm, owns stock portfolio trading on various stock exchanges across the globe decides to manage financial exposure of her portfolio using our service with specific requirements. The framework will support its customization through description points at the service layer and business logic layer.

Subscriber S3, an insurance company, owns portfolio which contains securities from various stock exchanges across the globe, real estate investment and loans decides to manage financial exposure of her portfolio using our service with specific requirements. The framework will support its customization through description points at the data layer, service layer and business logic layer.

A sample customization choices for the three subscribers is shown in figure 11.

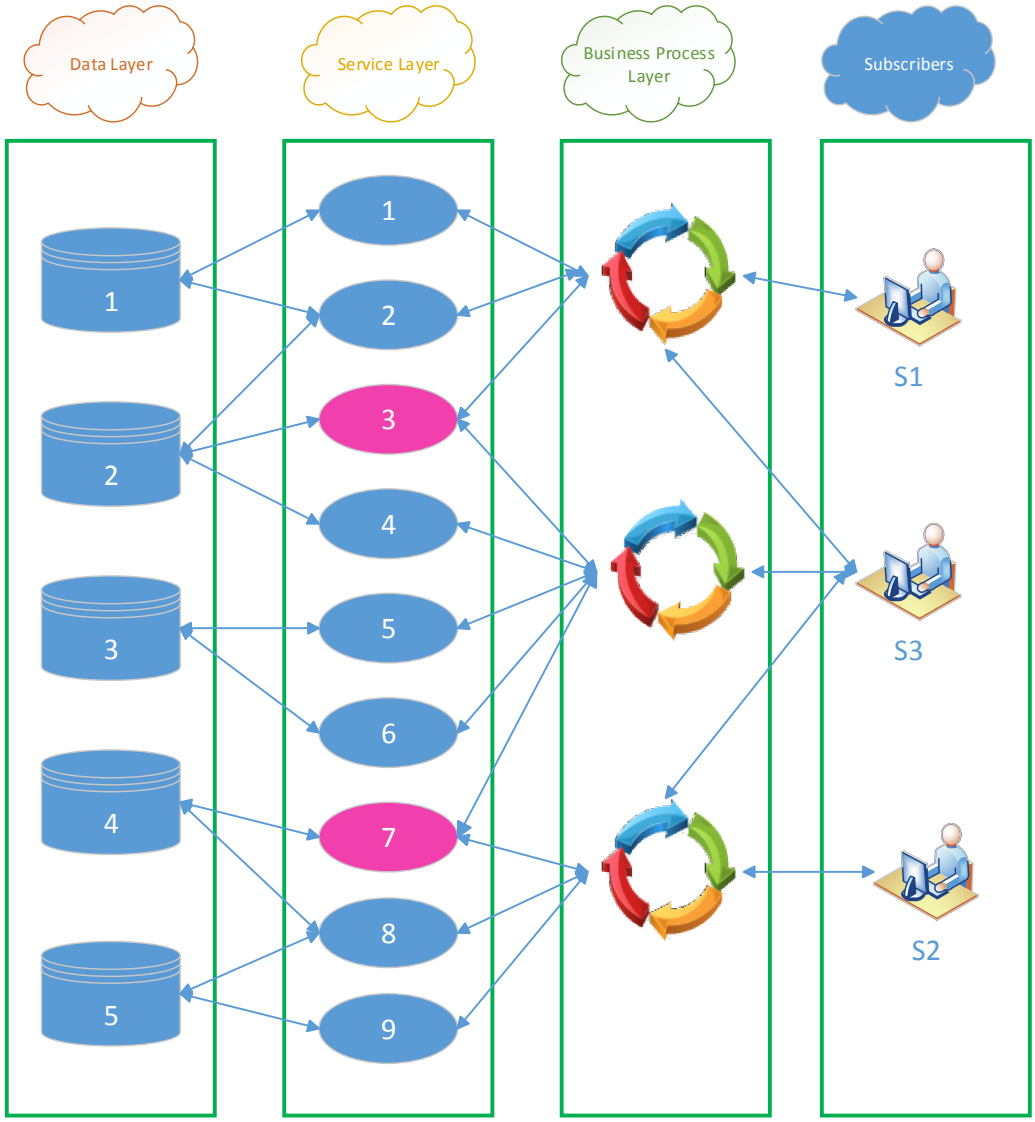

Figure 11: Sample implementation of variability model

3.3.2 Communication Paradigm:

There exist a cross-relationship between the layers, the data layer serves as input into the service layer and also store results from the service layer. Services however makes up the business process workflow and data exchange occurs during business processes i.e. in other to fulfill a particular business process, services applicable to the process are employed during which data is exchanged and (or) transformed.

Figure 12 depicts a typical interaction across layers.

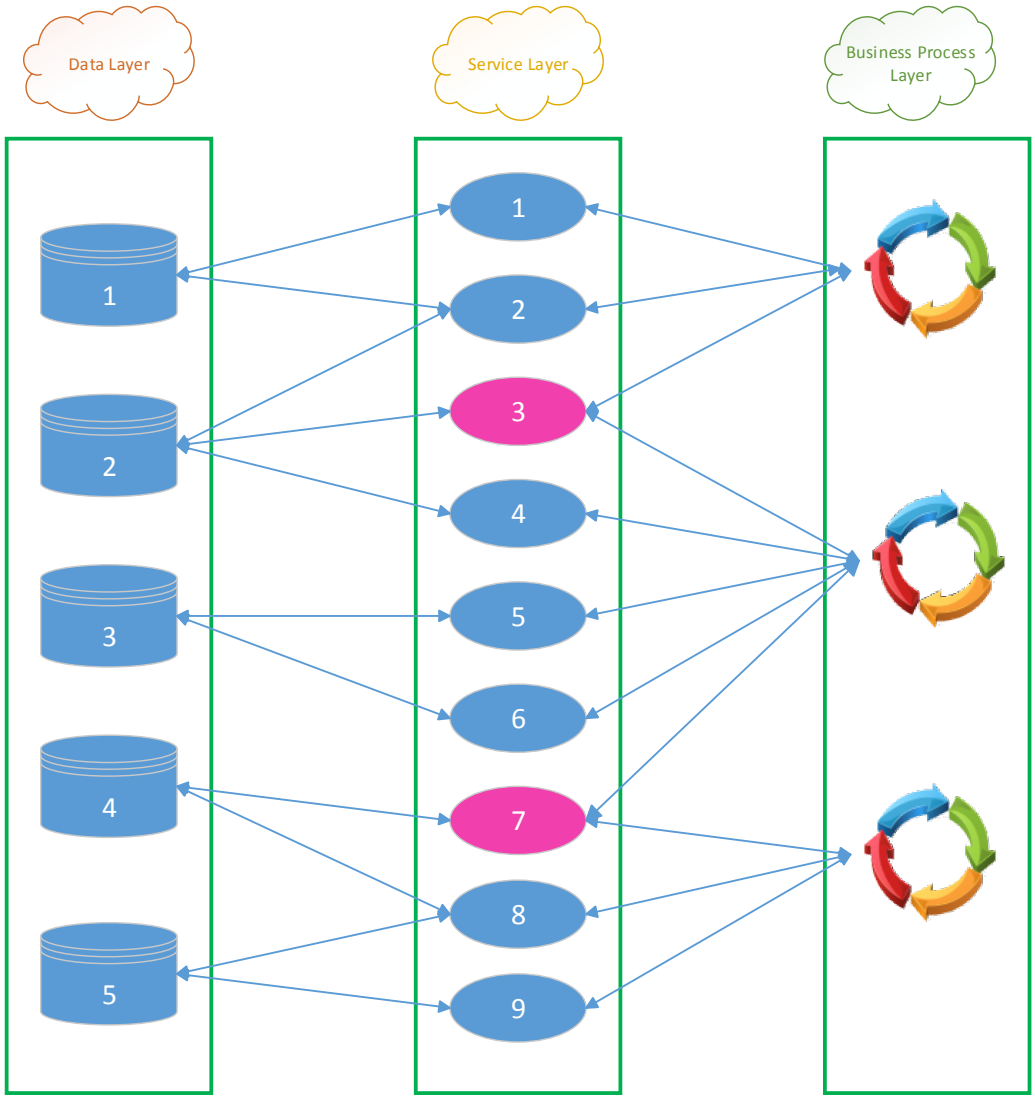

Figure 12: Inter Layer Communication

3.4 Financial Evaluation

With much said about the process variability framework, there is need to consider if such endeavor is economically wise. A typical SaaS vendor who wishes to invest in such project will like to see the profitability of his/her investment.

Economics of SaaS could be sometimes tricky for various reasons such as, investment are mostly dynamic dependent on varying requirements, most of the assets are also not fixed due to multi-tenancy nature of SaaS and as a result evaluating standard

financial metrics will require a bit of creativity and introducing variability into SaaS does not make it less complicated.

Depending on the goal of a SaaS venture, profitability can be evaluated from three perspective. Profitability can be evaluated as per customer profitability (micro economics), overall profitability (Accounting), and profitability per employee [15].

3.4.1 Per-Customer Profitability:

In this approach, economics of a business is evaluated on a single customer level. It involves customer acquisition and monetization. Calculating profitability revolves around two factors which are:

1. Cost of acquiring the customer (CAC)
2. The Lifetime value (LTV) of the customer i.e. how much a customer can be monetized.

Generally, lifetime value of a customer should significantly supersede the cost of acquisition as anything short of that will jeopardize the profitability of the business, in fact, as a rule of thumb for start-ups, LTV should be three times greater than CAC [16].

3.4.2 Overall Profitability

The overall profitability approach is to compare revenue to expenses. Revenue in this contest will be from service subscriptions while expenses will include marketing cost, development cost, maintenance cost and other acquisition cost such as hosting etc.

In general, metrics considered includes monthly rate of revenue (MRR), this considers the average monthly revenue by computing average number of customers thus:

Total number of customers – Total number of unsubscribed customers + Total number of new customers.

This value is then multiplied by average revenue per customer (ARPU) to get the

monthly revenue rate.

On the expense side, metrics considered includes cost of goods and services (COGS) which is cost incurred in providing the service such as cost of setting up servers, software developers wages etc. [19].

Expenses is also considered which includes marketing and sales cost.

An overall profitable venture will have its revenue greater than COGS and expenses together.

3.4.3 Profitability per Employee

This approach look at the factors contributing to profitability on a per employee basis, and uses other performing companies in the industry as a yardstick for evaluating performance companies.

This involves calculating expenses per employee by taking the total of all expenses, not just salaries, and dividing it by the number of employees.

A profitable company will have revenue per employee rise higher than expenses in the long term, however, with a consideration for gross margin ratio i.e. the total sales revenue minus cost of goods sold, divided by the total sales revenue in percentage.

3.5 Financial Viability of the Model

In evaluating the financial viability of our model, we will consider OpenERP S.A., the software vendor of the Odoo Apps (formerly Openerp).

Odoo, formerly known as OpenERP is a suite of open-source business apps written in Python. Its implementation conforms to standard expectations of ERP systems, while providing additional modules beyond the coverage of traditional ERP systems.

The choice for Odoo apps is driven by its modular architecture which as well allows for further customization at the business process level which fits the bill for our research area.

With limited access to information, the scope of our model will be limited to implementation and will not cover development cost because:

a. Odoo is an Open source application released under AGPL license, its code is open to community for reviews and enhancement, hence, development is not centralized and estimating its development cost might be a white elephant project
b. OpenERP S.A. model for turnover solely depends on implementation and support as any interested user can pick up the software and deploy without necessarily using the SaaS version.

3.5.1 Approach:

We will be comparing implementation of Odoo with a singular module on two different cloud platform, compare the cost.

The two chosen platforms are

a. Amazon EC2 compute engine which provides a one year free tier registration
b. Google cloud platform compute engine which provides a $360 free tier 60-days subscription

Odoo consist of various apps in 6 categories listed below:

a. Sales management: These include modules such as CRM, point of sales, quotation builder
b. Business operations: These include modules such as project management, inventory, manufacturing, accounting and purchase
c. Marketing: These include modules such as mass mailing, lead automation, events, survey, forum, live chat

In order to keep things simple and implementation within the free tier provided by

these cloud platform, a single module (Warehouse Application) has been instantiated and metrics on resource usage gathered over time will be used to estimate cost over a period of time.

Using one month usage data gathered from the Amazon EC2 instance, an OpenERP SA partner (BJI) who intends to implement the model for Odoo might wish to evaluate its profitability and if possible forecast a break-even period.

Odoo comes with a variability process model which implies such partner can monetize implementation by service components and customization i.e. the service components that will be used in fulfilling the business process as a result, a three year projection was derived to help management decide on the viability of embarking on such project using the following assumptions:

i. BJI has decided to offer Odoo in three categories, Category 1 offers one module, Category 2 offers two modules and Category 3 offers three modules
ii. BJI has existing clients who are interested in subscribing into the various categories
iii. An average of 20% of the clients require Odoo customized to fit their business needs
iv. BJI envisages a 50% average growth rate in the first four quarters and will not engage in any marketing activities in the first financial year in order to cut down on expenses
v. Churn rate is zero over the first three years i.e. KI keeps all the customers generated from marketing / sales over the first three years.
vi. In order to remain competitive, services are offered at a monthly flat rate of $15 (fifteen US dollars) per user per module and $129 (one hundred and twenty nine US dollars) per module customization as offered by Odoo

vii. All clients are assumed to have a single user instance

viii. Revenue is calculated on a quarterly basis i.e. Quarterly rate of Revenue QRR

ix. Average variable cost (i.e. cost attached to service components per user) will be used for the three default variants which implies revenue for each customer remain the same per variant.

This is important because variability cost will defer for all subscribers depending on the service components used and calculation base on individual revenue becomes more complicated, hence we will be using the overall variability approach.

Data was gathered base on one month single user deployment of Odoo on Amazon EC2 and projected over three years, profitability has been calculated using projected quarterly customer subscription through the 3 year period.

Also the data gathered from AWS has been used in generating infrastructure cost and while expenses were generated based on the marketing strategies discussed.

Details	Usage	Unit
AWS Service Charges		
Data Transfer		
US West (Oregon) Region		
AWS Data Transfer USW2-USE1-AWS-In-Bytes		
$0.00 per GB - US West (Oregon) data transfer from US East (Northern Virginia)	0.000001	
Total:		
AWS Data Transfer USW2-USE1-AWS-Out-Bytes		
$0.000 per GB - data transfer out under the monthly global free tier	0.00000044	
Total:		
AWS Data Transfer USW2-USW1-AWS-In-Bytes		
$0.00 per GB - US West (Oregon) data transfer from US West (Northern California)	0.0000005	
Total:		
AWS Data Transfer USW2-USW1-AWS-Out-Bytes		
$0.000 per GB - data transfer out under the monthly global free tier	0.0000005	
Total:	0.00000244	GB
Bandwidth		
$0.000 per GB - data transfer in per month	0.025	
$0.000 per GB - data transfer out under the monthly global free tier	0.028	
$0.000 per GB - regional data transfer under the monthly global free tier	0.0000002	
Total:	0.0530002	GB
Region Total:		
Elastic Compute Cloud		
US West (Oregon) Region	3 *24 hrs * 30Months days	
Amazon Elastic Compute Cloud running Linux/UNIX		
$0.00 per Linux t2.micro instance-hour (or partial hour) under monthly free tier	228	HRS
Total:		
EBS		
$0.00 per GB-month of General Purpose (SSD) provisioned storage under monthly free tier	2.522	GB-Mo
$0.00 per GB-Month of snapshot data stored under monthly free tier	0.441 GB-Mo	

Figure 13: One Month Snapshot of Amazon EC2 Usage Data

Plan Attributes	2015				2016				2017			
Name	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Compensation												
Software Developers	1	1	1	1	1	2	2	2	2	3	3	3
Developer Comp.	$ (32,000.00)	$ (32,000.00)	$ (32,000.00)	$ (32,000.00)	$ (40,000.00)	$ (40,000.00)	$ (40,000.00)	$ (40,000.00)	$ (40,000.00)	$ (40,000.00)	$ (40,000.00)	$ (40,000.00)
Outsources Marketing					$ (32,000.00)	$ (32,000.00)	$ (20,000.00)	$ (15,000.00)	$ (12,000.00)	$ (12,000.00)	$ (12,000.00)	$ (12,000.00)
Sales people					$ (15,000.00)	$ (15,000.00)	$ (15,000.00)	$ (15,000.00)	$ (15,000.00)	$ (15,000.00)	$ (15,000.00)	$ (15,000.00)
Support Contract Capacity					200	200	200	200	200	200	200	200
Support People					1	1	1	1	2	2	3	3
Support Comp					$ (13,500.00)	$ (13,500.00)	$ (13,500.00)	$ (13,500.00)	$ (13,500.00)	$ (13,500.00)	$ (13,500.00)	$ (13,500.00)
Total Internal Comp	$ (32,000.00)	$ (32,000.00)	$ (32,000.00)	$ (32,000.00)	$ (68,500.00)	$ (108,500.00)	$ (108,500.00)	$ (108,500.00)	$ (122,000.00)	$ (162,000.00)	$ (175,500.00)	$ (175,500.00)
Tax and Benefit Factor	17%	17%	17%	17%	17%	17%	17%	17%	17%	17%	17%	17%
Total Taxes and Benefits	$ (5,440.00)	$ (5,440.00)	$ (5,440.00)	$ (5,440.00)	$ (11,645.00)	$ (18,445.00)	$ (18,445.00)	$ (18,445.00)	$ (20,740.00)	$ (27,540.00)	$ (29,835.00)	$ (29,835.00)
Total Compensation	$ (37,440.00)	$ (37,440.00)	$ (37,440.00)	$ (37,440.00)	$ (112,145.00)	$ (158,945.00)	$ (146,945.00)	$ (141,945.00)	$ (154,740.00)	$ (201,540.00)	$ (217,335.00)	$ (217,335.00)
Other												
Rent					$ (1,500.00)	$ (1,500.00)	$ (1,500.00)	$ (1,500.00)	$ (4,000.00)	$ (4,000.00)	$ (4,000.00)	$ (4,000.00)
Travel & Related					$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)
Website Development		$ (15,000.00)	$ (15,000.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)	$ (1,200.00)
Total Other	$ -	$ (15,000.00)	$ (15,000.00)	$ (1,200.00)	$ (3,900.00)	$ (3,900.00)	$ (3,900.00)	$ (3,900.00)	$ (6,400.00)	$ (6,400.00)	$ (6,400.00)	$ (6,400.00)
Legal Expenses												
Corporate legal	$ (2,000.00)			$ (5,000.00)	$ (5,000.00)							
Patent Expenses		$ (5,000.00)	$ (5,000.00)	$ (5,000.00)								
Total Legal	$ (2,000.00)	$ (5,000.00)	$ (5,000.00)	$ (10,000.00)	$ (5,000.00)	$ -	$ -	$ -	$ -	$ -	$ -	$ -
Product Infrastructure AWS EC2	3 *24 hrs * 30 days											
AWS TIER	m3.medium	m3.medium	m3.medium	m3.medium	m3.medium	m3.xlarge	m3.xlarge	m3.xlarge	m3.xlarge	c3.4xlarge	c3.4xlarge	c3.4xlarge
Compute Engine Cost	$ (0.07)	$ (0.07)	$ (0.07)	$ (0.07)	$ (0.07)	$ (0.28)	$ (0.28)	$ (0.28)	$ (0.28)	$ (0.84)	$ (0.84)	$ (0.84)
Bandwidth Cost	$ (0.12)	$ (0.12)	$ (0.12)	$ (0.12)	$ (0.12)	$ (0.12)	$ (0.12)	$ (0.12)	$ (0.12)	$ (0.12)	$ (0.12)	$ (0.12)
Storage Cost	$ (0.10)	$ (0.10)	$ (0.10)	$ (0.10)	$ (0.10)	$ (0.10)	$ (0.10)	$ (0.10)	$ (0.10)	$ (0.10)	$ (0.10)	$ (0.10)
Compute Engine Usage (hrs)	2160.00	2160.00	2160.00	2160.00	2160.00	2160.00	2160.00	2160.00	2160.00	2160.00	2160.00	2160.00
Bandwidth Usage (GB)	525.00	780.00	1170.00	1755.00	5495.70	11522.97	18401.94	26259.71	35249.07	45559.94	57440.33	71235.98
Storage Usage (GB-Mo)	54.50	80.00	119.00	177.50	551.57	1154.30	1842.19	2627.97	3526.91	4557.99	5746.03	7125.60
Other Variable Cost Per User					$ (1.75)	$ (1.75)	$ (1.75)	$ (1.75)	$ (1.75)	$ (1.75)	$ (1.75)	$ (1.75)
Total Other Variable Costs					$ (641.17)	$ (1,344.35)	$ (2,146.89)	$ (3,063.63)	$ (4,112.39)	$ (5,315.33)	$ (6,701.37)	$ (8,310.86)
Total Server Costs	$ (219.65)	$ (252.80)	$ (303.50)	$ (379.55)	$ (865.84)	$ (2,102.99)	$ (2,997.25)	$ (4,018.76)	$ (5,187.38)	$ (7,737.39)	$ (9,281.84)	$ (11,075.28)
Total AWS Bill	$ (219.65)	$ (252.80)	$ (303.50)	$ (379.55)	$ (1,507.01)	$ (3,447.33)	$ (5,144.15)	$ (7,082.40)	$ (9,299.77)	$ (13,052.72)	$ (15,983.22)	$ (19,386.14)
Expenses Summary												
Total Expenses	$ (39,659.65)	$ (57,692.80)	$ (57,743.50)	$ (49,019.55)	$ (122,552.01)	$ (166,292.33)	$ (155,989.15)	$ (152,927.40)	$ (170,439.77)	$ (220,992.72)	$ (239,718.22)	$ (243,121.14)
Cumulative Expenses	$ (39,659.65)	$ (97,352.45)	$ (155,095.95)	$ (204,115.50)	$ (326,667.51)	$ (492,959.84)	$ (648,948.98)	$ (801,876.38)	$ (972,316.15)	$ (1,193,308.87)	$ (1,433,027.08)	$ (1,676,148.23)

Figure 14: Three-Year Projection of Expenses

Plan Attributes	2015				2016				2017			
Name	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Marketing Driven Lead Generation												
Growth rate in Inbound Links						100%	100%	100%	100%	100%	100%	100%
Number of Inbound Links					50	100	200	400	800	1,600	3,200	6,400
Click Through from Inbounds						15%	15%	15%	15%	15%	15%	15%
View from Inbound Links					0	15	30	60	120	240	480	960
Marketing Driven Visitors					10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
Total Page Views					10,000	10,015	10,030	10,060	10,120	10,240	10,480	10,960
MD Percent Visitors Trying Product					10%	10%	10%	10%	10%	10%	10%	10%
MD Number of Free Trials					1,000	1,002	1,003	1,006	1,012	1,024	1,048	1,096
MD Percent Conversion to Plan A					20%	20%	20%	20%	20%	20%	20%	20%
MD Percent Conversion to Plan B					10%	10%	10%	10%	10%	10%	10%	10%
MD Percent Conversion to Plan C					5%	5%	5%	5%	5%	5%	5%	5%
MD Total Conversion Percentage					35%	35%	35%	35%	35%	35%	35%	35%
Social Driven Lead Generation												
Social Factor					20%	20%	20%	20%	20%	20%	20%	20%
Social Driven Visitors					23	73	154	245	350	470	607	766
SD Percent Visitors Trying Product					40%	40%	40%	40%	40%	40%	40%	40%
SD Number of Free Trials					9	29	61	98	140	188	243	306
SD Conversion Advantage					2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
Customer Acquisition Analysis												
New MD Plan A Subscribers					200	200	201	201	202	205	210	219
New MD Plan B Subscribers					100	100	100	101	101	102	105	110
New MD Plan C Subscribers					50	50	50	50	51	51	52	55
New SD Plan A Subscribers					9	29	61	98	140	188	243	306
New SD Plan B Subscribers					5	15	31	49	70	94	121	153
New SD Plan C Subscribers					2	7	15	25	35	47	61	77
Total New from Marketing					350	351	351	352	354	358	367	384
Total New from Social					16	51	108	172	245	329	425	536
Total New Customers					366	402	459	524	599	687	792	920
Customer Counts												
Number of Plan A Customers	20	30	45	67	209	439	701	1,000	1,343	1,736	2,188	2,714
Number of Plan B Customers	10	15	22	33	105	219	351	500	671	868	1,094	1,357
Number of Plan C Customers	5	7	10	15	52	110	175	250	336	434	547	678
Total Number of Customers	35	52	78	117	366	768	1,227	1,751	2,350	3,037	3,829	4,749
Pricing												
Plan A (1 Module) Price	$ 45.00	$ 45.00	$ 45.00	$ 45.00	$ 45.00	$ 45.00	$ 45.00	$ 45.00	$ 45.00	$ 45.00	$ 75.00	$ 75.00
Plan B (2 Modules) Price	$ 90.00	$ 90.00	$ 90.00	$ 90.00	$ 90.00	$ 90.00	$ 90.00	$ 90.00	$ 90.00	$ 90.00	$ 180.00	$ 180.00
Plan C (3 Modules) Price	$ 135.00	$ 135.00	$ 135.00	$ 135.00	$ 135.00	$ 135.00	$ 135.00	$ 135.00	$ 135.00	$ 135.00	$ 450.00	$ 450.00
Plan A Average Variability Cost	$ 387.00	$ 387.00	$ 387.00	$ 387.00	$ 387.00	$ 387.00	$ 387.00	$ 387.00	$ 387.00	$ 387.00	$ 387.00	$ 387.00
Plan B Average Variability Cost	$ 774.00	$ 774.00	$ 774.00	$ 774.00	$ 774.00	$ 774.00	$ 774.00	$ 774.00	$ 774.00	$ 774.00	$ 774.00	$ 774.00
Plan C Average Variability Cost	$ 1,161.00	$ 1,161.00	$ 1,161.00	$ 1,161.00	$ 1,161.00	$ 1,161.00	$ 1,161.00	$ 1,161.00	$ 1,161.00	$ 1,161.00	$ 1,161.00	$ 1,161.00
Revenue Distribution												
Revenue From Plan A	$ 900.00	$ 1,350.00	$ 2,025.00	$ 3,015.00	$ 9,421.20	$ 19,753.67	$ 31,546.18	$ 45,016.65	$ 60,426.98	$ 78,102.75	$ 164,115.23	$ 203,531.37
Revenue From Plan B	$ 900.00	$ 1,350.00	$ 1,980.00	$ 2,970.00	$ 9,421.20	$ 19,753.67	$ 31,546.18	$ 45,016.65	$ 60,426.98	$ 78,102.75	$ 196,938.28	$ 244,237.64
Revenue From Plan C	$ 675.00	$ 945.00	$ 1,350.00	$ 2,025.00	$ 7,065.90	$ 14,815.25	$ 23,659.64	$ 33,762.49	$ 45,320.23	$ 58,577.07	$ 246,172.85	$ 305,297.05
Variations Revenue From Plan A	$ 1,548.00	$ 2,322.00	$ 3,483.00	$ 5,185.80	$ 16,204.46	$ 33,976.31	$ 54,259.43	$ 77,428.63	$ 103,934.40	$ 134,336.74	$ 169,366.92	$ 210,044.37
Variations Revenue From Plan B	$ 1,548.00	$ 2,322.00	$ 3,405.60	$ 5,108.40	$ 16,204.46	$ 33,976.31	$ 54,259.43	$ 77,428.63	$ 103,934.40	$ 134,336.74	$ 169,366.92	$ 210,044.37
Variations Revenue From Plan C	$ 1,161.00	$ 1,625.40	$ 2,322.00	$ 3,483.00	$ 12,153.35	$ 25,482.23	$ 40,694.57	$ 58,071.47	$ 77,950.80	$ 100,752.55	$ 127,025.19	$ 157,533.28
Total Variability Reveynue	$ 4,257.00	$ 6,269.40	$ 9,210.60	$ 13,777.20	$ 44,562.28	$ 93,434.85	$ 149,213.44	$ 212,928.74	$ 285,819.60	$ 369,426.03	$ 465,759.03	$ 577,622.02
Total Revenue	$ 6,732.00	$ 9,914.40	$ 14,565.60	$ 21,787.20	$ 70,470.58	$ 147,757.44	$ 235,965.44	$ 336,724.52	$ 451,993.79	$ 584,208.60	$ 1,072,985.40	$ 1,330,688.07
Cumulative Revenue	$ 6,732.00	$ 9,914.40	$ 14,565.60	$ 21,787.20	$ 70,470.58	$ 218,228.01	$ 454,193.45	$ 790,917.97	$ 1,242,911.76	$ 1,827,120.36	$ 2,900,105.76	$ 4,230,793.83

Figure 15: Three-Year Quarterly Projection of Revenue

Plan Attributes	2015				2016				2017			
Name	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Marketing Driven Lead Generation												
Quarterly Revenue	$ 6,732.00	$ 9,914.40	$ 14,565.60	$ 21,787.20	$ 70,470.58	$ 147,757.44	$ 235,965.44	$ 336,724.52	$ 451,993.79	$ 584,208.60	$ 1,072,985.40	$ 1,330,688.07
Quarterly Expenses	$ 39,659.65	$ 57,692.80	$ 57,743.50	$ 49,019.55	$ 122,552.01	$ 166,292.33	$ 155,989.15	$ 152,927.40	$ 170,439.77	$ 220,992.72	$ 239,718.22	$ 243,121.14
Quarterly Profit /Loss	$(32,927.65)	$ (47,778.40)	$ (43,177.90)	$ (27,232.35)	$ (52,081.43)	$ (18,534.90)	79976.29276	$ 183,797.12	$ 281,554.02	$ 363,215.88	$ 833,267.18	$ 1,087,566.93
Total Number of Users	35	52	78	117	366	768	1227	1751	2350	3037	3829	4749
Net Cumulative Revenue	$ 6,732.00	$ 16,646.40	$ 31,212.00	$ 52,999.20	$ 123,469.78	$ 271,227.21	$ 507,192.65	$ 843,917.17	$ 1,295,910.96	$ 1,880,119.56	$ 2,953,104.96	$ 4,283,793.03
Net Cumulative Expenses	$ 39,659.65	$ 97,352.45	$ 155,095.95	$ 204,115.50	$ 326,667.51	$ 492,959.84	$ 648,948.98	$ 801,876.38	$ 972,316.15	$ 1,193,308.87	$ 1,433,027.08	$ 1,676,148.23
Cumulative Profit/Loss	$(32,927.65)	$ (80,706.05)	$(123,883.95)	$(151,116.30)	$(203,197.73)	$(221,732.63)	$(141,756.33)	42040.78981	$ 323,594.81	$ 686,810.69	$ 1,520,077.87	$ 2,607,644.80

Figure 16: Three-Year Projection Highlighting profit point and Break Even Point

Plan Attributes	2015				2016				2017			
Name	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Marketing Driven Lead Generation												
Quarterly Revenue	$ 2,475.00	$ 3,645.00	$ 5,355.00	$ 8,010.00	$ 25,908.30	$ 54,322.59	$ 86,752.00	$ 123,795.78	$ 166,174.19	$ 214,782.57	$ 607,226.36	$ 753,066.05
Quarterly Expenses	$ 39,659.65	$ 57,692.80	$ 57,743.50	$ 49,019.55	$ 122,552.01	$ 119,492.33	$ 109,189.15	$ 106,127.40	$ 123,639.77	$ 127,392.72	$ 146,118.22	$ 149,521.14
Quarterly Profit /Loss	$ (37,184.65)	$ (54,047.80)	$ (52,388.50)	$ (41,009.55)	$ (96,643.71)	$ (65,169.75)	$ (22,437.15)	$ 17,668.38	$ 42,534.42	$ 87,389.86	$ 461,108.15	$ 603,544.91
Total Number of Users	35	52	78	117	366	768	1227	1751	2350	3037	3829	4749
Net Cumulative Revenue	$ 2,475.00	$ 6,120.00	$ 11,475.00	$ 19,485.00	$ 45,393.30	$ 99,715.89	$ 186,467.89	$ 310,263.67	$ 476,437.85	$ 691,220.43	$ 1,298,446.79	$ 2,051,512.84
Net Cumulative Expenses	$ 39,659.65	$ 97,352.45	$ 155,095.95	$ 204,115.50	$ 326,667.51	$ 446,159.84	$ 555,348.98	$ 661,476.38	$ 785,116.15	$ 912,508.87	$ 1,058,627.08	$ 1,208,148.23
Cumulative Profit/Loss	$ (37,184.65)	$ (91,232.45)	$ (143,620.95)	$ (184,630.50)	$ (281,274.21)	$ (346,443.95)	$ (368,881.10)	$ (351,212.71)	$ (308,678.30)	$ (221,288.44)	$ 239,819.71	$ 843,364.62

Figure 17: Three-Year Projection Highlighting profit point and Break Even Point with No Customization

CHAPTER 4 – DISCUSSION AND FINDINGS

4.1 Introduction

This chapter discusses the findings of the model discussed and its financial implication. It is expedient to say that there were assumptions in order to arrive at theses derivations, there were assumptions both in the model and financial calculation.

4.2 The Model:

The model shows that we can introduce variability into the process layer thereby generating distinct product line base on varying requirements dynamically. Separating the layers (Data, Services and Business Logic) and introducing variable descriptors is not a trivia process as the intricacies of software architecture creeps in.

The management of variability in software product lines exceeds the definition of variations, traceability and configurations, various assumptions about the variability and related models are made by the stakeholders all over the product line, hence, the rationale behind its specification should be adequately considered [19].

As a result this research involves some assumptions which should be explicitly considered in the implementation of this model, this includes:

1. SaaS applications implemented using this model will be considered not only on the service and the business process level but at an architectural level i.e. how it communicates and integrates with other third party services have been put into consideration
2. Such application should put into consideration, how it connect third party services with subscribers / customers with regards to constraints. For instance, a customer who wish to use a features offered by a particular third-party service should be able to do so within the constraint of his/her requirement.
3. As such application evolve over time through changes, its adaptive nature can

be retained and relevant product line applications can still be generated via composition of available services

4. Domain knowledge for all product line domain is well known and readily available before and during development.
5. Validation of runtime use-case and its selection allows for non-deterministic Choices i.e. it allow for expression of different use-cases and also its runtime execution.
6. Such application should be flexible enough to acquire new or additional services for enabling new features and capabilities.
7. Implementation of dependencies have been highly considered i.e. how the selection of a certain variant impacts the value the entire system property such as performance, availability, reliability and other quality attributes as availability of possible numerous variance could make precise determination of dependencies a daunting task.

As a matter of fact, variations should be defined as a tuple of various attribute such as time, class, scope, level, option and role which can be defined as follow:

Time: Design time, runtime or configuration time

Class: Quality or Functional

Scope: Global or local

Level: Service, Business Logic or Architectural

Option: Mandatory or optional

Role: user, administrator etc.

A combination of these attributes can then be used to define variations within the application such that dependencies, access, and options are explicitly defined.

4.3 The Financial Implications

Using the case study of Odoo apps, and projections we can derive various interpretation depending on the SaaS vendor's goal.

For instance, in the case of a SaaS vendors whose goal is to break even at the end of third quarter can adjust various investment options such as cutting down on extent of customization thereby reducing variable expense.

The data and financial model used was adopted from Galvao et al [20] and modified to suit our purpose. We are interested in the economically viability of this project and will be considering time to profit and break even period.

Figure 18 shows a break-even analysis of the proposed investment, we can notice that investment using the variability model broke even at the fourth quarter of the second year.

Using the costing model as defined by BJI above, it is evident that customization generates more revenue than standard description, we can conclude that the availability of variability gives user more options which when used increases the average per-customer revenue. While this is true, additional expenses was incurred in providing the services which could have come at a higher cost than the specified value.

In general, to have a holistic view of the break even analysis, the investor organizes his/her expenses into fixed and variable and adjust them until it aligns with the desired goal, for instance, an investor can decide to introduce more variations that he believes will be a source of revenue at a lower variable expense and reduce his fixed expenses or vice versa.

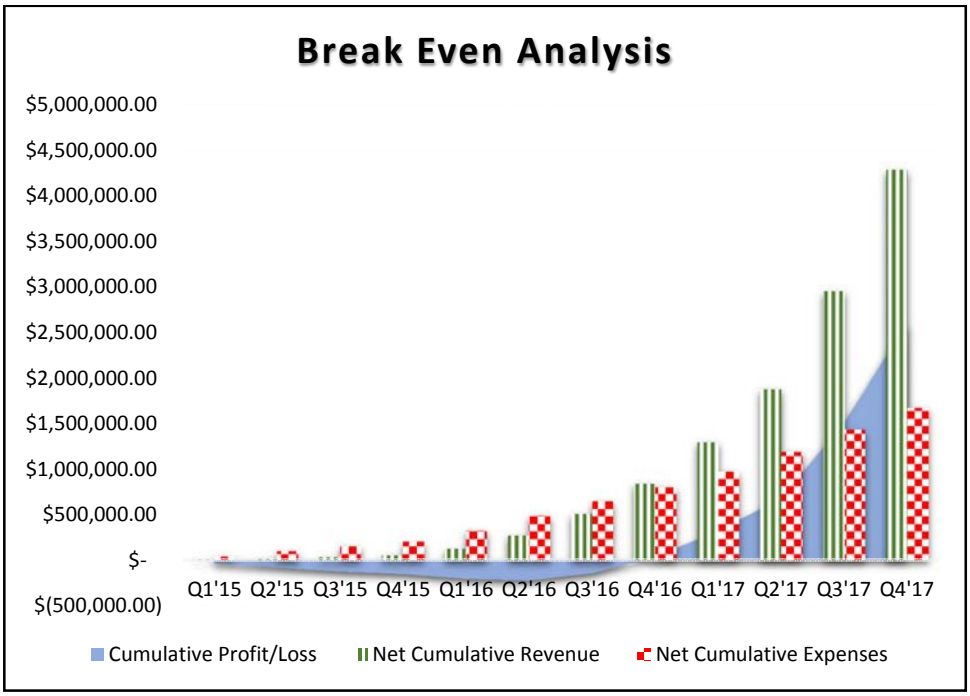

Figure 18: Break-Even Analysis

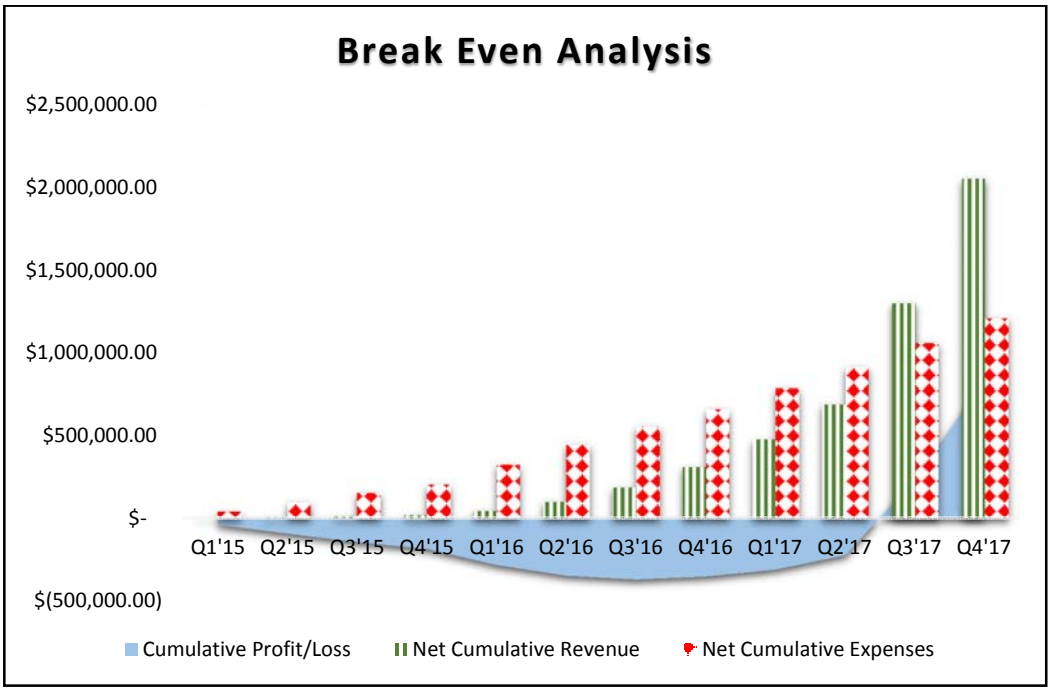

Figure 19: Break Even Analysis with No Customization

Figure 20 shows a profit / loss analysis of the proposed investment, while it is evident the time to profit and break-even period are relatively close, this can be accounted for

by the injection of more revenue due to customization and in a typical SaaS deployment, such might not be achievable.

Although the values in this research takes into consideration some assumptions as listed earlier by BJI, this can be subjective to a reverse derivation in a true use-case scenario, however, the crucial point worth stating is that introduction of variability into SaaS model has significant financial implication which should be considered during implementation as this could determine the profitability of the investment of otherwise.

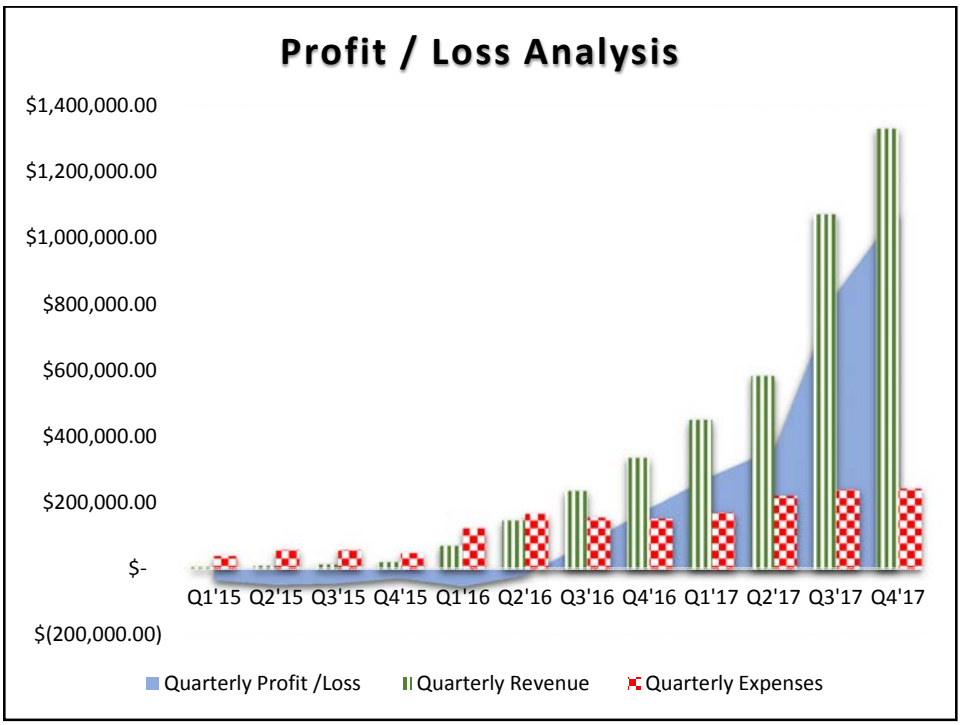

Figure 20: Profit / Loss Analysis

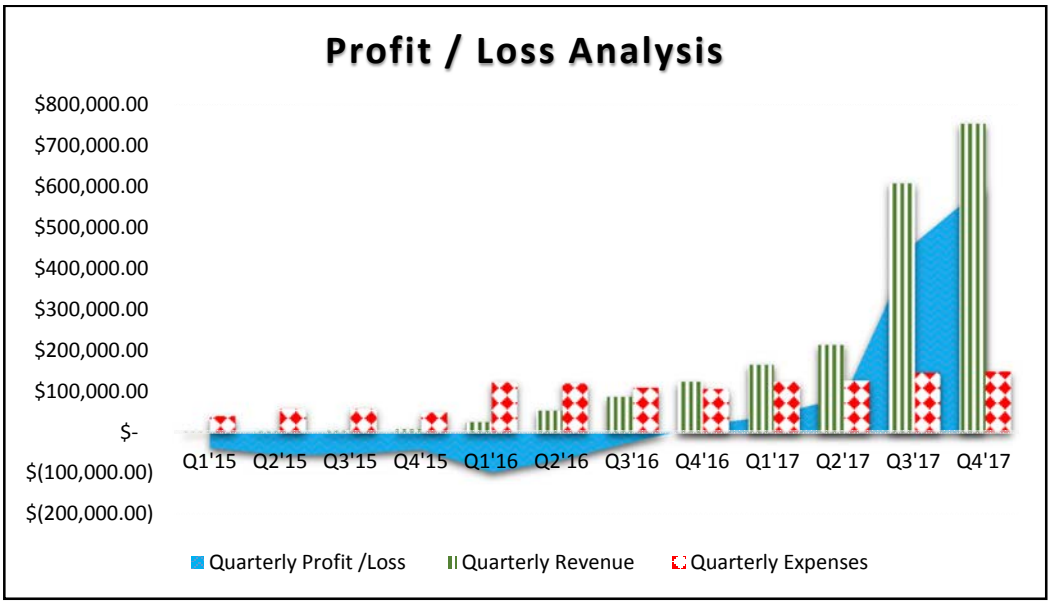

Figure 21: Profit / Loss Analysis with No Customization

CHAPTER 5 – CONCLUSION

5.1 Introduction

This chapter presents a comprehensive summary and conclusion of the research findings, research limitations and opportunity for further research.

5.2 Discussion

The purpose of this study was to design a model for examining the possibility of process level variation for a perceived Software as a Service (SaaS) solutions and predicting its corresponding financial exposure.

We have been able to introduce a multi layered variability model for introducing process variability by defining variation points using domain knowledge at the service layer and business process layer.

The introduction of variability spans beyond defining variation points and binding at runtime as this has a ripple effect on the entire system architecture as product line realization are done at runtime, hence, some design time assumptions might not hold as expected.

In order to avoid such scenario as much as possible, it will be helpful to explicitly define such assumptions within the software, therefore the variability definition carries such assumption and the system is aware at runtime.

We explored a possible tuple for defining such variability specifying attributes such as

Time: Design time, runtime or configuration time

Class: Quality or Functional

Scope: Global or local

Level: Service, Business Logic or Architectural

Option: Mandatory or optional

Role: user, administrator etc.

Which will help eliminate some assumptions while binding.

We also subjected a use-case scenario to a financial model to determine the profitability of such project, we carried out a break-even analysis as well.

It was discovered that the use of process variation via customization has significant financial impact on the time to profit and break-even point. In our case study, it has a positive impact on the profitability, however, parameters can be adjusted and the impact could be positive or negative depending on the company's goal.

One major limitation is the inability to get a real life case study within the time frame of this research as a real life case study will involve software engineering process for a Vertical SaaS solution such as odoo, monitoring and collating software metrics and financial metrics for a number of such application, and conducting analysis on historical metrics collated over time.

Although, the analysis carried out in this paper suffice for the purpose from implementation point of view, but this limitation deprives us the opportunity to explore other factors such as development cost which could have huge impact on the profit / loss and break-even analysis. In an ideal developmental situation, a software vendor will as well want to consider these factors.

5.3 Future Works

There are areas for further improvement and amendment on the model. One potential addition would be the use of data mining techniques to predict variation base on early user input i.e. A SaaS application that can propose a variability process to a new subscriber using the selection of the first few parameters to predict his needs based on patters from historical user specification.

While OpenERP SA, has done a great job in deploying a variability process SaaS application, there are little evidence that the application supports a full multi-tenancy

architecture which makes it fall short of a true SaaS application.

It will be interesting to see a commercial full SaaS application deployed with variability process descriptors in the future as this will enable us explore its real financial implication.

References

[1] Mietzner, R, and F Leymann. "Generation of BPEL Customization Processes for SaaS Applications from Variability Descriptors." 2008 IEEE International Conference on Services Computing, 2 (2008): 359-366.

[2] Salih, N. K, and Tianyi Zang. "Variable Service Process by Feature Meta-model for SaaS Application." 2012 International Conference on Green and Ubiquitous Technology, (2012): 102-105.

[3] Schroeter, Julia, Sebastian Cech, Sebastian Götz, Claas Wilke, and Uwe Aßmann. "Towards Modeling a Variable Architecture for Multi-tenant SaaS-applications." Proceedings of the Sixth International Workshop on Variability Modeling of Software-intensive Systems, (2012): 111-120

[4] Turnova, E, A Tusanova, and J Paralic. "Learning Support Tool for SMEs to Measure Their Cloud Investments." 2013 IEEE 11th International Conference on Emerging ELearning Technologies and Applications (ICETA), (2013): 393-398.

[5] West, Richard and Stephen L. Daigle. "EDUCAUSE Center for Applied Research Bulletin." Total Cost of Ownership: A Strategic Tool for ERP Planning and Implementation 6 January 2004.

[6] Harper, David. "An Introduction to Value at Risk (VAR)" 27 2010 May.

[7] McClure, Ben. "The Capital Asset Pricing Model: An Overview." 24 November 2010.

[8] Andrew, Beattie. "A Guide to Calculating Return on Investment." 14 October 2010.

[9] Ritobaan, Roy. "Horizontal vs. Vertical: SaaS Economics and Software Design" 10 April 2013.

[10] Boris Wertz "Is The Vertical Approach To Enterprise Software Enough To Help

You Win The Market?" 23 March 2013.

[11] David, Jackson. "The question of vertical vs. horizontal SaaS." 20 February 2014.

[12] Joel, York. "SaaS Model Economics 101, Competitive Advantage in SaaS." 2009.

[13] Marco, Sinnema and Sybren, Deelstra "Classifying variability modeling techniques" July 2007.

[14] http://en.wikipedia.org/wiki/Business_Process_Execution_Language

[15] Frederick, Chong and Gianpaolo, Carraro. "Architecture Strategies for Catching the Long Tail" April 2006.

[16] David, Skok. "SaaS Metrics – A Guide to Measuring and Improving What Matters." February 17, 2010.

[17] Joel, York. "SaaS Metrics Guide to SaaS Financial Performance." September 2010.

[18] David, Skok. "SaaS Metrics 2.0 – A Guide to Measuring and Improving What Matters." January 16, 2013.

[19] http://www.investopedia.com/terms/c/cogs.asp.

[20] Galvao Lourenco da Silva, I. and van den Broek, P.M. and Akşit, M. "A model for variability design rationale in SPL." August 23, 2010.

[21] Clearfactr "Financial Modeling: How to model a SaaS company with ClearFactr." February 14, 2014.