

Product Management in Software as a Service

by

Karthikeyan Rajasekharan

B.A.Sc. (Honors) in Computer Engineering
Nanyang Technological University, Singapore, 1997

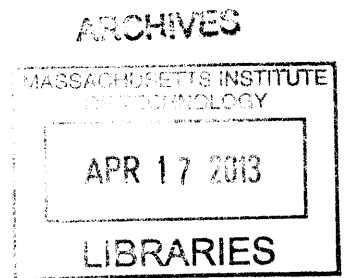
Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

Feb 2012
[June 2012]
© 2012 Karthikeyan Rajasekharan
All rights reserved



The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created.

Signature of
Author _____

Karthikeyan Rajasekharan
Fellow
System Design and Management Program

Certified
By _____

Michael Davies
Thesis Supervisor
Senior Lecturer, Engineering Systems Division

Accepted
By _____

Patrick Hale
Director
System Design and Management Program

Product Management in Software as a Service

by

Karthikeyan Rajasekharan

Submitted to the System Design and Management Program in
February 2012 in Partial fulfillment of the requirements for the Degree of
Master of Science in Engineering and Management

ABSTRACT

Software product management within Software as a Service is key domain of interest given the recent advances in Cloud Computing. This thesis explores the product management challenges within this domain. It makes a contribution to understanding how factors such as architecture, customer experience measurement, customer driven feature prioritization, editorial action of product manager and development process affect product success in the SaaS domain. SaaS product management dictates different priorities from traditional software and product managers and organizations must adapt to these changes to innovate.

Thesis Supervisor: Michael Davies

Title: Senior Lecturer, Engineering Systems Division

Acknowledgements

I would like to express my deepest gratitude to my advisor, Michael Davies for his support and encouragement. I have learnt a lot under his guidance. His probing questions made writing this thesis, a fruitful learning experience for me.

My heartfelt thanks go to Aunty Lee Fei Chen and Uncle Tan Kong Khoon for their unwavering support and encouragement. They are wonderful mentors and I am truly blessed for their interest in me.

I thank my wife, Gayathri, for her patience and love when I disappeared for long hours working on the thesis. I thank my parents (R.Usha and B.M. Rajasekharan) for all their support throughout the MIT experience.

I thank my mentor Andy Wilkinson, who was very supportive of my efforts to pursue my education at MIT. He provided me with both the opportunity and the guidance needed to handle technology diffusion challenges.

A huge shout-out to my teammates Shirish Nilekar, Sunny Cheung and Ritesh Shukla who worked with me on the Ask.com case study. I would like to thank Megan Reitan from Ask.com for providing the data required to make the study possible.

Table of Contents

1. Introduction	7
2. Innovation in the Software as a Service Industry	10
Background	10
Evolution of the Software Business into a Services Business.....	10
Is SaaS primarily a business model Innovation?.....	12
Product and Diffusion Process Innovations	13
Pure play SaaS Vendors versus Traditional Software Vendors	14
Product Portfolio Management in SaaS.....	15
Becoming a platform – Addressing new markets.....	15
Impact on other traditional IT Services Industry.....	16
Will all Software become Services?.....	16
Chapter Conclusion	17
3. Architecture within Software as a Service	18
Classic Model View Controller Architecture.....	18
High Level SaaS Application Architecture	19
Unique Architectural Challenges of SaaS	21
Role of Multi-tenancy	22
Impact of Architecture on Product Management / Product Evolution.....	26
Feature Development Cost: Cost Per Tenant vs. Cost Per Feature	26
Chapter Conclusion	29
4. Role of the Development Process in Software as a Service	30
Changing Nature of Development in SaaS.....	30
Agile Methodologies in SaaS.....	31
Case Study of SCRUM in Ask.com Product Development	32
Research Approach	33
Findings	34
Impact on Fraction Correct and Complete	34
Impact on Morale	34
Impact on Requirements Risk.....	35
Impact on Quality	35
Experience effects in SCRUM	36
Using System Dynamics Model to study SCRUM.....	37

Chapter Conclusion	42
5. Product Feature Ideation and Prioritization through Co-Creation	44
Traditional Product Development Process.....	44
Collaborating with Customers	45
Case Background.....	47
Source of Data:	47
Pre-Analysis Data Preparation	47
Data issues and Resolutions	49
Exploratory Data Analysis.....	49
Analysis of Dataset	52
Affinity Analysis to validate the tagging taxonomy	52
Classification using data mining techniques.....	56
Chapter Conclusion	60
Key takeaways.....	60
6. Using Customer Experience Data in Software as a Service	61
Product Usage data in SaaS.....	61
Customer Usage/Experience as a measure of feature performance.....	63
Defining Customer Experience	63
Instrumenting For and Measuring Customer Experience.....	64
Curse of Dimensionality.....	65
Chapter Conclusions	66
7. Conclusion	67
Bibliography	69
Appendix.....	72
Survey Questionnaire.....	72
Data Set Preparation.....	74

Table of Figures

Figure 1: Service, Product and Process evolution [Source: Cusumano, Kahl, Suarez 2006]	11
Figure 2: MVC Architecture [Source: Krasner, Pope 1998]	18
Figure 3 : SaaS high level architecture [Source: Chong , Carraro 2006]	20
Figure 4: Comparison of degrees of Multi-tenancy	25
Figure 5: Cost per Feature vs. Cost Per Tenant [Source: Carraro 2008]	27
Figure 6: Speed of feature development vs. Multi-tenancy	28
Figure 7: Fraction Correct and Complete	34
Figure 8: Impact of SCRUM on morale	34
Figure 9 : Impact of SCRUM on Requirements Risk	35
Figure 10 : Experience Effects of SCRUM	36
Figure 11: SD Model for SCRUM vs. Waterfall	37
Figure 12 : Zoomed view of Iterations in SD Model	38
Figure 13: Zoomed view of Task Completion in SD Model	39
Figure 14: Zoomed View of Defects in SD Model	39
Figure 15 : Non linear relationship of requirement uncertainty	40
Figure 16: Fraction Correct and Complete Table Function	41
Figure 17: Simulation Results across various iteration counts	41
Figure 18: Product Development Stage Gate	44
Figure 19 : Plot of votes per Idea	50
Figure 20: Plot of votes with Implementation	50
Figure 21: Box Plots of votes and implementation	51
Figure 22: Scatter Plot of Comments vs. Votes	52
Figure 23: Lift Charts of different data mining techniques	59
Figure 24: Causal loop of feature prioritization in SaaS	67
Figure 25: A holistic view of product management in SaaS [Causal Loop Diagram]	68

1. Introduction

"I think it [Cloud Computing] is one of the foundations of the next generation of computing" Tim O Reilly

Delivering software as a service is trend that has been fueled by advances in software technology (such as virtualization, multi-tenancy), lower costs of hardware (decreasing storage costs, compute cycle costs) and better connectivity (cheaper and ubiquitous internet access). In this model software vendors create and distribute applications on a pay as you go basis via the Internet. This is replacing more traditional software and services licensing models in the software industry.

This new paradigm raises new questions about the role of product management. Product delivered in an on-demand basis place different emphasis on the feature completeness, market reach tradeoffs. For the Service provider, the SaaS model raises new questions about development resource allocation to optimize application efficiency, performance, usage and diffusion. While this model provides rich playing field for analyzing customer requirements, preferences and feature usage, it presents interesting challenges in interpreting the new available data.

Is the lead user always right? What is the balance between feature iterations and platform evolutions? How does the software product manager glean unstated latent needs from analyzing the usage data? What are the implications of architecture on the speed of experimentation and delivery? What processes are best suited to be used in this rapidly changing environment?

Based on the analysis done through the course of this thesis, a few clear trends emerge.

Firstly, innovation in the SaaS industry is disruptive to software that involves high degrees of user interaction. Software such as workflow management, database driven programs benefit from the TCO economics that SaaS provides.

Secondly, Architectural decisions are dependent on the total addressable market. Multi-tenant architectures reduce per tenant costs but increase feature roll out and development costs. There are varying degrees of multi-tenancy and it has to be chosen with appropriate consideration to the total addressable market size that the SaaS provider is targeting.

Thirdly, Agile process methodologies work best for Software as a Service offering as the underlying risk that is being managed is requirements risk (not understanding what the end user needs) and not necessarily technical risk. Platform development doesn't lend itself to such an agile model.

Fourthly, User co-creation is a good approach to increase relevancy of new features.

However, the product manager must be conscious of the long tailed distribution of the idea generation process and also develop efficient taxonomies to classify ideas.

Fifthly, monitoring user experience is a good way to identify user frustrations. It is however easy to be overwhelmed by the dimensionality of the problem. To handle this effectively product managers must include Customer Experience measurements and taxonomies within the design cycle. This allows for efficient monitoring of diffusion and also helps to determine end of life of features.

Finally, the product manager's role as an editor and interpreter has not changed significantly. However, the pace of change has accelerated. The product manager still has to

look out for technological shift and anticipate latent needs on behalf of the end user. There is still no perfect measure of latent needs and it requires judgment and intuition on the part of the product manager. However, increased customer engagement in ideation, measurement of product usage provides new tools to make this process easier. In the following chapters, we will analyze each of these factors, their role in SaaS platform and how we arrive at the recommendations above.

2. Innovation in the Software as a Service Industry

“The customer simply has a job to be done and is seeking to “hire” the best product or service to do it.” (Clayton Christensen, Anthony Ulwick)

Background

Software as a Service is a technological evolution that has been fueled by advances in software technologies (such as virtualization), lower costs of hardware (decreasing storage costs, compute cycle costs) and better connectivity (Cheaper and ubiquitous internet access). This presents a scalable, shared computing architecture in which software companies (Software as a Service providers) distribute applications and computing resources to enterprises as required via the Internet. Forrester Research, Inc. defines it as *“A standardized IT capability such as software, application platform, or infrastructure, delivered via Internet technologies in a pay-per-use and self-service way.”* This approach is replacing traditional software in the enterprise and promises to transform computing into a service, thereby, bringing greater efficiency in enterprise IT systems [Kirkos 2010].

Evolution of the Software Business into a Services Business

The Software business differs from the conventional manufacturing focused industries, as software is malleable and becomes whatever function or application it addresses. Software is used in multitude of ways from word processing, to collaboration, to calculating taxes.

[Cusumano 2004] The costs of production in traditional software industry is similar to the pharmaceutical industry; the first product costs a significant amount of time and resources to produce but the costs of mass production (e.g. making new CDs) are virtually zero. The development of the enterprise software industry went through several stages in the past

1. Mainframe related software development.

2. Client – Server related software development
3. Internet fueled software development

During each of these discontinuities there was burst of software product innovations followed by focus on the process of developing software. This is inline with what has been proposed by the dominant design model [Utterback 94]. For instance, in the Internet fueled software development stage, there were rapid product innovations such as web servers for the efficient delivery of content on the World Wide Web. This was followed by a phase where costs of web servers and establishing presence on the Internet became extremely low. The focus since then has been on engineering practices such as Agile, Iterative methodologies to improve the process of software development to meet customer needs.

The product and process innovation phases do not complete the picture for the software industry. Following the product and process lifecycle stage, the software industry is likely to move to into a third stage where the major economic value generator and focus is on services [Cusumano, Kahl, Suarez 2006]

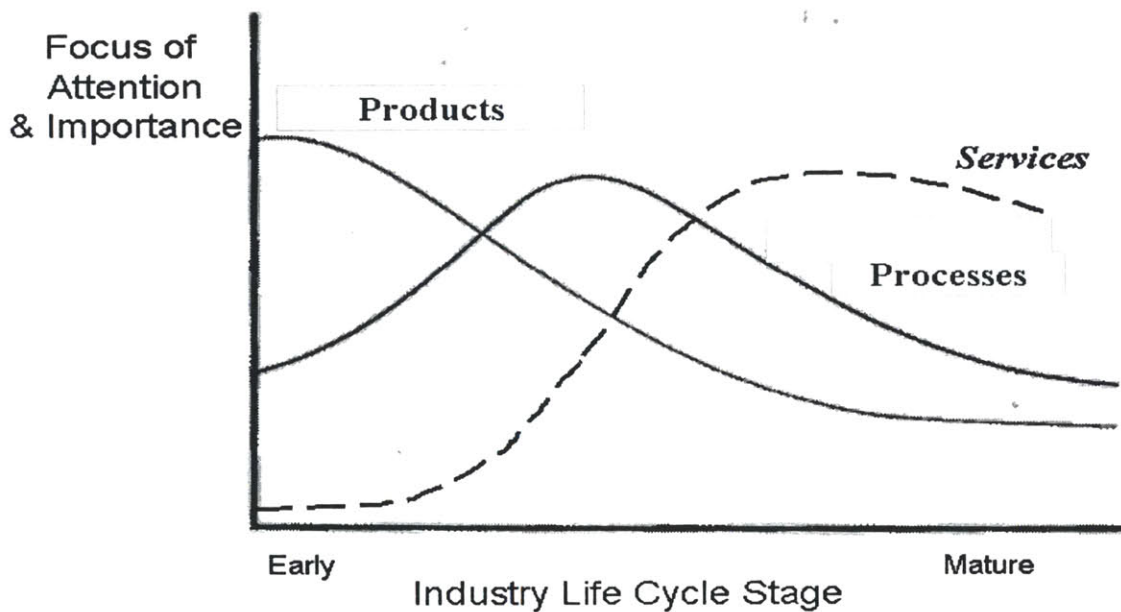


Figure 1: Service, Product and Process evolution [Source: Cusumano, Kahl, Suarez 2006]

There are three primary sources of revenue in the software Industry

1. License Fee (The upfront acquisition fee that the customer pays to procure the software)
2. Maintenance Fee (The yearly fee that the customer pays for upkeep of the software)
3. Services Fee (Specialist consultancy provided to install, configure & adapt the software)

The importance of each of these buckets varies with the stage the industry sub-segment is in.

Small and Medium sized Industries could not afford to pay the large upfront license fees or maintenance fees that the traditional enterprise software demanded. This was a latent untapped market that was poorly served by the traditional software Industry. Small companies could afford to pay for services, as these were operational expenses rather than large capital investments.

Secondly, software was a means to an end and not the primary value process for these companies. These companies valued outcomes and didn't consider IT a core process i.e. they could live with fewer features. These influences when combined with cheaper connectivity and the maturing of the industry (entering the services phase) provided fertile conditions to create a new delivery model for software, namely the software as a services (SaaS) paradigm. A recent study on the drivers of SaaS adoption supports these notions. [Benlian, Hess, BuxMann 2009]

Based on current trends, monthly subscription based fee with hosting may be the most likely strategy for the future [Cusumano 2007].

Having established how the Software as a Service industry came to being, we shall now focus on the different aspects of innovation within the Software as a Service Industry.

Is SaaS primarily a business model Innovation?

The notion of business model innovation has been a topic of recent focus and several authors have dealt with it in significant detail. [Meyer 2007] proposes that software as a service, such as Salesforce.com is a business model innovation. While there is certain aspect of business model

innovation in the SaaS paradigm, the author would argue that this doesn't complete the whole picture. For instance, the notion of "Pay as you Go" existed prior to SaaS. Time sharing schemes of mainframe computer resources were available. The notion of renting, versioning or feature based licensing were all variations of the "Pay as you Go" model and were in existence well before the advent of SaaS. What separates the SaaS paradigm from purely business model innovation are the networks effects that SaaS leverages. These include user participation, higher server efficiency through pooled usage etc. Secondly, to be able to deliver applications at the scales required, there has been a whole range of software infrastructure innovations such as Multi-tenancy; browser based rich user interfaces. Without these innovations the paradigm would not be possible. Thirdly, the notion of a dataset that grows richer with collaboration also makes the SaaS model more than business model innovation.

Product and Diffusion Process Innovations

From a SaaS service provider point of view, the paradigm where end users execute applications on hosted servers presents interesting opportunities to optimize application efficiency, performance and provides new avenues for analyzing customer requirements and preferences. There are two types of delays in the transmission of technology, adoption delays and realization delays. Adoption delays are related to price/performance, risk and market structure. Realization delays are dependent on "opportunity", usability and adaptability of the technology [Barras 86] Adoption delays can be minimized in SaaS model as price is already included in the subscription model, there is lower risk as there is no migration plan required to exploit the newer features. Realization delays are much easier to measure and quantify with the SaaS paradigm. In the past, fairly expensive processes were required to understand user penetration of individual improvements or features that are included in each release of software. For instance, to this day most users of Microsoft Word only use a very small subset of the features of the

software. The reason for this reduced penetration is that product developers, engineers and marketers at Microsoft have not had good means of analyzing the entire product eco-system. When the information around the usage of a product and its drawbacks are localized and the cost of transferring this information to the vendor is significant, user driven product innovation may be a better alternative [Von Hippel 1998]. Software companies had to setup dedicated user Interaction labs to obtain such information. In the SaaS model, when the user executes software on a hosted platform, it is fairly easy for the SaaS service provider to get deep analytics on usage. The rich actionable dataset promises to change the landscape of product development and roll out in the software Industry.

Pure play SaaS Vendors versus Traditional Software Vendors

Traditional software vendors are attempting to adapt to the SaaS paradigm. But the transition to this paradigm is not easy. Most software companies rely on maintenance revenues as the primary source of their economic value generation. Software product firms overall saw about 50% of their revenues come from maintenance and services [Cusumano 2008]. Moving away from this source of maintenance revenue requires significant upheaval of the business for the software vendors. Such transitions are possible but not easy.

The difficulty in making this transition can be understood with a simple thought experiment. Most enterprise servers are idle 93% of the time. Salesforce.com supports roughly 150,000 customers with 5000 servers. If each of the 150,000 customers were to purchase their own enterprise software and databases that would amount to 150,000 software licenses and maintenance agreements. In effect, the SaaS model has shrunk the traditional software revenue for a database vendor to 3% of its potential size. This speaks to the core of the innovators dilemma argument put forth by Prof Clayton Christensen. [Christensen 1997] However, it

doesn't mean the end of the traditional software vendors as Clayton Christensen would argue, there might be change in leadership but the traditional vendors can adapt to this new paradigm. Microsoft's move into Azure and their development of office 360 are examples of this transition.

Product Portfolio Management in SaaS

The idea of user collaboration and user community driven development has been central to the development of the SaaS Industry. The author was the lead architect for Salesforce.com Asia Pacific. Salesforce.com has 3 product releases a year and almost 30% of new features in each release is from user submitted ideas. The filtering of those ideas is done by the user community, which votes for each other's ideas. Leveraging the community of users so deeply in the development cycle is an innovation that is unique to the SaaS Industry. While other industry's listen to the customer base, the speed at which this is incorporated is what sets the SaaS Industry apart. It also allows the technology manager at the SaaS company to better manage the product portfolio. Development expenses can be better targeted at those features that users will find most useful.

Becoming a platform – Addressing new markets

SaaS industry is presently in the stages of making a transition to an Industry platform as firms are opening up their SaaS software infrastructure to other players and complementary product vendors. Google originally developed their application engine infrastructure to power the various services such as Gmail, Maps etc. Now that it has matured, the Google App Engine offering allows enterprises to take advantage of the same Infrastructure that hosts Google's SaaS offering to derive newer applications. Presently, there is fight for the dominant design of the platform that will allow custom SaaS application development. The contestants include Salesforce with Force.com and its AppExchange platform, Microsoft with Azure, Google with Google App Engine, VMware with its VSphere technology to name a few. All these companies

are adopting an “Open But not Open strategy” [Cusumano 2010]. This is a combination of open innovation (The open part of the strategy) and specific APIs that make switching difficult (The Not Open part of the strategy).

This mode of innovation helps to address software markets where the size was not big enough to attract large vendors. An analysis of amazon.com’s sales in 2003 showed that a significant portion of amazon’s sales came from selling books that were hard to find / niche oriented that traditional book sellers couldn’t cater for [Brynjolfsson et al 2003]. Similarly, these SaaS platforms are being used by complementary software vendors to address niche markets such as ERP software for non-profits. It is unclear whether a clear dominant platform will emerge or whether there would be use case driven balkanization of platforms.

Impact on other traditional IT Services Industry

IBM is looking to leverage cloud and SaaS trends in services. However, while SaaS present a new opportunity for services, much like the migration to the Internet did, it also promises to endanger traditional IT outsourcing operations. Companies who have arbitrated the cost of labor between geographies as the primary value driver face an uncertain future. When software is delivered as a service there is lesser demand for traditional services as both product and service is combined into a single package by the SaaS vendor. While there is opportunity for traditional IT consulting, these are smaller consulting engagements. While the likes of IBM can navigate these waters due to their extensive R&D investments, companies like Infosys and Satyam will likely face an uncertain future, as they do not invest significantly in software R&D.

Will all Software become Services?

While there is momentum behind delivering software as a service, the author doesn’t believe that all software will be delivered as services. Applications that require a fair amount of user interaction and collaboration such as workflow and Enterprise Resource Planning applications

lend themselves to the services model. Applications that are computation intensive are not readily amenable to be delivered as services. Data Mining applications, Video editing applications are computation intensive and require specialized knowledge to benefit from and are less likely to be delivered as services in the near future. But this is merely a function of delivery maturity and cost economics. In time, these will also be delivered as services.

Chapter Conclusion

Delivering software as a service is an evolutionary (not revolutionary) paradigm that been fueled by cheaper network access, excess computation power and advances in software architecture. This paradigm has taken traditional shrink-wrapped software and converted them into user centered service driven industries. In this process, the nature of innovation within the industry has been driven by higher user participation. Platform innovations have emerged that allow participants to address previously untapped markets. This trend may also upset existing IT consulting industries by packaging services and product in a single solution. However, not all software will be delivered as services. Presently, higher user interaction applications are driving the industry growth.

3. Architecture within Software as a Service

"Architecture doesn't come from theory. You don't think your way through a building." Arthur Charles Erikson

Classic Model View Controller Architecture

At its core SaaS is software deployed as a hosted service and used over the Internet. Typical business applications that can be address by SaaS are those that business process and workflow oriented and have a fair degree of human interaction with the application. These types of application have been architected around the design paradigm of a model-view-controller architecture.

The MVC architecture began as a user interface paradigm in the Smalltalk environment. [Krasner, Pope 1988].

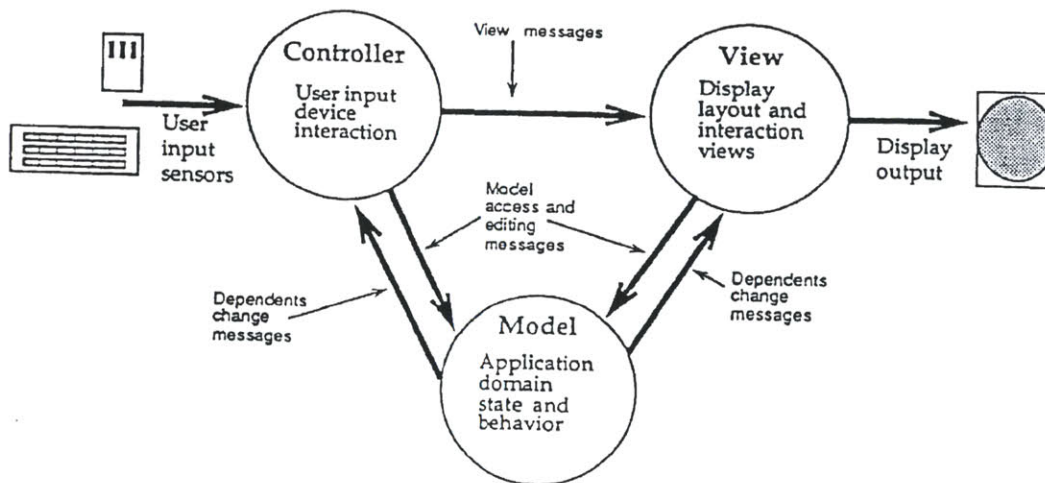


Figure 2: MVC Architecture [Source: Krasner, Pope 1998]

This architecture separated concerns in three key specific areas.

- a) **The Model:** This functions as the encapsulation of the domain specific objects that the application has to deal with. It is responsible for persistence of state of the objects being manipulated upon
- b) **The View:** This is responsible for displaying and rendering of the output for a particular invocation in a user acceptable format
- c) **The controller:** The controller's job is to control both the manipulation of the underlying model and also a particular view of the resultant data

The MVC architecture separated the presentation from persistence and allowed loosely coupled software architectures to be made possible.

This underlying paradigm has since been adopted in several iterations of the software development industry including Java based application server architectures, .Net based architectures to the latest web design frameworks such as Ruby on Rails.

High Level SaaS Application Architecture

Following the extension of the MVC paradigm described earlier, SaaS application typically follows the high level architecture shown below

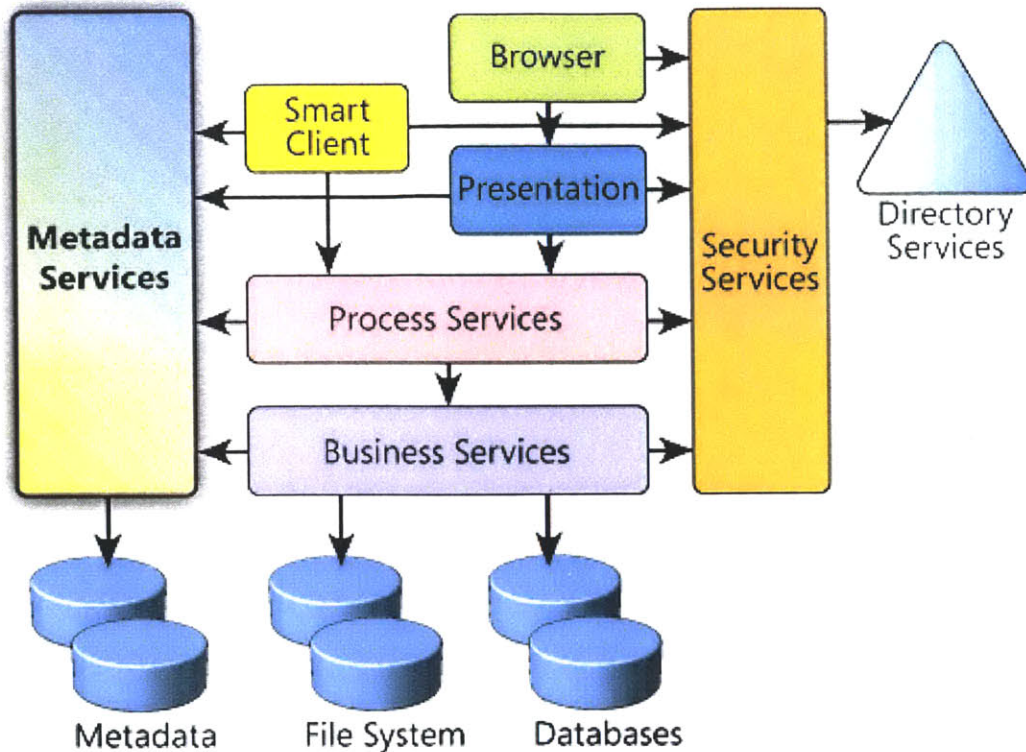


Figure 3 : SaaS high level architecture [Source: Chong , Carraro 2006]

The key service that enables all of these offerings is the Metadata Services. In architectural sense it is akin to the model layer of the MVC paradigm we described earlier. However, unlike traditional software where the model, view and the controller often co-existed on the same server, within the SaaS environment, horizontal scalability is a key requirement. This is required both from the perspective of providing fault tolerance and from the perspective of end user perceived performance.

The controller and the View modules (business services/process services) can be scaled in a relatively straightforward manner, as they don't manage state persistence (perform

immutable write to disk). Hence, the key component that requires architectural effort at horizontal scalability is the Model layer / Meta data services layer.

The Metadata services layer is typically built on top of a relational database. However, the partitioning of databases in a distributed system is difficult task from a maintenance and operational standpoint.

Unique Architectural Challenges of SaaS

Horizontal Scalability of Persistence

Maintaining consistent state in distributed systems (horizontal scaling) is a difficult proposition. The CAP theorem first conjectured by Eric Brewer [Brewer 2000] and then proved by [Lynch, Gilbert 2002] places some architectural limits on the ability of a SaaS vendor to maintain state within a SaaS application. The theorem states that between Consistency, Availability and Partitioning Tolerance, at most only two of these properties can be satisfied by any shared data system.

Metadata driven approach to domain modeling and Configurability

SaaS applications typically have to provide the flexibility for end-users (tenants) to configure the application to their specific needs and requirements. The requirements/configurations made by one such user should not affect those of another. In short, the application has to be polymorphic. This places a heavy emphasis on meta-data driven approach to data modeling. The key is to balance this meta-data approach without sacrificing performance.

Maintenance is part of the architecture

The architectural design of a product feature must also look at the maintenance and lifetime value of the feature. In SaaS, the maintenance overheads are borne by the provider and not by the user. *“A wrong architectural choice might entail that the SaaS application becomes a maintenance nightmare”* [Bezemer Zaidman 2010]

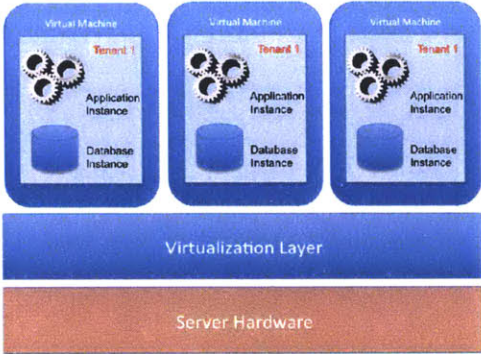
Role of Multi-tenancy

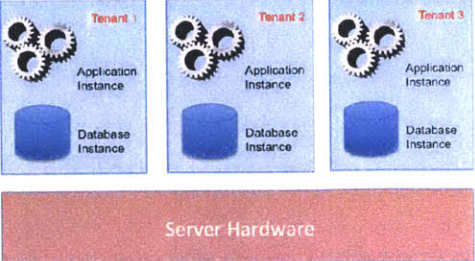
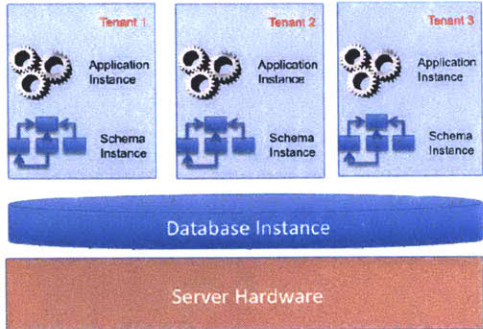
The economics of SaaS application depends the architectural concept of multi-tenancy. *A multi-tenant application lets customers (tenants) share the same hardware resources, by offering them one shared application and database instance while allowing them to configure the application to fit their needs as if it runs on a dedicated environment.* [Bezemer Zaidman 2010]

This concept can be implemented in a few different ways

- a) Virtualization as a means of partitioning hardware resources
- b) Shared application, separate persistence layer instances (database instance)
- c) Shared application, shared persistence layer but separate objects (tables)
- d) Shared application, Shared tables

These implementation approaches each have their individual advantages and disadvantages

Implementation approach	Observations
<p data-bbox="168 331 610 443">1) Virtualization driven multi-tenancy</p>  <p>The diagram illustrates a multi-tenant architecture based on virtualization. At the base is the 'Server Hardware' (represented by a brown block). Above it is the 'Virtualization Layer' (represented by a blue block). On top of the virtualization layer are three separate 'Virtual Machine' containers. Each VM contains an 'Application Instance' (represented by a gear icon) and a 'Database Instance' (represented by a cylinder icon), both associated with a specific 'Tenant' (labeled 'Tenant 1').</p>	<p data-bbox="725 331 1390 590">In this approach, the hardware is split into separate virtual machines and within each instance a separate instance of the database and the application are configured per tenant.</p> <p data-bbox="725 632 902 667">Advantages</p> <ul data-bbox="725 709 1357 890" style="list-style-type: none"> Provides very good isolation between tenants Easier to setup initially Replication is extremely bandwidth heavy <p data-bbox="725 932 948 968">Disadvantages</p> <ul data-bbox="725 1010 1398 1415" style="list-style-type: none"> Memory cannot be shared across tenants and hence the per tenant costs increase Maintenance requires modifying individual machines Multi-tenant only at the machine level but not at the application level
<p data-bbox="168 1455 638 1491">2) Separate Database Instances</p>	<p data-bbox="725 1455 1390 1713">In this approach, a single hardware runs several database instances and application instance individually and each tenant has access to once such instance</p> <p data-bbox="725 1755 902 1791">Advantages</p>

 <p>Tenant 1 Application Instance Database Instance</p> <p>Tenant 2 Application Instance Database Instance</p> <p>Tenant 3 Application Instance Database Instance</p> <p>Server Hardware</p>	<p>Good isolation of tenants</p> <p>Disadvantages</p> <p>Maintenance is poor as bulk changes to application across all instances cannot be done, very little gain from shared resource pooling</p>
<p>3) Separate Schema approach</p>  <p>Tenant 1 Application Instance Schema Instance</p> <p>Tenant 2 Application Instance Schema Instance</p> <p>Tenant 3 Application Instance Schema Instance</p> <p>Database Instance</p> <p>Server Hardware</p>	<p>In this approach, all tenants on a server use the same database but separate schemas (tables).</p> <p>Advantages</p> <p>Better utilization of hardware than separate database instances</p> <p>Better ability to support maintenance changes across multiple tenants</p> <p>Disadvantages</p> <p>Connection pooling of database needs to be handle expertly</p> <p>Security is done at the application level</p>
<p>4) Shared Schema approach</p>	<p>In this approach, the database schema is shared across all the tenants. Each table has a separate column (Tenant ID) that is used to ensure that data for the right tenant is used. Extension of the base schema are done through Metadata approach</p>

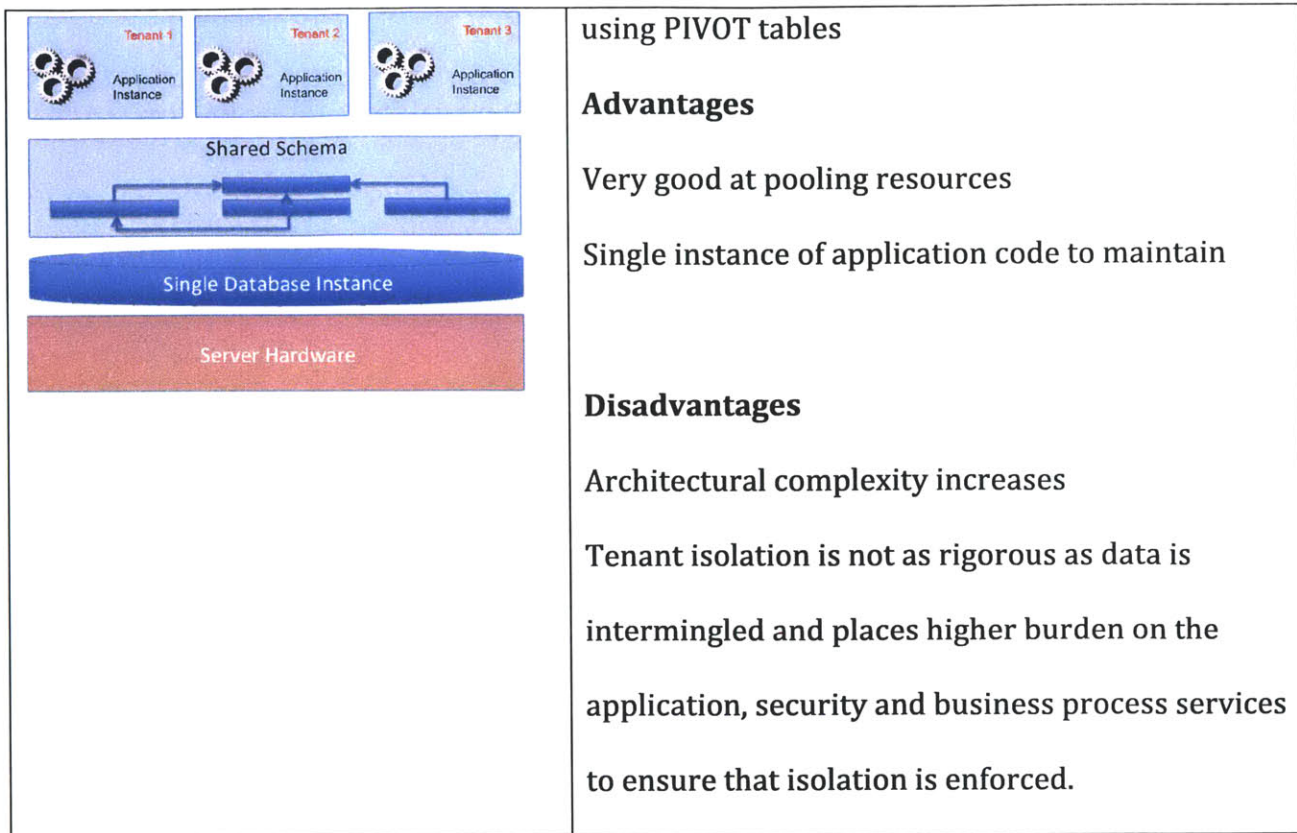


Figure 4: Comparison of degrees of Multi-tenancy

[Aulbach, Jacobs 2007] have analyzed the relative performance merits of each of these approaches and conclude that the shared process approach is perhaps the most optimal for the SaaS applications. They claim that the *“Shared-table swings too far in the other direction, potentially compromising performance, customer migration, security, and typing.”* However, the shared table approach has been shown to be very successful in large implementation such as those of force.com [Weissman, Bobrowski 2009]

Presently in the marketplace, there are a wide variety of vendors claiming to be multitenant and each has one implemented one variation of these schemes. Force.com is an example of the shared schema (table) approach. On the other end, VMware is touting the virtualization approach given its isolation benefits.

The author feels that there is no one fixed approach that is ideal for all SaaS applications.

The nature of the multi-tenancy adopted is dependent on the scope and scale of the solution that is being built.

Impact of Architecture on Product Management / Product Evolution

The choice of architecture impacts the product evolution of a SaaS application along the following parameters

1. Cost of provisioning new features
2. Speed of new product feature development

Feature Development Cost: Cost Per Tenant vs. Cost Per Feature

The cost of providing and developing new SaaS features is significantly higher when the scale of the application increases. As the degree of multi-tenancy goes up so does the cost of implementing a new feature. In effect, there is a multi-tenancy tax on each new feature that needs to be taken into account whilst making product management decisions.

This impact can be viewed ***Cost per tenant*** is the cost of providing a new feature to single client. ***Cost per feature*** is the cost of implementing the specific feature [Katzan, Dowling 2010]

Multi-tenancy incurs a higher cost per feature but lower cost per tenant. While increased isolation provides lower cost per feature but increases cost per tenant.

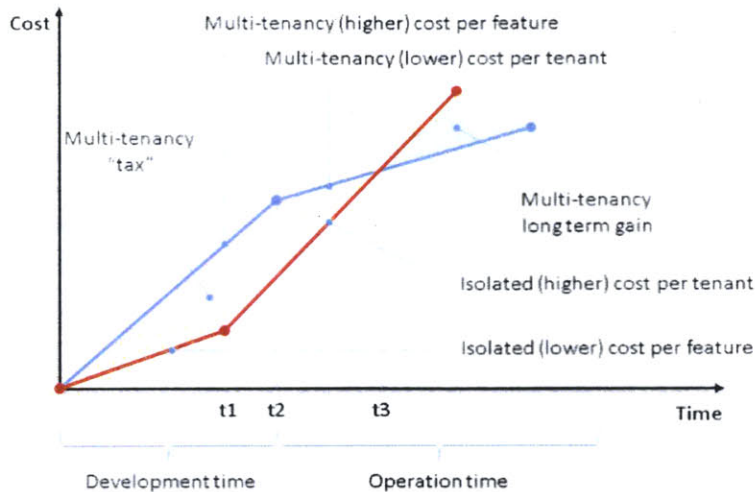


Figure 5: Cost per Feature vs. Cost Per Tenant [Source: Carraro 2008]

Additionally, the higher architectural complexity of multi-tenancy requires different development priorities and tradeoffs. Since bulk of the complexity arises from consistency, the product manager has to analyze in detail the nature of the consistency demanded by a particular feature. For instance, a status update on communication-oriented feature (such as tweet) could be engineered to lower consistency guarantees than say a financial transaction update (which may require full ACID compliance).

As each new feature is being considered issues about data residency and ownership and information security will play higher importance as part of the Market Requirements Document (MRD) that is typically owned by the Product Manager.

Speed of new feature development

The lower the degree of multi-tenancy, the quicker it is to develop a new product or feature. However, this speed comes at a cost, as the SaaS application is not able to effectively exploit the economies of scale that it enjoys with pure multi-tenancy. However,

depending on the market being addressed a purely virtualization based approach (lowest multi-tenancy) may be the ideal approach.

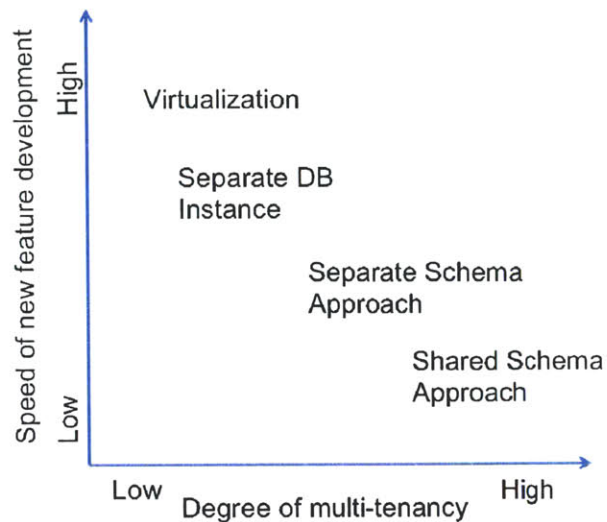


Figure 6: Speed of feature development vs. Multi-tenancy

The speed of feature development is but one aspect of the overall SaaS product management challenge. Some of the difficulties in the development speed of pure multi-tenancy can be better managed by separating features into two separate categories

- Platform Features
 - These are features that are application independent and focus on the overall platform on which the SaaS application is run. For instance, implementing a NO-SQL storage for eventual consistency that is shared across all applications would be one such feature.
- Application Features
 - These are feature that specific to the application domain that the SaaS application is being engineered for. For instance, implementing a balance

check feature within an Accounting application would be an example of this feature.

Separating feature in this fashion provides the product manager better control over resource allocation and roadmap scheduling.

Chapter Conclusion

The architecture of the SaaS application indirectly affects the Product Manager and his ability to manage the new feature roll out and development. Most SaaS applications are a variation of the Model View Controller paradigm and bulk of the complexity related to SaaS applications are at the Model (persistence) layer.

SaaS applications exhibit varying degrees of multi-tenancy with different security, ease of development, maintenance and resource efficiency. The product manager must carefully consider the implications of the underlying architecture whilst generating requirements specifications and take into consideration lifetime value of a feature and trade it off against the cost per tenant implied by the underlying architecture.

4. Role of the Development Process in Software as a Service

"No member of the development effort faces greater change in moving into the SaaS world than the product manager" Tom Grant, Forrester Research

Changing Nature of Development in SaaS

The nature of software development changes within the SaaS model. The development process sees greater involvement from the product manager with the development team.

The product manager often becomes an integral part of the process rather than just the owner of the Market Requirements Document.

The key element of risk in SaaS product feature development is market acceptance and end user perceived quality. Since there is no upfront license fee and customers can discontinue product use at anytime, market relevance of product features is extremely important.

Achieving this requires quicker turnaround in development times. Quality of feature rolls out is critical as losing a customer has a negative impact on new customer adoption (as SaaS relies heavily on relationship marketing). Traditional development models such as the CMMI do not adapt themselves well to this highly dynamic environment. In a study performed on 72 small software firms, it has been demonstrated that Agile methodologies have a positive effect on product development effectiveness where CMMI does not.

[Peltonen, Fruhwirth, Ronkko 2010]. SaaS firms primarily engage in relationship marketing rather than transactional marketing. In this paradigm, customer acquisition cost, customer lifetime value and churn are the key metrics for a sustainable business [Tyrväinen, Selin 2011].

These changes in the nature of the software business have required SaaS companies to handle development process differently.

Agile Methodologies in SaaS

Traditional waterfall development strategy has proved to be hard to use for software projects that have dynamic requirements. In response to the needs of managing software projects many different iterative and incremental development techniques emerged.

Collin Murray in his thesis "lean and Agile Software Development A Case Study" states "The term waterfall is attributed to Winston Royce, through his paper "Managing the Development of Large Software Systems", published in 1970. The misinterpretation of Royce's model led to the standard accepted practice of waterfall development that is used today. SaaS projects lend themselves well to iterative and incremental development. This move away from waterfall has been covered under the umbrella term "agile"

The key emphasis of the Agile methodologies are

1. Continuous delivery as opposed to big bang delivery
2. Changing requirements are embraced rather than frowned upon
3. Individual interactions are deemed more important than strict documentation

These principles align themselves very well with the needs of the SaaS product development approach. In this scenario, the Product Manager morphs into the Product owner in the agile methodology and is focused on continuously refining the feature set with the development team. Rather than focus on large MRDs, the product manager focuses on targeted user stories that address features of highest relevance to the end customer.

Within this paradigm, the traditional activities of requirements collection and functional specification all exist. However, the approach is focused on smaller increments and new features are often divided into smaller deliverables. Fixed feature sets are abandoned as these typically lead to schedule slippage. The product manager engages in constant reprioritization and refining the features, determining the minimal viable set for the release. Several leading SaaS companies have made the transition to Agile methodologies. Salesforce made a big bang transition to the Agile methodology and experienced a 37% increase in product release productivity and went from 1 major release each year to 3 major releases each year. [Mencke 2008]

Case Study of SCRUM in Ask.com Product Development



To study the application of newer agile project management methodologies in SaaS, the ask.com product development team will be used. The study will focus on the advantages, disadvantages that SaaS projects will benefit from agile methodologies.

Ask.com has adopted of SCRUM across the organization starting with software development. They graciously agreed to open up their internal teams and data to us for this study. Ask.com has teams spread across the globe and practice SCRUM in distributed teams. They also have dependencies between scrum teams. Some of the challenges Ask.com faced which lead them to adopt SCRUM were

1. Quicker time to market driven by dynamic business environment and changing business model
2. Quality issues with waterfall development for their company
3. Help rally the product development team in a very competitive business environment

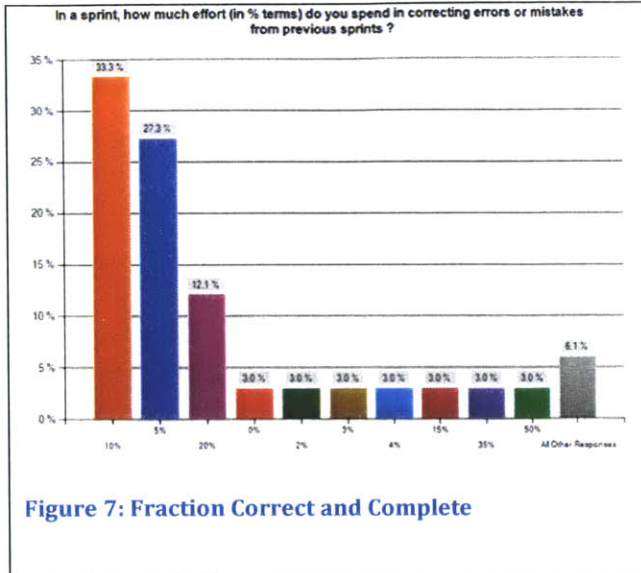
Research Approach

The following approaches were adopted as part of this case study.

1. Literature Survey on SCRUM
2. Regular conference calls with sponsors for sharing of SCRUM PM data
3. Survey of SCRUM practitioners at Ask.com
 - a. List of 17 questions designed to (Actual Survey is in the Appendix)
 - i. Measure Fraction Correct and Complete
 - ii. Measure effect of SCRUM on human factors
4. Assess the impact of
 - a. Effort Estimation variance
 - b. Requirements Risk variance
5. Use data as basis for a System Dynamics Simulation of SCRUM to understand how Agile helps projects with requirements uncertainty as the key risk

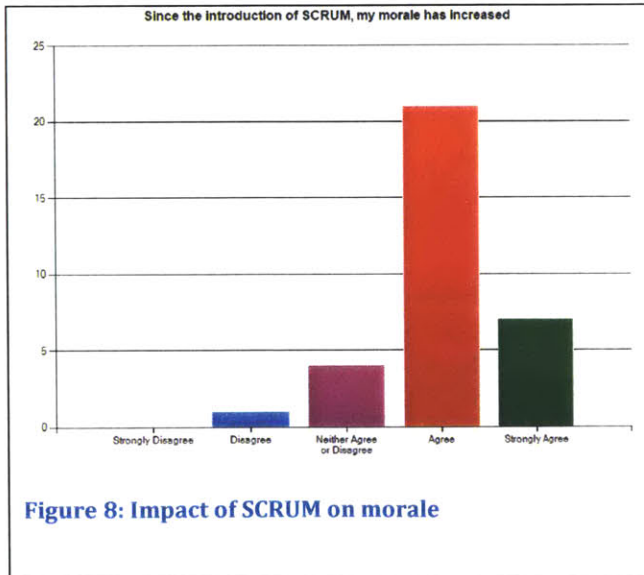
Findings

Impact on Fraction Correct and Complete



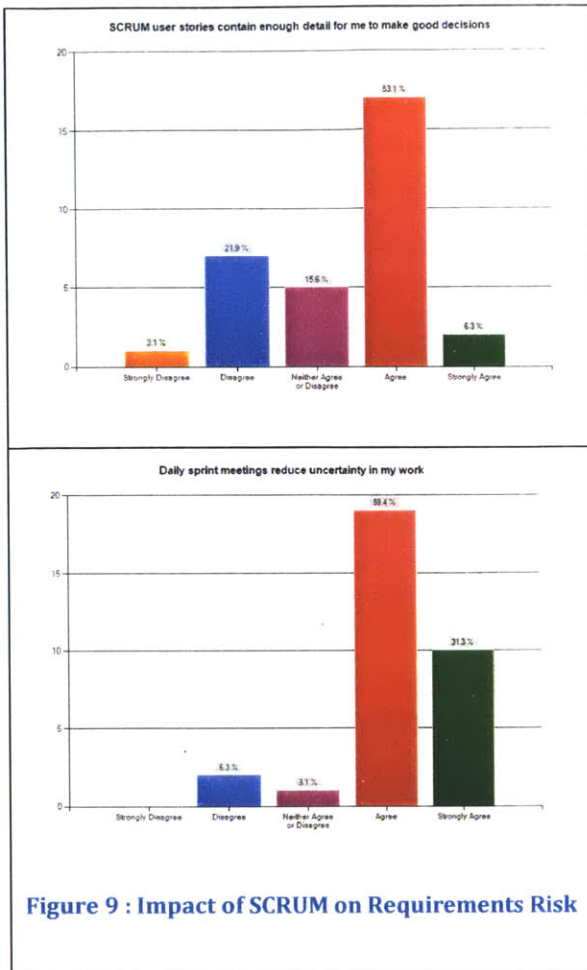
As a proxy measure for fraction correct and complete, respondents were asked to estimate how much of their time in a current sprint, they spent fixing errors from the previous sprint. The median value was 7.5 %. This is a good indication that bulk of the activities is done correctly during a sprint.

Impact on Morale



Close to 85% of the respondents, stated that team morale had improved since the introduction of SCRUM. This is line with other findings from other authors. [Mencke 2008]

Impact on Requirements Risk



The results on managing requirements risk were interesting. The product team was asked if SCRUM stories contained enough detail for making good quality decisions. A significant portion of the respondents (40%) was neutral or didn't agree with this. However, more than 90% of the respondents maintained that the daily sprint meetings reduced uncertainty in the work being performed.

It is likely that the daily meeting between product managers and the development helps refine feature sets and requirements as the build takes place. Written documentation is static and within the dynamic environment of a SaaS platform, the constant communication and feedback helps reduce development risk and increases quality.

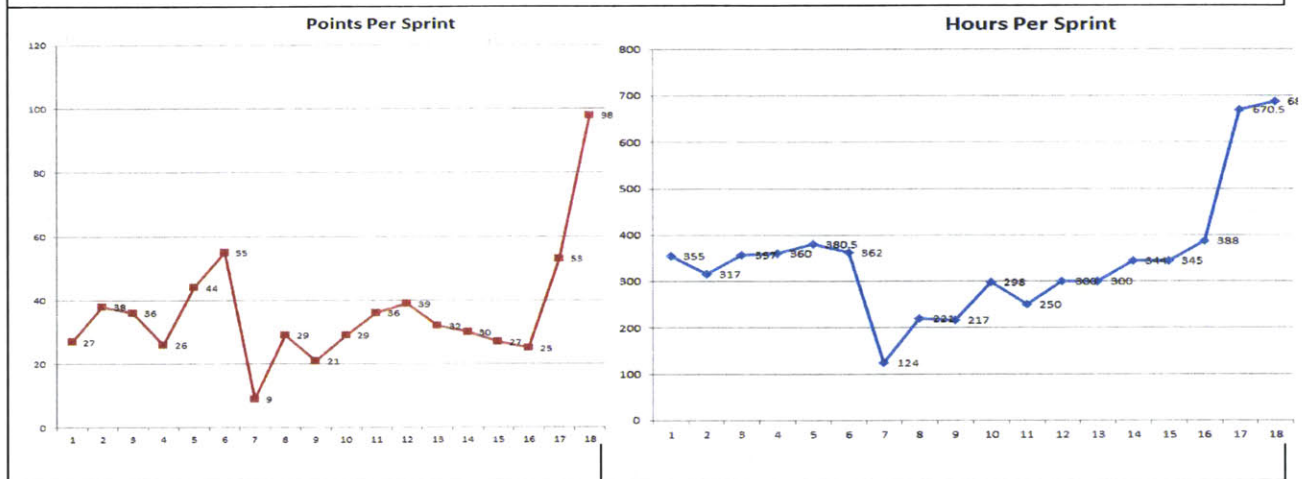
Impact on Quality

- **Before SCRUM:**
 - 6/1/2010 – 7/1/2010 -> 518 bugs created
 - **After SCRUM:**
 - 6/1/2011 – 7/1/2011 -> 174 bugs created
- ~300% Reduction in defects**

This impact on quality of product development is significant. For SaaS, the key marketing tool is word of mouth and the quality delivered acts both a source of reducing customer churn and allowing network effects of new customer adoption through word of mouth.

Experience effects in SCRUM

Complexity Measures



Experience and Resource burn down across sprints

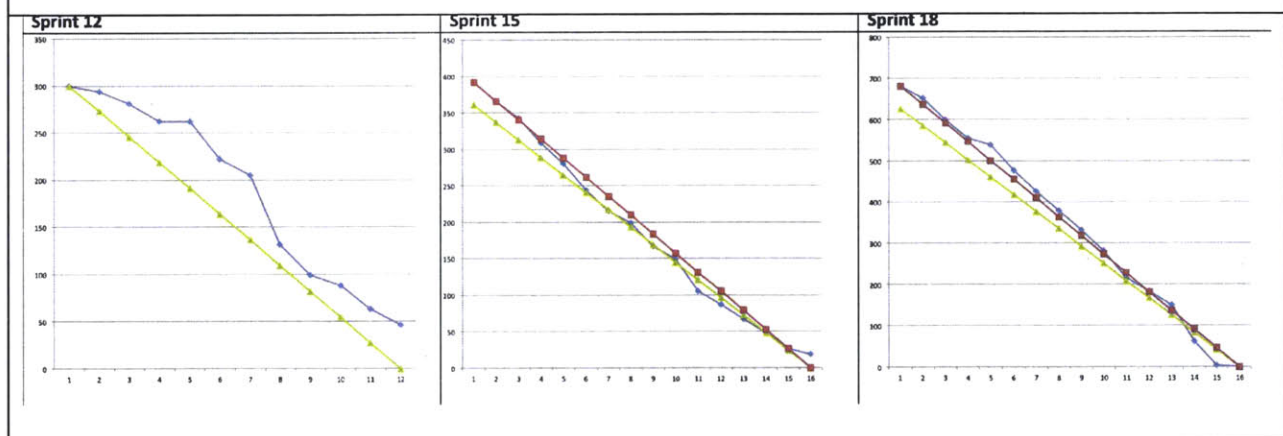


Figure 10 : Experience Effects of SCRUM

The charts above show the increasing complexity of tasks as the team matures. Both the points (a measure of complexity of tasks) and the hours per sprint have increased. The corresponding sprint burn down is getting accurate. For instance at sprint 12, the hours were 300 and complexity was 39 But sprint 18, the hours are 687 and 94 points but the

burn down is nearly as much as the team predicted. **The key finding is that it takes a while before things get better and management must stick with the course.**

Using System Dynamics Model to study SCRUM

A system dynamics model to capture the impact of SCRUM on requirements uncertainty on product completion time as we vary the number of iterations (a.k.a sprints) from 1 (i.e. waterfall model) to 20 was built. The results show that the impact of requirements uncertainty are reduced as the number of sprints increases.

The Model

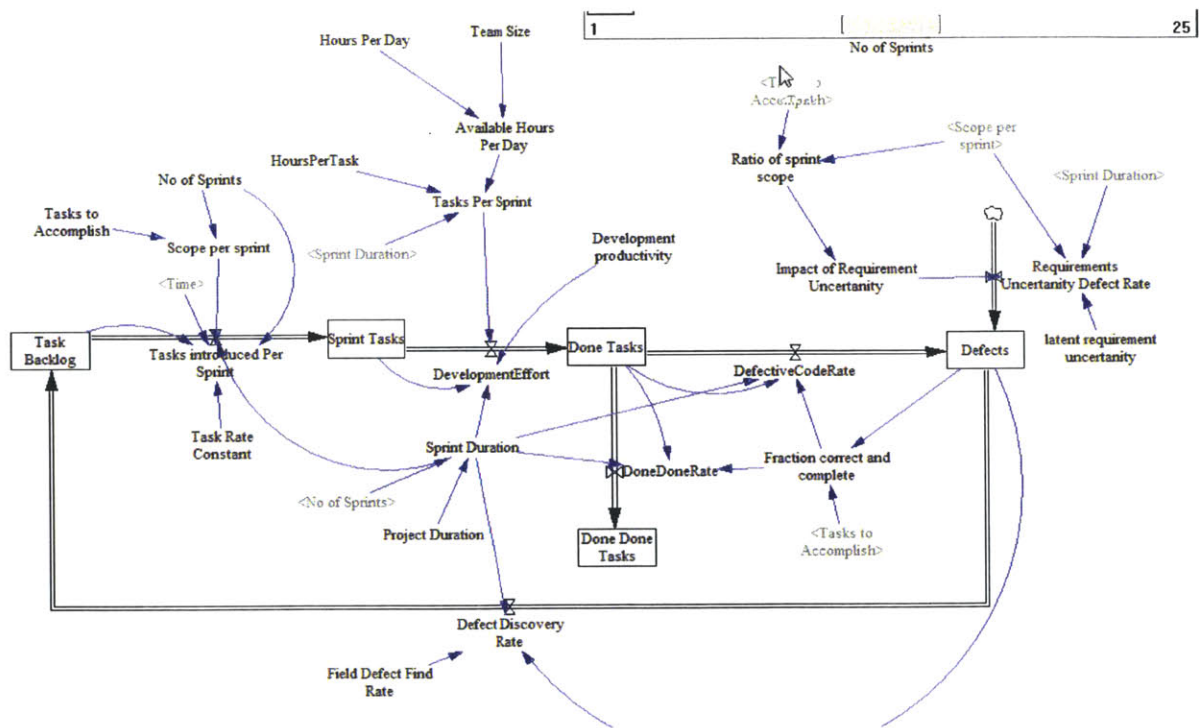


Figure 11: SD Model for SCRUM vs. Waterfall

The Rationale

Here is a brief explanation of the stocks and important variables in this model.

Iterations

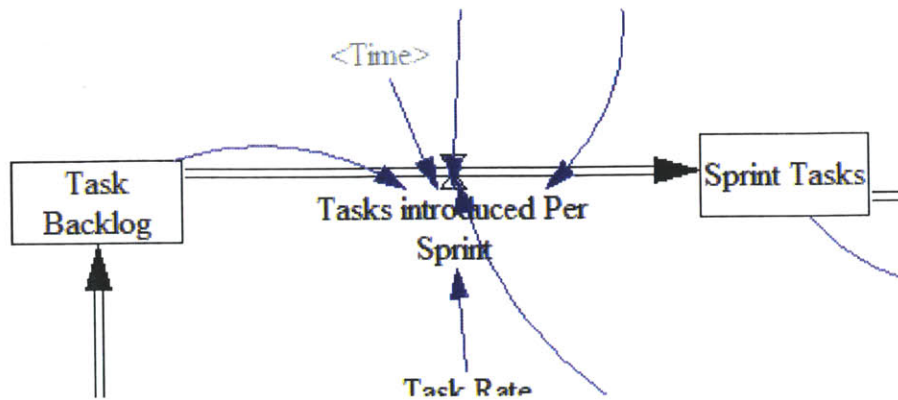


Figure 12 : Zoomed view of Iterations in SD Model

A **Task Backlog** is a stock of tasks to finish for a project. It has an initial value of 1000 and rework discovered during each sprint (the defects) is fed back into the task backlog after it goes through a defect discovery mechanism.

Sprint Tasks is a stock of tasks which a SCRUM team works on in each sprint. Tasks move from the **Task Backlog** to the **Sprint Tasks** stock at the beginning of each sprint at the rate of **Tasks Introduced Per Sprint**. (This variable is explained more later)

The **Tasks Introduced Per Sprint** rate variable denotes the number of tasks that are handled in each sprint. This is modeled via a PULSE TRAIN as follows:

$$\text{PULSE TRAIN}(0,1,\text{Sprint Duration},1000)*(\text{IF THEN ELSE}(\text{Time}>((\text{No of Sprints}-1)*\text{Sprint Duration}), \text{MIN}(\text{Scope per sprint}, \text{Task Backlog}), \text{Scope per sprint}))/\text{Task Rate Unit Constant}$$

The **Scope Per Sprint** variable in the formula above denotes the amount of tasks ideally addressed per sprint during the planned project duration. After the planned project duration, we put a IF-THEN-ELSE condition, in addition to a MIN() constraint in order to handle the reducing number of tasks in the task backlog.

Task Completion and Impact of Uncertainty

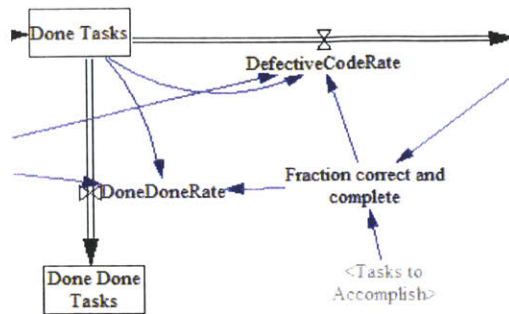


Figure 13: Zoomed view of Task Completion in SD Model

The **Done Tasks** are tasks, which are considered complete. Tasks move from the stock of **Sprint Tasks** Stock to **Done Tasks** stock depending on the **Development Effort**. Not all **Done Tasks** are 100% done - some of these might be defective in which case they go to the stock of **Defects**. The remaining tasks which are done correctly go to the stock of **Done Done Tasks**. Tasks from the stock of **Defects** move back to the **Task Backlog** depending on the **Defect Discovery Rate**.

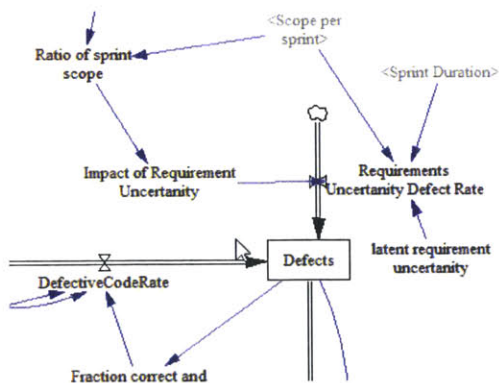


Figure 14: Zoomed View of Defects in SD Model

The **Requirements Uncertainty Defect Rate** represents the rate at which defects are generated in the system due to uncertainty in requirements. The uncertainty itself is

represented by a constant called latent requirement uncertainty. The total Requirements uncertainty depends not only on this latent uncertainty, but also on the sprint duration and the scope of each sprint. This is modeled as follows:

$$\text{Scope per sprint} * \text{Impact of Requirement Uncertainty} * \text{latent requirement uncertainty} / \text{Sprint Duration}$$

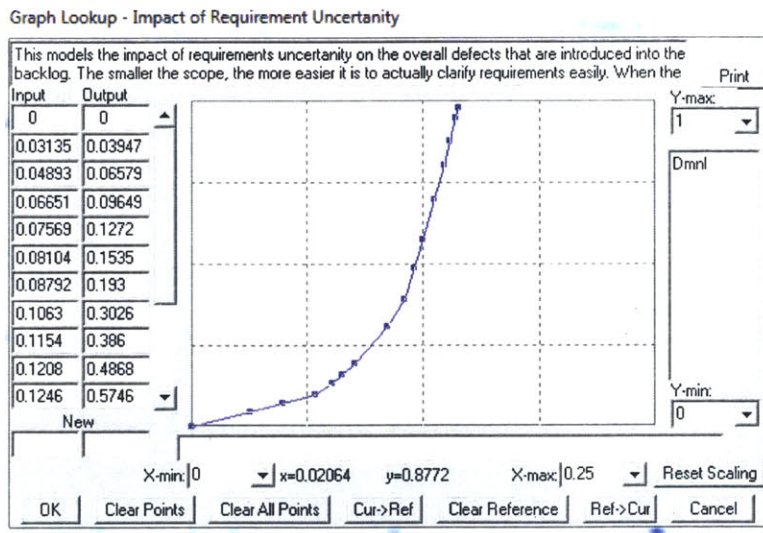


Figure 15 : Non linear relationship of requirement uncertainty

Finally, we modeled the **Fraction Correct and Complete (FCC)** as a table lookup of the ratio of the number of outstanding **Defects** to the total **Tasks to Accomplish**. As the number of defects goes up, the work pressure also mounts and the FCC goes down.

Graph Lookup - Fraction correct and complete

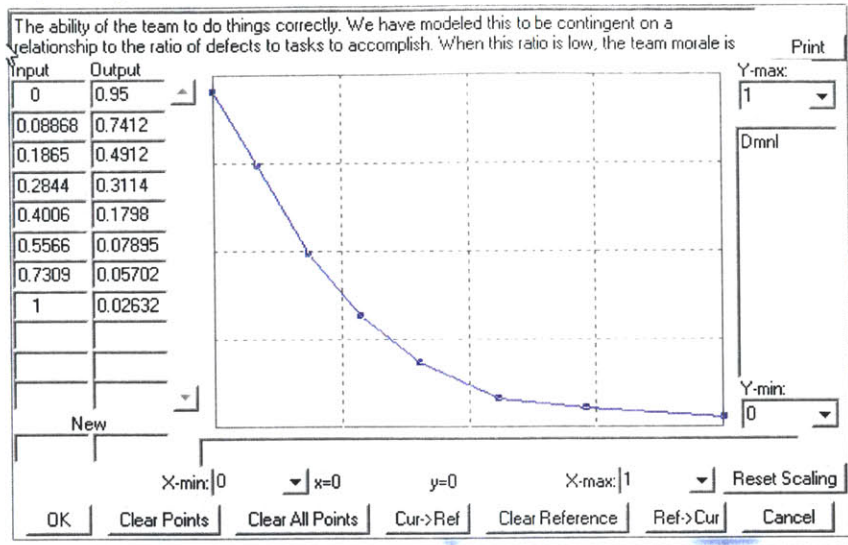


Figure 16: Fraction Correct and Complete Table Function

The Simulation Results

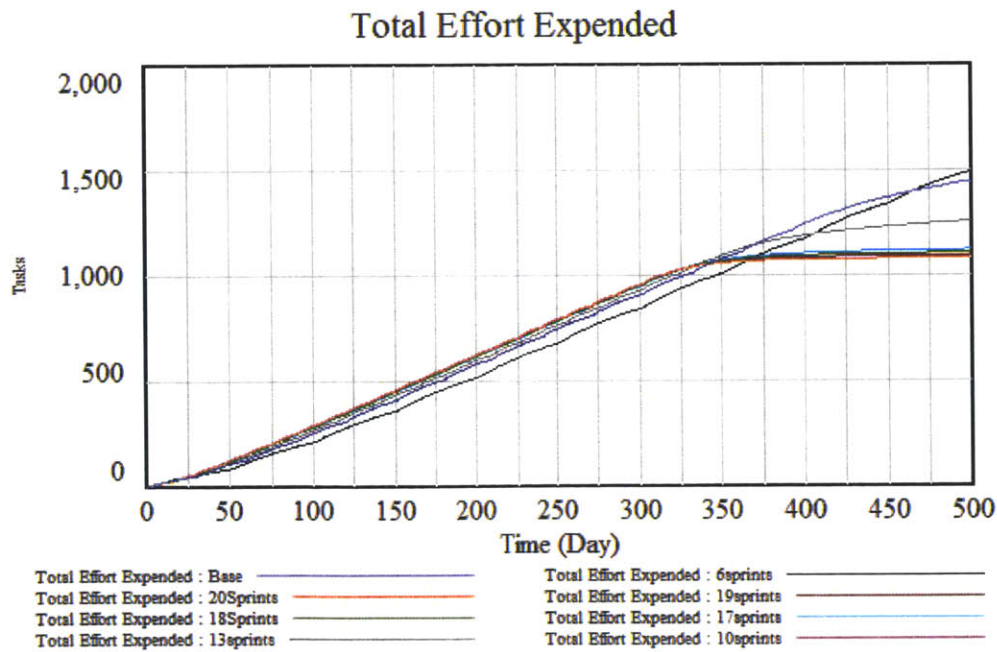


Figure 17: Simulation Results across various iteration counts

Based on the model, the author varied the number of sprints and looked at the effect on overall project effort. At between 17-20 sprints, the system stabilizes and the project finishes with only a 10% variance in the project duration and efforts.

When the number of sprints is reduced, the FCC rates starts to drop and the defects start to grow significantly. As a results the “Error on Error mechanism kicks in and the project doesn't stabilize. Taking the product scope and running it through iterations makes a significant difference when the key risk is market risk (understanding customer needs) and not technical risk.

Possible Improvements to the Model

1. The impact to team size is not directly reflected in this model. Increasing time sizes in a SCRUM (beyond a certain number) leads to a decrease in the quality of communication and hence less reduction in requirements uncertainty. This aspect can be modeled better.
2. Learning effects are not included in this model. When tasks move from **Done** to **Defects**, currently it is assumed that it takes the same amount of time to re-do the task. This is not always true in real life as there is some learning going on and the next time around, it will very likely take less time to finish. The model can be improved to handle this learning effect.

Chapter Conclusion

SCRUM and other agile methodologies have taken over the traditional waterfall model in the SaaS world. This is because the primary role of the SaaS company is to focus on the features that resonate the highest with the largest section of customers. The market risk

involved is much higher and methodologies such as SCRUM help mitigate this risk better than the traditional waterfall approach. The system dynamics model built in this section shows how the SCRUM methodology is superior in managing development uncertainty related to requirements risk. In this agile world, the Product Manager works much more closely with the development team than before and is responsible for continuously refining features sets during the development cycle. Past practices such as slipping schedule to meet a feature set are now replaced with pruning feature sets to meet a fixed schedule date.

5. Product Feature Ideation and Prioritization through Co-Creation

"In a world of abundant knowledge, not all smart people work for you."

Henry W Chesbrough

Traditional Product Development Process

Product Management is typically viewed as stage-gate funnel. The steps involved are ideation, scoping, business case development, development, testing and launch [Cooper 2010]. Different authors [Ulrich and Eppinger 2003] label the stages of the funnel using different terminology but the core stage gate approach remains the same.



Figure 18: Product Development Stage Gate

The engagement of the customer varies with each stage, but typically a customer subset is involved in the ideation stage and through the testing and launch stage. The traditional approach to ideation requires sampling the customer base with questionnaires and surveys where customer tell surveyors what they want and what frustrates them. As well intentioned as this process is, it suffers from the following flaws

1. It views value creation as a firm-centric activity and customers as passive recipients of new product innovation
2. Surveys and questionnaires require sampling and this approach doesn't always elicit the right ideas

3. The traditional ideation process produces a bias towards the most advanced users. These users are not necessarily representative of the greatest number of customers
4. The importance attributed to an idea is dependent on the product manager's own situational biases and there is no external filter to validate the relevance of ideas prior to testing and launch

The root cause of many failed product initiatives is due to the inability to capture the customer inputs that are needed to successfully manage innovations [Ulwick 2003]. The traditional approach to ideation and new product development is ill suited for the dynamic nature of the SaaS world.

Collaborating with Customers

As opposed to closed innovation, that assumes a strategy of vertical integration and exclusive control [Chesbrough 2003], co-creation strategies are better suited to the service innovation that SaaS entails. *"People are inherently creative and want to engage with organizations; they don't want to have products and processes imposed on them"* [Ramaswamy and Gouillart 2010]. Through Internet enabled virtual communities, customers are not only able to participate in ideation but are also able to communicate, share and shape their experience with the product platform.

Customer engagement through virtual platforms provides enhanced reach, interactivity, flexibility and speed for collaborative product innovation. One key area where the SaaS firm can benefit from is the area of ideation. Through virtual communities, the SaaS enterprise has a new interaction channel through which many more customers could be incorporated in the ideation process without incurring significant cost penalties associated

with traditional survey mechanisms. Secondly, by engaging a much wider section of the target base, the risk of being hijacked by advanced users can be reduced. Finally, the virtual community can be tapped to filter ideas by their expressions of support through voting, comments etc. These provide invaluable knowledge about the exact nature of the customer need rather than the product manager having to guess based on survey and questionnaire. Using Customer suggestions to make product management decisions is a recent trend that has been embraced in the Software as a Service industry.

Creating virtual communities that can actively participate in ideation requires trust building that includes dialogue, access, risk-assessment and transparency. This methodology (DART) proposed by [Prahalad and Ramaswamy 2004]. Similar approaches are actively being used by many SaaS companies.

When an organization opens up for suggestions, it is important to understand what key influences a product manager must take into account to implement user suggestions. Care must be taken to ensure that the wisdom of crowds doesn't degenerate into the tyranny of the masses. Adopting co-creation strategies actually helps to improve the perception of firm's innovativeness.

In addition to improving mutual learning mechanisms between customer and provider, customer relationships are better enabled and the co-creation process becomes an effective marketing channel to better communicate innovative aspects of the product offering. This enhances customer loyalty with the SaaS platform, thereby customer churn risk and increasing lifetime value of the customer. [Stoyan Tanev et al 2011]

The co-creation ideation process is still in its nascent period and this area of SaaS platform can benefit from the application of data mining and machine learning techniques to enhance the product manager's effectiveness in identifying relevant features for his or her SaaS application.

Case Background

The trend in the SaaS industry has been to focus on customer led innovation where the features enhancements to a product are suggested by the end users themselves and the leading proponent of such an approach is Salesforce.com and its Idea exchange platform. In this case study, we will use data extracted from Salesforce.com's idea exchange platform and apply data mining techniques to aid in the identification of feature suggestions that are promising. The output will be classification based on historic data that is currently available to gauge/model product management decisions.

Source of Data:

Salesforce.com is a leading player in the Software as a Service field and it has led the field in the area of crowd sourcing as a way to handle new feature introduction. Typically in any release, a fair number of the features that are released come from end user suggestions. Salesforce.com's Idea Exchange is an online platform where the suggestions from end user are posted. This is a public database where the suggestions are listed and tagged with the feature / area that it belongs to. It is also tagged with information such as user, number of comments that it received and the community votes such a suggestion received. For each of the user it is also possible to view characteristics of the user such as the number of comments they have made, how many answers to technical problems they have provided in the community.

Pre-Analysis Data Preparation

The dataset was not directly available in a structured form; it was however accessible through a web interface where the user can query for the suggestions. The public access to the suggestions is restricted to 1000 suggestions and the analysis is based on those 1000

records. Please refer to the appendix for the layout of the webpage where the data was extracted from. The author wrote a web-crawler using PERL language to crawl the Salesforce web page and extract data from the page to the following structured format.

Data Field	Type	Explanation
No	Numerical	The recorded
Tags	Categorical	A list of all the functional tags associated with the idea.
Title	Text	The summary of the idea
Votes	Numerical	The number of community votes that the idea garnered
Comments	Numerical	The number of comments that the idea generated from the community
Status	Categorical	The current status of the idea, such as no action, implemented, under consideration etc.
Date	Date	Date when the idea was submitted
User_Name	Categorical	User who submitted the idea
UserideaCount	Numerical	The number of ideas this particular user has submitted
UsercommentCount	Numerical	The number of comments on the forum this user has made
UserreviewCount	Numerical	The number of reviews this user has contributed
Userreview_CommentCount	Numerical	The number of comments on reviews (discussion threads) this user has contributed to
UseranswerCount	Numerical	The number of answers to forum questions the user has contributed
UserbestAnswerCount	Numerical	The number of answers contributed by the user that were vote the best answer

UservoteCount	Numerical	The number of votes that this user has cast
----------------------	-----------	---

Data issues and Resolutions

The following issues were encountered with the data and were resolved using the techniques in data mining.

1. The Tag field has many values and needed to be converted to a binary array. Each Tag was assigned a separate column and if an idea has a tag it was given the value "1" else "0". This effectively produced about 52 binary variables, each dealing with a different Functional area or Tag (an idea can belong to several functional areas)
2. The status response variable contained too many classes. What is of interest is whether the feature was implemented or not. So a new variable called Response was introduced that was binary. It took on the value 1 if the feature was implemented or under consideration etc. It was zero if there was no action on product management on the idea. This would become the dependent variable that was tracked.
3. A new variable tracking the ratio of the votes to the comments was introduced for use as an explanatory variable.

Exploratory Data Analysis

The final data set had 1000 rows of data with 65 explanatory variables. TIBCO Spotfire was used to explore the data set prior to applying classification algorithms.

Plot of votes per idea

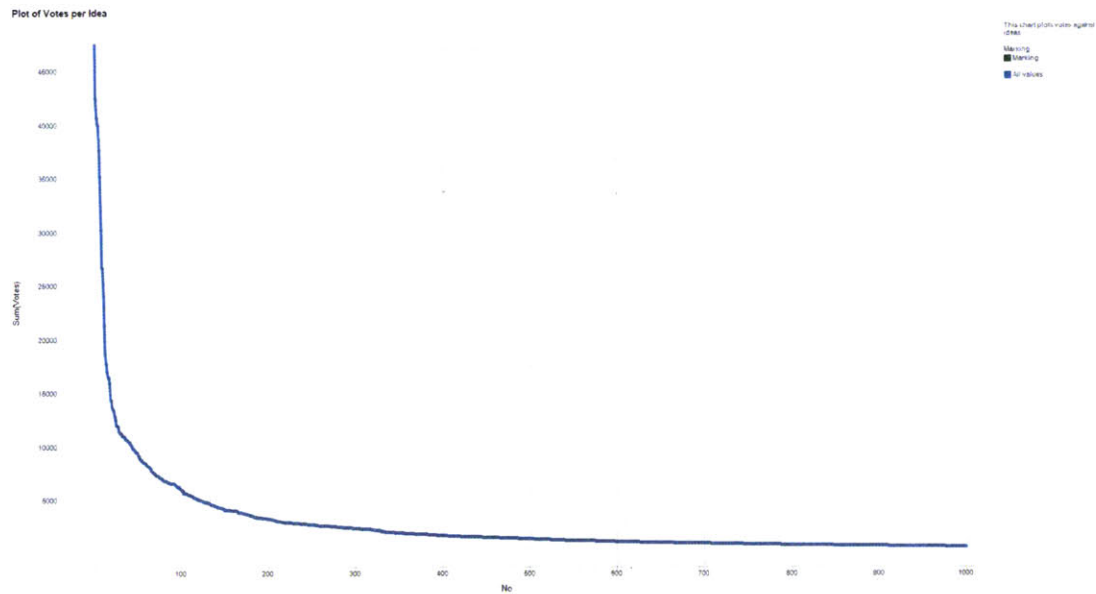


Figure 19 : Plot of votes per Idea

As shown in the above graph, the distribution of votes seems to follow a power law with a very long tail. This kind of a distribution is to be expected given the nature of the crowdsourcing and user collaboration on the Internet. It was very satisfying to see the pattern emerge from the data as opposed to intuitive guesswork.

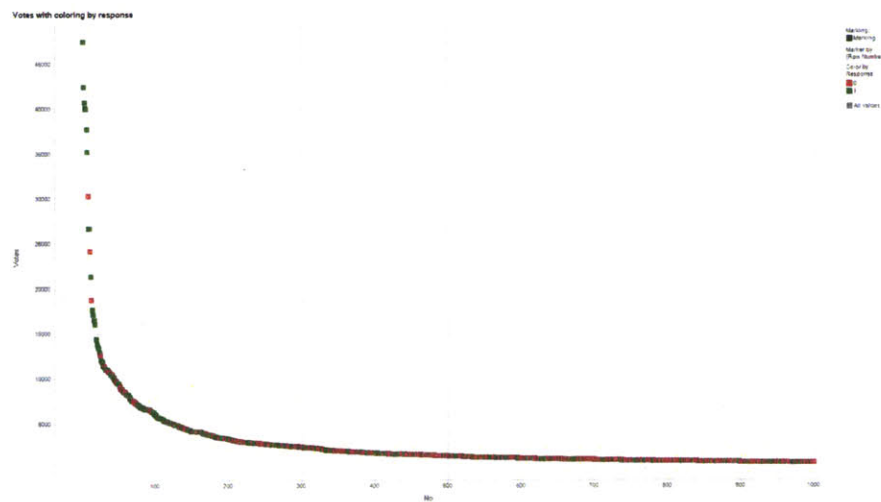


Figure 20: Plot of votes with Implementation

In the graph above, the colors represent the dependent variable, Response. A Red color indicates the situation where the feature was ignored and green indicates when the feature was implemented. As the data shows, it is important not to ignore the tail of the distribution and implement features across the entire tail as they represent higher number of customers.

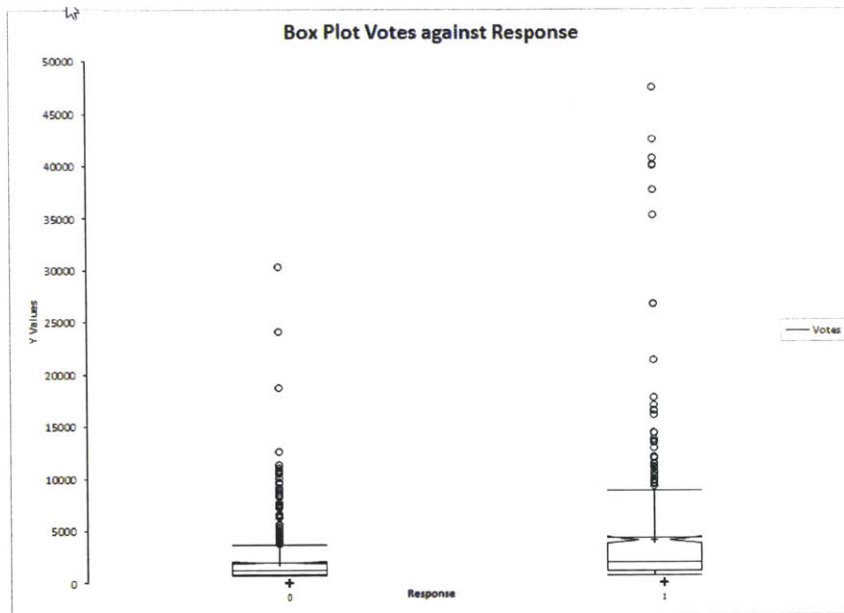


Figure 21: Box Plots of votes and implementation

The box plot of votes against response indicates that there is generally a higher trend in the number of votes for ideas that are implemented (Which is to be expected as this is user led innovation). The mean votes for non-implemented ideas are 1958 and that for implemented ideas is 4058. However, given the long tail distribution, the means are sensitive to the outliers.

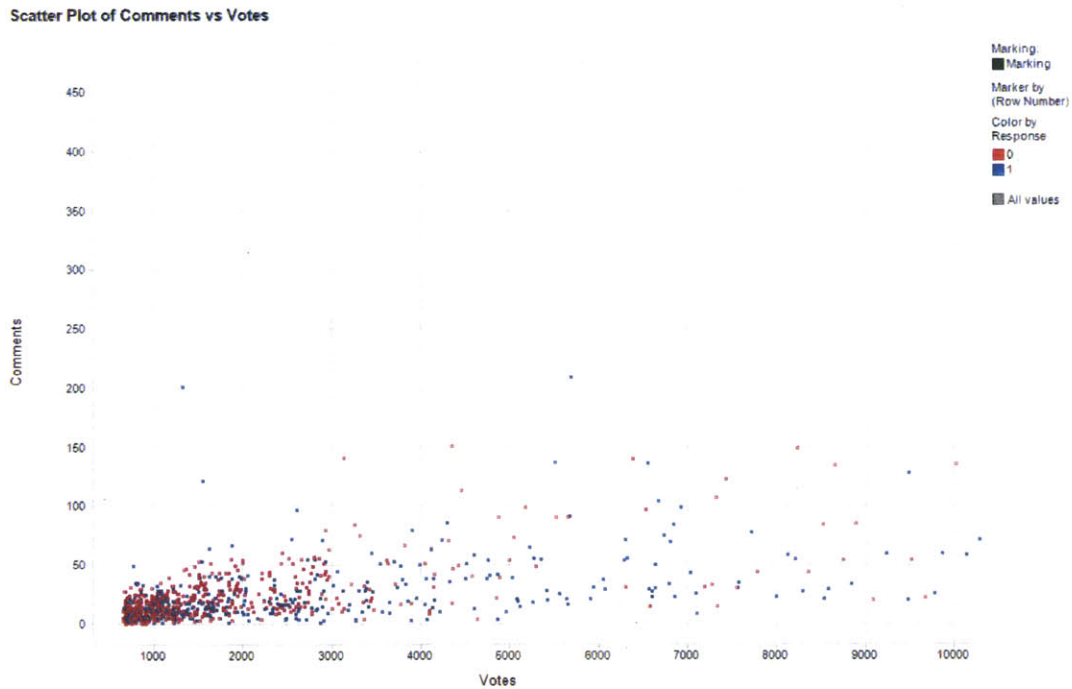


Figure 22: Scatter Plot of Comments vs. Votes

Analysis of Dataset

Affinity Analysis to validate the tagging taxonomy

The binary matrix generated by the Tagging field was used to run affinity analysis to extract rules. **The results indicated a large amount of redundancy in the Tagging Taxonomy.** This is likely due to the fact that the Tagging taxonomy evolved as the product grew and a proper review of the Tag cloud has not been undertaken. For instance, the Tags Applications, Large Enterprise, Large Enterprise Ideas, Force.com platform seem redundant and can be eliminated.

Unlike traditional affinity analysis which is used to club items of economic value together, in my case, **very strong affinity actually indicates poor tagging diversity and can be used to fine-tune the tag hierarchy.**

Rul e #	Conf. %	Antecedent (a)	Consequent (c)	Support t(a)	Support t(c)	Support t(a U c)	Lift Ratio ↓
10	100	Administration_AND_Sharing, Large_Enterprise=>	Force.com_Platform, Large_Enterprise_Ideas	138	649	138	1.540832
11	100	Customization, Large_Enterprise_Ideas=>	Force.com_Platform, Large_Enterprise	206	649	206	1.540832
12	100	Administration_AND_Sharing, Large_Enterprise_Ideas=>	Force.com_Platform, Large_Enterprise	138	649	138	1.540832
13	100	Customization, Large_Enterprise=>	Force.com_Platform, Large_Enterprise_Ideas	206	649	206	1.540832
21	100	Customization, Large_Enterprise_Ideas=>	Force.com_Platform	206	672	206	1.488095
22	100	Administration_AND_Sharing, Large_Enterprise=>	Force.com_Platform	138	672	138	1.488095
23	100	Administration_AND_Sharing, Large_Enterprise_Ideas=>	Force.com_Platform	138	672	138	1.488095
24	100	Customization, Large_Enterprise, Large_Enterprise_Ideas=>	Force.com_Platform	206	672	206	1.488095
25	100	Administration_AND_Sharing=>	Force.com_Platform	143	672	143	1.488095
26	100	Customization, Large_Enterprise=>	Force.com_Platform	206	672	206	1.488095
27	100	Customization=>	Force.com_Platform	210	672	210	1.488095
28	100	Administration_AND_Sharing, Large_Enterprise, Large_Enterprise_Ideas=>	Force.com_Platform	138	672	138	1.488095
46	100	User_Experience=>	Large_Enterprise	226	944	226	1.059322
47	100	Administration_AND_Sharing, Force.com_Platform, Large_Enterprise_Ideas=>	Large_Enterprise	138	944	138	1.059322
48	100	Large_Enterprise_Ideas, Reports_AND_Dashboards=>	Large_Enterprise	147	944	147	1.059322
49	100	Force.com_Platform, Large_Enterprise_Ideas, Reports_AND_Dashboards=>	Large_Enterprise	143	944	143	1.059322

50	100	Applications, Force.com_Platform, Large_Enterprise_Ideas=>	Large_Enterprise	228	944	228	1.059322
51	100	Applications, Large_Enterprise_Ideas, Sales_Force_Automation=>	Large_Enterprise	150	944	150	1.059322
52	100	Customization, Force.com_Platform, Large_Enterprise_Ideas=>	Large_Enterprise	206	944	206	1.059322
53	100	Large_Enterprise_Ideas, Sales_Force_Automation=>	Large_Enterprise	151	944	151	1.059322
54	100	Applications, Large_Enterprise_Ideas=>	Large_Enterprise	445	944	445	1.059322
55	100	Force.com_Platform, Large_Enterprise_Ideas=>	Large_Enterprise	649	944	649	1.059322
56	100	Large_Enterprise_Ideas, User_Experience=>	Large_Enterprise	227	944	227	1.059322
57	100	Delivered_Ideas=>	Large_Enterprise	150	944	150	1.059322
58	100	Delivered_Ideas=>	Large_Enterprise, Large_Enterprise_Ideas	150	944	150	1.059322
59	100	Delivered_Ideas, Large_Enterprise_Ideas=>	Large_Enterprise	150	944	150	1.059322
60	100	Administration_AND_Sharing, Large_Enterprise_Ideas=>	Large_Enterprise	138	944	138	1.059322
61	100	Customization, Large_Enterprise_Ideas=>	Large_Enterprise	206	944	206	1.059322
62	100	Administration_AND_Sharing, Force.com_Platform, Large_Enterprise=>	Large_Enterprise_Ideas	138	945	138	1.058201
63	100	Customization, Large_Enterprise=>	Large_Enterprise_Ideas	206	945	206	1.058201
64	100	Force.com_Platform, Large_Enterprise, Reports_AND_Dashboards=>	Large_Enterprise_Ideas	143	945	143	1.058201
65	100	Applications, Large_Enterprise=>	Large_Enterprise_Ideas	445	945	445	1.058201
66	100	Force.com_Platform, Large_Enterprise, User_Experience=>	Large_Enterprise_Ideas	226	945	226	1.058201
67	100	Delivered_Ideas=>	Large_Enterprise_Ideas	150	945	150	1.058201
68	100	Delivered_Ideas, Large_Enterprise=>	Large_Enterprise_Ideas	150	945	150	1.058201
69	100						

70	100	Applications, Large_Enterprise, Sales_Force_Automation=>	Large_Enterprise_Ideas	150	945	150	1.058201
71	100	Applications, Force.com_Platform, Large_Enterprise=>	Large_Enterprise_Ideas	228	945	228	1.058201
72	100	Administration_AND_Sharing, Large_Enterprise=>	Large_Enterprise_Ideas	138	945	138	1.058201
73	100	Large_Enterprise, Reports_AND_Dashboards=>	Large_Enterprise_Ideas	147	945	147	1.058201
74	100	Large_Enterprise, Sales_Force_Automation=>	Large_Enterprise_Ideas	151	945	151	1.058201
75	100	Customization, Force.com_Platform, Large_Enterprise=>	Large_Enterprise_Ideas	206	945	206	1.058201
76	100	Large_Enterprise, User_Experience=>	Large_Enterprise_Ideas	227	945	227	1.058201
77	100	Large_Enterprise=>	Large_Enterprise_Ideas	944	945	944	1.058201
78	100	Force.com_Platform, Large_Enterprise=>	Large_Enterprise_Ideas	649	945	649	1.058201
127	51.24	Applications, Large_Enterprise, Large_Enterprise_Ideas=>	Force.com_Platform	445	672	228	0.76244

Classification using data mining techniques

Now the data set was partitioned into Training, Validation and testing data (500 Training, 300 Validation and 200 Testing) and different data mining techniques were applied to determine if the dataset could potentially be mined for features that should be implemented. The techniques that were used are

1. Discriminant Analysis
2. Logistic Regression
3. K nearest neighbors
4. Classification and Regression trees

The table below compares the Lift charts of the respective techniques for the Validation and Test partitions.

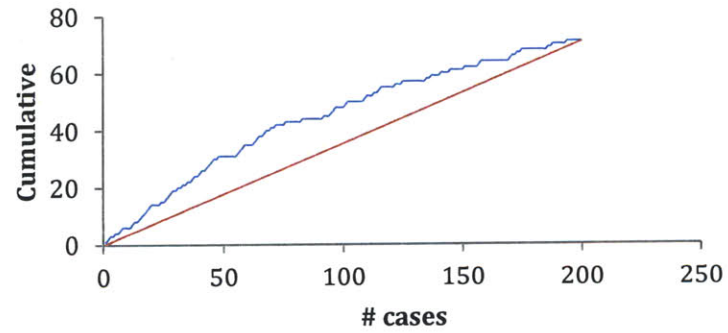
Technique

Test Lift Chart

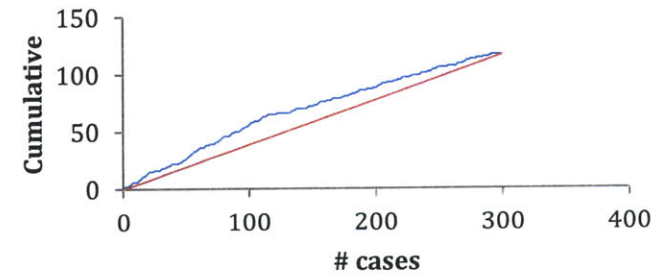
Validation Lift Chart

Discriminant analysis

Lift chart (test dataset)

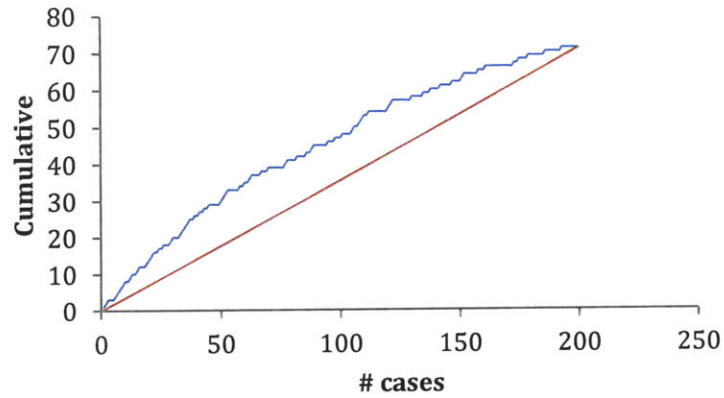


Lift chart (validation dataset)

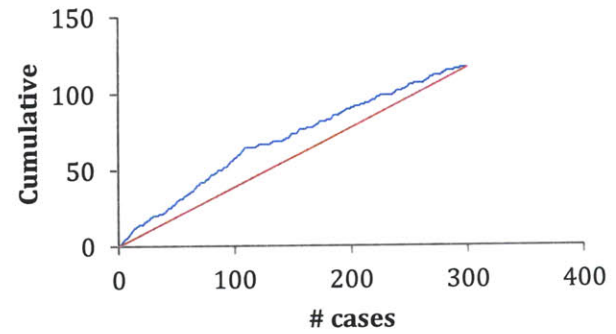


**Logistic
Regression**

Lift chart (test dataset)

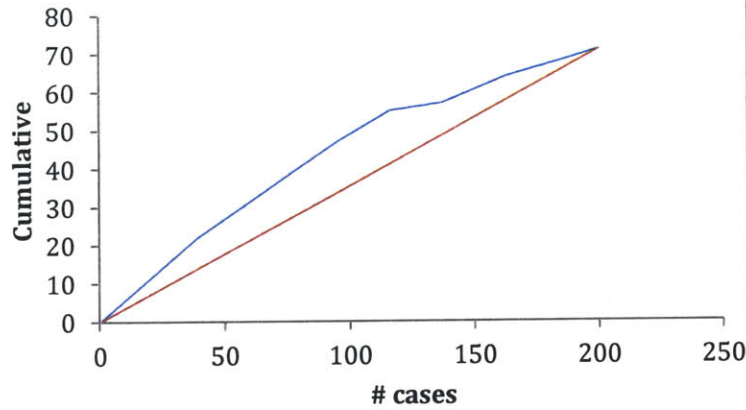


Lift chart (validation dataset)

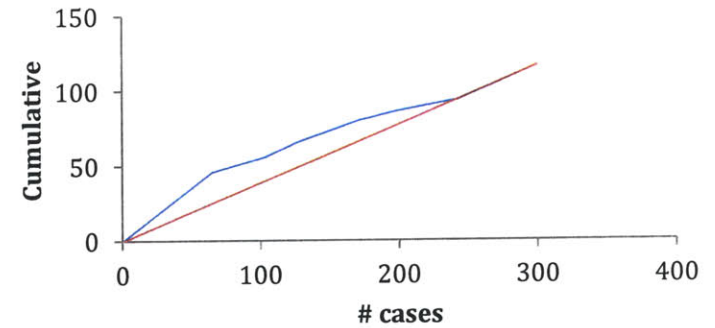


**Classification
Trees**

Lift chart (test dataset)



Lift chart (validation dataset)



**K-Nearest
Neighbors
(Best K = 5)**

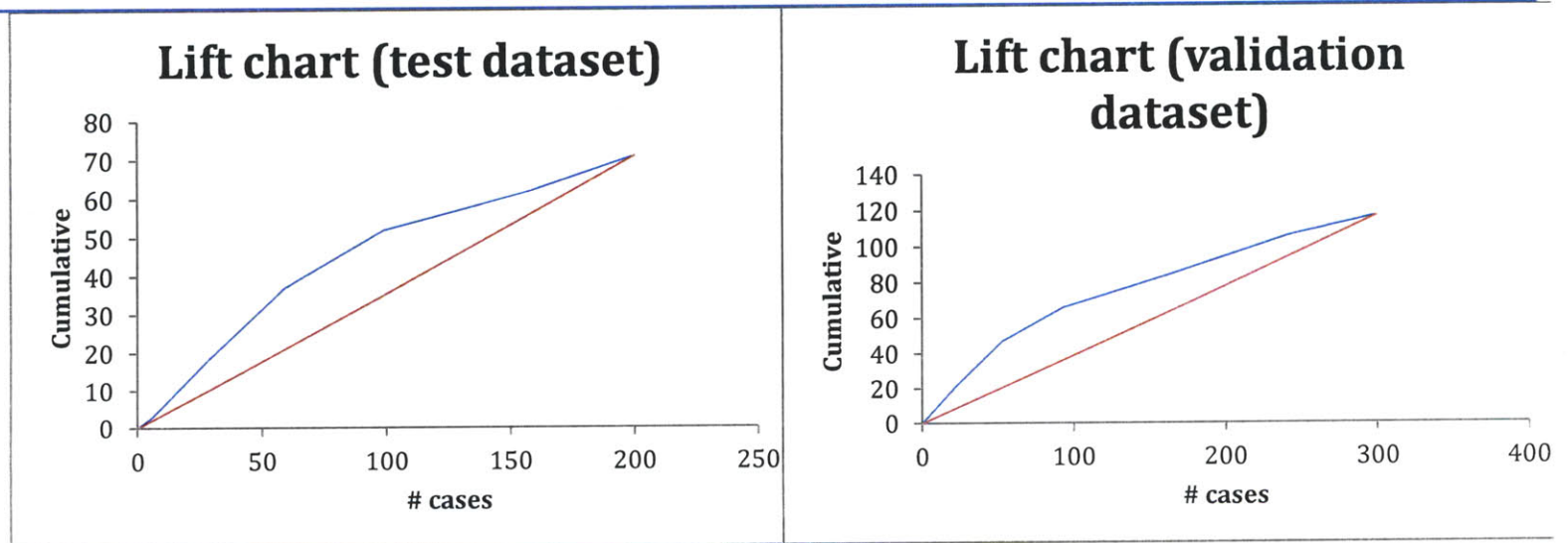


Figure 23: Lift Charts of different data mining techniques

Of the techniques investigated, the K-Nearest Neighbors with a value of K=5 provides the best ability to predict whether a feature should be implemented or not.

Chapter Conclusion

Engaging end users to suggest features for product iterations is a good approach to ensuring that the product is sensitive to market demands.

Key takeaways

1. When soliciting user ideas, the nature of the idea distribution tends to follow a long tail distribution. It is important for a product management organization to take care of the tail end of the distribution and not just focus on the most vocal users. As the Salesforce.com data indicates success likes in being able to discriminate and satisfy different sections of users.
2. It is important to develop a tagging taxonomy for feature suggestions and actively manage the tagging taxonomy. As product features emerge, use techniques such as affinity based rule extraction to validate and improve the tagging taxonomy,
3. It is useful to use data mining techniques to get an indication for the value of the suggested idea for product management. Such an approach could save product management search time and focus on the important ideas more quickly. This could be implemented as a dashboard that the product manager can look at to establish feature priorities based on user suggestions. Although Logistic Regression or K nearest neighbors looks good techniques to handle this, it may be better to adopt a bagging approach for determining an idea's value.

6. Using Customer Experience Data in Software as a Service

"Data is the new plastic", Om Malik

Product Usage data in SaaS

Within the SaaS platform, there is a significant potential to monitor customer usage and behavior. Unlike traditional software development, the product is being used on the servers run by the provider thus allowing for a wide range of client behavior analysis. This however must be done with significant consideration to client privacy as SaaS business model relies on trust. The end user has traded convenience for some privacy but this should be taken into account while designing monitoring schemes.

Product usage as the central measure of client behavior is nothing new. This has been dealt with in fair amount of detail in the consumer market. Product usage within SaaS can be used critically in a couple of junctures. Firstly, it is applicable during the testing and validation stage prior to feature rollout. Ask.com perform such testing on platforms such as Ustesting.com to perform this. This allows the product manager to study end user behavior with the actual feature developed and make adjustments as necessary. Secondly, in the post roll out stage, the product usage can be monitored to glean intelligence on feature viability and also as a source for new feature ideas that are not readily expressed by the end user.

Product usage in the SaaS world can be done on two dimensions,

1. Usage Frequency. A measure of how often the client makes use of the application

2. **Usage Variety.** A measure of how deep the usage of the product within the client organization is.

(Adapted from Consumer behavior concepts, presented in [Ram and Jun 1989])

Usage Frequency has been traditionally measure in web-site analytics as “Visits” to a page.

Third party analytics companies such as Google analytics often support tracking this.

However, unlike a web site where the primary value is content delivery, SaaS application focuses on interactivity to accomplish business oriented tasks. Purely, looking at “Visits” to a page is not sufficient to measure feature usage.

Usage Variety is a measure how varied the end-user relationship to the application is? For instance, does the end user modify business objects in the SaaS application through multiple modes of usage (Web browser, mobile phone, tablet). Does the end user link internal systems through vendor provided APIs?

It is important to conduct usage measurement in SaaS on two different levels

- a) **Intra-tenant behavior.** How do users within a single client organization (tenant) make use of feature portfolio within a SaaS application?
- b) **High-level tenant level behavior.** I.e. how does the tenant organization behave in relation to other tenants within the same SaaS infrastructure?

Within the traditional website analysis paradigm, the data for performing usage analysis is typically collected in a couple of ways

- a) **Tagging or Injecting application pages**
- b) **Looking at the log files of the web-server in the backend to look for interaction patterns**

For the case of SaaS monitoring, while both of these are applicable. It is also important to directly look at the business objects that are modified by the end user in their interactions with the application. A SaaS application as discussed earlier is a modern day incarnation of the MVC paradigm. While one can instrument the view (through tagging), look at log files (requests arriving at the controller), it is also crucial to see how the model layer is manipulated as well.

Customer Usage/Experience as a measure of feature performance

The usage of customer engagement data for making product management decisions in SaaS is a relatively new concept. [Latner and Ricardo 2011] proposed a framework where the SaaS product features are segmented according to their location relative to the value to cost trend line into: most valuable features, outperforming, underperforming and fledglings. The rationale in this approach is to potentially discard the underperforming features. However, the author contends that the scenario isn't that straightforward. While the mechanism proposed is good for portfolio analysis, it is not a foregone conclusion that underperforming features should necessarily be discarded. Underperformance might be related to other factors such as lack of awareness, lack of complementary services to exploit the feature etc. Thus the editorial action and judgment of the Product manager is still required in making these decisions.

Defining Customer Experience

It is important to take a systems thinking approach to customer usage measurement and approach the problem holistically from all layers of the SaaS application. Secondly, it is important to have a singular user authentication (identification) mechanism that will tie a

user across the SaaS application’s service front end, virtual community and the actual application. This will allow the product manager to not only measure a user’s experience but also be able to cluster the user with other representative users. This can be used later in the process of feature evaluation.

Instrumenting For and Measuring Customer Experience

Measuring the user perceived experience is important and current work has targeted these in separate silos. [Duan et al 2011] have proposed a tenant behavior analysis methodology wherein the mining of the usage data is done through the business object layer and collaboration indices around the business object in question are measured as a proxy for the feature usage or behavior of a tenant. This is an interesting approach, however it is but one part of a much larger analysis framework that is required.

The author proposes the following framework as a basis for instrumentation and measurement of customer experience through usage data.

	Instrumentation Tool	Analysis techniques	Value to Product Manager
Model	Business Object Tracking database. Each business object operation is written into a separate data store (NOSQL storage is ideal as this is write heavy)	Sequential pattern matching for object access User – Object graph to ascertain how users within a tenant collaborate	Identify collaboration patterns across class of users for new feature development Ascertain stickiness of an application or feature across a tenant organization
View	JavaScript Injection and tagging of product feature calls	Page start and Page end time analysis Sequential pattern extraction	Navigation patterns whilst a user is within the SaaS application can be ascertained

			<p>This information helps to validate user profile and helps gauge user suggestion and service requests</p> <p>Measure such as Time to Interact can be used to ascertain usability of features</p>
Controller	URL Logging	Time series analysis of View URL invocations	<p>Ability to project season trends on how product usage varies.</p> <p>Ability to predict when a particular feature will be used by an end user</p>

The key element in the above framework is the singular user identification that can be used across all the layer and services to ascertain client behavior.

Curse of Dimensionality

Based on the above framework, it is possible for the product manager to be overwhelmed with amount of data that is generated by the instrumentation. It is important that the product manager remain firmly in control of the mechanisms that he/she will think will provide new insight. This is a fine balance between exploratory data gathering and analysis and targeted instrumentation and tagging. Ultimately, some of this is determined by the organization's resource availability to perform such analysis. In light of the complexity of this problem, new startups such as Totango.com are now providing this analysis as a SaaS in its own right. The product managers the author interviewed presently are overwhelmed

by the amount of data that can be collected and often rely on their experience and editorial action to manage this complexity.

Chapter Conclusions

Usage data within the SaaS platform is a double-edged sword. One on hand it allows unprecedented insight actual customer interactions with the product. One the other hand, the sheer volume of data that can be collected can be overwhelming. Product Managers need to think about the user experience and craft instrumentation that measure specifically those user experience measures that they would like to track (due to lack of resources). Alternately, product managers in resource constrained SaaS companies can resort to third party services such as Totango and user testing.com to provide the analysis and feedback on these issues. The future of user data based portfolio optimization is not far off.

7. Conclusion

“When the wind blows strong, some people build walls, others build windmills”

Chinese Proverb

Innovative organizations live and die by the quality of their product management. Product Managers are responsible for feature prioritization, development resource allocation, vision and direction of the product platform. Being a Product Manager for a SaaS application is not the same job as that for an on-premise product.

The author started the work on this thesis based on the premise that Software as a Service is an 80/20 representation of the features that an on-premise application. In this view as more end users use the application, the relevancy of the product feature portfolio can be enhanced in a positive feedback loop. The following causal loop diagram shows this premise.

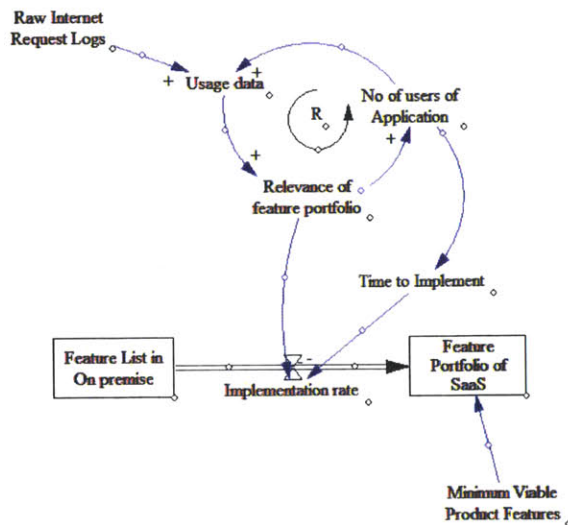


Figure 24: Causal loop of feature prioritization in SaaS

Based on the analysis that the author has done in this thesis, a more refined model of the role of product management within Software as a Service emerges

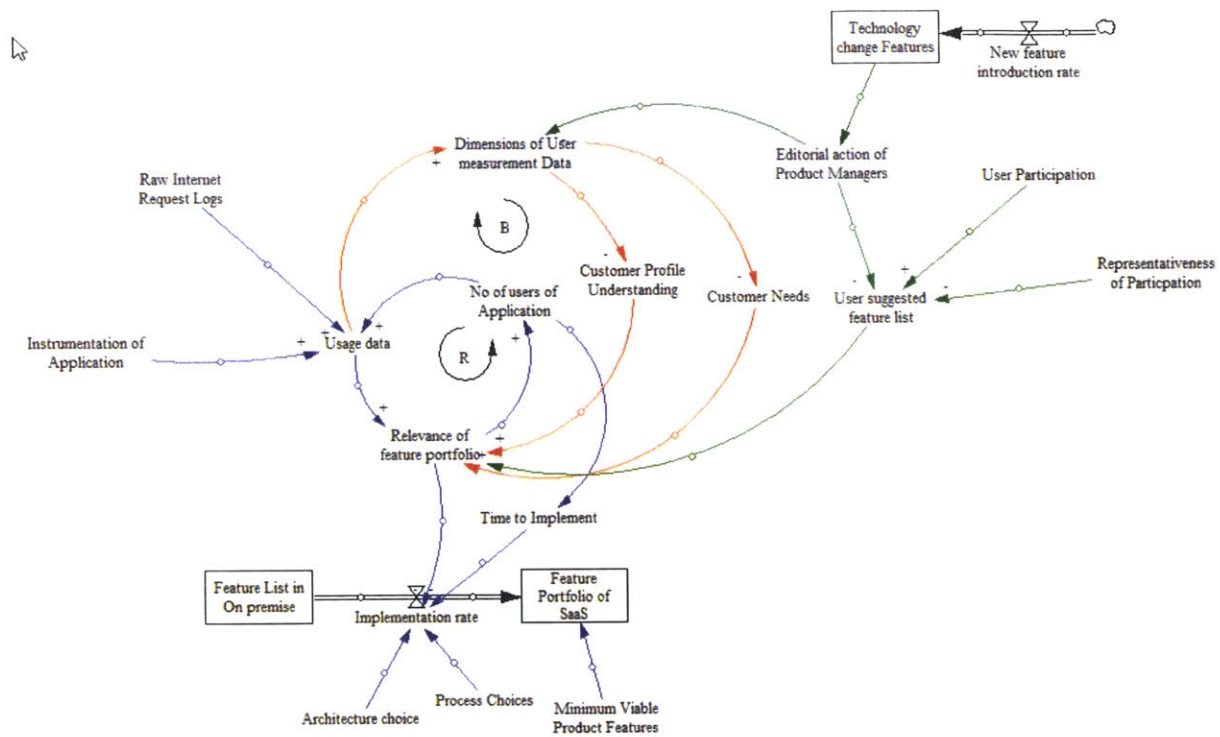


Figure 25: A holistic view of product management in SaaS [Causal Loop Diagram]

In this refined view we find that there are 5 separate individual areas (architecture, process, customer experience data, customer participation and editorial action of product manager) that contribute to the efficiency of product management operations within the SaaS. Each of these individual actors is important and their relative importance changes with context of operations. In a nutshell, SaaS dictates different product management priorities [Grant 2008] and requires product managers to adapt to this new innovative environment.

Bibliography

[Kirkos 2010] Alex Kirkos, Disruptive Technology Business Models in Cloud Computing, SDM Thesis, 2010, MIT

[Cusumano 2004] Michael A Cusumano, The business of Software, Simon and Schuster

[Cusumano, Kahl, Suarez 2006] Michael Cusumano, Steve Kahl, Fernando F Suarez, Product, Process and Service: A new Industry Lifecycle Model

[Utterback 1994] James Utterback, Mastering the Dynamics of Innovation, Harvard Business School Press

[Meyer 2007] Marc H Meyer, The Fast Path to Corporate Growth

[Cusumano 2007] Michael A Cusumano, The changing Labyrinth of Software pricing, Communications of the ACM.

[Cusumano 2008] Michael A Cusumano, The changing software business moving from products to services, Computer, IEE Computer Society

[Benlian, Hess, BuxMann 2009] Dr. Alexander Benlian, Dr. Thomas Hess, Dr. Peter Buxmann, Drivers of SaaS Adoption – An empirical study of different application Types, Business and Information system Engineering

[Christensen 1997] Clayton Christensen, The Innovator's Dilemma, Harvard Business School Press

[Barras 86] Richard Barras, Towards a theory of Innovation in Services

[Von Hippel 1998] Eric Von Hippel, Economics of Product Development by Users: The Impact of "Sticky" Local Information

[Brynjolfsson et Al 2003] Erik Brynjolfsson, Yu (Jeffrey) Hu, and Michael D. Smith Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers

[Krasner, Pope 1988] A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System, Glenn E. Krasner and Stephen T. Pope

[Lynch Gilbert 2002] Nancy Lynch and Seth Gilbert, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services", ACM SIGACT News, Volume 33 Issue 2 (2002), pg. 51-59.

[Brewer 2000] PODC Keynote, July 19, 2000 Towards Robust Distributed Systems Dr. Eric A. Brewer

[Chong, Carraro 2006] Frederick Chong and Gianpaolo Carraro, Microsoft Corporation, Architecture Strategies for Catching the Long Tail, <http://msdn.microsoft.com/en-us/library/aa479069.aspx>

[Bezemer Zaidman 2010] Cor-Paul Bezemer, Andy Zaidman. Multi-tenant SaaS applications: maintenance dream or nightmare? Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)

[Aulbach, Jacobs 2007] Ruminations on Multi-Tenant Databases, BTW Proceedings, volume 103 of LNI, Pages 514-521 Dean Jacobs, Stefan Aulbach

[Weissman, Bobrowski 2009] Craig D. Weissman, Steve Bobrowski Proceedings of the 35th SIGMOD international conference on Management of data, The design of the force.com multitenant internet application development platform

[Katzan, Dowling 2010] Harry Katzan, Jr., William A. Dowling, Software-As-A-Service Economics, The Review of Business Information Systems. Littleton: First Quarter 2010. Vol.14, Issue. 1; pg. 27

[Peltonen, Fruhwirth, Ronkko 2010] Examining the Effects of Agile Methods and Process Maturity on Software Product Development Performance, Mikko Rönkkö, Juhana Peltonen, and Christian Frühwirth, ICSOB 2011, LNBIP 80, pp. 85–97, 2011

[Tyrväinen, Selin 2011] How to Sell SaaS: A Model for Main Factors of Marketing and Selling Software-as-a-Service, Software Business, Pasi Tyrväinen and Joonas Selin, Lecture Notes in Business Information Processing 2011

[Mencke 2008] A Product Manager's Guide to Surviving the Big Bang Approach to Agile Transitions, Rasmus Mencke, Agile 2008 Conference

[Cooper 2010] Winning at New Product, Creating Value through Innovation, 4th Edition

[Ulrich and Eppinger 2003] Product Design and Development

[Ulwick 2003] The Strategic role of customer requirements in Innovation, Anthony W Ulwick, Strategyn Inc, 2003 - marketing4entrepreneurs.org

[Sawhney, Verona, Prandelli 2005] Collaborating to create: The Internet as a platform for customer engagement in product innovations, Journal of Interactive Marketing

[Chesbrough 2003] A better way to Innovate, Henry W Chesbrough, Harvard Business Review 2003

[Prahalad and Ramaswamy 2004] The future of Competition: Co-creating value with Customers, Prahalad, C.K., Ramaswamy, Venkat, Harvard Business School Press 2004

[Ramaswamy and Gouillart 2010] Building the Co-Creative Enterprise, Venkat Ramaswamy, Francis Gouillart, Harvard Business Review, October 2010

[Stoyan Tanev et al 2011] How do value co-creation activities related to the perception of firm's innovativeness? Stoyan Tanev, Tony Bailetti, Steve Allen, Hristo Milyakov, Pavel Durchev, Petko Ruskov, Journal of Innovation Economics 2011, Page 131

[Ram and Jun 1989] The link between involvement, Use innovativeness and Product Usage, S Ram, Hyung-Shik Jung, Advances in Consumer Research, Volume 16, 1989

[Latner and Ricardo 2011] Feature Performance Metrics for Software as a Service Offering: The case of Hubspot, ESD Working Papers, MIT

[Duan et al 2011] Tenant behavior analysis in software as a Service environment, Ning Duan; Xin Zhang; Wei Sun; Li Li; Ke Hu; IEEE International Conference on Service Operations, Logistics, and Informatics (SOLI), 2011

[Grant 2008] SaaS dictates different product management priorities, Tom Grant, Forrester Research

Appendix

Survey Questionnaire

SCRUM Survey Page 1

SCRUM Survey

* 1. I believe that there is more transparency in the organization since the introduction of SCRUM

- Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 2. In a sprint, how much effort (in % terms) do you spend in correcting errors or mistakes from previous sprints ?

* 3. Code quality has improved since the introduction of SCRUM

- Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 4. My productivity has improved since the introduction of SCRUM

- Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 5. Management's response to my feedback in sprint retrospectives is satisfactory

- Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

Enter Comments in the text box below

* 6. Since the introduction of SCRUM, I work less overtime hours

- Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 7. Since the introduction of SCRUM, my morale has increased

- Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 8. I think that the SPRINT duration is adequate for my work

- Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 9. SCRUM Methodology is applicable to all kinds of software projects

- Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

[Next](#)

SCRUM Survey Page 2

SCRUM Survey

* 10. SCRUM user stories contain enough detail for me to make good decisions

Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 11. The tools we use for SCRUM help in its implementation

Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 12. I have received adequate training on SCRUM

Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 13. Daily sprint meetings reduce uncertainty in my work

Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 14. In recent SCRUM sprints, my task duration estimates are accurate (<10% variance)

Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 15. I think SCRUM methodology will be effective with teams over 20 people

Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

* 16. SCRUM methodology is effective even when teams are not co-located

Strongly Disagree Disagree Neither Agree or Disagree Agree Strongly Agree

Any Additional Comments are welcome

* 17. My experience level with SCRUM

Less than 6 Months 6 Months to 1 Year 1 Year to 2 Years 2 Years to 3 Years More than 3 Years

18. General Comments on SCRUM Overall

Data Set Preparation

 promote

Multiple Contacts to be allowed on one activity, be it a task or event! Please! Title

 demote

Under Consideration Status

47530 points Votes

Idea Details

Our sales reps have meetings where they could literally speak to thirty contacts or have tasks where they could involve 30 contacts. We too, want to track that activity. We want our reps to get credit for all thirty of those talks but we all know our sales people, they don't want to take 20 to thirty minutes to log the same activity 30 different times. Having the ability to clone activities, tasks and events, and change the contact would be a huge improvement. The best solution would be the ability to add contacts in the same fashion you can add contact roles to an opportunity. This would mean that when running a report for the rep for their activity on the account this activity would show 30 times, once for each contact. The history would also show on the contact record. I cannot tell you how much easier this would make our reps lives!

***** UPDATE 7/26/2011 *****

We are in development on this idea. We do not have a firm ETA yet but are hoping for a pilot in the Spring '12 release (Q1 CY2012). Of course this is subject to change.

Thank you,
Kayvaan Ghassemieh | Senior Product Manager | Salesforce.com

Comments User Tags

246 comments Posted by **mticanuck** Applications, Calendar & Activity Management, Large Enterprise Ideas, Large Enterprise More than a year ago

A PERL Script based web-crawler was used to crawl through the web page and extract all the publically available ideas.