Feature Performance Metrics in a Service as a Software Offering

by

Avi Latner

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

May 2011

[June 2011]

Signature of Author_____

Avi Latner
Fellow
System Design and Management Program

Certified by_____

Dr. Ricardo Valerdi
Research Associate, Lean Advancement Initiative
Center for Technology, Policy, and Industrial Development

Accepted by_____

Patrick Hale
Director – System Design and Management Fellows Program
Senior Lecturer – Engineering Systems Division

Feature Performance Metrics in a Service as a Software Offering

by

Avi Latner

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

Abstract

Software as a Service (SaaS) delivery model has become widespread. This deployment model changes the economics of software delivery but also has an impact on development. Releasing updates to customers is immediate and the development, product and marketing teams have access to customer usage information. These dynamics create a fast feedback loop between developments to customers. To fully leverage this feedback loop the right metrics need to be set. Typically SaaS applications are a collection of features. The product is divided between development teams according to features and customers access the service through features. Thus a framework that measure feature performance is valuable.

This thesis provides a framework for measuring the performance of software as a service (SaaS) product features in order to prioritize development efforts. The case is based on empirical data from HubSpot and it is generalized to provide a framework applicable to other companies with large scale software offerings and distributed development. Firstly, relative value is measured by the impact that each feature has on customer acquisition and retention. Secondly, feature value is compared to feature cost and specifically development investment to determine feature profitability. Thirdly, feature sensitivity is measured. Feature sensitivity is defined as the effect a fixed amount of development investment has on value in a given time. Fourthly, features are segmented according to their location relative to the value to cost trend line into: most valuable features, outperforming, underperforming and fledglings.

Finally, results are analyzed to determine future action. Maintenance and bug fixes are prioritized according to feature value. Product enhancements are prioritized according to sensitivity with special attention to fledglings. Underperforming features are either put on "life-support", terminated or overhauled.

This page is intentionally left blank

Acknowledgements

I'd like to thank Research Associate Ricardo Valerdi, my advisor, for encouraging me to apply to conferences with this research and for knowing when to advise and when to be hands-off.

I'd also like to thank Pat Hale, Director of System Design and Management (SDM), and the SDM staff for creating a collaborative environment where students have the freedom to pursuit their own engineering and management related interests. There are many interesting places at MIT, more than one could explore in 16 months, but probably no other place is quite like SDM.

A special thanks goes to my wife and best friend Inbal, who is also my "chief-editor" for papers and blog articles. She is the reason this thesis is under 50 pages (otherwise, she wouldn't read all of it).

Finally I'd like to thank the people at HubSpot for being open to share knowledge and receive knowledge. Especially Yoav Shapira, an executive at HubSopt and SDM alumnus, who opened the door for me, and Brad Coffey who gave me hours of his time as well as many good ideas. Other contributors at HubSopt include Joss Poulton and Dan Dunn.

This page is intentionally left blank

Contents

Equations' Index

Figures' Index

Table's Index

Abbreviations' Index

BG	Blog Grader
CMS	Content Management System
COCA	Cost of Customer Acquisition
COGS	Cost of Goods Sold
KWG	Keyword Grader
LG	Link Grader
LTV	Customer Life Time Value
PG	Page Grader
R&D	Research and Development
SaaS	Software as a Service
SG&A	Sales General and Administration
SM	Social Media
MRR	Monthly Reoccurring Revenue
MVFs	Must Valuable Features

1 Introduction

In the past decade, with the increase of internet bandwidth, decrease in hardware costs, and a paradigm shift in business models, software delivery has been increasingly done by software as a service (SaaS) model. The software application is hosted at the vendor, licensed for a reoccurring subscription fee and accessed through a web browser. SaaS is not only a change in the business model but a change in the product development paradigm.

Since software typically resides on vendor's servers, it is easier for them to release updates at more frequent intervals. Coupled with agile development practices, applications are updated almost continuously without traditional version control. This also allows the vendor to collect valuable information about customer usage patterns. The information available is unprecedented in scope and immediate in availability. With a continuous deployment model and immediate customer response the feedback loop between development and customers has never been faster.

However, in order to fully leverage the fast feedback loop, companies must use the right performance metrics. Since most software suites are a collection of several features or applications and the development team's work is segmented according to features, performance should be measured and evaluated in at a feature level. In the context of this paper, features refer to high level functional subsections of the product. Some companies name these subsections applications or product-lines instead of features. Features are a fairly static segmentation of the product. Although new features may arrive and old ones be terminated the change is not rapid. Features do not refer to quick alterations which are referred to as feature enhancements in this thesis.

It is important to focus development efforts on the features where the investment will make the most impact on software usage and company profitability. For startups, often with limited cash flow and high uncertainty, this focus is all the more important.

In a SaaS model, much like other subscription based models, vendors use relationship marketing where customers are viewed as having a long term relationship instead of a series of discrete transactions. Hence, the marketers' objective is to acquire and keep customers.

This paper proposes a framework by which to prioritize product development efforts. This framework is explained in the introduction in sections 1.2 to 1.7. Section 1.1 is about HubSpot whose data was used to refine and implement the framework. Section 2 provides a background on the methods used in the framework. Section 3 describes the methodology. Finally sections 4 to 6 include the results of implementing the framework in HubSpot by way of analysis and conclusions.

By following the steps proposed in the framework a company can gain awareness of relative importance of product features and examine how well their past decision making is aligned with feature importance. A step beyond that would be for a company to make future prioritization decisions in light of these findings.

Two hypotheses are examined as a base for the framework. The first is connection between customer feature usage and customer retention. If this connection is demonstrated then usage information can serve as a predictor of customer retention. This premise is essential for

capturing feature value. The second hypothesis explores the correlation between feature investment and feature value.

Feature performance changes over time for external reasons such as emergent customer preferences and internal strategic priorities. If the methodology explicated in this thesis is followed, a company's focus may shift causing a change in performance assessment. Features that were once under-invested may receive more investment and, as a result, may increase in value. Another feature may mature and exert its full potential suggesting a shift in investment. Therefore, it is suggested that a company repeat this framework and re-evaluate the situation every time there is a significant change in the business environment (new competitor entering the market, new uses by customers, new capability available). Feature value, which mainly captures external shifts in the way the product is used, should be calculated as frequently as once a month. If the measurement process is fully automated this may even be done once a week. Measures that are also dependent on internal shifts in investment such as feature profitability and sensitivity should be calculated less frequently, perhaps once a quarter. Since investment is based on cumulative data a few months have to pass before significant change may be observed.

1.1 About HubSpot

This research was done using data collected at HubSpot, an online inbound marketing service. The service includes several features that help websites get found online by more qualified visitors, show customers how to convert more visitors into leads, give customers tools to close those leads efficiently, and provide analytics to help make smart marketing investments.

Inbound marketing, a term coined by HubSpot's founders (Haligan et al. 2010), refers to marketing techniques aimed at helping a company being found by customers, as opposed to outbound marketing, where companies reach out to the customer. The term is meant to convey a paradigm shift in sales and marketing in the age of web 2.0 and social media. According to HubSopt, traditional marketing executives specialize in purchasing advertisement in traditional channels: television, radio etc. Public relations firms are hired to get news of the company into the press. However, publishing content directly on the web through blogs, social media and websites is a cheaper alternative to those traditional methods. Good blog content and online conversations find their way into press without expensive public relation companies.

Sale representatives use direct emailing and cold calling to pursue customers. However potential customers are becoming increasingly better at blocking out unwanted sale calls and emails. Instead, in inbound marketing, the customers find the service they want actively on the web and then approach the company. It is based on this marketing and sale theory that HubSpot built an online SaaS offering by which marketing managers and small business owners can practice inbound marketing without being experts on web technologies and social media.

HubSpot's web offering is made of 17 features which can be segmented into four types. The first is content management tools that help maintain dynamic websites and blogs: *Content Management*, *Social Media*, *Blog*, and *Landing Page*. The second type is analytics that help track and analyze traffic: *Sources*, *Reach*, *Visits by Page* and *Competitors*. The third type is grader tools that use HubSpot algorithms to grade thing such as websites and blogs. For example a website's ranking in Google search depends on the number of links into the website

and their relative importance. *Link Grader* is a grader tool that grades each link according to its contribution to search engine optimization. Other than Link Grader this group includes *Keyword Grader*, *Blog Grader* and *Page Grader*. Many of the grader tools are also offered in the free HubSpot version along with other features, such as *Twitter Grader*, that are exclusively in the free version and are therefore not included in this analysis. The fourth type is features that help manage leads and turn them into customers, they are: *Leads*, *Prospects*, *Email*, *List Manager* and Lead *Nurturing*. HubSpot offers several products that differ in level of support. All products offer the same 17 features.

HubSpot is an interesting case study as it has a diversified service with many features, a rapidly expanding customer base, open management and it implements the latest development methodologies such as agile and lean startup. HubSpot development and other business functions are located at one site. This made HubSpot an ideal case to research since people could be interviewed in person. However, the framework introduced in this thesis is extendable to other companies including companies with distributed development activities. This research is less concerned with the specifics of each feature and more with a generalized method to evaluate and prioritize work across a feature portfolio set.

1.2 Usage as a Predictor of Retention Hypothesis

If customers that frequently use a given product are less likely to discontinue service we can use usage data of a given feature as a proxy for the impact that the feature has on customer retention. This connection is tested by the following hypothesis:

- H_0: No difference exists between retention of customers who use a feature and those who do not
- H_1: Customers who use a feature are more likely to retain service subscription

1.3 Value and Investment - the Loose Correlation Hypothesis

It is assumed a company, even without having implemented this framework, would have some understanding of feature value and would therefore strive to align development effort to feature value. If this assertion is true we should see a correlation between development effort and feature value. This assumption is formulized as the following hypothesis:

- H_0: No correlation between development effort and feature value
- H_1: A positive correlation exists between development effort and feature value

1.4 Measuring Relative Feature Value

SaaS product revenue is a function of the customer base and the product price. Customer base is a function of the number of new subscribers and the attrition rate. Therefore a measure of a feature value should actually be a measure of the impact that a feature has on these key parameters: customer acquisition and attrition rate.

As explained in detail in section 2.4, retaining existing customers is more important than acquiring new ones. Hence our research gives more value to retention rate impact than to

acquisition impact using a weighted average of 70% to 30% respectively. The exact weight should be a managerial decision and is a way to focus a company's priorities. A company more concerned with high attrition should choose a weighting, such as this one, with high preference towards retention. Whereas a company concerned with slow adoption should choose a more evenly weighted measure closer to 50% to each parameter.

This choice of parameter is validated through the hypothesis articulated in section 1.2. Our other measure of feature retention value is expert opinion survey done amongst business development and support staff within the company. Surveyed employees were asked to rate the top five most valuable features in the product to the customers. With no preference to either measure they were given equal importance.

Feature effect on customer acquisition is computed using two equally weighted measures. The first is the customer usage data in the first 30 days after subscription to the service started, since the usage in this early period reflects the features that made the customer subscribe to the service. The second is the expert opinion of the sales representatives. Salespeople were asked to rank the five most important features in closing a sale. The equations below summarize the measure calculations.

Equation 1 – Value Score Computation

$$\text{value}_i = 0.7 \cdot \text{retention score}_i + 0.3 \cdot \text{adoption score}_i$$

$$\text{retention score}_i = 0.5 \cdot \text{usage}_i + 0.5 \cdot \text{support poll}_i$$

$$\text{adoption score}_i = 0.5 \cdot \text{early usage}_i + 0.5 \cdot \text{sale poll}_i$$

Where *i* denotes a feature out of *n* features in a given SaaS offering.

Analyzing usage data in this way is valid in cases where all the features are client-facing, meaning that customers utilize the features by actively accessing them. When a SaaS product contains back office features, such as a product for network maintenance that has an automatic remotely triggered fix feature, a different measure must be used. Another example is user-automated reports that run without user interference. For example, Salesforce.com found a strong connection between user adoption of automated reports and retention. In their case a feature's value formula should also measure the amount of user automation. If value is calculated in weekly or monthly basis it is advised to repeat the value measures without surveying expert opinion. Expert opinion could be used to determine value of individual features but it should not be the primary basis for this analysis.

1.5 Measuring Feature Investment

Capturing cost is in principle straight forward. Many of the costs are feature specific whereas the value of individual feature is more difficult to quantify since the customer pays for the service as a whole. The lion part of the cost of a feature is development costs. Most of the development effort is spent on building, enhancing or debugging a given feature. Other

reoccurring monthly costs are also feature-specific. The costs of goods sold could be servers and storage. Cost is the accumulation of investment on a feature over the product history.

1.6 Measuring Feature Profitability

Profitability is, of course, a function of value and cost. To measure profitability the value and cost measures have to be comparable, that is measure the same set of features in the same units. One way to have value and cost in comparable units is to allocate all the revenue according to value and to do the same for costs. With costs that are not feature specific a pro-ratio allocation of costs could be used. If that is the case than difference between total feature value and total cost value should be the gross margin. However doing that would be a time consuming effort. Since this paper aims at giving a practical measure it is important that the measure be as simple as possible without compromising accuracy. Hence, it is advised to keep value and cost at relative terms by dividing 100% of value and cost amongst the features.

The profitability in equation 2 is not similar to the known accounting profitability, revenue minus cost. However, in our case, the value and cost are given in relative terms and they both add up to 100%. Using equation 2 we "force" the total profitability to be the gross margin and it is divided between all the features.

Equation 2 – Profitability Score Computation

$$\text{profitablity}_i = \text{value}_i - \text{breakeven value}_i$$

$$\text{breakeven value}_i = \text{cost}_i \cdot (1 - \text{Gross Margin})$$

Where *i* denotes a feature out of *n* features in a given SaaS offering.

1.7 Measuring Feature Sensitivity

Feature sensitivity is defined as the effect a fixed amount of development investment has on value in a given time. It is a measure of how effective recent development investments have been in improving features. Sensitivity is a dynamic measure that captures change between two time periods. One should use two measures of value and cost that are significantly apart, perhaps four or six months. Since feature value and cost described in sections 1.4 and 1.5 are measured in relative terms the average sensitivity would be zero and many of the features will have negative sensitivity. It is our experience that overall zero sensitivity can be counter-intuitive to some business managers. To prevent that, the value in a given time can be multiplied by the growth in customer base. This way than the average sensitivity score would be equal to the customer base growth rate. The overall sensitivity score will than reflect the product performance as a whole. The equation set below summarizes the sensitivity measure.

Equation 3 – Sensitivity Score Computation

$$\text{sensitivity}_i = \frac{(\text{value}_i)_t \cdot N_t - (\text{value}_i)_{t-1} \cdot N_{t-1}}{(\text{cost}_i)_t - (\text{cost}_i)_{t-1}}$$

Where

i denotes a feature out of *n* features in a given SaaS offering
t and *t-1* denote two subsequent time periods
N_t denotes the customer base in time period *t*

An alternative to using the customer base is using monthly reoccurring revenues. The total sensitivity score will change but feature's standing relative to other features will not.

2 Literature Review

Faced with the challenge of creating a framework for measuring feature performance in SaaS there are three main domain areas to research. The first is to understand best practices for building a measurement system regardless of the function that this system is set out to measure. Section 2.2 deals with setting up a measurement system. The second is looking at common software measurements specifically, measures that relate to feature value and performance. Sections 2.3 and 2.5 look at different aspects of software performance. The third area is about understanding the business implication of SaaS and how they shaped SaaS metrics in industry. Sections 2.1 and 2.4 examine SaaS and SaaS metrics.

2.1 Software Delivery – Shifting Towards a SaaS Model

A continuous trend in developing countries, and specifically in the US, is the move from an industrial economy to a service economy. Quinn (1992) shows that this trend has accelerated in the past 50 years to a point where services account for 80% of the US economy. As product companies are moving towards service areas, the line between a product and a service becomes ever more blurred. Even cars have added a service component such as GM's Onstar technologies. The Royal Society innovation in service report (2009) sites the extreme example of Rolls-Royce that moved from selling airplane engines to leasing propulsion services. Airliners pay per hour of use and Rolls-Royce's revenues throughout the service life-cycle exceeds the original sale price many times over.

We see a similar trend in the software sector where increasingly software packages are delivered over the web as a subscription service. With the prevalence of high speed internet access SaaS has become increasingly common. Dueby and Wagle's (2007) industry review claims that non-mission critical applications, with low data security, low need for customization and that require little integration with on-premise enterprise applications have already migrated towards SaaS. Such applications include Customer Relationship Management (CRM) and payroll. The next wave of applications to migrate is procurement, logistics and supply chain management software. Small and medium businesses are faster to adopt SaaS.

SaaS is primarily a change of business model for delivering and paying for software. Instead of a one-time charge software is offered on a reoccurring subscription base. Under the SaaS model software is delivered and accessed through a web browser reducing lead delivery time and eliminating the need for local installation. The change in payment method effects the way value is captured. As discussed in section 2.4, with SaaS value is achieved by maximizing customer retention.

However the impact of SaaS goes beyond licensing agreements and cash flow statements. SaaS also has product development implications. With the software residing with the publisher, it is easier for the publisher to release updates. Coupled with agile development, applications are updated almost continuously without version control. Vidyanand (2007) shows that SaaS publishers are incentivized to release features sooner, tend to invest more in product development, and that SaaS leads to higher quality and profit for the vendor.

Perhaps most important for this thesis, since the vendor hosts the application it can collect valuable information about customer usage patterns. This thesis is based on this new wealth of data. For SaaS providers this is a strategic advantage. Data from all the customers is available in real-time. Providers can use this information to improve product offerings and make quick iterations based on immediate feedback.

2.2 Performance Metrics – Creating a Measurement System

Constructing a measurement system should start with clear and defined objective. Then that objective is translated into measurable metrics. Hubbard (2010) points to a common mistake in measurement where companies measure that which is most convenient and not necessarily the most important. In these cases companies look at available and easily accessible information and start tracking that information. Hubbard claims that everything, including intangible attributes, is measurable. Therefore, companies should strive to create metrics which reflect the seemingly intangible business objectives. Value, one of the measurements in this thesis described at section 1.3, is not easily defined at a feature level. Defining what feature value means was the largest step towards measuring it. Similarly, with feature sensitivity coming up with a definition made measuring easier.

Another important aspect is weighing the cost of measurement against the value of the obtained information (Hubbard 2010). Many times what one measures is an indirect measure or a proxy that is less costly than a direct measure. To take Hubbard's example, Eratosthenes (276-194 B.C.) measured the circumference of earth by observing the length of shade cast at the library of Alexandria. Once a year, the sun would light the bottom of a well in Syene in south Egypt. That meant that this well was directly below the sun at that time. The shade cast by Alexandria's library at the same day at noon created a 7.5° angle. By multiplying the distance between the cities by 50 (=365/7.5), Eratosthenes got to an accurate estimate of the earth's circumference. In this example, Eratosthenes' approach was easier and faster than embarking on a trip around the word to directly measure its circumference. In the same manner, usage data collected for this research is not the "true" value of a feature but rather an indirect measure, relatively easy to capture, that is shown to be a good proxy.

Balance Scorecard (Kaplan et el. 1996), a well know management measurement systems, exemplifies the top-down approach in measurement. The method starts with stating strategic management objectives and then creates score cards to track performance. There are scorecards for financials, appearance to customers, internal business processes, and learning and growth. Each scorecard has lower level objectives derived from the overall vision and strategy. For each objective, a measure is set with a defined target.

With a clear objective at hand one can progress to creating an analysis model. McGarry et al. (2002) in *Practical Software Measurement* provide guidance on how to build and implement a model. This framework was used in building the model as well as implementing it in HubSpot. Beyond developing a measurement plan, the plan architects, must ensure that the company set a process in place to continually measure and then evaluate and improve. A systematic

feedback loop between measure and objective is what turns a measure into a management tool. Figure 1 illustrates the measurement process.

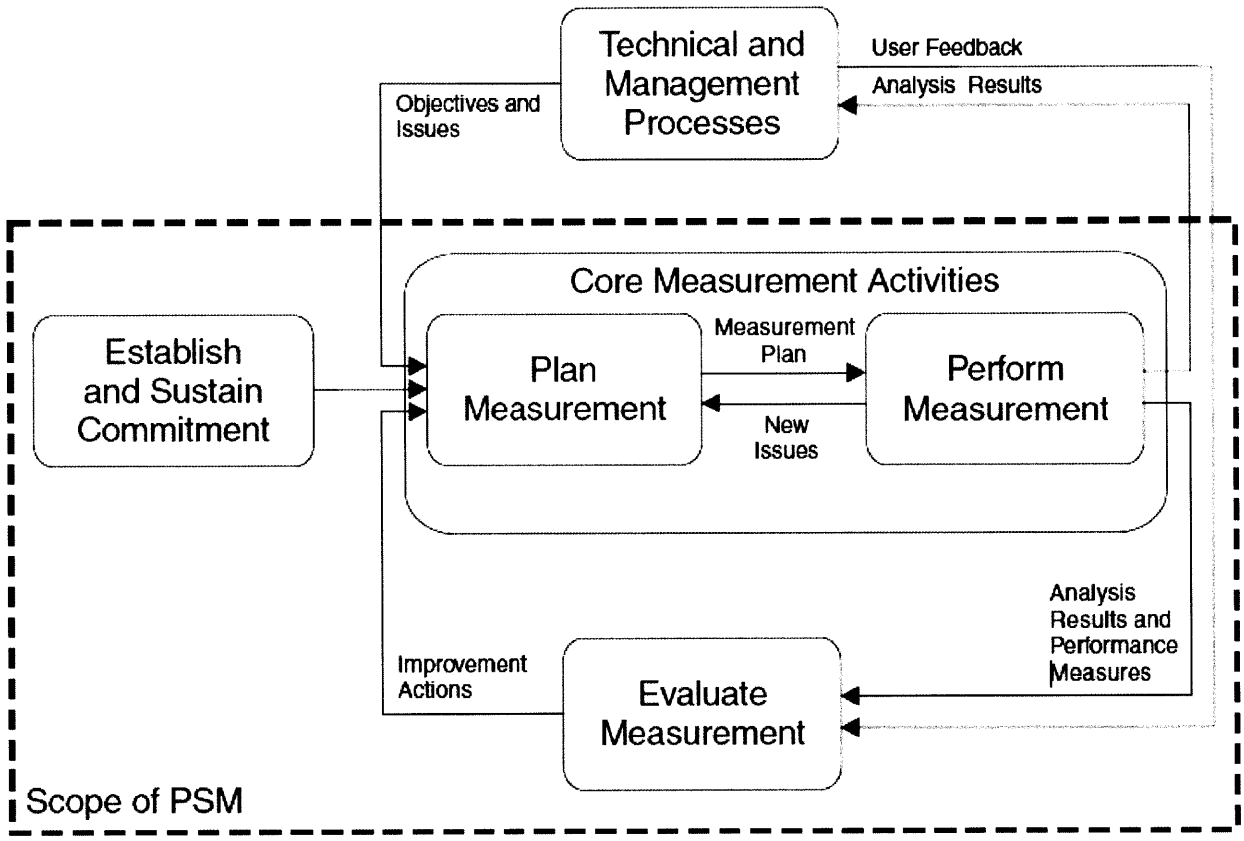

Figure 1 – Measurement Process (McGarry et al. 2002)

With a measurement process in place Practical Software Management describes, as shown in figure 2, how to create a measurement product. At the most granular level, there are systems-entities with attributes. In our case the entities are features, customers, user stories and COGS. At the level above it, there are base measures that are a measure of a single attribute gathered by a specific method. In our case, these are: feature usage; customer retention, acquisition and time elapsed since acquisition; story points; and feature COGS consumption. At the third level there are derived measures that are a function of two or more base measures. In our case the based measures are used to compute feature value, feature cost, feature profitability (which uses value and cost as inputs) and feature sensitivity. Finally the information is gathered into a model with specific decision criteria. In our case, the decision criteria are used to prioritize future development and assign resources to features.

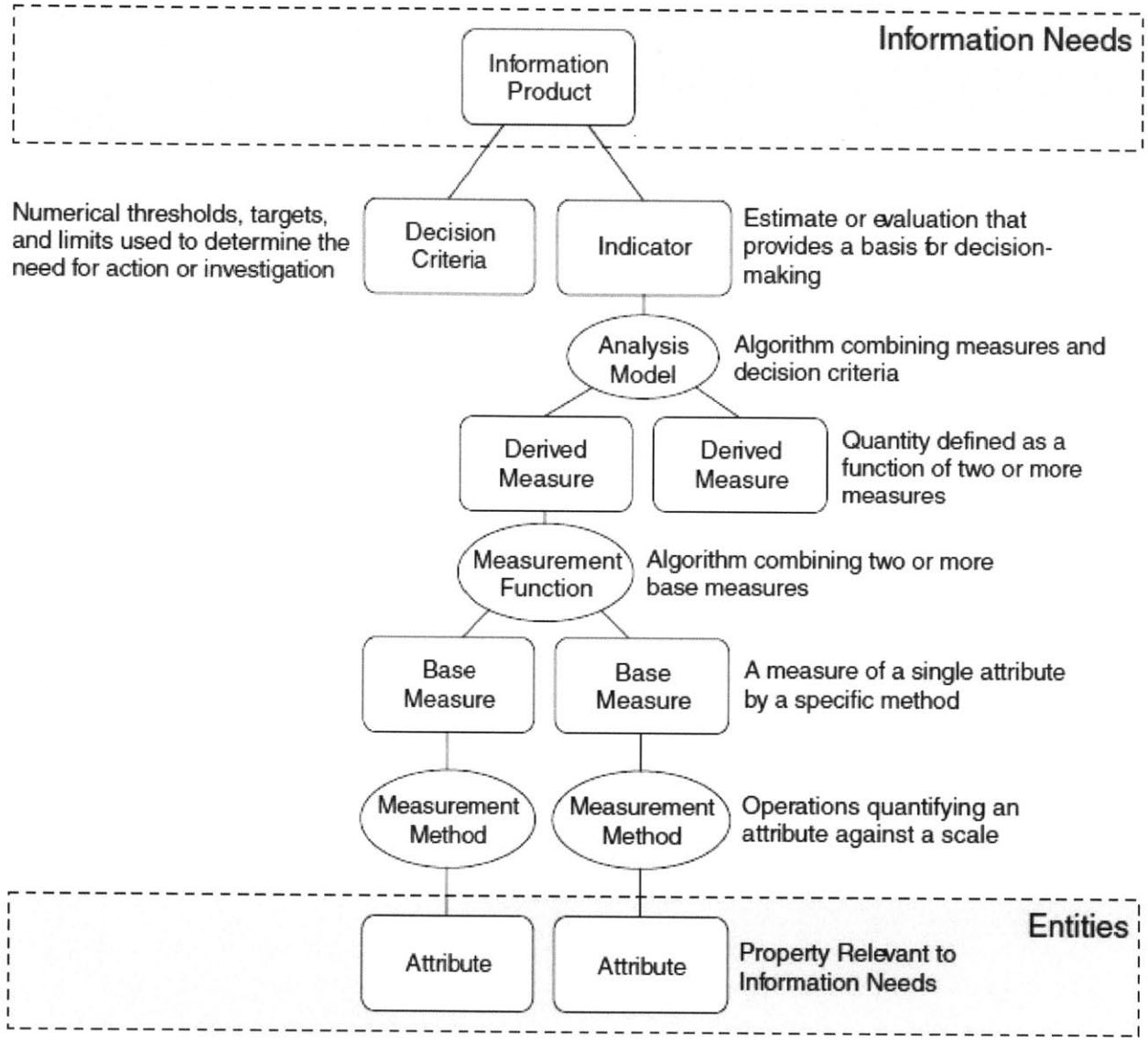

Figure 2 - Measurement System Construct (McGarry et al. 2002)

2.3 Software and Web Performance Metrics

There are numerous types of software related metrics and there is a tremendous body of research. Rather than addressing all of it, this section aims to map the general types of metrics widely applied to software. The first part of this section surveys the most common software metrics. The second part of this section specifically considers metrics for web and website performance. Research in web metrics is especially relevant since SaaS is delivered via the web.

Software metrics cover several aspects of software. They can measure the size of the software, its complexity, the efficiency of a development process or the programmers and the quality of the end product. Some metrics are of a predictive nature and are meant to reduce uncertainty and evaluate the effort and expected time involved in a software project.

The most straightforward measure of a program's complexity is lines of code (LOC). For that reason it is sometimes used as an input to derived measures like project effort. It is also used to normalize other measures, such as defects that are measured per fixed unit of LOC, and programmer productivity by LOC per time period. LOC is an alluring measure for its simplicity but it can hold many inaccuracies. Not all LOC are equal. For example assembly code requires many more lines than the lines that a higher level language requires for the same procedure. A given procedure can be implemented in several ways, each with a different length. The complexity for each way is the same, since they implement the same procedure. However, many times the shortest way is the least complicated. Even defining a line of code is not simple. For example, should it contain remarks? How do styling and formatting differences such as indents effect LOC count?

There are alternatives to LOC. Helstead (1977) measures complexity as a function of the number of operands and operators. Equation 1 shows Helstead's complexity based formula for effort. McCabe (1976) measures the cyclomatic number which is a graph theoretic complexity measure. The cyclomatic number is the number of linearly independent paths in a graph. McCabe recommends limiting the size of a graph to ten paths.

Equation 4 – Helstead's Program Complexity

$$E = \frac{n_{1+}\, N_2 * (N_1 + N_2) * log_2(n_1 + n_2)}{2n_2}$$

where

n_1 denotes the distinct number of operators in a program
n_2 denotes the distinct number of operands in a program
N_1 denotes the total number of operator occurrences
N_2 denotes the total number of operands occurrences

A common application for software metrics is in predictive models which use known attributes to predict the effort needed to complete a software project. One approach is to estimate by analogy to completed projects of a similar nature. Another is to induce effort by expert opinion using methods such as Delphi or surveys. A third way, perhaps the most empiric and scientifically grounded, are parametric models. These models are an indirect way of assessing effort using a base input (such as LOC) and a set of 'environmental' attributes. The most well known is COCOMO authored by Boehm (1981). COCOMO uses project, product and personnel attributes as cost drivers. For example, effort depends on the required software reliability. Many other parametric models stemmed from COCOMO. Equation 5 is the basic COCOMO formula.

Equation 5 – Parametric Effort Evaluation

$$E = a(KLOC)^b$$

Where the coefficient a and the exponent b are functions of 15 cost drivers

As already discussed, LOC has many variances that makes it difficult to standardize. Moreover, when predictions are made, in the beginning of a project it is almost as hard to estimate the expected program length as it is to predict the effort it would take. An alternative approach is functions points, described by Albrecht and Gaffney (1983). Function points are standard units of a program's size and they are a count of five measures: internal logical files, external interface files, external outputs, external inputs and external inquiries. Function points practitioners converted COCOMO and other parametric models to work of function points as a base measure.

Quality with regards to software is most commonly measured in defects per LOC. However, defects found are more of a measure of development process than its output. To measure product quality it is more interesting to know how many defects were not found but that is much harder to do. Kitchenham and Pfleeger (1996) point out that quality has different meanings depending on the viewpoint. A user thinks of quality in terms of reliability, which is the probability that a system performs the expected function, and usability. A manufacturer looks at the number of defects during the construction time. Kitchenham and Pfleeger also mention a value based quality view which is perhaps the closest to the approach taken in this paper. Another way to evaluate quality through the lenses of customer value is the Kano model (1984) which measures satisfaction against needs fulfilled differentiating between three curves: basic, performance and excitement.

For all the research done and wide uses of software metrics, few metrics have been demonstrated to be predictable or closely related to product or process attributes. Kitchenham et al. (1995) propose a validation method for software metrics and points to theoretical and practical shortcomings of many prevailing methods. Kearnay et al. (1986) claim that complexity measures are not well supported by experimentation and that these measures do not take into account other more important variables of real life projects such as programmer experience. Mohanty (1981) found software system project estimates varied widely depending on the model used.

The underlining fact is that software development is a creative process and therefore highly uncertain. Large variance between projects is inherent to software production. These inaccuracies and shortcomings should not discourage managers from using software metrics. Often, even though they are flawed, software metrics are the best way to reduce uncertainty related to software projects. However, in light of these shortcomings, software metrics should be used carefully as a decision support tool and not completely replacing subjective judgment.

Most web based internet measurement systems in the literature are applicable for e-commerce. Such are Van-Morsel (2001) quality of experience (QoE) and quality of business (QoB) metrics. QoE are quantifiable measures of the user experience such as successful

download completion rate, customer friendliness and end-to-end response time. QoB are business metrics such as cost of service and cost per transaction. Another research on web metrics (Palmer, 2002) examines how download delay, responsiveness, navigatability, site content and interactivity affect user's likelihood of return. Palmer's model is most applicable to websites and needs some adjustments to fully apply to software delivered through the web. Some of these metrics can be applied to SaaS as well. However most of the business metrics are applicable at the SaaS package level and not at the feature level.

2.4 SaaS Performance Metrics and Customer Loyalty

SaaS vendors turn to older industries that practice relationship marketing and adopted their metrics. In relationship marketing vendors view their relationship with the customers as a long term relationship instead of a series of discrete transactions (Berger and Nasr, 1998). This view has been increasingly adopted in the service sector where companies, such as credit vendors or airline operators, offer loyalty programs. Hence, the marketers' objective is to acquire and keep customers (Kotler and Armstrong, 1996). Marketers look at the customer lifetime value that is abbreviated as CLV or in some cases LTV.

In order to calculate customer lifetime value one has to project the expected net cash flows from the given customer and then calculate their net present value. In the general case projecting future cash flows can be challenging since one has to estimate the likelihood of additional purchases. However, in the case of a single SaaS product the projection is easier since the customer pays a reoccurring subscription fee. The total future cash flows is calculated from extrapolating the fee over the customer life. Customers' average life can be derived from customer base and the retention rate, another key metric in subscription based models. Some use the inverse measure of attrition rate or CHURN rate which are synonymous terms.

It is often the case in subscription models that there is a large beginning cost associated with acquiring the customer. This cost is called cost of customer acquisition (COCA). A funnel metaphor describes how out of the many potential customers reached by advertizing and brand awareness activities only a few become leads. Out of this few leads fewer yet become customers. The total marketing and sales costs in a given amount of time divided by the number of customer obtained in that time is the cost of customer acquisition. There is also a smaller reoccurring operational cost associated with keeping the SaaS product running. Per customer the ongoing cost may be the cost of goods sold divided by the number of customers. Equation set 6 below summarizes these calculations. Companies often look at the break-even point that is the time it takes to regain the cost of customer acquisition. Pricing decision is made based on breakeven point. The second equation in equation set 6 is used to calculate break-even point, by making $CLV=0$ and finding t that satisfies this condition.

Equation 6 – Customer Life Time Value

$$COCA = \frac{SG\&A_t}{\Delta N_t}$$

$$CLV = \sum_t \frac{Fee - \frac{COGS_t}{N_t}}{(1+r)^t} - COCA$$

$$t = \frac{1}{CHURN_0}$$

$$Revenue = \overline{N_t} * CLV$$

Where

r denotes discount rate

N_t denotes Customer base in time t

ΔN_t denotes new customers added in time t

Retention can be affected by many factors including a dynamic competitive landscape, changing customer preferences and the state of the economy. Bolton (1998) shows that the duration of retention is positively related to customer satisfaction. Thomas (2001) even discusses the connection between the acquisition process and retention. The two are often treated independently but are actually dependent variables. Since customer satisfaction affects retention and is largely a variable that can be controlled by the service, product providers go to great lengths to measure customer satisfaction. One option promoted by Reichheld (2003) is net promoter score. Customers are asked: "How likely is it that you would recommend our company to a friend or a colleague?" This question was chosen out of a variety of survey question as the answer was most correlated to actual behavior. Based on customer's response on a 0 to 10 rating scale, the customers are grouped. Having answered 9 or 10 the customer is a "promoter". With an answer equal to or lower than 7 the customer is a "detractor". The percentage of "detractors" is subtracted from the percentage of "promoters" and the result is the net promoter score. According to Reichheld, world class companies have a score of over 75%.

SaaS product revenue is a function of the customer base and the product price. As mentioned previously, the customer base is a function of the number of new subscribers and the attrition rate. Since companies have a limited amount of resources, one might ask what is more important to focus on: new acquisitions or retaining existing customers. Furthermore, there may also be a tension between the two factors. For the purpose of this study this is of particular significance since feature value is measured as a weighted average of acquisition and retention impact. Studies suggest that retaining existing customers is more important than acquiring new ones. One reason for that is that service discontinuers create a negative word of mouth that is more convincing than the positive word of mouth of continuing customers (e.g. Mahajan et al., 1984; Oliver, 1997). Another reason, pointed out by Madhavan and Bhattacherjee (1998) is that new customers come at high adoption cost of sales and marketing. Since, as previously mentioned, customer acquisition cost is many times greater than the monthly cost of maintaining an existing customer.

Beyond customer lifetime value and customer satisfaction, SaaS practitioners, especially in startups, measure the SaaS "magic number". This metric was introduced by Josh James, then CEO of Omniture, and documented by Lars Leckie (2008). This metric is a ratio of revenue change to sale and marketing expenses.

Equation 7 – SaaS "Magic Number"

$$Magic\ Number = \frac{QRev_t - QRev_{t-1}}{ExpSM_{t-1}}$$

Where

$QRev_t$ denotes the quarterly revenue in period t
$ExpSM_t$ denotes the quarterly sales and marketing expenses in period t

Acquiring a costumer is an investment which takes several time periods to turn into profit. Therefore, a growing company that invests heavily in sale and marketing can have a negative cash flow for months. The SaaS "magic number" is meant to be a litmus test to identify when to grow the sales organization and when to stop hiring sale representatives. If the number is beyond 0.7 then the sale's team should be expanded. If the number is between 0 and 0.7 then more questions need to be asked and if it is negative then something is wrong. A beneficial aspect of this metric is that the data can be taken from a companies' 10K which makes it easier to benchmark against publicly traded companies. Other versions came later such as Joel York's (2010). One should note that these metrics were not published in any academic publication and therefore did not pass the desired level of scrutiny. Nevertheless, these metrics should be considered as they represent important industry trends. However, they should be taken into consideration with healthy skepticism.

In summary, existing SaaS metrics tend to look at a customer view or an organizational view with respects to sales and marketing. The customer metrics look at customer satisfaction, customer cost of acquisition and customer present value. The organizational metrics look at sales and marketing perhaps because SaaS is viewed a business model. There are no metrics that focus on the product development aspects of SaaS. Moreover, most of the work done on SaaS is outside the academic circles. This is natural with new business developments. Academic work tends to be slower as it carefully builds its case on evidence and empirical studies. As SaaS becomes more widespread, it is time for academia to step in, and industry can in turn benefit from academic research.

2.5 On Agile, SCRUM and Agile Performance Metrics

Understanding Agile methods is relevant to this thesis for two reasons. Firstly, the framework proposed is most effective for products with short time to market such as products using continuous deployment, an aspect of Agile. Secondly, since the framework was implemented at HubSpot, which uses these techniques, it was paramount to understand agile concepts in order to gather data, standardize the data and communicate effectively with employees.

Agile is a software development process based on quick iteration and an emphasis on flexibility. Agile grew out of a perceived need to create a better process for delivering product that satisfies customers. The method's advocates claim that traditional processes, that of waterfall and similar methods, are too rigid. These methods have a long requirement and specification phase and by the time coding starts it is costly to change design. However, customers find it hard to articulate their need until they see a working product which in traditional methods happens only after development.

Agile has a cultural even philosophical underlying theme. Since it grew to contrast traditional methods it is built on somewhat of an anti-culture. The developer is in the center of agile development and is empowered to make design decisions with little as possible specifications and documentation. Under agile even naming convections changed. Things such as requirements, effort, product manager, development team, project manager and project plan got new name and a slightly different meaning. Requirements are user stories, effort is story points, a product manager is a product owner, development team is a scrum team, project manager is a scrum master and project plan is a sprint.

Extreme programming and Scrum are two preceding methods that were brought together along with a few other methodologies into Agile. The Agile manifesto (Beck et al., 2001) is a short document that solidified twelve principals that are considered to be the base of the method. These principals include: early delivery of software, welcoming changes in requirement even in late stages, face-to-face meetings and a supporting and motivating environment for individuals.

This section is not a comprehensive summary of Agile. Rather its aim is to mention and explain the aspects most relevant to the framework presented and to the specifics of collecting data and implementing the method in HubSpot. One such aspect is Scrum teams. These are small cross functional teams performing the development tasks (Schwaber and Beedle, 2001). A Scrum team has a Scrum master who is the management representative, in charge of facilitating discussion, schedule and resources. A team also has a product owner who represents the voice of the customer and is responsible for prioritization. Finally, Scrum team also has developers, QA and designers.

The main method of communication between team members is a daily meeting called Scrum meeting, done standing up. These are short status meetings where progress is assessed. Planning is done for short cycles of development called Sprints that are typically a month long. A sprint ends with executable product functionality.

Scrum teams work to achieve tasks that are assigned to them at the beginning of a Sprint. These tasks are taken out of the Sprit Backlog (or Product Backlog). Tasks in the backlog are called user stories. User stories are very much like functional requirement that are described in a natural language. User stories can only be functional, that is described in a business meaningful way, and not in a technical manner (Cohn, 2004). It is the Scrum team developers that translate user stories to technical terms without a having defined documentation to support it. The developers are responsible to estimate the effort required per a given task. It is part of the Agile belief is that only the ones "doing the work" have a right to estimate the time it takes to do it.

With an executable function at the end of every sprint the software product gets updated quickly. When the software is SaaS delivered then the production code (that is, the software version that customers use) is also updated very frequently. Some have taken this approach further, developers' check-in code to production at the end of every day. This method is called continuous deployment. To do it successfully, companies have to put in place automated tests that run on a pre-production environment before changing the production code. Continuous-deployment along with minimal viable product (launches the absolute necessary and continues development after launch) and some terminologies borrowed from lean production methods are part of an Agile offspring methodology called Lean Startup (Ries, 2011; Blank, 2006). Lean Startup is centered on reducing the waste of creating products that customers don't want. HubSpot practices many of the Lean Startup principals although they probably started it independently before Lean Startup was developed.

Agile, and its siblings, have gained tremendous popularity. Perhaps because some of the most successful .com age software companies are using it (many of the worst are probably using it too). However, for a system that has been widely used there are surprisingly few quantitative studies that measure the system's efficiency. Highsmith and Cockburn (2001) provide a summary of the development of Agile methods and explain how they grew as a reaction to market changes. Even this well sited article does not have an empiric base to support its arguments. Moreover, this article and other like it (Martin, 2000) are strong proponents of these methods without acknowledging any weakness, limitations or cases were other methods are more suited. Add white papers about it from Yoav.

Concurrent Engineering is another development approach that started at about the same time as Agile and is also meant to reduce time to market for product development. In contrast to Agile, Concurrent Engineering was evaluated empirically (Lawson and Karandikar, 1994) and some researches acknowledges both its strength and limitations (Smith and Eppinger, 1998).

A common metric in Agile Sprints is velocity, a measure of how much work a Scrum team can do in a Sprint, measured at Story Points per Sprint (Cohn, 2004). Actual and planned velocity is compared in a manner similar to Earned Value Calculations (Alleman et al., 2003).

3 Research Methodology

This research was done using data collected at HubSpot. Usage information was collected from HubSpot's 3,000 customers over a period of four months. For each feature the percentage of users that used the feature was calculated. If a user accessed a feature at least once in the week prior to the measure than it counted as an active user. In order to eliminate seasonal volatility the usage was based on the average of four measures taken at the end of four consecutive months.

Development cost was collected from evaluating sprint logs from the earliest point available. In total 27 sprints were taken into account. Each sprint stands for a calendar month. The logs had indication of the user stories that were done in a given sprint and their story points. The research involved the analysis of over a thousand storylines and assigned each storyline to a feature. Some user stories were assigned to more than one feature and others representing back-end work or cross functional work were not assigned.

Converting story points into a consistent measure of effort created some difficulty. Story points' scale changed over time and in between scrum teams. Also team sizes changed over time. In order to calculate the total amount of effort per feature for the entire period the sprint efforts were adjusted according to relative team size and the number of story points per team per sprint. The equation below describes the conversion from story points to effort and the aggregation of user story effort into features.

Equation 8 – Feature Cumulative Development Effort

$$effort_i = \sum_{k}^{sprints} \frac{scrum_j}{\sum_j scrum} \times \frac{sp_i}{\sum_i sp}$$

Where

sp_i denotes the total number of story points to feature *i* in a given sprint and scrum team
$scrum_j$ denotes the size of scrum team *j* in a given sprint
k is an index for a given sprint running from 1 to 27

Even after having figured out the development effort for feature there was still some work to do to get a full feature cost. Cost of goods sold (COGS) is not captured in the backlog. COGS have been previously calculated by an internal financial team and apportioned to features. Sine COGS where in dollar terms the development effort was also converted into a monitory estimate and then the two were added up. In the results I present two cost analyses. The first shows a cost score based only on development effort. The second shows a cost score base on development effort and COGS.

For feature sensitivity the study looked at two measurements six months apart, from sprint 21 and 27. Value was calculated based on usage and early usage data without expert opinion. Usage was also calculated based on one month data and without an average of several months mainly because some features go back only as far as sprint 21. This also makes the value measurement more sensitive to changes over time.

User retention data used to test the usage as a predictor of retention hypothesis (section 1.2) used a pre-made internal report published at sprint 27 and based on a year worth of user retention data. The report uses a sample of 2,843 customers.

Expert opinion was collected in a survey with 60 sales and support representatives participating. Surveyed participants were asked two questions. The first question, used as a measure for customer acquisition, was: "Please rank only the five most valuable applications in the product when you are prospecting/to the customers you close. Most valuable app gets five votes, second most valuable app gets four votes, etc." "App" is the company's terminology for feature and "closing" is the terminology used for subscribing a new customer. The second question, used as a measure for customer retention, was: "Please rank only the top most valuable applications in the product to our Ollie/Mary customers. Most valuable app gets five votes, second most valuable app gets four votes, etc." Ollie and Mary are marketing personas. They represent the two customer types that the company is targeting. The voice of the customer in HubSpot is always described in terms of Ollie, Mary or both.

4 Results

4.1 Results of Usage as a Predictor of Retention Hypothesis

The hypothesis testing connection between customer usages to retention was applied to five features separately and on the product as a whole. In all cases, but one, the null hypothesis was rejected at a 90% confidence level. For the product as a total the null hypothesis was rejected at a 99% confidence level. Meaning, users who are not using a feature are much more likely to discontinue service than active users. The only feature were the null hypothesis could not be rejected at a significant level is list manager. That is due to a very low population of customers that use list manager. Although features where tested separately they are not necessarily independent variables. In fact that data implies a correlation between feature usages.

Table 1 – Usage Connection to Retention

Feature	Attrition Rate Active Users	Attrition Rate Non-Active Users	% Usage	P Value
Blog	0.47x	2.89x	30.53%	~0
Content Management	1.3x	2.51x	39.71%	0.0376
Leads	2.03x	3.53x	44.50%	~0
Landing Page	0.76x	2.9x	23.41%	~0
Lead Nurturing	0.77x	2.22x	10.15%	0.054
List Manager	0.97x	2.58x	2.29%	0.356
Product	x	2.69x	82.35%	0.0003

Table 1 provides the results of the statistical tests of the hypotheses. To protect sensitive business information retention is stated in relative term to 'x' the attrition rate for active users of the product as a whole.

4.2 Results of the Loose Correlation Hypothesis

The relation between value and cost is *value=0.64*development effort + 0.021;* R^2*=0.44*. The null hypothesis is rejected at a 90% confidence level showing that there is a positive relationship between value and development effort.

The results show that there is a clear and statistically significant correlation between effort and value. However, $R^2=0.44$ is not a tightly coupled relation. The relatively low correlation reflects uncertainty. If there was a perfect alignment between value and investment ($R^2=1$) that would suggest that the market's behavior is completely predictable and that there are no difficulties with task prioritization. Such certainty is unnatural in any competitive market scenario, before

applying a change it is hard to know what the effect would be. Uncertainty is even greater in new markets and startup environment. Social media marketing is certainly a very novel industry and HubSpot in many cases creates a market for a new service. As such it is only in retrospect that they know the success of a given feature enhancements.

Because SaaS software development in Agile is a relatively short process, HubSpot tends to develop and then assess customer reaction. In other industries, extensive preliminary research is done before starting development of a new feature or capability. For example, in oil expeditions preliminary drilling is done to assess the future oil field yield. Because of the high capital expenses to drill for oil, extensive research before starting is warranted. At HubSopt, with a relatively low capital expense for development, the cost of information does not justify market research efforts. It is simpler to go ahead and build. Or as one HubSpot executive puts it "throw spaghetti on the ceiling and see what sticks".

One would assume that in mature markets, with less uncertainty, the correlation between investment and value would be higher. Also, in industries with high capital expenditure, that conduct extensive market research before launching a feature enhancement the correlation between value and investment may also be higher.

4.3 Value Score Results

As explained in section 1.4, value is calculated by four equal measures. Two measures represent the impact on customer acquisition and the other two values represent the impact on customer retention. The support poll and sale poll are expert opinion with the two other measures are automated results queried out of the companies databases. The interim measures and the overall value score are given in table 2.

Table 2- Value Score and Value Score Components

Feature	Usage Score	Support Poll Score	Early Usage Score	Sale Poll Score	Value Score
Leads	10.82	21.09	6.81	15.36	14.49
Sources	9.06	17.68	12.05	21.82	14.44
Content Management	9.96	9.41	17.14	2.54	9.73
Landing Page	5.91	14.78	5.58	9.25	9.47
Keyword Grader	6.80	9.19	9.67	14.87	9.28
Blog	8.17	5.37	9.89	8.91	7.56
Social Media	6.15	2.21	11.09	3.63	5.14
Competitors	6.72	2.50	1.14	9.41	4.81
Page Grader	4.52	1.98	7.23	3.66	3.91
Lead Nurturing	2.93	4.92	3.18	2.61	3.62
Blog Grader	6.45	0.92	4.21	0.59	3.30
Prospects	4.56	2.94	2.22	0.93	3.10
Link Grader	5.84	0.96	2.26	1.11	2.88
Visits by Page	5.71	0.39	3.29	1.01	2.78
Reach	4.36	0.40	3.37	0.40	2.23
Email	0.96	3.44	0.42	2.48	1.98
List Manager	1.06	1.80	0.45	1.41	1.28
Total	**100**	**100**	**100**	**100**	**100**
Best fit	Normal N(5.9, 2.7)	Lognormal LN(6.4,12.6)	Exponential EXP(0.17)	Lognormal LN(6.9,10.5)	Lognormal LN(6.2,5.8)

The components of the value score are also shown graphically in figure 3. They are displayed again, in a graphic manner, to emphasis the different distribution of scores in each component. The usage score is based on the full customer population over the course of four months. That score takes into account over 10,000 data points, more than any of the others and perhaps that is the reason it has the lowest deviation around the mean of 5.9. The early usage score is based only on new customers which is a population of around 100 data point in each of the four months. The standard deviation is higher. One reason may be the considerably smaller sample.

Another reason is that new users are gradually learning the product and as they learn they use more of the product functionality. The components derived from polls give a much more polarized distribution. A handful of features get a high score and the rest get very little scores. Experts overvalue the valuable features and undervalue the features with low value in comparison to usage information. Hence, both polls have a higher deviation. This is probably a side effect of the poll question. Respondents were asked to rank the five most valuable features and not to rank all the features in descending order. However, the poll question was chosen because people find it hard to distinguish at low resolution. It is much easier for people to single out a few items than to accurately rank many closely performing items. In that sense, the high polarization is an artifact of a human tendency. Another take away is that the experts asked are in agreement with each other, more than the usage data would suggest they should be.

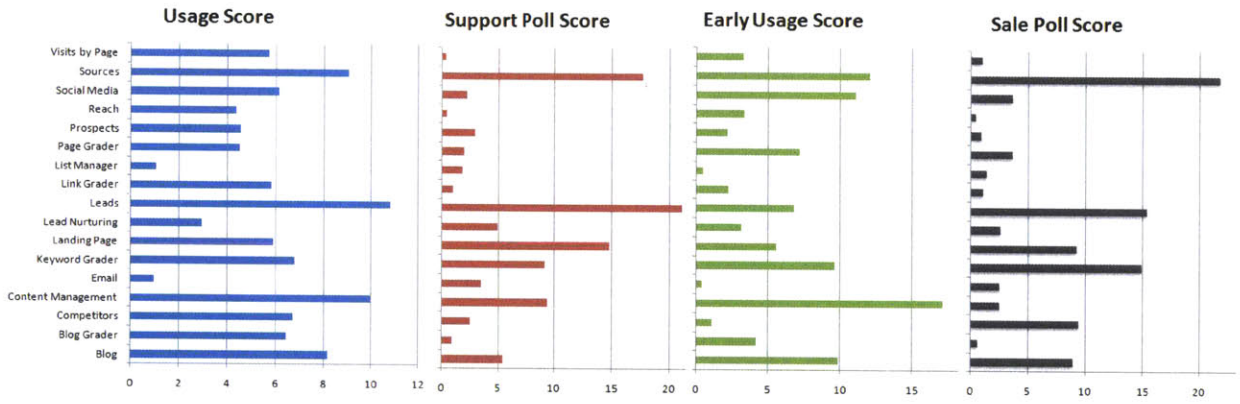

Figure 3 – Value Score Components

Another interesting outcome of the value analysis is the appearance of long tail phenomena. A relatively small set of features drive most of the value, while the majority of features "drag" behind with relatively low value. In figure 4 the features are sorted in descending order and the value outline (in blue) is somewhat reminiscent of a long-tail distribution (in red). Also, the median score is 3.9 with is lower than the average at 5.9. Running an Anderson-Darling best fit analysis (Lewis 1961) on the data came with a lognormal distribution. Lognormal is a "skewed" distribution that creates long-tail curves.

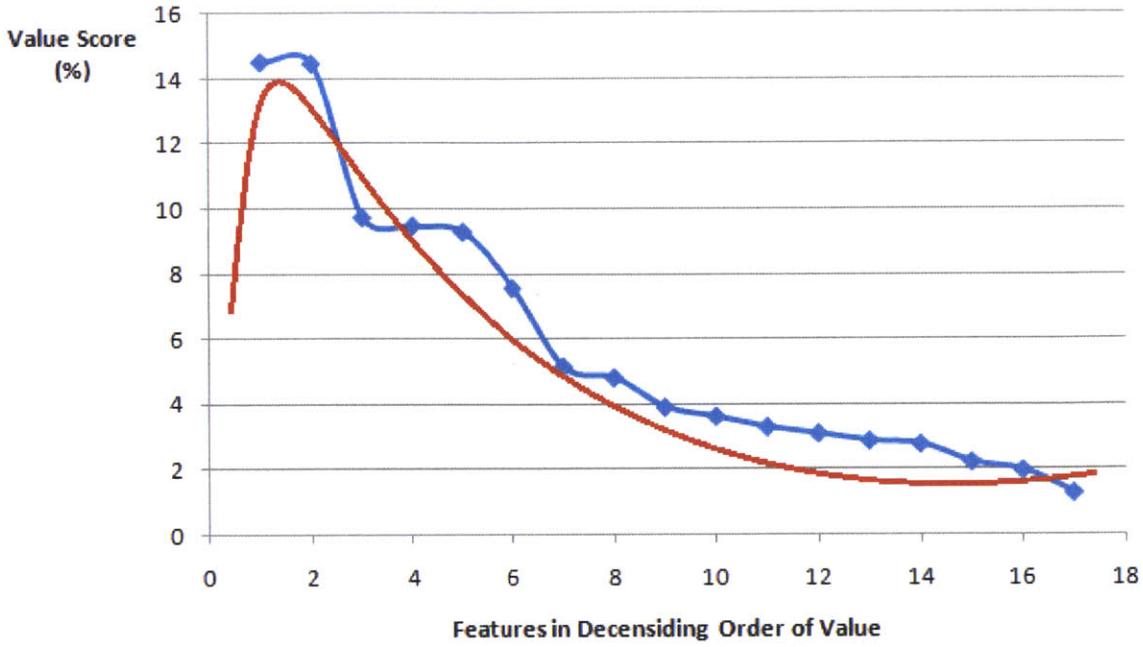

Figure 4 – Feature Score Follows a Long Tail Distribution

4.4 Cost and Profitability Score Results

Cost information is an accumulation of development effort and COGS from the beginning of documented backlog work at HubSopt. Cost is calculated twice, once just by looking at development effort and again by including COGS. Depending on the type of insight desired it is useful to look just at development cost per feature or all the direct costs per feature. Table 3 shows relative costs as well as relative value scores and profitability. Profitability in table 3 is calculated with direct development cost only.

Table 3 – Feature Cost and Profitability

Feature	Relative Value Score	Relative Development Cost Score	Relative Development & COGS Cost Score	Profitability
Leads	14.49	17.08	13	8
Sources	14.44	8.72	8	11
Content Management	9.73	8.63	6	6
Landing Page	9.47	2.22	2	9
Keyword Grader	9.28	7.96	16	6
Blog	7.56	4.21	4	6
Social Media	5.14	8.07	7	2
Competitors	4.81	10.03	8	1
Page Grader	3.91	6.89	7	1
Lead Nurturing	3.62	9.75	9	0
Blog Grader	3.30	3.66	4	2
Prospects	3.10	1.86	1	2
Link Grader	2.88	1.96	6	2
Visits by Page	2.78	1.36	3	2
Reach	2.23	1.60	2	2
Email	1.98	3.74	3	0
List Manager	1.28	2.26	2	0
Total	**100**	**100**	**100**	**60**

With value and cost at hand the best way to evaluate features' relative standing is to look at a scatter plot with of value against cost, as seen in figure 5. The top left corner of the scatter plot is the most desirable place for a feature. Likewise, the right lower corner is the least favorable place on the plot. Figure 5 also has the regression line that progresses from the lower left corner to the top right. Features above the regression line perform better than average (6 features are above the line) and features below it do worse than the average (11 features). One can see that the feature farther away than any other above the line is *Sources*, the most valuable feature.

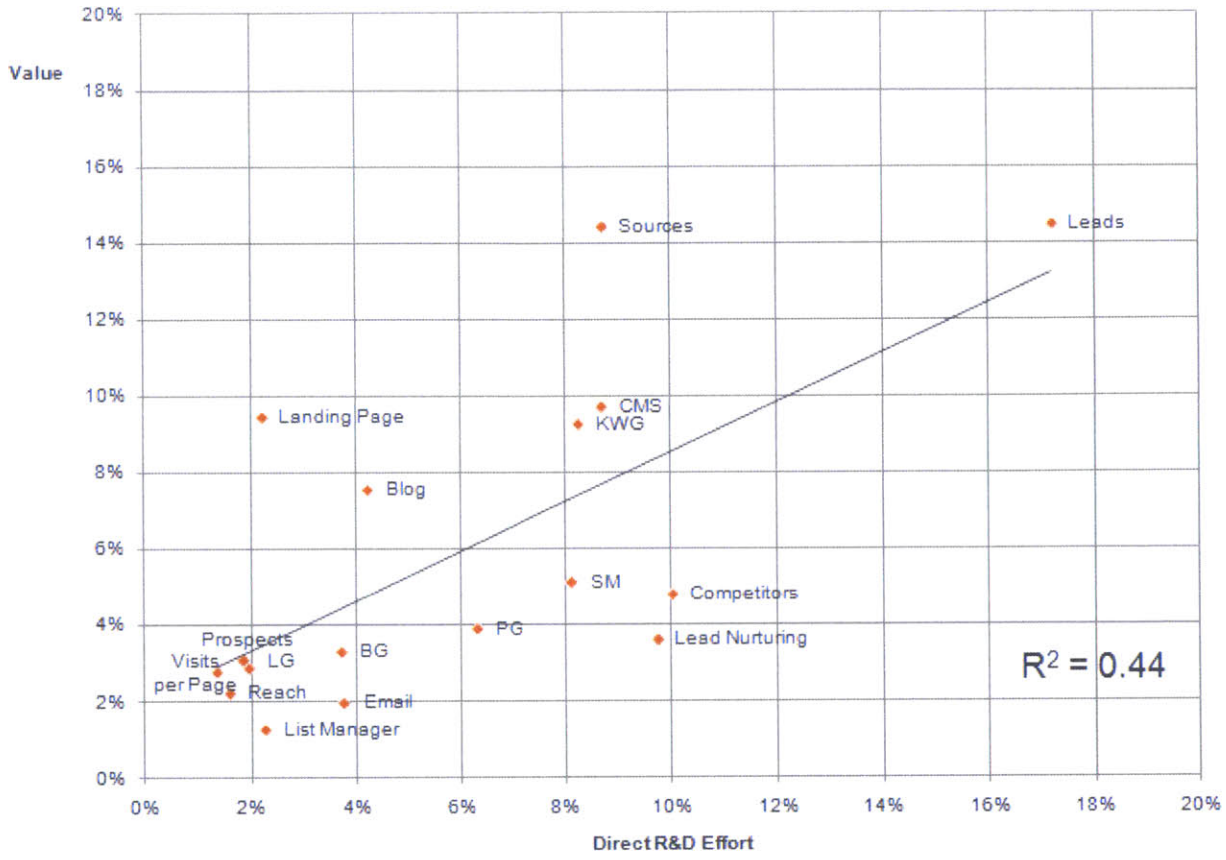

Figure 5 – Value to Cost Scatter Plot with Linear Regression

By looking at the scatter plot in figure 5 one can tell the profitability of a feature relatively to another feature. However, figure 5 does not tell us which features are profitable. To solve that, a brake-even slope is added to figure 6. Since the gross margin in HubSpot at the time was around 60% the slope is represented by the line *value = 0.4*cost*. To understand why this line is used refer to section 1.6. One way to think about profitability score is that the 60% gross margin is divided between the features.

The vertical distance between a feature and the slope equals the feature's profitability. Features that are in the "green" area are unprofitable. In our case, there are four features that are around zero profitability: *Lead Nurturing*, *Competitors*, *Email* and *List Manager*. The most profitable feature is *Sources* with 11% profitability. Here too, a long tail phenomenon exists: the six must profitable features hold 75% of the profits and the other 25% is thinly spread between 11 features.

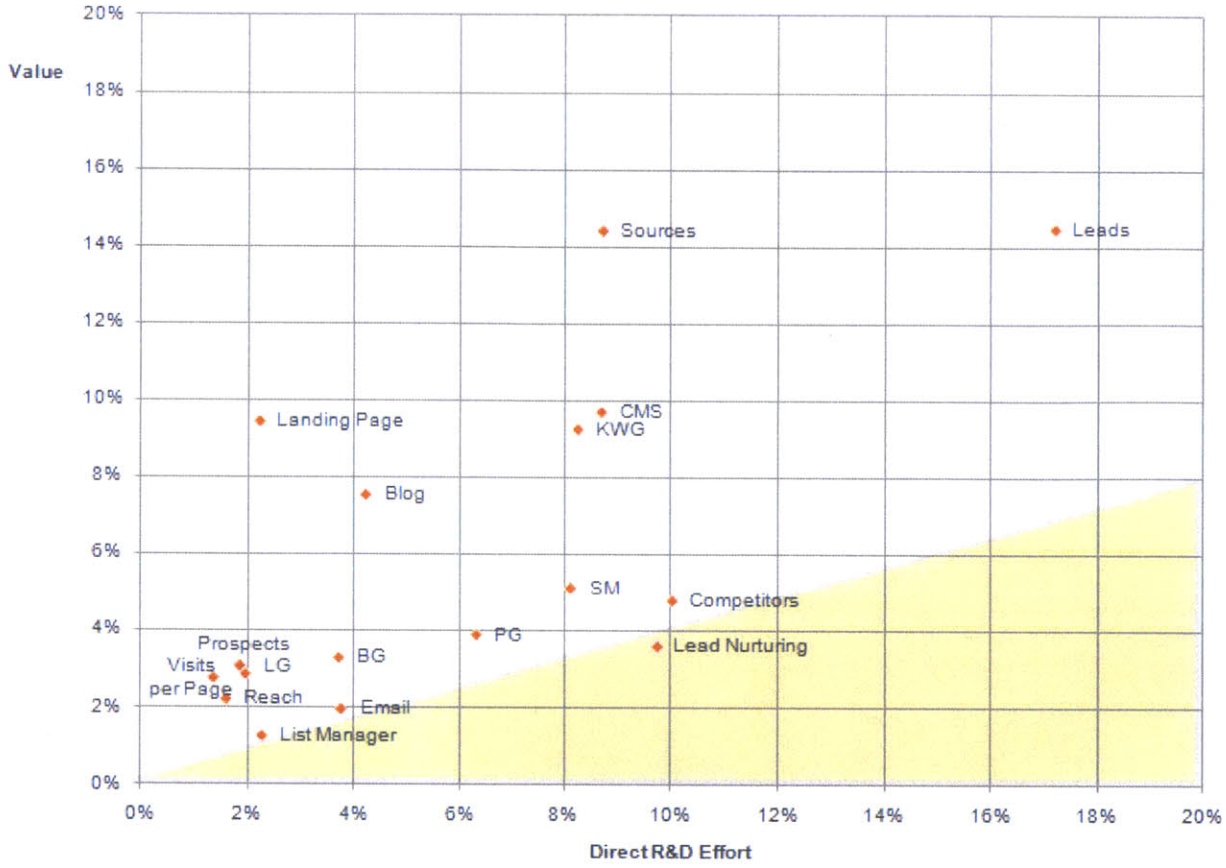

Figure 6 – Value to Cost Scatter Plot with Profit Indication

Development effort is about 80% of the total costs. Since COGS is spread pretty evenly among features and is only 20% of the total direct, adding COGS does not dramatically affect the overall standings of most features. There are two exceptions, *Link Grader* and *Keyword Grader*. In the case of *Keyword Grader*, a relatively high cost feature even without COGS, the change is very significant. Adding to the usual storage costs that all features have, *Keyword Grader* has a unique additional cost. The feature uses an information feed, on word popularity, acquired from Google. Figure 7 shows value to cost when cost includes development investment and COGS. The right strategic way to examine profitability is by looking at all costs just as shown in figure 7. However, in some cases, there might be more options to change development efforts than COGS or the two cost factors can be under different management. If that is the case, it makes sense to also look at profitability that considers development only. Moreover, looking at both scatter plots, figure 5 and figure 7, really allows one to appreciate the added impact COGS has on feature costs.

Figure 7 – Value to Cost Scatter Plot with COGS

4.5 Sensitivity Score Results

Sensitivity is a measure how effective recent development investment has been. It is defined as the effect a given addition of investment has on feature value. In our case the formula used was slightly different than the one in equation 3. Instead of normalizing sensitivity by dividing the number of customers by the number of customers in the previous period, normalization is done by dividing the monthly revenue (MRR) by the monthly revenue of the previous period. If all the customers pay the same price then the two methods are exactly equivalent. Since HubSopt has two products that differ in the level of support and in price the two measures are not equivalent. Using monthly revenue is more accurate but may require information that maybe, in some cases, harder to obtain.

Table 4 shows feature sensitivity score. The feature with the highest score, landing page, has the best sensitivity rating. It means that, in the time period tested, its value relative to investment increased the most. The average sensitivity score, in the HubSpot case, is positive and it will be so in any case that the monthly revenue (or number of customers, depending on normalization method) increased between the two periods. Even a feature that decreased in relative value (100% divided between all features) between the two periods can have a positive sensitivity, if the total increase in monthly revenue is larger than the decrease in value. Therefore a feature with negative value, like *Lead Nurturing*, had a significant decrease in value larger the organization increase in monthly revenue of 24%.

Table 4 – Feature Sensitivity Score

Feature	Sensitivity	Feature	Sensitivity
Blog	1.11	Reach	0.18
Landing Page	0.8	Email	0.14
Link Grader	0.48	Competitors	0.09
Sources	0.29	Keyword Grader	0.03
Visits by Page	0.28	Social Media	-0.05
Leads	0.25	Prospects	-0.06
List Manager	0.24	Page Grader	-0.06
Content Management	0.21	Lead Nurturing	-0.13
Blog Grader	0.21	**Average**	**0.24**

It is also beneficial to look at a graph like the one in figure 8, which has a graphical display of feature sensitivity. In this graph the horizontal axis shows accumulative development investment. The vertical axis shows the reoccurring monthly revenue. Each line represents a feature. The left edge of a line is a feature's state in sprint 21. The right edge of a line is a feature's state in sprint 27. To protect confidential financial information such as the total monthly reoccurring revenue and development costs values in the graph are given in relative terms. "X" and "Y" represent base values (both are at the ball park of thousands of dollars) and MRR and accumulative direct-development costs are given as multiples or "X" and "Y" respectively.

Let's look, for example, at *Leads* (orange line with black square edges) that had a development cost of $5.6X and a share of $2.4Y of the revenue $24Y reoccurring revenue in sprint 21. By sprint 27 the total development cost of Leads rose to $9.5X and the share of revenues rose to $4.4Y out of around $30Y reoccurring revenue in sprint 27. The slope of the Leads line is its sensitivity 0.25 which is about the same as the average sensitivity of 0.24. The horizontal projection is the increase in cost and the vertical projection is the change in value. The graph shows more information than just the sensitivity score. We know that *Leads* improved in quite a lot in value but that it also had intensive investment at the same time. By contrast, *Visits by Page*, with a sensitivity of 0.28 had a small gain in value but per a very small investment. In fact, there was likely no planned strategic investment in *Visits by Page*, and the work done was a reaction to an unplanned need that arouse during that period.

Another example of the value of a graphic illustration is comparing *Social Media* to *Page Grader*. Both have a similar negative sensitivity of -0.05 and -0.06 respectively. The graph shows that *Social Media* deteriorated despite having a considerable amount of investment during the six month period, and in that sense the performance of *Social Media* is a larger disappointment.

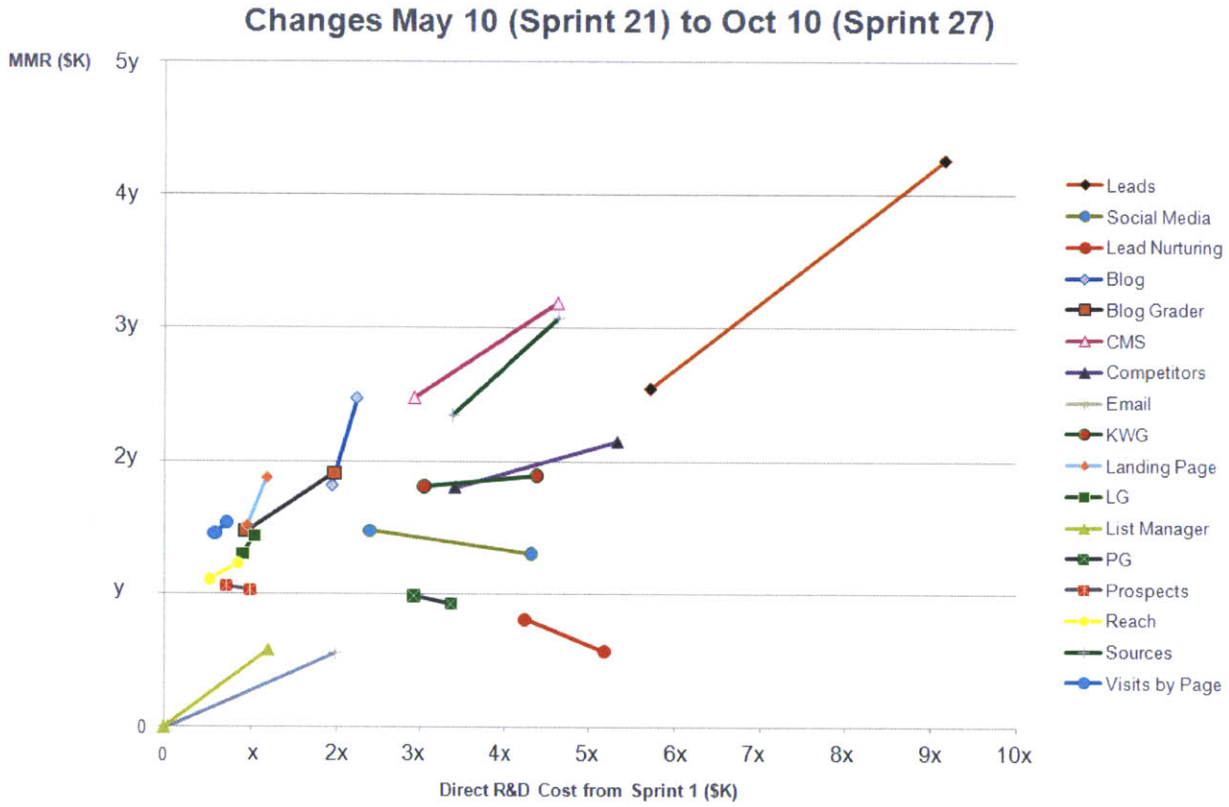

Figure 8 – Feature Sensitivity Analysis

After examining preliminary sensitivity results, company personnel had an assumption that features that were part of the strategic planning performed better than those who were not. Every few moths the company's management comes with a product roadmap. The roadmap has a high level plan that includes which features are going to be worked on and by what capacity. Some development work arises without being part of a longer term strategic planning. These could be bug fixes, or small enhancements that came as user stories and found their way into a sprint planning. It could also be that strategic changes in one feature affect another. For example, part of a feature improvement could be altering a back-end or middleware piece that is used by multiple features. Even thought the work initiated in one feature, some of the effort would be attributed to other features. To check this assumption, the product roadmap for the six months between sprint 21 and sprint 27 were evaluated. The results are presented in table 5. The table shows that improved features are just as likely to be part of the roadmap as they are likely not to be part of the roadmap. With the exception of one feature, *Social Media*, all the deteriorated features were not on the roadmap. A third of the features not on the roadmap deteriorated while only 12.5% of the roadmap features deteriorate. This sample size is too small to check for statistical significance, but it does imply that features on the roadmap are less likely to deteriorate. Moreover, since the product roadmap came before the deterioration it suggests a direction of causality. That is that "neglecting" features causes deterioration.

Table 5 – Relation between strategic planning to Sensitivity Score

	Features in Product Roadmap	Features not in Product Roadmap
Improved	3	4
Unchanged	4	2
Deteriorated	1	3

Improved features are features with a sensitivity score of 0.24 or higher. Unchanged features group includes all the features with a positive sensitivity lower than 0.24 and deteriorated features have a negative sensitivity score. Table 6 shows which features where included in each group.

Table 6 – Relation between Strategic Planning and Sensitivity Score Detailed

Group	Features
On roadmap & Improved	Leads, sources and List Manager
On roadmap & unchanged	Competitors, Keyword Grader, Email and Content Management
On roadmap & deteriorated	Social Media
Not on roadmap & Improved	Blog, Landing Page, Link Grader and Visits by Page
Not on roadmap & unchanged	Reach and Blog Grader
Not on roadmap & deteriorated	Prospects, Leads Nurturing and Page Grader

5 Using the Framework to Prioritization Feature Enhancement, Maintenance and Bug Fixes

The proposed performance scores are a useful tool in product development. The relative standings of features are to be considered when dividing feature ownership between development teams. Resource allocation between teams and within the teams should also use this analysis tool. Bug fixes should be prioritized according to value. The highest value features should be highest in priority. Routine maintenance should be assigned according to value as well. When a features has a cost score that is significantly higher than value score it may be a sign that the ongoing maintenance is not justifiable and the maintenance approach should be changed.

Feature sensitivity is an indication of how the recent investment is working. As long as a feature's sensitivity is high it makes sense to continue development investment. Once the sensitivity score dampens it should be perceived as an indication that development investment becoming is less effective and it is time to consider alternative investments. It is interesting to look at features with low cost, value and profitability that also have a high sensitivity score. These features are avenues for potential growth.

Another useful way to asses features based on these scores is segmenting the features. This is most easily done by looking at a cost to value scatter plot, as seen in figure 9. In our case all the features fit nicely into four segments: Most Valuable Features (MVFs), Outperformers, Underperformers, Fledglings.

The most valuable features are a segment of features that are high in value and investment. These features are recognized as important by the company. In our case these are *Leads* and *Sources*. This group should be the highest in priority for bug fixes and regular up-keep. As long as the sensitivity is positive they should also be considered for enhancements.

Outperformers are a segment of features that are doing very well relative to the investment in them. In the scatter plot described above they will appear closest to the top left corner. In this case they are *Blog*, *Content Management* and *Landing Page*. At times, things that are most valuable to the customers can be relatively simple to implement. A common bias, at least in development departments, is overvaluing the most complex and hardest to implement features. Identifying and communicating the outperformers may help mitigate this bias. Companies should also study outperformers to try and understand what is it about them that made them so profitable.

By contrast underperforming is a segment containing features that in retrospect do not justify their investment. In this research they are *Lead Nurturing*, *Social Media*, *Competitors* and *Page Grader*. Out of this group the features with zero or negative profitability need re-examination. If value covers the cost of goods sold and there is little maintenance development anticipated the feature could be kept on 'life support'; that is kept alive while avoiding investment as much as possible. Otherwise the feature should either be terminated or overhauled. Since in our case the company operates in high gross margins that keep growing and none of the features is significantly below zero profitability I would not recommend to terminate any of the features. It

is clear however, that *Social Media* feature, despite considerable investment is not picking up. Social media is a major part of HubSpot that differentiate it from competitors. HubSpot's freeware (free version) social media tools, such as twitter grader, are valuable to the company and generate much of its publicity. The free version tools are not included in this study. It is important to HubSpot to have a dedicated social media feature in the paid version as well; hence what they are doing, at the time the data was collected, needs re-assessment.

In the fourth segment, fledglings are features that have had little investment and provide little value. They include *Link Grader*, *Landing Page*, *Blog Grader*, *Visits by Page*, *Prospects* and *Reach*. It is interesting to note the existence of a long tail effect; most of the value derives from a few of the features and the fledgling group is the largest one. However, this group also holds the most potential as it may include promising features that have yet to mature. For example, *Link Grader* and *Visits by Page* have an above average sensitivity and therefore they are ideal candidates for future investment.

Another use of these scores is post-rollout evaluation of new features. If a feature has no profitability, or negative profitability, that means, that developing this feature may have been a mistake. It might be that the new feature is still increasing in value and given enough time will show profitability. If that might be the case, it is advised to re-examine profitability score as time passes by. As long as the sensitivity score remains high the new feature is still growing in profitability. Even if a new feature is profitable, but has a below average profitability, it might have been a mistake to develop the feature. That depends on if the company had other alternative investments that were not selected.

HubSopt, for example, launched *List Manager* and *Email* features in sprint 21. You can see in figure 8, that they are at the (0, 0) coordinate in sprint 21. At the end of six months, in sprint 27, the features have around zero profitability. Although both of them continue to grow *Email* has a below average sensitivity score meaning that it grew slower than the product as a whole. Looking back it is not clear that adding these features was the right decision, particularly *Email*. Perhaps interfacing with third party email campaign tools would have been a more successful strategy. Another measure in sprint 33 will give a clearer answer.

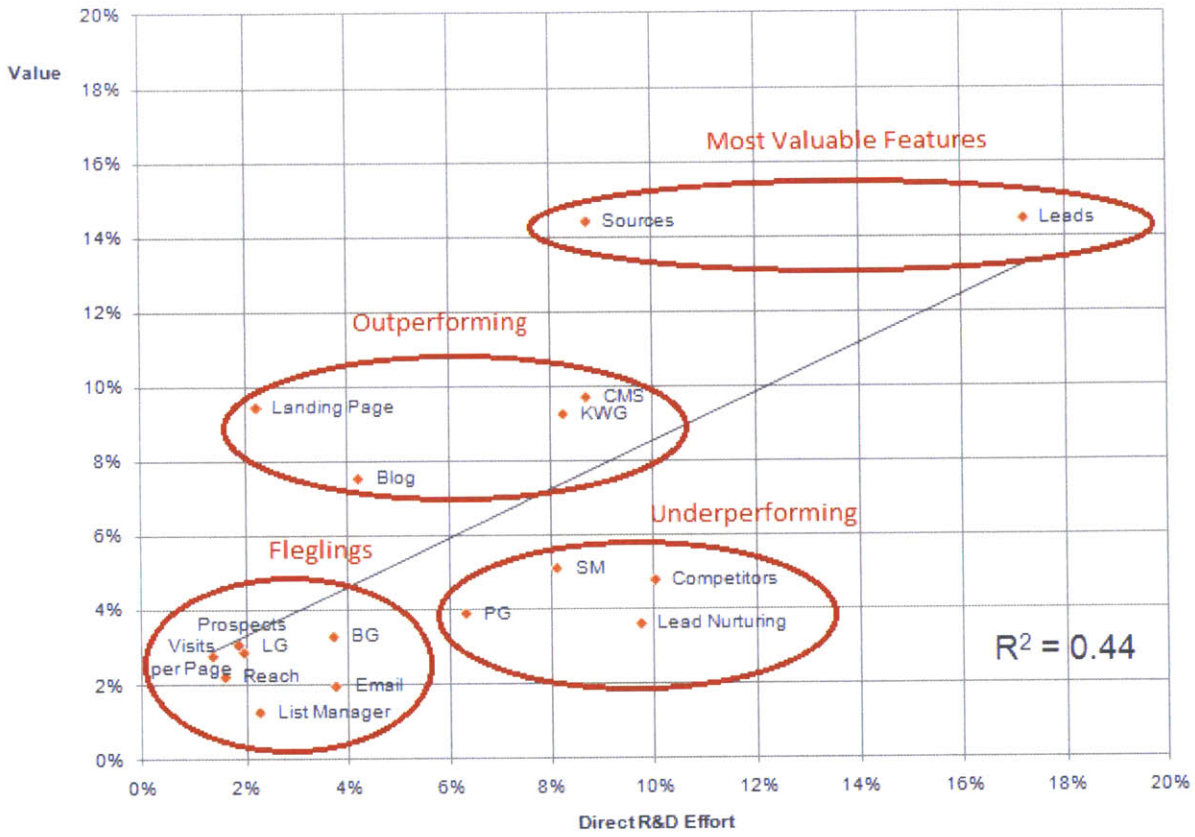

Figure 9 – Feature Segmentation

6 Conclusions and Future Work

This paper surveys SaaS companies' operation, their development methodologies and the measures they use. The thesis also looks at how measurement systems are built and implemented and other common measurement systems in software development, marketing and management. Then using this knowledge, the thesis innovates by introducing a measurement framework designed to measure feature performance in SaaS companies. With the prevalence of SaaS companies this framework has a large target audience.

The idea of measuring value (or revenue), cost and profit is perhaps the most basic principle accounting; at the company level revenue, cost and profit are defined and governed by accounting rules. It is a common practice to apportion those values down to the customer level. However, using the same notion of value, cost and profit at a feature level is the main contribution of this research. Defining value, at a feature level, required extensive research since customers don't reveal their preferences by buying features; rather customers pay for the product as a whole. Even looking at costs per feature, which is less complicated, was never done at HubSopt before this research. If HubSpot, an extremely successful company, has never done it, I'd expect that many companies never have. Besides applying these intuitive measures of value, cost and profit to the feature level, this paper aimed for an easy and cost effective way to do so. Instead of accounting for all the costs and revenues the measures look at relative scores eliminating the need to match with the accounting books.

This thesis also introduces the concept of sensitivity at the feature level. This work shows how the performance metrics should affect decision making using HubSopt as an example. The metrics are incorporated into a framework that is a cyclic process of implementing measures, collecting data and reviewing decision making.

The framework provided in this paper is applicable to other industry cases. It could benefit a software product if these conditions apply: (1) the product is provided as SaaS, (2) the product has a defined set of features and (3) have a large enough number of features to justify a quantitative measurement system.

This thesis does not prove that implementing the framework improves bottom line performance. Ideally, after integrating the framework the paper should revisit the company and check the impact it had on feature prioritization. However, in management studies it is rarely the case that a scientific study with a control group is possible. Companies are not Petri dishes and there are too many variables to allow isolating just one variable (Rosenzweig 2007). Even with this limitation it would be useful, as future work, to compare a company before and after the framework's implementation.

While it is true for every SaaS company that value derives from customer acquisition and customer retention, it is not always suitable to measure impact on customer acquisition and customer retention by looking at usage data. It would be interesting to implement the framework to other types of SaaS products. For example, products with features that work without active user engagement or feature that need heavy customization to automate some processes.

One aspect that was not fully taken into consideration is feature "stickiness", that is features that once they are in use customers tend to stick with the service because the switching cost is high. Potentially, there could be a feature that is relatively low in use but has a high "stickiness" factor. Features such as *Content Management*, *Blog* and *Leads* are considered at HubSpot to be very "sticky" features, although the data that shown in table 1 does not point to that. In order to fully asses features' stickiness one has to isolate every feature and measure its connection to retention. Note that users tend to use more than one feature. This examination could be done using logistic regression (Agresti 2007). If the logistic regression model finds different levels of "stickiness" between features, than the retention score should be captured as a combination of usage and "stickiness" sub-scores.

I was not able to perform a full logistic regression test because of the way HubSpot collects its data. HubSpot's data is housed at Salesforce.com and not in an in-house data-warehouse. Salesforce.com does not allow free querying of data, one has to use Salesforce.com report generator. The report generator is limited in data-mining capabilities compared to an SQL based querying tool.

A possible direction for expanding this framework is adding marketing costs to the cost score at the feature level. It turns out, that at HubSopt many of the marketing efforts are dedicated to specific features. HubSpot holds several webinars a week; each one is dedicated to a specific subject that maps into a feature or a set of features. Looking at these webinars could be a good proxy for allocating marketing costs to features. It would then be interesting to compare feature development sensitivity (as in table 4) and feature marketing sensitivity. I would also examine the connection between development and marketing effort and look for synergies in feature value creation. I expect that in many companies, at least a part of marketing is done at a feature level. However, it is probably far more common that development efforts are done at the feature level than marketing, and that is partly why this research focuses on development efforts.

Another interesting area for further research is checking the correlation between feature investment and value across various companies. Then examining to see what parameters impact the correlation. Specifically, I would check the impact of the following parameters: industry clock speed (how much innovation), company maturity and financial performance. I would expected that companies with very low correlation ($R^2<0.2$) are at a "chaotic" state and have poor financial performance. They are also more likely to be in a fast changing industry and be a new company. In contrast, maybe companies with high correlation ($R^2>0.7$) are at slow moving industries, are well established and have a sound financial record.

Having a clear evaluation scale for features brings focus to a company. During strategic planning, the product roadmap should be done according to this evaluation. Frequent measurement allows to experiment, trial often and fail often, and to obtain a quicker understanding of customers' value and what is the subsequent development cost.

7 References

Agresti A. Building and Applying Logistic Regression Models. Hoboken, New-Jersey: Wiley. 2007.

Albrecht A., Gaffney J. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE transaction on Software Engineering. 1983. Vol. SE-9 no. 6.

Alleman G., Henderson M., Seggelke R. Making Agile Development Work in a Government Contracting Environment – Measuring Velocity with Earned Value. IEEE Computer Society: Agile Development Conference. 2003.

Beck K. et al. Manifesto for Agile Software Development. Available at < http://agilemanifesto.org/> Accessed on: Feb 16, 2011.

Berger D., Nasr N. Customer Lifetime Value: Marketing Models and Applications. Journal of Interactive Marketing. 1998. Vol. 12 pp 17-30.

Blank S. Third Step to the Epiphany. Lulu.com. 2006.

Boehm B. Software Engineering Economics. Englewood Cliffs, New-Jersey: Prentice-Hall. 1981.

Bolton R. A Dynamic Model of the Duration of the Customer's Relationship with a Continuous Service Provider: The Role of Satisfaction. Marketing Science. 1998. Vol. 17 pp 45-65.

Cohn M. User Stories Applied for Agile Software Development. Boston, Massachusetts: Addison Wesley. 2004. pp 87-96.

Dubey A., Wagle D. Delivering Software as a Service. The McKinsey Quarterly. 2007, May.

Halligan B., Dharmesh S. Inbound Marketing: Get Found Using Google, Social Media, and Blogs. Hoboken New-Jersey: Wiley. 2010.

Helstead M. Elements of Software Science. Elsevier North, New York: Holland. 1977. Chapter 2 pp 9-18.

Highsmith J., Cockburn A. Agile Software Development: the Business of Innovation. IEEE Computer. 2001. Vol. 34 no. 9, September.

Hubbard D. How to Measure Anything: Finding the Value of Intangibles in Business. Hoboken, New-Jersey: Wiley. 1962. pp 119-136.

Kano N., Nobuhiku S., Fumio T., Shinichi T. Attractive Quality and Must-be Quality. Journal of the Japanese Society for Quality Control. 1984. Vol. 14 pp 39-48.

Kaplan R., Norton D. Using the Balanced Scorecard as a Strategic Management System. Harvard Business Review. 1996.

Kearney J., Sedlmeyer R., Thompson W. et al. Software Complexity Measurement. Communications of the ACM. 1986. Vol. 29 no. 11, November.

Kichenham B., Pfleeger S., Fenton N. Towards a Framework for Software Measurement Validation. IEEE Transactions of Software Engineering. 1995. Vol. 21 no. 12, December.

Kitchenham B., Pickard L., Pfleeger S. Case Studies for Method and Tool Evaluation. IEEE Computer Society. 1995. Vol. 12 no. 4, July.

Kotler P., Armstrong. G. Principles of Marketing, 7Th ed. Englewood Cliffs, New-Jersey: Prenlice-Hall. 1996.

Oliver L. Satisfaction: A Behavioral Perspective on the Consumer. New York, New York: McGraw-Hill. 1997.

Lawson M., Karandikar H. A Survey of Concurrent Engineering. Concurrent Engineering Research and Applications. 1994. Vol. 2 no. 1, March.

Leckie L. Magic Number for SaaS Companies. Available at: < http://larsleckie.blogspot.com/2008/03/magic-number-for-saas-companies.html>. Accessed on: April 15, 2011.

Lewis P. Distribution of the Anderson-Darling Statistic. The Annals of Mathematical Statistics. 1961. Vol. 32 no. 4, December.

Madhavan P., Bhattacherjee A. Understanding Post-Adoption Behavior in the Context of Online Services. Information System Research. 1998. Vol. 9 no. 4 December.

Mahajan M., Muller E., Kerin R. Introduction Strategies for New Products with Positive and Negative Word-of-Mouth. Management Science. 1984. Vol. 30, December.

Martin R. eXtreme Programming Development through Dialog. IEEE Software. 2000. Vol. 17 no. 4, July.

McGarry J., Card D., Layman B., Clarck E., Dean J. et al. Practical Software Management: Objective Information for Decision Makers. Indianapolis, Indiana: Addison-Wesley. 2001. pp 13-26.

McCabe T. A Complexity Measure. IEEE Transactions on Software Engineering. 1976. Vol. 2 no. 4, December.

Mohanty S. Software Cost Estimation: Present and Future. Software: Practice and Experience. 1981. Vol. 11 no. 2, February.

Vidyanand C. Software as a Service: Implications for Investment in Software Development. IEEE System Sciences. 2007, January.

Palmer J. Web Site Usability, Design, and Performance Metrics. Information System Research. 2002. Vol. 13 no.2, June.

Quinn J. Intelligent Enterprise: A knowledge and Service Based Paradigm for Industry. New York, New York: Simon & Schuster. 2002.

Reichheld F. The One Number You Need to Grow. Harvard Business Review. 2003.

Rhind D., Burland J., Gann D., Harding D., Howells J. et al. Hidden Wealth: the Contribution of Science to Service Sector Innovation. The Royal Society. 2009.

Ries E. Lessons Learned: What is Lean about the Lean Startup? Available at: < http://www.startuplessonslearned.com/2009/12/what-is-lean-about-lean-startup.html>. Accessed on: Feb 16, 2011.

Rosenzweig P. The Halo Effect. New York, New York: Simon & Schuster. 2007.

Schwaber K., Beedle M. Agile Software Development with Scrum. Upper Saddle River, New Jersey: Prentice-Hall. 2002.

Smith R., Eppinger S. Deciding between Sequential and Concurrent Tasks in Engineering Design. Concurrent Engineering. 1998. Vol. 9 no. 1, March.

Thomas J. A Methodology for Linking Customer Acquisition to Customer Retention. Journal of Marketing Research. 2001. Vol. 38 no. 2, May.

Van Moorsel A. Metric for the Internet Age: Quality of Experience and Quality of Business. Hewlett-Packard Company: Fifth Performability Workshop. 2001.

York J. SaaS Metrics: Joe'l Magic Number for SaaS Companies. Available at: <http://chaotic-flow.com/saas-metrics-joels-magic-number-for-saas-companies/> Accessed on: Feb 16, 2011.