MANAGING DISTRIBUTED TEAMS USING AGILE METHODS: AN IMPLEMENTATION STRATEGY FOR REGULATED HEALTHCARE SOFTWARE

by

Dylan Marks

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF BUSINESS ADMINISTRATION

In the Management of Technology Program
of the
Faculty
of
Business Administration

SIMON FRASER UNIVERSITY

Summer 2010

APPROVAL

Name: **Dylan Marks**

Degree: **Master of Business Administration**

Title of Project: **Managing Distributed Teams Using Agile Methods: An Implementation Strategy for Regulated Healthcare Software**

Supervisory Committee:

_____

Aidan Vining
Senior Supervisor

_____

Rick Colbourne
Second Reader

Date Approved: _____

ABSTRACT

Omega is a small medical software company focusing mainly on highly customized software solutions around patient communities, telemedicine and workflow optimization. The company has been in operation for nearly 10 years, with many successful project implementations, but has had little growth in this period. A set of recommendations are established for setting up an offshore team for software development, as well as moving infrastructure to the cloud to decrease costs. An analysis of strategy and process revision is required to ensure that this risky transition is effective.

This paper will analyze risks and objectives in regards to moving to offshore development for a portion of software development. It will identify necessary corporate structure, roles and processes to ensure efficient development with virtual teams. It will outline the use of 'agile' methodologies for software development in regards to offshore teams, as opposed to traditional project management methodologies. In addition, it will establish an analysis of costs and return on investment for moving to cloud for server hosting and corporate IT infrastructure

Keywords: software; development; operations; outsource; offshore; agile; lean; platform; process; medical; compliance; cloud

DEDICATION

I dedicate this project to my lovely wife, Cara.

ACKNOWLEDGEMENTS

Thanks to the staff and faculty of the Segal Graduate School of Business at Simon Fraser University, and to the amazing group of fellow students that I had the chance to learn and evolve with. Thanks as well to the executive team and employees at 'Omega' for all of their support.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

Agile	A set of software development methodologies that minimize waste and increase flexibility
ARRA-HITECH	Health Information Technology for Economic and Clinical Health Act
Backlog	Prioritized list of all future development tasks and software features
Bug	A defect in software
CFRs	Code of Federal Regulations
Distributed Teams	Teams separated by geographical distance
EDMS	Electronic Document Management Server
EMR	Electronic Medical Record
FDA	Federal Drug Administration
5:15s	Short status updates sent by employees every week, taking 15 minutes to write and 5 minutes to read.
HCIT	Health Care Information Technology
HIPAA	Health Information Portability and Accountability Act
Iteration	A fixed interval of development that finishes with delivery of incremental value to the customer, typically between 2-4 weeks in length. Also referred to as a 'Sprint'.
Offshoring	Relocating a business process from one country to another
QA	Quality Assurance
Real Options	

Refactoring	Changes to software that aren't reflected in the external operation of the software but help to improve maintainability and efficiency
SaaS	Software-As-A-Service. Software hosted on a remote server that does not require a user to install it locally
Scrum	An agile project management methodology characterized by fixed iterations of development and daily 'scrum' meetings.
SharePoint	Microsoft's document management server (EDMS)
Sprint	A fixed interval of development. See 'Iteration'.
SOP	Standard Operating Procedure
Technical Debt	Equivalent to the future cost of software support and maintenance due to aging or badly designed source code
TCO	Total Cost of Ownership: includes the hidden costs and associated support costs
Unit Testing	Automated tests used to validate software functionality. Used to ensure new changes do not introduce defects elsewhere in the system
Whole Product	Marketing strategy that attempts to meet all of the needs of customers so that they do not need to go to competitors to meet this need
Waterfall	Counterpoint to agile methodologies in software development, where development proceeds in ordered stages with documentation and specifications written completely before development begins

1: INTRODUCTION

Omega[1] is a small medical software company focusing on highly customized software solutions around patient communities, telemedicine and workflow optimization. The company has been in operation for nearly 10 years, with many successful project implementations, but has had little overall growth in this period.

The company has just recently restructured their product portfolio and established their strategic placement within the market. The company is encountering significant sales opportunities but does not currently have the personnel resources or server infrastructure to meet demand. The company needs to formulate an implementation strategy in order to scale quickly to match this demand, even though short-term cash supply is low.

Omega has requested an implementation strategy for setting up an offshore software development team and scaling the company with an outsourced infrastructure. An offshore development team will work alongside local software engineers to build additional functionality for Omega's healthcare software products. Since the company has never used outsourced development for research and development, it needs to establish new employee roles and processes to ensure project success. Being able to efficiently bring on new developers, train them and establish an effective project management methodology will be critical for the company's success.

A high percentage of the company's costs centre around the hardware infrastructure Omega uses to build solutions and deliver web-based services to its customers. This

[1] Note that the company name has been changed to protect confidentiality. Omega is not the name of a real company and any similarities to another company are purely coincidental.

hardware is mission-critical, but because of the company's small size, it is difficult to build out a cost-effective infrastructure without single points of failure. Outsourcing hardware infrastructure to a 'cloud computing' environment will operationalize capital costs and help ensure scalability and stability.

Chapter 2 reviews Omega's positioning within the healthcare IT industry. It will dissect the strategic direction of the company's product portfolio and outline the additional software development needed to strengthen their competitive position. Chapter 3 presents an implementation plan for adding an offshore development team, including contractual relationships, incentive schemes, communication difficulties and handling different cultural contexts. Chapter 4 discusses systems for knowledge management and training new employees. The project will explore strategies for successfully managing distributed teams through agile methodologies in Chapter 5. This project will conclude with a discussion on outsourcing the company's hardware infrastructure in order to decrease operational costs and minimize risk.

Some of the primary areas addressed in this paper are:

- Establishing a legal and professional relationship with an offshore team
- Ensuring proper incentive structures both internally and for the offshore team
- Setting up a project management methodology that will ensure delivery of projects on scope and on budget
- Structuring the organization with the required roles and dealing with human resources change management
- Navigating cultural differences and coordinating information systems

- Introducing agile software development methodologies and the use of lean manufacturing methods within information technology
- Analysis of decreasing operational costs through outsourcing infrastructure to cloud services

2: CORPORATE AND INDUSTRY OVERVIEW

2.1 Omega: Company Background

Omega is a small medical software company with headquarters in Phoenix, Arizona. The company offers customized software solutions focused around patient communities, telemedicine and clinical workflow optimization. The company has been in operation for nearly ten years, with many successful project implementations and until recently has been entirely founder-funded without the need for investor capital. Although it has experienced periods of moderate organic growth in the past, this growth is minimal compared to the overall industry. Ten years is an extraordinarily long time in the context of the healthcare technology industry. It has survived through two technology industry crashes by keeping costs low and bootstrapping with revenue from client projects, but has not had a cohesive product strategy to cross the chasm to mainstream use.

Until recently, Omega has focused largely on customized services and solutions, requiring development work for each new customer. There has been little focus in product offerings, which has caused difficulties in defining the market and having a directed sales approach. In the last year, the company has refined and simplified its product portfolio. This has yielded a significant increase in business opportunities that require improvements on their software. The medical software industry is in a period of intense change, and Omega wishes to capitalize on its experience to gain market share.

The company is considering turning to an offshore team for a portion of its software development, as well as moving some of its infrastructure to the cloud to decrease costs. These actions will help to ensure that the company can expand quickly and meet the

growing customer demand in the burgeoning healthcare software market. An analysis of implementation strategy and process revision is required to ensure that this risky transition is effective.

2.2 The Healthcare Information Technology Industry

In order to place this analysis in context, a brief overview of the current healthcare information technology (HCIT) industry would be valuable. The US medical software industry is an emerging field with a unique business landscape. Some of the primary features of the heterogeneous US health care system are multiple insurance providers with diverging billing requirements, competing hospitals with disparate IT infrastructures, and little standardization to share patient health data. A strong demand for software solutions will minimize these high administrative costs. Because of the multitude of approaches to delivering health care that each hospital group takes, it is difficult to prepare software that specifically meets the needs of all clients.

Medical IT still has a long way to grow and mature. A study by the US Department of Health indicated "that one-quarter of office-based physicians report using fully or partially electronic medical record systems (EMR) in 2005, a 31% increase from the 18.2% reported in the 2001 survey" (Burt, 2005). Of that quarter, far fewer had a fully implemented EHR system: "only 9.3% of physicians" (Burt, 2005) reported having the minimum baseline number of features required for a full EMR system. Although these numbers are likely much higher, the market remains unsaturated.

Omega must continue to innovate and match the features of competitors in order to maintain relevancy. The health care IT market in 2010 is highly turbulent with many new entrants. This is partly due to the Obama healthcare bill and stimulus package that placed a large amount of grant money on the table for hospitals and practices to adopt electronic

medical record (EMR) software (Goldman, 2009). As of summer 2010, one listing online showed 292 certified EMR competitors (EHR Scope, 2010; EHR Scope, 2010). Many other smaller healthcare software companies have not achieved certification or have built solutions internally for a single hospital.

This project will look at ways to ensure Omega stays ahead of the competition by expanding quickly, utilizing technical resources, and utilizing effective project management tactics to increase the velocity and reliability of new development.

2.2.1 Compliance: FDA and HIPAA

A primary factor in selling software in the healthcare market is the requirements around security and best practices enforced by government agencies. The United States government enforces privacy through the Health Information Privacy and Accountability Act (HIPAA) that dictates a set of rules to ensure the protection of patient data, including requirements such as not sending identifying data through email. Just recently, the United States government has introduced the ARRA – HITECH regulations for health care information breach notification. In the event of a breach of patient health information, Omega would need to publicly report and notify every affected individual. A breach is defined as "an impermissible use or disclosure under the Privacy Rule that compromises the security or privacy of the protected health information such that the use or disclosure poses a significant risk of financial, reputational, or other harm to the affected individual" (HHS.gov, 2009). Such an event can decisively damage a company's reputation within the insular world of healthcare information technology.

The company must also comply with rules for software development outlined by the Federal Drug and Alcohol (FDA) administration. *Title 21 CFR Part 11* of the Code of Federal Regulations provides guidelines for the use of electronic systems in the medical industry.

"Because of its complexity, the development process for software should be even more tightly controlled than for hardware, in order to prevent problems that cannot be easily detected later in the development process" (FDA, 2002). This places a heavy burden of paper trails, audits and administrative overhead to enforce software quality, but this overhead helps to stabilize a highly complex industry.

Although complying with these burdensome regulations can stifle innovation, there are strategic benefits to Omega. The regulatory requirements act as a barrier that helps to decrease the threat of new entrants (Porter, 1998, p. 13). It signals to potential customers that the company is reliable and trustworthy, helping to differentiate it from other competitors.

Internally, these regulations push employees to act in a careful and secure way in order to protect private information since "software engineering needs an even greater level of managerial scrutiny and control than does hardware engineering" (FDA, 2002). The documentation requirements prompt the company to think in a risk-aware way. This paper will address these compliance requirements, since they influence Omega's project management methodology, approach to infrastructure outsourcing, and use of distributed development teams.

2.3 Strategic Factors for Firm Innovation

2.3.1 Omega's Platform Strategy

The company uses a platform-based approach to delivering software. A single code base, the Omega platform, acts as a starting point for all software customizations. Omega developed this unified software platform over the past 8 years, and it now comprises tens of thousands of lines of code with legacy features and a long history of implementation on

software projects. Omega's business analysts use this platform to build disparate software solutions including:

1. Patient communities for chronic disease management
2. Tissue bank software for tracking and shipping specimens
3. Telemedicine software allowing remote specialist physicians to review an electronic medical record and use video conferencing to consult with a patient
4. Tracking, notification and paging solution to ensure throughput of patients in testing and surgery centres

In the past, Omega customized each application for the information needs and technical requirements of a particular health organization or department. As the company initializes each new project, the software engineers look for ways to generalise and abstract the features to add them to the underlying platform. This allows the company to use these features in future solutions with minimal additional configuration. The company uses "radical customizability overlaid onto a constant and reliable foundation, dramatically shortened times to market, relatively small production runs, and an intense focus on customer service" (Moore, 1999, p. 12). For example, since the patient community and telemedicine product lines share the same code base, it is trivial to include the video conferencing module in a 'patient community' application, even though Omega did not initially develop the conferencing module for the patient community product.

The result of this platform strategy is highly complex and interrelated code. Developers must maintain many legacy interfaces to ensure backwards compatibility. Each new feature added to the software takes much longer than it would if the company

developed a 'one-off' solution. Developers must abstract each feature with configuration settings and add them to the library.

This approach helps to minimize the number of moving parts. Once the quality assurance team validates a new release of the platform with one application, developers can quickly roll it out to other customers without full rework. This spreads the cost of development across multiple projects. By standardizing the underlying code across multiple products, the cost of delivering customized services to a new customer decreases.

The complex platform requires highly skilled engineers with an understanding of how the different pieces interact. Developers must maintain upgrade scripts to automate moving between versions of the platform, further complicating the addition of new features. The company must dedicate significant resources to train new developers in this complexity while staying aware of the risk that employee turnover carries

Over the past year, Omega has strategically refocused by decreasing the number of services it offers and by marketing directly to specific submarkets in the healthcare industry. The company is shifting towards a product-based strategy rather than a customized service-based approach where each customer receives a unique solution. Non-customized software "can be transported and replicated at essentially zero marginal cost and its use by one party does not preclude use by another" (Kogut, 2001, p. 248). The company hopes to sell more of these non-customized products with a lower development cost in order to increase sales and revenue. Omega will continue to use its existing platform strategy with all of its products sitting on top of a core software framework, but will attempt to standardize offerings to capitalize on economies of scale.

A majority of new software solutions using the Omega platform can be prepared using business analysts rather than programmers. This allows Omega to have a high ratio of

projects to developers. At any one time, the company has 6-8 projects in the active development pipeline with only 2-3 software engineers currently on staff. Current projects in the sales pipeline have a high level of customization requiring new software development. Omega requires a larger technical workforce to meet this demand, which may be difficult. The company faces a period of low cash flow as it restructures its product portfolio and attempts to enter new product areas. This analysis will be looking at ways to both decrease current infrastructure costs as well as bring on more developers to help strengthen the company's market position.

2.3.2 Strategic Use of Technical Debt

Before going into detail on the implementation of an offshore development team, a framework needs to be set up for analysing cost of development and code quality. Code quality is both highly important and difficult to measure. Managers are in the awkward position of only seeing the surface of their software and not the quality of the underlying code. If the quality of the underlying code is low, it can introduce significant business risks and liabilities.

A piece of software might have a very nice visual design but could also have a hastily planned architecture that will result in future maintenance headaches. Issues do not become evident until much later when bugs arise and developers need to work on fixes and updates to production systems. The manager begins to notice schedule delays and increases in reported defects. Technical debt is the hidden costs of future support and redevelopment of legacy software (Cunnigham, 1992).

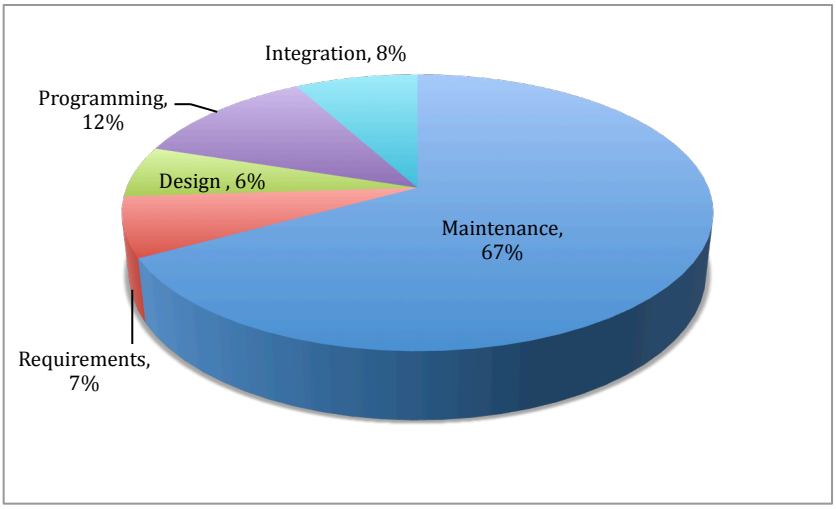
This technical debt is analogous to other financial measurements that affect the profitability of a company. In the same way that companies can over-leverage regular debt,

"development organizations let their technical debt get out of control and spend most of their future development effort paying crippling interest payments" (Atwood, 2009).

Rushing software development schedules and cutting corners is equivalent to taking a loan against the future value of the software. Inexperienced developers can make mistakes in designing software that will cause future maintenance problems. Complications pile up to swamp a company in constant support problems. One loose metric for level of technical debt is 'Lines of Code' (LOC). As the code base becomes larger, it becomes more difficult to maintain. If managers understand the concept of technical debt, they can effectively manage it and understand its influence on firm competitiveness (O'Connor & Bowe, 2010).

Omega's platform has a legacy code base of approximately 20,000 lines of code. The company must be careful not to take on too much technical debt. Omega faces the potential of "a legacy code base in which so much work goes into keeping a production system running (i.e., 'servicing the debt') that there is little time left over to add new capabilities to the system" (McConnell, 2007). Making this technical debt visible to the executive helps to ensure a proper development strategy. Using project management software that tracks bugs and the time spent on maintenance helps to make this technical debt visible. Later sections will discuss these project management tools for quantifying technical debt.

Figure 1: Relative Costs of Software Lifecycle



Adapted from (Schach, 1999, p. 11)

When estimating the cost of new software development, maintenance costs are the most difficult to plan for. Figure 1 above shows relatively how much of the cost of software development centres around maintenance. When developers build software once in a careful, well-structured way, business analysts can use it in other solutions without additional development work. Whenever adding new features to the platform, Omega must dedicate additional time to modularizing it and ensuring compatibility with the framework. This helps to decrease maintenance costs, since future product sales do not require additional development.

Although decreasing maintenance costs, upfront development costs are much higher with a modular design. However, the firm dramatically decreases incremental (or marginal) costs as it uses the software in multiple solutions. In other words, the firm is able to capitalize on economies of scope, in that it is "less costly to combine two or more product lines in one firm than to produce them separately" (JC Panzar, 1981, p. 268). By

approaching new software development as the creation of a product, the marginal costs from development go down with every new sale.

The downside of a long-term strategy is that by building new features and products in a robust manner from the start, the company sacrifices flexibility and speed to market. In certain situations, one can willingly take on technical debt in order to build momentum or to be a first entrant to a new market. Hard-coded settings and shortcuts allow a product to get to market that much faster: "Startups can benefit by using technical debt to experiment, invest in process, and increase their product development leverage" (Ries, 2009). The executive team can knowingly take on technical debt to invest in new markets. The concept of investing in new markets takes us to a similar subject: the use of real options analysis for calculating the value of innovation.

2.3.3 Using Real Options in Software Development

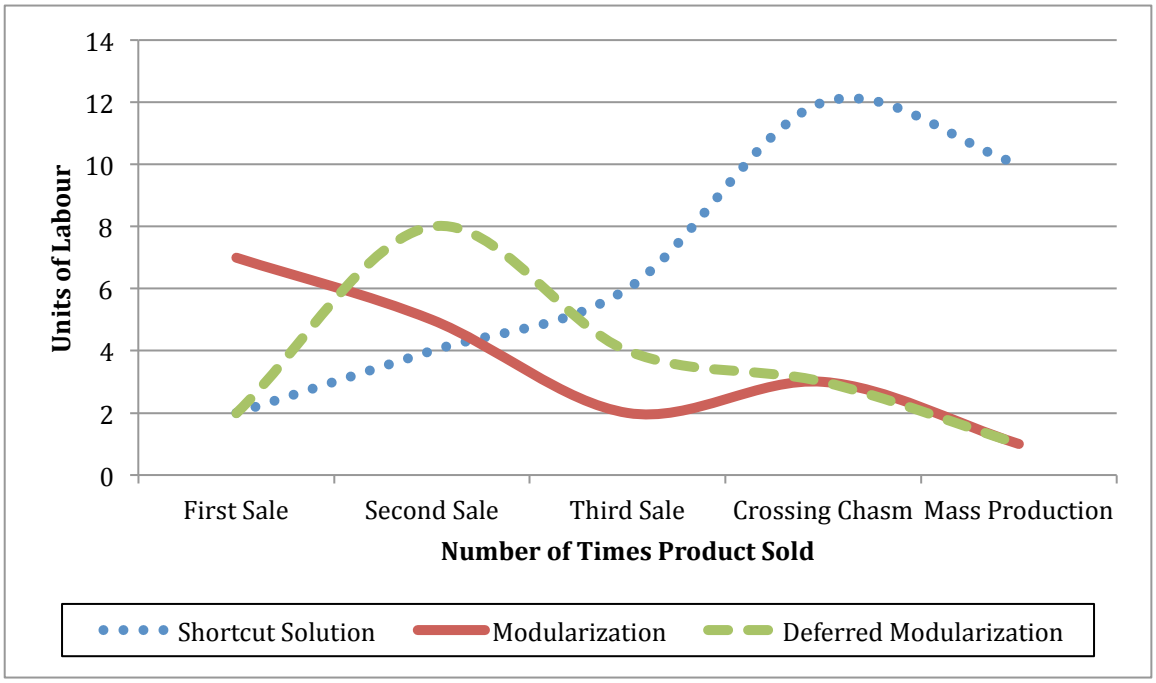
In finance, real options analysis helps to quantify the value of flexibility. It is a set of equations that "treats flexibility in capital investment decision-making as an option and values it as such" (Sullivan, 1999, p. 221). The traditional valuation of investments (such as using net present value calculations) does not take into account the benefit of being able to defer decisions until a future date (Luehrman, 1998, p. 2). The concept of real options is valuable for quantifying the investment in software development. Rather than investing fully in a new product, developers can create a proof of concept for limited cost, with the understanding that they will likely need to rebuild it in order to sustain mass production and long-term use. This investment is analogous to the 'exercise price' of buying a Real Option in finance. The exercise price is the initial cost that gives the firm flexibility to enter a market in the future. Although using the actual quantitative calculations is not practical in

day-to-day decision-making, management can apply the concept on a psychological level to accept the intrinsic volatility of the development process.

Using a real options approach to the product roadmap defers decisions so that the firm can respond quickly to new market forces. In this way, new features on the roadmap are not committed to until the development team is ready to start working on them. Software development is "largely a process of decision-making under uncertainty and incomplete knowledge, including threats of competitive entry" (Sullivan, 1999, p. 219). By using a real options approach, the company can respond fluidly to changes in the market and customer needs.

When designing for the long-term, it is not always easy for the software developer to foresee how system requirements will change, and thus "a modular structure that makes one set of potential changes easy by localising their impact might not localise the impacts of changes that turn out to be needed in practice" (Sullivan, 1999, p. 234). This is another reason for using proof of concepts and deferring the development of modular structures until a second sale. The figure below compares the approach of building a shortcut solution with the upfront costs of modularizing a new feature so that it can be used in future solutions with limited cost. Omega takes a middle ground with the approach of 'deferred modularization', where the initial development is a proof of concept to enter the market, with modularization only being applied when the market has been tested and other customers are interested in the feature.

Figure 2: Balancing Development Costs During Life of Software Product



Source: Author

This concept of using real options in software development aligns closely with agile project management. Chapter 5 will discuss agile methodologies in more detail, but the foundation concept is that teams should break software development down into small iterations, since "when you use an early version of some software you really begin to understand what features are valuable and what parts are not" (Fowler M. , 2005). This real options analysis is a starting foundation for the discussion of this approach to project management.

2.3.4 Testing the Market with a Broad Product Portfolio

The use of a real options strategy for product development means that Omega offers a very broad yet shallow set of products. By testing the market with a number of different product offerings, the company can be highly flexible and respond to the specific needs of a customer. It is also the foundation of a platform approach to product development. Since the company bases each unique product off a single platform, analysts can interchange

different features and modules specifically for a customer's needs. The customer can choose from a wide shopping list of available modules, with unique data collection forms for specializations such as neurology, oncology, nursing, or stroke.

Omega has a long list of features that it would like to offer on the platform. Since each of the applications that the company sells uses the same platform, one can mix-and-match modules depending on business need. For example, one customer might upgrade their patient registry to include a video conferencing module and pager notification system for emergencies. Another customer might focus on tissue bank software-, but still be able to add in the ability for calendaring and for patients to log in and check their test results. By having many modules available, Omega can provide a menu of possible choices to up sell to their existing customers.

The 'Whole Product' strategy attempts to avoid a customer needing to go to a competitor to fulfil requirements that the firm's product cannot meet (Moore, 1999, p. 68). By having a broad range of modules available, as well as the ability to provide the complete services to solve the customer's problems, one can start to become the standard in the market. The difficulty in implementing a whole product approach is that it requires an incredible range of functionality to be included in the platform. As well, by offering customized configurations and specialized services for each customer, the marginal cost of each software sale is much higher than with turnkey solutions that have no customization.

Omega's strategy of testing the market with proof of concepts is similar to Microsoft's approach in early years: "Microsoft's relationship with real options is well known, like the famous trade show where the Microsoft stand 'looked like a bazaar'" (Matts, 2007). Rather than focus on one product, Microsoft hedged its bets by offering a broad array of possible products that might become successful. This contradicts the common wisdom in

regards to small companies, which recommends an intense focus on a single product as a 'lead bowling pin' to establish a presence in an industry (Moore, 1999, p. 39). The primary requirement for success with this 'bazaar' approach is finding a way to exercise entry into new markets with a minimal upfront investment. Omega is able to use this real options approach because it can quickly build specialized applications for niche markets using the company's platform.

Regardless of the speed of development, pursuing such a strategy requires extensive technical resources. Omega has decided to establish an offshore team to accommodate this need. The next section will look at some of the possible product improvements that would make the most sense for an offshore team to address.

2.3.5 Outsourcing and Commodity Software

The open source software (OSS) movement is a primary force for commoditization in the software market. This disruptive force serves to undermine the profit margins for non-open source companies (Carroll, 2003). Software can become commoditized in the same way as hardware, and "prices will fall to zero or near zero for any kind of standardized product" (Cusumano, 2008, p. 24). With information-based products such as software, "the fixed costs of production are large, but the variable costs of reproduction are small" (Shapiro & Varian, 1999, p. 21). This naturally pushes the price of non-differentiated software towards the cost of reproduction, which for software is very low: "the producers with no sales all have an incentive to undercut the competition, and there is no natural floor on prices except the $1 a copy reproduction costs" (Varian, 1995, p. 1).

Omega uses a mix of customization services alongside standardized product offerings to offset this force of commoditization and differentiate itself from competitors. It offers all of its software as a service (SaaS), so that customers do not need to install or host

the hardware themselves. This provides Omega with ongoing revenue from support contracts, and helps to differentiate the company from competitors. By integrating services and products, Omega can avoid commoditization as well as "generate new revenues and profits, even as the product business declines" (Cusumano, 2008, p. 26). Omega must find the proper combination of standardized product offerings and differentiating services.

The tendency towards commoditization in the software industry is a disruptive force that could undermine Omega's revenue. Yet it also allows Omega to expand the types of services it offers by integrating with third party commoditized software. Omega can integrate with third party software and services either through strategic partnerships, or by integrating the source code with the rest of Omega's platform. Omega has leveraged third party partner services to offer features such as video conferencing, credit card processing, electronic medication prescriptions and content management. As the number of competitors offering these services grow, they tend to become standardized. For instance, with a large number of different video conferencing applications now available, they have become "interchangeable commodities" (Murdock, 2009) that Omega can swap out depending on the customer. Omega can reach the goal of providing a whole product by pulling together all of these different services and packaging them up as a full solution for a customer.

2.3.6 Integrating Third Party Software and Services

Omega cannot always leverage available open source alternatives when looking to add new features. Often user agreement licenses tied to open source software would not be appropriate for integrating into a commercial product. These licenses explicitly require that a company open source their own software if they integrate with these packages. In other

cases, the open source packages are not available in the Microsoft-based programming language that Omega uses.

Duplicating or integrating open source commoditized software would be an excellent candidate for outsourcing since “outsourcing activities should not be idiosyncratic, critical or surrounded by too much uncertainty" (Bartholemy, 2001, p. 68). The outsourced team would need a defined scope and clear boundaries of success. Developers can copy an entire piece of software without actually copying a single line of the source code (Carroll, 2003), thus not breaking any intellectual property constraints. The following sections on setting up the offshore team will discuss models of outsourcing and the question of complexity in more depth.

Purchasing closed-source software from third parties is another potential way of quickly adding value to Omega’s platform. As with open source software, the most important deciding factor of integrating with these third party packages is ensuring that the terms of use do not cause difficulties in the future. The best license for purchasing source code is a one-time upfront fee with no recurring licensing terms. Otherwise, this legally connects the two companies, and any development on top of the third party source code opens up the possibility of opportunism by the licenser.

Another drawback of integrating third party solutions into Omega’s platform is that integration increases technical debt. Developers will need to do a lot of work to make sure that the different code bases interact cleanly and do not have any conflicts. Security analysis is also very important to protect patient privacy. Omega recently integrated an open-source blogging engine into their platform, and development costs were higher than expected because different modules did not interact well together. Developers needed to review thousands of lines of code to conduct proper due diligence. Integrating third party

software can save huge amounts of development time in order to offer a whole product. Yet Omega still faces a large amount of development work in order to meet the entire needs of a customer, requiring an effective implementation of strategy.

This concludes the brief analysis of Omega's strategic positioning within the medical software market and its method of delivering value to customers. This analysis outlined the value of using customized services to avoid the disruptive force of commoditization. It discussed the whole product strategy that attempts to meet the complete needs of a customer and the use of a platform approach to achieve economies of scale. It noted the advantages and dangers of leveraging technical debt to test out a market. All of these strategic approaches require substantial technical resources. Currently, Omega does not have enough developers to follow this strategic direction. It needs to find a way to add more employees while carefully moderating costs. The next chapter will present an implementation plan for expanding these technical resources with outsourcing and a distributed offshore development team.

3: DECIDING BETWEEN OUTSOURCED CONTRACTS AND VERTICAL INTEGRATION

In the next chapters, we will look at the alternatives for outsourcing, the ways to mitigate risks involved with an offshore team, and the approaches for successful implementation of a distributed development team.

At the time of this analysis, Omega's executive team had already decided to turn to offshore development to meet its growing needs for technical resources. It will set up a team based in a country with a lower cost of living and thus lower labour costs. The company pursues this strategy in order to achieve economies of scale while keeping operational costs low at a time when the company has high expectations for growth but limited current cash flow. The company is in the initial stages of setting up this team. This analysis will not question the underlying decision to use an offshore team, but will look at some of the risks involved and ways to mitigate them. The following section will look at the primary modes of cooperation in outsourcing and what legal arrangement is most appropriate for different types of projects. The company will continue to expand technical resources in the future, and this analysis will help establish a framework for choosing the type of outsourcing relationships.

3.1 The Hidden Costs of Outsourcing and Offshore Development

Expectations should be set at the start that cost savings for offshore development will not reflect the possibilities implied on paper. At the outset of setting up an offshore team, executives might optimistically expect savings to match the difference in labour costs, since a full-time equivalent in India will cost around 40% less (Davison, 2003). This

expectation does not take into account hidden costs. In reality, "studies show that only about half of IT outsourcing contracts deliver the promised 20-30% cost saving" (Kakabadse, 2002, p. 189). A careful structuring of the contractual arrangement as well as the establishment of a strong project management process will help ensure that the company can realize some of the cost savings that offshore development seems to offer.

Some of the hidden costs:

- Communication costs: Language barriers and lack of business context for the offshore team can result in delivery of software that does not meet the underlying intent of specifications, resulting in extensive rework.
- Temporal delays: The offshore team works opposite hours of the US team, so any questions will cause delays in resolution.
- Transaction costs: This includes both the cost of legal fees and the amount of work finding a reliable outsourcing partner.
- Technical debt: Poorly designed software accumulates debt, in that the cost of initial development might be low, but unexpected maintenance costs will arise. Any analysis of the value of outsourcing should take into account this hidden cost, which is equivalent to the future cost of maintenance and support for the product.

Working with a team in a widely divergent time zone with different cultural and technical worldviews introduces significant risk. It is difficult to forecast the true extent of the hidden costs listed above. This analysis will look at techniques and strategies to mitigate this risk.

3.1.1 Government Regulation and the Medical Industry

Compliance with government regulations comprises one unique cost associated with outsourcing in the medical industry. As a company following both HIPAA and FDA requirements, Omega must make sure that the offshore vendor is willing to submit to audits and "provide sufficient transparency" (Davison, 2003) in order to comply with government regulations. Consulting with a lawyer specializing in medical law will be of utmost importance here.

Part of the legal contract with the outsourced vendor should address patient privacy requirements, the importance of security, and the requirement that the offshore team notify their management if they inadvertently encounter private health information. Omega should provide documented standards of operation to the offshore team. These standards should focus on permission levels to production data and ensure that developers do not use real patient information in testing scenarios. The section below on the project management process will address the transparency required to meet these regulations.

3.2 Deciding Between Contracts and Long Term Relationships

One of the primary costs of using a development team in a different country is the transaction cost of selecting a vendor or navigating the complicated process of hiring. Omega's management has a strong network of corporate partners that it is leveraging to help find an appropriate team and provide advice on meeting the legal requirements for contracting teams in foreign countries. There are different possible models for bringing on an offshore team, including using a contract-based consultancy company as intermediary, or actually setting up an office in the country and hiring directly.

In order to derive value from the financial investment of hiring an outsourced team, one needs to establish the legal foundation for how the two organizations cooperate. There

are two main types of contracts: fixed price contracts and time and material contracts (Gopal & Sivaramaktrishnan, 2003). In searching for an offshore vendor, Omega broke this down into three alternatives for structuring the relationship:

1. Establish project-based fixed price contracts with an outsourcing company
2. Hire employees in the country using a proxy onshore management company
3. Hire employees in the country and set up a company office directly

The latter two options are effectively time and material contracts, while the third option creates a long-term employee relationship. The third option is equivalent to the vertical integration of the company's supply chain. This employee relationship helps to align incentives with the contracting firm. Naturally, there are trade-offs between choosing one of these models of cooperation, and the decision of which approach to use depends on the type of work that the firm needs to outsource.

With fixed price contracts, a large part of one's costs will centre on having to write detailed specifications and negotiating terms. The transaction costs become even higher when factoring in the cost of communication breakdown, in the event the two parties understand the project requirements in different ways. This type of relationship is most effective with simple, standardized types of development that do not require extensive training of the outsourcing vendor.

Conversely, one can set up a remote office in the country and hire the team as employees. An employee relationship is a more integrated, internalized approach, but requires more work at the onset to set up a legal relationship in the country. This vertically integrates the company rather than outsourcing part of the product development. It will require negotiating the subtleties of the country's legal environment. Once established

however, this approach helps to do away with the high transaction costs of project- and task-based contracts.

Outsourced Contract Relationship

- Quick setup
- Low start-up up cost
- High operational costs
- Best for short-term investment and commoditized tasks

Company Branch/Employee Relationship

- High start up cost
- Lower operational cost
- Requires a higher investment and ties the company closely with the chosen location.
- Best for long-term investments and complex innovative development

3.2.1 Fixed Price Contract Relationships

Omega works with its own customers in a contractual, project-based approach to delivering services. The company derives 80% of revenue from offering fixed price service contracts with its clients. The Phoenix-based team has extensive experience in developing detailed statements of work for client projects. This might seem, on the surface, the most effective way to avoid price overruns, since the parties agree on price at the start of the project. The problem with a piecemeal contract approach is that defining the specifics of new research and development is extremely difficult. The vendor cannot completely understand the extent of work for a research and development project until work has

started. Once the development team begins to actually build proofs of concept, the true effort required starts to become clear. This results in changes to scope and cost overruns.

A fixed price contract attempts to shift all of the risk to the outsourcing company. Yet, there is no alignment of incentives to ensure that the third party does not cut corners and deliver the lowest possible denominator in terms of quality of the output. Using an independent outsourcing company opens up the possibility of opportunism, with this intermediary being the sole provider of production for Omega and thus holding all the bargaining power. Opportunism is "any behaviour by a party to a transaction designed to change the agreed terms of a transaction to be more in its favour" (Vining & Globerman, 1999, p. 646). One primary source of opportunism with outsourcing is the hidden cost of technical debt. If the outsourcing vendor does not have incentives aligned with the long-term health of the company, then the outsourced developers might save time by not designing maintainable software.

With software, especially Omega's highly complex platform model, the quality of the core software framework is of utmost important for future projects. With a fixed price contract, the outsourced company will want to deliver as fast as possible with as few resources as possible. It does not need to think about on going maintenance and support of the delivered software. This future cost of support can often be far higher than the original development cost if the quality of the code is low.

Omega has decided not to use a contract-based third party for outsourcing development for the reasons outlined above. This seems to follow the trend in the industry, since "the empirical evidence supports the idea that product complexity raises the probability of internalization" (Vining & Globerman, 1999, p. 647). Since Omega requires

development on legacy software with high complexity, an outsourced approach would be problematic, since the company will need to sink a lot of time into training the vendor.

There may be small development projects to pass out to a third party with clearly defined parameters that will avoid the constant cost overruns that often come with contract-based development. The analysis will go into more detail in a following section on the most effective way to use these spot-market contracts.

3.2.2 Using an Onshore Management Company

Another approach that is becoming more popular is using a management company with a team lead based in the US that holds an employee relationship with the offshore office. These management companies have already invested in the legal costs of setting up a legitimate business, often have a deep understanding of the cultural and language, and have had experience with achieving productivity out of a remote team.

Using a management company, one can benefit from the quick start-up that a contract-based approach offers, since the management company has already established a legal relationship with the offshore employees. One can also benefit from the lower transaction costs of not having to prepare a contract for each task. Omega lacks international expertise as well as contacts within the offshore countries, and so would find value in an intermediary that has experience handling the "cultural, professional and operational complexities of managing relationships across borders" (Mahnke, 2008, p. 64). The management company can also begin to establish economies of scale as multiple dedicated teams share the same office space and infrastructure in the offshore country. By using a US-based management company to handle the employee relationship, Omega is able to benefit from the stronger legal protection available in US corporate law. Since the

management company is located within the United States, it is liable for breach of any nondisclosure agreements (NDA) and contracts.

The management company would benefit by taking a percentage fee of the employee wages for the handling of the legal and financial overhead of managing the offshore employees. One way for Omega to align the management company with Omega's objectives would be to establish an equity relationship. Setting up this type of a relationship does add another layer of legal complexity to the arrangement but is critical in order to harmonize the incentives of the manager and Omega.

The executive team previously negotiated with a management company offering this sort of relationship. However, after a review of costs and taking into account the misalignment of incentives, it will instead pursue an employee relationship by hiring a team of developers directly. Omega was not willing to part with the necessary amount of equity required to ensure that the management company had sufficiently aligned incentives to make this approach worthwhile.

3.2.3 Offshore Employee Relationships

Because of the high level of complexity in Omega's product development requirements, a vertically integrated employee relationship seems to be the most viable approach. This approach minimizes the level of opportunism, since "employees within organizations have better and more numerous opportunities to 'pay back' (and, therefore, discourage) opportunistic fellow employees" (Vining & Globerman, 1999, p. 646).

By establishing a dedicated team in a closer relationship, one can train more deeply and integrate the team as a branch of the company. Since Omega's software framework is highly complex with a large amount of legacy code, minimizing staff turnover will be valuable for maintaining efficiency. The projects that Omega will need work completed on

have a high level of asset specificity, and "asset specificity and added uncertainty pose greater needs for cooperative adaptation" (Williamson, 2008, p. 9). Long-term relationships result in economies of learning: "the benefits of the set-up costs can be realized only so long as the relationship between the buyer and seller of the intermediate product is maintained" (Williamson, 1979, p. 240). Employees become more efficient the longer they work with the company's software.

Setting up something closer to an employee relationship for the offshore team would be beneficial for this reason. Legal costs produce the main roadblock to using this type of relationship. Omega has been fortunate to have a technology partner in its network with an established office in Bangalore, India. This partner company has underutilized office space, providing Emerge MD with the opportunity to share resources. Omega can rent this space along with the services of the Indian project manager to accelerate the rate of hiring a team. This relationship is the best of both worlds: the quick initiation of an outsourced contract with the long-term stability of a permanent employee.

3.3 Risk Analysis of Using Offshore Resources

3.3.1 Intellectual Property Protection

Since Omega has a large percentage of value tied up in software assets, handing over source code to an offshore team is a delicate issue. The company's software is not unique enough to protect through legal patents. Instead, Omega relies on 'hidden secrets' to maintain competitive value. How does one make sure that the remote team, operating under a legal domain where the company has no foundation, not simply walk away with the source code and sell it to a competitor or start a competing venture? The offshore group could easily go out to their own customers to sell the software and Omega would likely never know about it.

Omega's competitive value comprises more than the software source code. Omega's capabilities arise from its unique network of clients, customers and industry contacts that it has built up over a decade in the industry. The company's executive and sales team boasts medical doctors with extensive contacts, and has a team of experienced business analysts who understand how the industry works. The company also has service agreements with a variety of major corporate partners and hospitals that a potential competitor cannot reproduce with only the source code. Competitors can easily copy software, but Omega's distinct position within the healthcare industry network offers a much stronger advantage.

An attorney specializing in offshore outsourcing can provide a sufficiently airtight nondisclosure agreement. However, nondisclosure agreements are difficult to enforce and challenging to prove without a prolonged legal battle. The most effective way of minimizing the threat of IP theft is through providing positive incentives to employees, such as offering a small equity amount in the form of stock options.

3.3.2 Software Development in Discrete Blocks with Interfaces

One approach used in the industry to protect intellectual property is to break down projects into discrete blocks of functionality. Unrelated teams would then develop each of these blocks independently.

One company interviewed for this paper split a project's scope up into independent sections to ensure no single outsourced vendor had access to the entire source code (anonymous, 2010). This would be a good way to approach a single use contract relationship. Rather than providing the entire code base to work with, the product development manager could give the offshore team discrete blocks of source code specifically on the area of interest. Management would provide the rest of the referenced code libraries as compiled packages without original source code. By establishing set

interfaces and rules for how these discrete components interact, the outsourced team would not need to have access to the full source code. This protects the company's intellectual property.

This final integration step can be extraordinarily complicated and drawn out unless the coordinator has carefully designed and specified the interfaces beforehand. Unfortunately, this approach creates much more work for the local development team. Before beginning a project, the local developers must dedicate significant time to designing airtight interfaces. They must write specifications with concise descriptions of all of the required interactions. After the outsource team delivers on scope, the local team must then integrate this work into the rest of the platform and resolve any conflicts with the rest of the code. Often difficulties arise, as developers need to adapt the code fragment to fit effectively with the rest of the software.

Such a fragmented approach greatly decreases the ability of the company to adapt to the changing market and needs of its customers. Upcoming sections of this paper will introduce the concept of agile project management methodologies for a more efficient and integrated way of working with the offshore team. These methodologies defer the amount of upfront design costs in order to maintain flexibility. The company can deliver with more agility when developers merge the outsourced code into the main product as quickly as possible.

3.3.3 Using Spot Markets for Discrete Development Tasks

The interface approach to development outlined in the previous section would work well for short-term contracts, rather than with an employee-relationship based arrangement. The company has already decided to vertically integrate by hiring offshore employees rather than use contracting. However, Omega will continue to grow and will

likely need flexibility to scale depending on specific project requirements. Having the ability to balance both vertically integrated and outsourced development gives the company additional flexibility.

Product management can do an occasional review of the product roadmap for possible development areas that are discrete, standalone modules such as described in the previous section. These modules would be excellent candidates for contract-based outsourcing when the company needs to accelerate research and development without a permanent increase in employees. A number of IT expertise marketplaces have gained popularity online, such as *elance.com, guru.com and odesk.com* (Sivers, 2010). These are essentially spot-markets where a firm can place a Request for Proposal and have independent consultants bid on it in an auction. As discussed in previous sections, the firm should limit requests to simple and straightforward projects in order to minimize the "substantial costs of bidding and evaluating bids" (Snir, 2000, p. 2).

3.4 Managing Change while Preventing Opportunism

3.4.1 Opportunism and the Prisoner's Dilemma

As the company adds new technical resources, management must look at ways of minimizing opportunism, since "problems of asymmetric information are common when buyers and sellers have incomplete information regarding their trading partners" (Snir, 2000).

The remote team would likely prefer longer-term projects, to help minimize the concern that the work will disappear after the project is completed. Small granular projects have no indication of future work so the team might decide to pad hours or attempt to expand scope in order to ensure they continue to have work. The executive team can mitigate this through transparency about the backlog of projects that are on the roadmap

and explain that once a project is finished there are many more projects waiting. Providing the remote team visibility into the amount of work pending can provide extra motivation to increase development velocity.

Promoting the possibility of future work also decreases the potential for opportunism. If the contract is a single event with no future interaction, then the contracted parties have less incentive to keep the terms, since there is no future gain from cooperating. As in the Prisoner's Dilemma, the equilibrium point shifts towards cooperation when the game is infinitely repeating. The outsourced parties have more incentive to cooperate to avoid the possibility of retaliation if there is (at least) the possibility of future rounds of the game (Axelrod, 1987, p. 8). Having multiple (iterated) rounds of the game decreases the potential payoff for selfish activity: "No matter what the other does, the selfish choice of defection yields a higher payoff than cooperation. But if both defect, both do worse than if both had cooperated" (Axelrod, 1987, p. 3). Making sure that the relationship with the offshore team is equivalent to an iterated prisoner's dilemma will help minimize the chance of opportunism.

Reputation is a major factor in minimizing opportunism. If the arrangement is public to the software industry and influential to others within the industry, the parties will want to ensure that they do not convey negative signals. This could affect their ability to do future business in the industry and results in a social cost.

Perhaps the most effective way to minimize opportunism is simple respect for the other party and acting reliably and professionally. "When the relationship between the client and its vendor is adversarial, the vendor will take advantage of gaps in the agreement" (Bartholemy, 2001, p. 66) and possibly take advantage of asymmetric information. Managers can minimize opportunism by providing basic respect to employees

and building positive teams. The following chapter will look at ways to minimize opportunism throughout the firm through positive incentive structures, building cohesive teams, and avoiding communication difficulties.

4: SETTING UP AND TRAINING THE OFFSHORE TEAM

The previous chapter looked at techniques and approaches for structuring the relationship with the new team, including ways of minimizing opportunism and expanding resources when needed through spot-market contracts. The analysis will now look more closely at the implementation plan, which requires a careful look at proper incentive structures, ways of building a cohesive team, and techniques for training the new team.

4.1.1 Aligning Incentive Structures

Omega needs to establish incentive structures that will help motivate employees to follow the company's strategic goals. Management has not yet used rewards systems to motivate employees, even though "a creative and thoughtful review of the bases for rewards can have a strong impact on the performance of an organization" (Stonich, 1981, p. 347). Management should think about incentives and building team cohesion with both the current local employees and with the new offshore team.

Employees often meet intense change with resistance, and might even subtly undermine or sabotage an implementation that pushes them out of their comfort zones. Management must ensure that any incentive scheme not only rewards the short-term success of individual projects, but also establishes incentive structures that reflect the long-term company strategy.

Poorly designed incentive structures can actually harm the firm by "hindering efforts toward strategic thinking and overemphasizing short-run results" (Stonich, 1981, p. 348). For example, if a developer only receives bonuses for delivering a project on time, they might cut corners that could result in serious maintenance costs in the future. In this

regard, management should associate rewards to both releasing a project on time and to metrics that track the number of support issues on the project in future quarters. However, such metrics around technical debt rely on robust and reliable reporting that is difficult quantify. Using such metrics actually encourages gaming the system rather than working for the benefit of the company.

Rather than individual bonuses for specific developers, management should reward the entire team. "Agile practices don't encourage heroism or the individual developer but instead focus on team accomplishments" (Spagnuolo, 2008), and so rewarding a specific member might actually cause competitiveness and undermine the team cohesion. Since Omega is a small company with little division between business units, the most effective incentive scheme would be stock options that reward employees for the holistic success of the entire company.

4.1.2 Creating a Positive Workplace and Maintaining Talent

Even if employees show outward compliance with an implementation, the work climate might begin to devolve. Outsourcing can create a negative climate for employees that are affected by the change, since "there no longer exists a supportive social contract between employer and employee, creating an environment of insecurity and limited commitment to the workplace" (Kakabadse, 2002). Using an offshore team at a drastically lower cost can undermine the morale and increase the anxiety of the local employees.

Managers should emphasize the value of all employees and reassure them of their future with the company. Retaining talent is of utmost importance, and management should strive to build a culture that embraces and promotes these individuals. Methods to offer this feedback include providing the opportunity for employees to retrain and evolve into

other roles, and treating employees as valuable team members rather than as disposable resources.

For example, Omega is moving to an outsourced hardware infrastructure. This could cause considerable stress for the network administrators who handle the maintenance and support of the existing infrastructure. How does one incentivize employees to work effectively on a project that could possibly put them out of a job? In the implementation of the cloud computing transition, managers should emphasize the organic nature of the network administrator role. This role will evolve to focus on maintaining the online systems. This would leave these employees more time to establish emergency preparedness plans or optimize systems.

As a small company in a highly volatile industry, Omega requires employees to use this sense of urgency to fuel a passion for constant innovation. Management "needs to avoid displacing intrinsic motivation with extrinsic motivation" (Spolsky, 2010). If positive incentives do not result in performance, certain employees that cannot embrace change may need to be let go in order to retain a lean and efficient firm. Management can inspire intrinsically motivated employees by establishing a positive corporate culture and respectful workplace. Even if restructuring removes specific roles, management can recognize the value of talented individuals and retain them.

This basic respect extends to the offshore team as well. Management and employees should not treat the offshore team as temporary workers but as legitimate and valuable team members with the same access to incentives and consequences as the rest of the local team. As well, the company should pay offshore employees a comfortable living wage.

As part of the initial relationship building between the two teams, a virtual staffroom can be set up on the company SharePoint portal for employees to post their

photographs, create conversations and establish relationships. Management will ask everyone to post a PowerPoint with personal information about his or her family and history. This can help create a collegial environment and begin to build team cohesion.

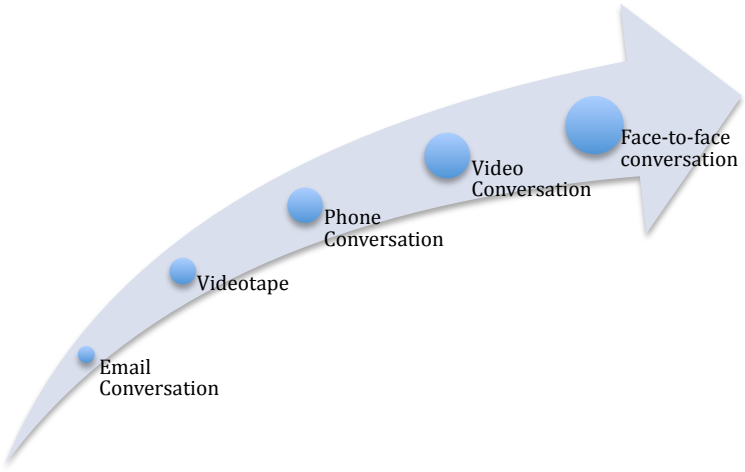
4.1.3 Seeding the Relationship: Modes of Communication

Bringing on an offshore team introduces new temporal and cultural complications. Perhaps the largest barrier to success with an offshore team is the temporal difficulty of working during opposite business hours. India is twelve and a half hours ahead of Omega's main office in Phoenix and so both teams will need to make some concessions to enable open communication channels.

Team members must strive to use the richest mode of communication possible in order to build team cohesion. Research shows video communication as the most effective mode of communication with distributed teams (Ehsan, 2008). Although email discussions are convenient, written communication lacks the subtle physical cues of body language, and richness of verbal connection that face-to-face conversation provides.

Face-to-face meetings are invaluable at the start of the relationship in order to form a rapport. Establishing a new virtual team with face-to-face interaction "create(s) a bond and develop(s) agreements on how you are going to work together" (Fisher, 2001, p. 35). A distributed team lacks the interactions that happen in the margins of the workday, such as lunchroom chats and workplace celebrations that cement bonds within the team.

Figure 3: Modes of Communication in Increasing Effectiveness



Source: Adapted from Cockburn A. , 2009

Sending lead managers to the offshore site adds to the hidden costs of outsourcing that serves to undercut the savings in labour costs. However, this investment creates crucial "opportunities to learn about one another and 'humanize' co-workers at the distributed sites" (David, 2008, p. 138). It can actually save money in the end, since "without decent personal relationships it is likely that you will run into problems due to miscommunication and lack of trust" (Fowler M. , 2006). These breakdowns in communication can derail projects and require a lot of time to re-establish. Omega should dedicate a budget for these opportunities for face-to-face contact.

4.2 Navigating the Offshore Cultural Context

Omega is considering India as a possible country for the offshore team. Omega's strong and trustworthy contacts in the industry already have software teams working in this country. By setting up a collocation structure, Omega can leverage this situation to help setup the employee relationship using the pre-existing office building and human resources structure.

This next section will provide a brief analysis of India and look at some possible cultural confusions and ways to avoid them, since "executives should not assume that cultural alignment would be insignificant or trivial" (Davison, 2003). Proactively anticipating cultural differences by decreasing communication barriers will be crucial for success between home and remote teams.

4.2.1 India: A Technology Services Powerhouse

India is the de facto leader in technology services outsourcing. With a powerful technology cluster, a positive political environment, and a large number of post-secondary institutions feeding a skilled workforce, India has leveraged a large, educated English speaking population to "become a major exporter of information technology services and software workers" with 62% of GDP derived from services (CIA, 2010). India offers a large pool of experienced programmers due to their mature technology cluster. These veteran programmers will be able to produce quality code without Omega having to provide basic training.

4.2.2 Cultural Intelligence: An Analysis of Cultural Differences

An understanding of the cultural and political context of the offshore team plays a key role in building a strong rapport and a positive work environment. This is referred to as 'cultural intelligence', which means bringing awareness to interactions with other cultures, establishing a set of skills that help to adapt to uncomfortable situations, and "gradually reshaping your thinking to be more sympathetic to other cultures" (Thomas, 2009, p. 13). North American culture is much more direct than South Asian cultures and so being able to adjust behaviour will improve this cross-cultural interaction. For instance, one major cause for communication gaps between Indian and US teams is the Indian 'yes', which is "a well-documented mind-set that leads Indians to readily agree to a requirement

asked for by a U.S. client, whether there is a realistic chance of achieving it" (Gibson, 2006). The intention of this 'yes' is to please, rather than to deceive. However, when projects become late because of unrealistic expectations or a lack of understanding, this can become a major obstacle to success.

Receiving feedback from the remote team will require directly asking for it because "disagreement with a respected person is unthinkable" (De Baar, 2007). Managers must emphasize that disagreements, critical feedback and second opinions are valued and crucial to the company's growth. Do not simply accept a 'yes' from the remote team, because "polite acceptance is often a sign of an important issue not getting discussed" (Fowler M. , 2006). Instead, specifically ask if they are happy with the plan or their work and inquire if they might have any ideas to improve on or creative solutions with which to approach the problem.

Another difference is the comfort with silence and empty spaces in a conversation. North Americans treat silence as a sign of discomfort, and try to fill these gaps, whereas "people from Asian societies, where long silences in conversations are considered normal and acceptable as participants reflect on the topic, do not realize how odd and intimidating silence in this situation seems to many Westerners" (Thomas, 2009, p. 12). Keeping this in mind during meetings will help ensure that the opinions of the remote team are not drowned out.

Managers should not rely on these cultural observations as rules or strategies, as they are generalizations. Such generalizations can emphasize rather than bridge difference, as they may not reflect the individual at hand. Often, such cultural generalizations can "establish a hierarchical relationship" and legitimatize "sophisticated stereotyping" (David, 2008, p. 138). Using social awareness is the best tool for developing methods that create

positive cultural interactions. Understanding that cultural differences will have an influence ensures communication does not break down and helps to maintain a positive and respectful atmosphere as long as it does not overshadow the individual on the other end.

4.3 Adding New Roles and Specialization

The next stage in setting up the offshore team is deciding the makeup of employee skills and roles. Research shows an increase in development and velocity with cross-functional teams that minimize specialization (Ford & Randolph, 1992). Each developer should handle all areas of application development, including working with the database tier, application tier and front-end. The team should try to minimize excessive specialization and promote cross-functional abilities.

Specialized roles create more dependency in the flow of work. The application developer must wait for the completion and sign-off of the database structure from the database administrator before starting work, and the tester must wait on all of the other team members to finish before starting testing. This lengthens the development 'supply chain' and results in bottlenecks with some roles having a larger amount of work for a particular feature implementation. "Individual code ownership and knowledge silos" can cause significant delays in projects when a person leaves a team or is busy with other work (Laribee, 2009). The analysis will go into more depth on these issues in the chapter on lean development. Rather than have separate graphic designers, database administrators, application programmers and quality assurance engineers, the company should look for more experienced employees that have abilities in all of these areas.

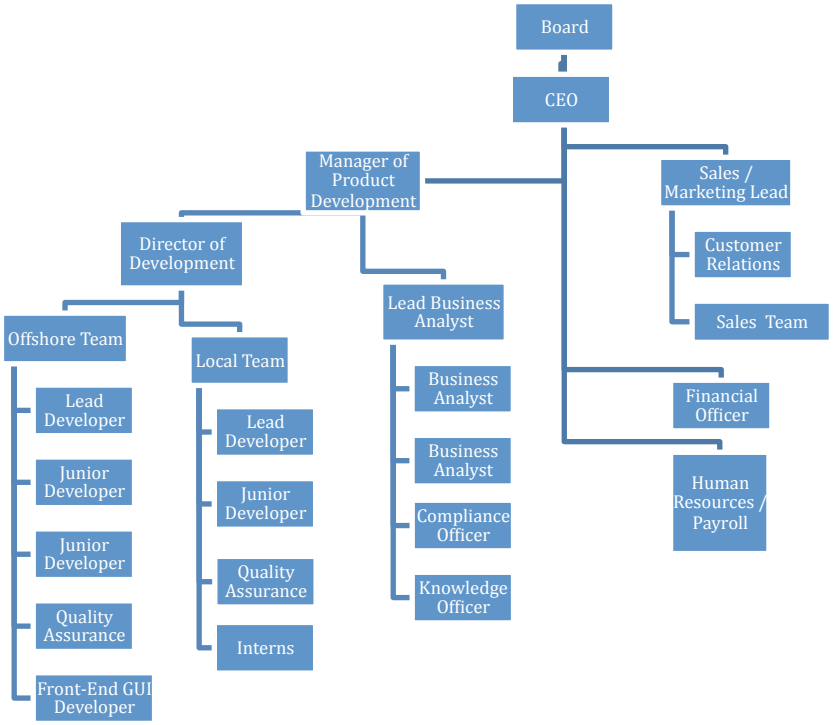
As the company grows, it might consider filling specialized positions in these different areas. Although each developer should be able to work on all development areas, some amount of specialization in particular technology would be valuable. For example, one

person in each team might focus on graphical user interfaces and user experience optimization, whereas another might be the specialist in writing automated test frameworks. However, the team should avoid over-specialization in order to minimize bottlenecks and single points of failure with having only one employee able to perform a certain task.

4.3.1 Organizational Structure and Coordination

The figure below displays the proposed organizational structure of the company with the added offshore team.

Figure 4: Formal Corporate Organization Structure



Source: Author

In practice, teams are broken down in a more flat structure based on projects. In the view of operations, one to two developers coordinate with a business analyst who acts as a proxy for the customer and a quality assurance specialist.

Figure 5: Operational Project-Based Organization Structure

Source: Author

By using project-based teams such as shown in the operational organization structure in the above figure, developers can gain a deep understanding of requirements and have direct responsibility for project success or failure. Unfortunately, it is rare that a developer is able to focus on a single project. Omega has a broad project portfolio, with many projects running simultaneously.

The product manager handles looking at the project portfolio from a top-level view, balancing strategic input from the executive team with overlapping deadlines and resourcing needs. Managing the project portfolio is one of the most complicated parts of coordinating the Omega team. Analysts and developers work on multiple teams at once, and handling priorities between all of these projects is a constant balancing act.

A Director of Development that has extensive experience in software architecture would be a valuable additional role. As the team grows, the product management and technical roles will need to diverge, with the product manager focusing more on strategy and positioning in the market, and the director of development more internally focused on optimizing and stabilizing the code.

It would be valuable for the offshore team, to have a project manager responsible for motivating and keeping the employees on track. As Omega only plans to hire four developers at the beginning, having a dedicated project manager may not be necessary. Instead, a lead developer with project management experience could handle managing timelines alongside programming. When Omega adds a second group of developers in the future, a project management role would be valuable to promote coordination and synchronous teams.

4.4 Knowledge Management and Training

4.4.1 Replication Strategies and Knowledge Management

Omega sells generic out-of-the-box software products as well as customized business process solutions. Although it is working towards productizing its software rather than doing custom development for each customer, there is still a market for offering custom workflow solutions. These tailored services require skilled consultants with knowledge of business process design as well as the technical skills to implement functioning solutions. Omega customizes each solution for the information needs and technical requirements of a particular health organization or department. It relies much more heavily on the use of rivalrous human capital that increases the marginal cost of each software solution. Finding ways to scale with the need for skilled human capital revolves around being able to capture and transmit the company's learning to new employees.

One body of research that might be useful to consider in Omega's approach to knowledge management is the concept of 'replication'. Replication, referred to as the 'McDonalds approach', "entails the creation and operation of a large number of similar outlets that deliver a product or perform a service" (Winter, 2001, p. 730). In order to be successful with this type of strategy, the company must be able to package up and

crystallize the complete set of processes and documentation that the company uses to deliver custom solutions to customers. These processes and document templates map out a specific customer's workflow. This process entails the company's core capability and competitive advantage. This is an important consideration for enabling the eventual sale of the company, with an acquiring company expecting proper formulized documentation in order to retain Omega's intrinsic value.

Although Omega is far from being at the point where it could replicate itself as a chain organization, a look at some of the procedures used for expansion by other industries might serve as inspiration for its own knowledge management and process development. Such chain organizations have mastered the ability to formulize and pass on business knowledge to new groups of employees. Scaling a company using this knowledge requires "the capability to recreate complex, imperfectly understood, and partly tacit productive processes in carefully selected sites, with different human resources every time, facing in many cases resistance from proud, locally autonomous agents" (Winter, 2001, p. 731). Omega's maturity in scoping and workflow definition documentation gives it a competitive advantage. Some of the specific tools that Omega uses to help define a company's workflow include:

- Control Document that lists specifics of all data elements that the customer needs to collect
- Structured workflow diagrams showing the flow of patients or activities throughout a hospital's clinical system
- Systematized approaches to interacting with the customer to drive the project forward in a timely manner

- Software platform that allows non-developers to easily customize the workflow and business logic based on a particular customers needs

As discussed previously, this use of systematized customization services helps to counteract the force of commoditization in the software industry: "managers need to think about how to 'productize' their services so they can deliver them more efficiently" (Cusumano, 2008, p. 26). This attention to and constant revision of the project management process is one of Omega's core capabilities. Management should focus on this since "productivity tends to increase because of reduced rework efforts and more attention to process controls" (Krafcik, 1988, p. 47). By constantly refining and crystallizing the approach to Omega's software platform, the company gets closer to delivering software at a productized cost with the profit margins available in the service-based industry.

4.4.2 Training New Employees

New employees would have a difficult time understanding these tools without training and proper documentation. With an eye on this replication strategy, new employee training should be refined. When a new employee has difficulty understanding how to accomplish a task, this signals that the trainer should expand or clarify the documentation materials. These diagrams and structured documents only derive their benefit if new employees understand how to use them and can actually leverage them.

Adding a new employee requires a number of steps to configure their environment. For Omega to maintain rapid growth by adding employees, a checklist should be prepared with all of the necessary procedures for initializing an employee.

Using an offshore team to assist the local analysts will help with this ability to scale. Local analysts can be co-located with the customer, able to walk the halls of the hospital and clearly understand customer need. With a set of well-structured documents describing

these customer needs, the analyst can then hand off a defined package to the offshore team that will create the functioning application.

Human resource knowledge entails a large part of the company's core capabilities. The medical software industry is complex, and preparing a customized solution for an Omega client requires an understanding of their software platform. Replicating and transmitting this knowledge becomes a crucial priority for adequately training new employees. Rather than having this tacit knowledge in the head of a single experienced employee, the team should strive to codify and solidify it with actively maintained documentation.

4.4.3 Encouraging a Culture of Documentation

Economies of learning play a major role in the efficiency of the firm. Naturally, more experienced programmers in the company can produce applications at least twice as fast as newer employees. With their understanding of the platform, they are able to reuse code libraries and avoid future support costs by building well-designed applications. Translating these abilities to new employees becomes a critical initiative, as employee turnover is financially and managerially expensive. A culture of documentation and the establishment of a library of detailed training resources will help capture the derived value from this learning.

Experienced employees should highlight and correct any inadequacies in the training materials when new team members ask questions or express confusion. This will keep training materials living documents that reflect the transparency of the company and the organic nature of the industry. Wiki software, an example of a living document, can help organize the communication that normally gets lost in email correspondence, since "wikis are by nature unstructured, and this lack of structure is part of the benefit" (Fowler M. ,

2006). Having user-friendly and simple software such as a wiki will minimize the friction required to update and edit project and company knowledge. Wikis can track the number of specific changes that each employee has made each day, and management can consider tying this to bonuses or other recognition. Management can endorse this method of knowledge transfer by cultivating a corporate culture that celebrates documentation.

4.4.4 Document Management and Compliance

There is a final important value in having a well-structured knowledge management strategy: FDA compliance. The company needs to have standard operation procedures that "provide for system validation, development and deploying systems according to a formal methodology, physical and logical security, backup and recovery, system operations, staff training, change control, contingency planning, and use of purchased systems" (Grunbaum, 2002, p. 2). The procedures listed in the quote above could be actual folders within the compliance section of the site.

Twice per year, the company should schedule a compliance review to identify any gaps and do an internal audit. Maintaining compliance is an ongoing task, and has value outside of the legal requirements. It is essential for identifying and then minimizing risk as well as ensuring that customers' private health information is secure. This is particularly important for Omega during this time of rapid change and growth: the addition of new management systems, an offshore team, and the transition to cloud based computing.

Omega needs to refine its processes for managing projects and producing software in order to ensure compliance with regulations in the medical industry. The following chapter will present some project management methodologies to make the development team more efficient as well as introduce tools that can help bring transparency to the organization through minimizing waste.

5: INTRODUCING AGILE PROJECT MANAGEMENT AND LEAN METHODOLOGIES

A group of software engineers introduced the 'Agile Manifesto' to the software industry in 2001 after becoming frustrated with the amount of wasted resources resulting from trying to do traditional upfront design in a complex and changing environment (agilemanifesto.org, 2001). This manifesto adopts the lean manufacturing concepts invented for the Toyota Production System and translates them to software development (Poppendieck & Poppendieck, 2003, p. 3). Agile acts as a counterpoint to the bureaucratic 'waterfall' engineering methodology where in-depth planning and specifications are locked down before any development can begin. Software development is a much more creative process than traditional engineering where "creative processes are not easily planned, and so predictability may well be an impossible target" (Fowler M. , 2005).

The Agile Manifesto argues for

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Omega will introduce a particular implementation of agile called Scrum. Scrum teams work on producing deliverable value in short blocks of time, typically two-week iterations or sprints. The goal is to decrease the amount of delay between strategic decisions and development, since "Scrum helps a company to compete by focusing the team on the highest-value client requirements" (Woodward & Surdek, 2010, p. 10). The defining

feature of Scrum from other methodologies is the iterative nature: with the team planning together in a constant feedback circle, "modifying its approach daily as it encounters new complexities, difficulties and surprises" (Schwaber, 2004, p. 6).

This chapter will begin with an analysis of using Scrum and the challenges in using this methodology with a distributed team in a regulated environment. An analysis of available project management software will finalize with a recommendation of a specific tool needed to manage distributed teams. It will then look at some of the causes of waste in software development by drawing analogies to 'lean' manufacturing methods popularized by Toyota.

5.1.1 Agile Project Management in a Regulated Environment

The FDA "wants clear-cut documentation that shows that you have a process, that you follow it, and that the process will produce a quality product" (Jacquette, 2001). They state that specifications must be predetermined and clearly documented. This adheres most closely to traditional waterfall software development, and appears to contradict the use of agile (Nadler, 2007).

One common misconception of agile development is that it does away with documentation completely. The business analyst will need to prepare a lot of the same documentation that traditional waterfall project management condones, "including a work plan, requirements specifications, design documentation, a test plan, and system test scripts" (Jacquette, 2001). The difference is that team members create only the currently needed documentation during the iteration rather than all at once at the start of a project.

5.1.2 Agile Project Management with Distributed Teams

The fact that Omega plans to set up an employee relationship with the offshore team will help with adopting agile. One advantage of using a dedicated team rather than fixed price contracts is that "you can work in agile mode and be much more flexible in regards to methodologies" (Kostukova, 2009). Team members do not need to fully detail and sign off on requirements specifications any work begins. Instead, the team can stay flexible and evolve the requirements as work progresses.

Omega has already set up a number of tools that make distributed teams possible and focus on accountability, including:

1. Employees work on remote desktops rather than locally installed software
2. Secure instant messaging server showing when team members are online
3. SharePoint portal for electronic sharing of project documents and technical documentation
4. Dedicated teleconference line

The team will need to supplement these tools with additional technology and processes to coordinate with the remote developers.

5.1.3 Daily Stand-Up Meetings to Remove Obstacles

In order to ensure proper governance and keep projects moving in the right direction, management will need to communicate regularly with the offshore team. The project management team will begin using daily scrum meetings. These meetings should be short and concise, with a fixed time limit of 15 minutes. This practice follows a primary recommendation of the Scrum methodology: "brief, to-the-point communication that looks for ways to remove obstacles and move the team forward through project execution"

(Woodward & Surdek, 2010, p. 13). These stand-ups should prompt each employee to answer the following questions:

1. What have I accomplished since last meeting?
2. What am I working on next?
3. Is there anything blocking me from accomplishing this work?

This meeting provides two major values: it keeps the lines of communication open to promote team support and cohesion, and it pushes forward momentum by focusing on accountability. It prompts each employee to say what they have accomplished in the previous day and re-focuses what they are working on next. This acts as a way to hold employees to their word as any missed goals become immediately obvious.

With the offshore team only being available in opposite hours of the day, having the stand-up meeting with all employees across the organization would not be viable on a daily basis. Many organizations using Scrum in a distributed team find that distributing the answers to the three stand-up questions in written form at the start of each day saves time during the videoconference meeting, since it helped "improve communication and reduce misunderstandings due to cultural and distance barriers" (Sutherland & Viktorov, 2006). Technical leads from each of the distributed teams will hold a "Scrum of Scrums" (Woodward & Surdek, 2010, p. 13) coordinated in off hours that will highlight any issues that might be blocking the offshore team. This check-in meeting will help bridge the gap between the local and remote teams to ensure coordination of goals.

5.2 Adopting Agile Project Management Software

The Agile Manifesto encourages "individuals and interactions over processes and tools" (agilemanifesto.org, 2001). Many agile teachers argue against introducing new

software at the start of an agile adoption (Langr, 2010). When introducing the process, it is best to use simple physical whiteboards, pieces of paper, and direct interaction between team members to drive the project management process, rather than being distracted with complicated software. However, due to the distributed nature of Omega's company profile the team needs online software to coordinate personnel, since "high velocity complex projects need an automated tool to track status across teams and geographies" (Sutherland & Viktorov, 2006, p. 3). The next section will discuss the value of tracking software, followed by a brief comparison of different tools and a final recommendation of a specific software solution.

5.2.1 The Value of Tracking Velocity and Cost

Until this point, Omega has not collected clear metrics showing how much time each employee spends on a specific project. The internal cost breakdown for each software sale is difficult to quantify, since most employees are working on multiple applications at once. Not enough data is available to measure which projects have had a return on investment. The company cannot focus on one project at a time since each project goes through natural latent periods while waiting for feedback from clients or for input from members of the team. Linking development costs to the revenue for specific product lines has not been possible until this point. A following section on managing the project portfolio will further address this issue of multitasking projects.

Previously, employees provided a status overview in the form of a weekly status report delivered to managers every Friday. This status report, called a '5:15', takes five minutes to read and fifteen minutes to write. It lists the projects that an employee worked on, an estimate of the percentage of the week spent on each project, and details on how they

accomplished tasks. Employees send these weekly status updates by email and thus are not stored in a database so that the executive team can track and trend velocity over time.

These status updates comprise the primary way for employees to convey important project information to the executive, and have worked well with the small team size where informal face-to-face communication can fill any information gaps. These 5:15s are simple to write and useful because they deliver a concise summary of project statuses. A full project management tool would more effectively track progress. Having a more comprehensive time tracking system will encourage greater communication and collaboration between developers, dissuading them from disappearing into their own workspace for the entire week until the final check-in.

A project management tracking system that forces employees to quantify time spent on specific tasks will help establish a set of metrics that can tie the costs of goods sold with the actual revenue of a specific project. Employees have expressed resistance in the past with explicitly track how much time they spend on various tasks. In order to gain acceptance for the use of granular time tracking, management should convey that they do not want this tracking to overly control employees or remove spontaneity and creativity from their jobs. Management can emphasize that the company needs to track the amount of time spent on tasks in order to effectively price future projects and see that revenue exceeds costs.

5.2.2 Selecting the Project Management Software

Omega wants to minimize the amount of locally installed and supported software within their environment. One part of this goal is finding a project management system that employees will not need to install on desktops. A large number of available project management solutions focus on agile development and are hosted services that do not

require local install. Before this analysis, Omega used a customized SharePoint site to manage tasks and application defects. Although effective in keeping a history of changes for FDA compliance, this site had limited utility in providing a view of project velocity and status. To fill this requirement, Omega needs a more effective project management tool.

Although a large number of products are available, focusing on agile development tools narrowed the selection down to four main choices: *Jira*, *VersionOne, Pivotal Tracker* and *TargetProcess*. All have similar positive reviews online and matching feature sets. As part of this project, the writer reviewed and extensively piloted each of these products and examined others available.

TargetProcess received positive ratings, but lacked responsiveness, which would have made the tool unpleasant to use. *Pivotal Tracker* was an effective tool and had no licensing feed. However, it lacked the depth of features to handle multiple projects and integrate with the company's ecosystem.

Management selected *VersionOne* for an in-depth pilot with other team members based on a recommendation from a colleague in a major accounting company. It could handle multiple simultaneous projects and had a plugin ecosystem to work alongside Omega's development software. In order to introduce the change gradually, the development team started using it alongside the existing SharePoint PM tool to test it in real-world use. Once key members became familiar with the tool, management expanded use to the whole team on a single project. In this limited pilot project, both developers and analysts used it to track tasks and assign bugs and issues. Deficiencies in using the tool for support and bug tracking became evident. Even though Omega already purchased two user licenses, it deferred purchasing licenses for the entire company. Management wrote these licenses off as a sunk cost, and the company migrated to another industry leader in project

management called *Jira*. This software package interoperates with other valuable tools such as a wiki for documentation, code review software for reviewing developer work, and hosting of source control.

5.2.3 Using Project Management Software with FDA Compliance

In order to use *Jira* in compliance with the FDA's recommendations in *21 CFR Part 11*, the company's compliance officer must verify the software for proper functionality. The compliance officer will prepare a checklist of functionality for the software, and store this checklist in Omega's secure compliance folder. The company will also need to modify the software to ensure that employees cannot delete or modify previous audit history. This is to certify the "authenticity, integrity, and, when appropriate, the confidentiality of electronic records, and to ensure that the signer cannot readily repudiate the signed record as not genuine" (FDA, 1997).

The compliance officer will need to perform some modifications to the initial installation of *Jira*, including adding a verification step to the workflow of issue tracking and resolution. Rather than simply marking an issue as resolved, developers will mark the issue as 'Pending Fixed'. Quality Assurance will then do a verification check. This check becomes part of the audit log of changes over time that the company requires for FDA compliance.

Omega uses an electronic document management system (SharePoint) for maintaining all of the necessary compliance documentation during the development of its medical software. This system has already been configured for the "protection of records to enable their accurate and ready retrieval throughout the records retention period" in accordance with *21 CFR Part 11* (FDA, 2002).

5.2.4 Peer Reviews and Quality Assurance

To help minimize the major risk of adding multiple new employees at the same time on the offshore team, management should set up a code review workflow. Developers maintain source code in source control software that maintains a history of changes over time. The offshore team will be working on a separate development branch in source control. After each check-in, the experienced local developers will review the code and provide feedback. This review will likely highlight areas for optimization or point out security flaws that the company should address. Code reviews are valuable for decreasing and avoiding technical debt, by catching any issues before they can enter production systems.

Code reviews are also valuable for FDA compliance. A developer triggers a workflow notification whenever checking in new code to the source control system. Another developer would then review this checked in code to try to find any possible issues or security weaknesses. When a reviewer signs off on these changes, it creates the audit trail that FDA compliance requires. In order to comply with FDA requirements, "testing should occur throughout the entire lifecycle of the project" (Jacquette, 2001) and should include documentation that this testing has occurred.

Unfortunately, peer reviews cost money in terms of opportunity costs. Research shows that this short term cost results in better software quality, lower risk, and more reliable release schedules in the long term (Fagan, 1976) and that "the earlier a defect is found in a product, the cheaper it is to fix it" (Cockburn & Williams, 2000, p. 5).

These reviews will help the local experienced developers train the remote team and maintain code quality. They will likely need to request changes to match the code standards of the company. Once the new offshore team gains experience with the standards, they can

review each other's work in order to decrease the burden on the technical leads and take advantage of local time zones for fast feedback. Having developers check each other's work will also help build cross-functional and self-disciplined teams.

5.3 Lean Methodologies in Software Development

The software industry has adopted the concept of 'lean' from the manufacturing industry, where Toyota introduced lean manufacturing with its Toyota Production System. This system found ways to eliminate waste through refining processes, minimizing work in process, and decreasing the storage of inventory (Krafcik, 1988). The following sections will draw analogies to these forms of waste within the software development process.

5.3.1 The Product Backlog: Aligning Business Objectives with Development

The heart of the development process in an agile methodology is the product backlog. This prioritized list of all possible additions, improvements and defects, creates a place to accumulate all ideas for new product development. This list promotes innovation by building a product roadmap and prioritizing R&D strategy. The product backlog in *Jira* is sortable, with the higher priority items consistently shuffled to the top of the list so that the team always knows what to work on next. Priorities can be set for a single project or across the entire company portfolio. In order for the development team to progress in a direction following corporate strategy, the executive team should sit down with the lead development manager at least once a month to review the product backlog to prioritize and order.

One of Toyota's primary ways of reducing waste in manufacturing is to minimize the amount of inventory produced and stored. This idea is relevant in the software industry as well, since "ongoing development projects are just like inventory sitting around a factory" (Poppendieck & Poppendieck, 2003, p. 2). In the software industry, the development team

can maximize customer value by only building things a customer actually needs. Building unnecessary features is equivalent to overproduction in manufacturing and results in waste. The product backlog stored in Omega's project management software helps to synchronize corporate strategy with actual development timelines.

5.3.2 Optimizing the Supply Chain: Installation and Migration Scripts

The development team can increase operational efficiency by using as much automation as possible. Developers need to be able to setup and install new applications from Omega's platform with minimal work. One of the new offshore developers should focus on refining the installation and migration scripts that create new instances of Omega's software. The company can derive enormous gains in productivity by being able to generate new versions of an application and move between development and production environments.

5.3.3 Continuous Improvement and Technical Debt

Agile development advocates constant improvement. Programmers refer to continuous improvement of the code as refactoring, where they restructure the software to increase maintainability. It is "a disciplined way to clean up code that minimizes the chance of introducing bugs" (Fowler & Beck, 1999, p. xvi). This work usually does not result in any exterior changes to the software, other than performance, yet will result in lower support costs in the future.

Refactoring can pay down technical debt. Management should not neglect development tasks that help reduce technical debt and resolve inefficiencies in the underlying software. During development, sometimes developers need to take shortcuts in development to meet tight deadlines. Whenever consciously cutting corners, developers should create new backlog tasks in the project management tool to record ways of

correcting these shortcuts in the future. “Tasks needed to pay off that debt are entered into the system along with an estimated effort and schedule” (McConnell, 2007) where these backlog tasks become visible and the team can prioritize them against new feature development. Recording these improvement tasks during initial work also helps to document why developers made certain decisions in software design.

5.3.4 Retrospectives and Improvement

In order to avoid repeating mistakes the team will participate in retrospective meetings at the completion of each sprint: “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly” (agilemanifesto.org, 2001). At the end of each sprint, the entire project team sits down to discuss how the sprint progressed, what caused problems, positive outcomes, and improvements for next time. The manager should encourage each team member to vocalize their opinions and by going around the room and asking each person to say something. The distributed team should use video conferencing during these retrospectives as a structured time to build cohesion.

There is often a tendency to bypass these retrospectives with constant time pressures and deadlines, but these discussions are crucial. Looking for ways to reframe the retrospectives by varying the format of the retrospective can be helpful to ensure it is productive (Derby, 2010).

5.3.5 Balancing Support Tasks with New Development

One obstacle in adopting fixed two-week sprint schedules is balancing the pressures of unplanned support and maintenance requests with new feature development. The company must address support requests in a timely manner to ensure excellent customer service. Omega maintains a policy of ensuring as few bugs on production systems as

possible. Production issues take immediate priority over new feature development. As well, customers might find issues that could compromise usability and stability of data. The initial theory for handling support was to have the local developer team take care of support tasks. However, this will likely cause a bottleneck, and miss "the sense of responsibility the team has for a system that it not only develops but also maintains—the team is acutely aware of the impact of getting things wrong" (Watts, 2008). Without the offshore team experiencing the negative consequences of low quality code that result in support requests, they are not able to learn from mistakes and find ways of increasing code quality and decreasing technical debt (Bartholemy, 2001).

Since the support team will need to deal with issues immediately, they cannot commit to two-week iterations ahead of time. The development manager can swap members of the support team every two weeks to ensure each person has the opportunity to work on new development. Moving new developers in and out of support roles cross-trains them so that they gain a deeper understanding of the entire software platform.

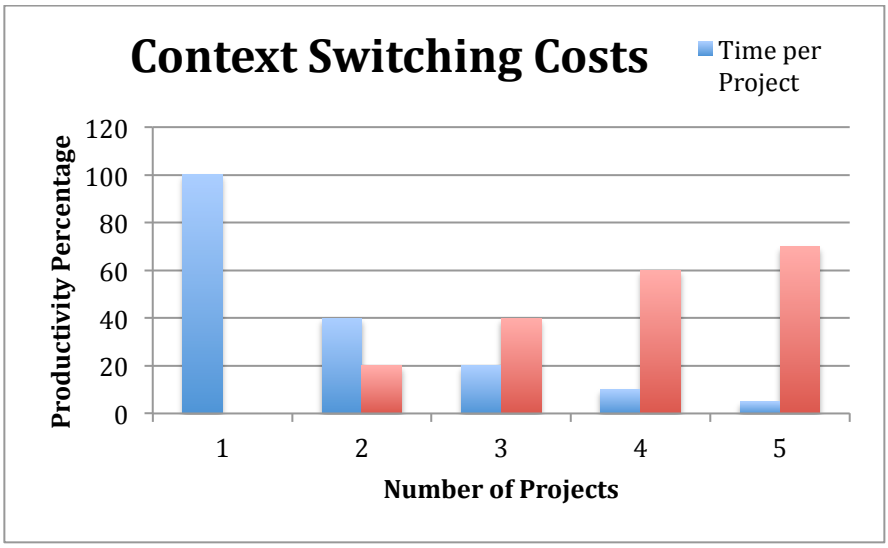
5.3.6 Multitasking and the Project Portfolio

Changing contexts between different projects causes a loss in productivity. Multitasking "is the act of stopping a task before it is completed and shifting to something else; in software development the term "thrashing" is often used to describe this practice" (Fox, 2007). This becomes an issue because employees spend more time shifting contexts than they do actually completing work (Piper, 2005).

In the past, Omega carried at least 6-8 active projects simultaneously with six business analysts sharing the resources of two developers. Analysts compete for developer attention with bug fixes and development tasks being the critical path for project completion. Projects often end up sitting at 90% complete as management pulls developers

to work on other tasks. Toyota saw this as a source of waste in their product development methodology (Poppendieck & Poppendieck, 2003, p. 2). The figure below illustrates the cost of switching between multiple projects.

Figure 6: Cost of Context Switching on Multiple Projects



Source: adapted from Roger Brown, 2010.

One major cost of multitasking projects is handling the synchronization of bug fixes from one project to another. When a developer fixes a bug in one project, she will need to merge this fix over to the other projects that have the same code version. The time spent handling this code merging increases for every active project running. With one active project at a time, these code updates only need to happen at the start of the new project, rather than constantly synchronizing versions between the projects. Omega developers report that handling this constant merging process consumes approximately 15-20% of their development time.

Implementing a strict limit on the number of active projects will require a significant change in management policies. Such a change will dramatically affect company

sales cycles and approaches to interacting with clients. Management should avoid special projects that defray from the long-term strategic roadmap: "keep your projects small, focused and attainable" (Piper, 2005). This way the development team can quickly release versions to a customer and shift quickly to a new project. Sales executives should only sell features that are already available in the company's product offering rather than offering to do custom requests for a single customer.

Doing two week fixed iterations forces the executive team to prioritize and focus on the most important tasks. It introduces rigour by protecting developers from distracting tasks such as creating demo sites for sales people or handling superfluous requests from panicked customers. When teams work on single projects at a time, the duration of projects become easier to forecast. As well, it allows the company to focus on strategic 'technology push' development instead of always following the whim of customers in a market pull approach, since "both market need and technical information are required for an inventive idea to be synthesized" (Voss, 1984, p. 147). In order to deliver both what the market requires (market pull) as well as invent new markets (technology push) the company needs to balance both.

Until this point, Omega primarily responded to market pull, which delivers on what a customer wants to pay for. However, the company also needs to turn down customers that would distract from pursuing the most profitable business. Omega can minimize this thrashing using fixed iterations, minimizing the work in process and establishing a solid product roadmap. If new projects require extensive developer resources, management should evaluate them in terms of the opportunity cost that draws away from more strategically appropriate and financially solid product development.

5.3.7 Project Management Software as a Kanban Tool

One system used in manufacturing to limit multitasking is the use of 'Kanban', which is a set of visual indicators to drive workflow. *Jira* has the equivalent of a Kanban board built into its project management software. The Kanban board visually represents different project tasks in a sprint as cards arranged in columns. The left column lists tasks and user stories waiting for implementation. As a developer begins working on something new, she drags the task from the left column into a middle column to mark it as active. This starts the timer tracking how much time a task takes to complete. When the developer finishes a task, she drags the card to the 'Completed' column on the right side.

This Kanban board gives a quick visual indication to the distributed team what tasks are active. "Any project or part of a project you are working on but is not complete would be inventory" (Miller, 2006), and decreasing the amount of outstanding inventory helps prevent waste. This board limits the number of task cards that can be in the middle active column. This dissuades developers from beginning a task and leaving it incomplete, and is an effective tool for minimizing work-in-process, a central tenet to lean methodologies.

5.3.8 Waste and Underutilized Hardware

Another form of waste within the company is the number of underutilized servers that the company must maintain, upgrade and service. These servers, especially testing servers, typically sit idle for a large part of the day and night. In manufacturing, maintaining inventory is the largest source of waste, and "in software development the analogy to inventory is excessive hardware" (Wenger, 2008). A solution to this underutilization is the transition to cloud based computing, where the company outsources server hosting to powerful third party grid computers that scale depending on the amount of processing power needed. With cloud computing, the company only pays for the traffic and processing

power that it needs. The next chapter will look at an implementation plan for outsourcing this infrastructure in order to decrease waste.

6: CLOUD COMPUTING: OUTSOURCING INFRASTRUCTURE

Omega is looking for ways to minimize the amount of resources it needs to dedicate to maintaining this infrastructure. The goal is determining what is core to the company, while passing on any peripheral tasks to an external service provider (Kakabadse, 2002). Even though server hosting and infrastructure is not the company's core focus, it takes up a significant amount of focus and labour resources. Cloud computing uses outsourced services to extend a company's computing infrastructure as "a way to increase capacity or add capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software" (Knorr, 2009).

Support will become a major issue for Omega in the future, particularly as it adds products and services that require 24-hour support. Having an offshore team residing in an opposite time zone will allow the company to provide this overnight support and fast turnaround on issues. Emergency support concerns typically arise because of failures in hardware that hosts applications. The offshore team would not be able to address hardware issues in a traditional data centre since this usually requires physically walking over to reboot a server or replace a hard drive. With a fully virtualized cloud server infrastructure, network administrators can perform this support from anywhere.

6.1 Omega's Server Hosting and IT Infrastructure

Omega currently maintains a large IT hardware infrastructure. This infrastructure of networked servers is required both for the company's internal use for handling day-to-day business needs, and to provide site and application hosting to its customers. This infrastructure includes the maintenance and running of multiple server farms for email

(Microsoft Exchange), databases (SQL Server), web servers (IIS) and hosted terminal services (Windows Remote Desktops).

Omega uses Microsoft technology for 90% of its software and hardware. This conveys the benefit of a stable, mature environment compared to open source software alternatives. The downside of Microsoft server technology is the steep cost of licensing. Hosting the entire required information technology infrastructure internally results in high costs for the company, and the executive has requested strategies on how to outsource at least some of this infrastructure to third parties to simplify support and maintenance and reduce cost.

The following sections will analyse different approaches to outsourcing Omega's IT infrastructure so that the company can decrease costs, increase reliability and security of their environment, and ensure the ability to expand as needed. The analysis will look at two separate areas:

1. Internal IT Infrastructure: The software and tools required for the functioning of the business, including email, document management, and software development tools
2. Hosting of software and applications for customers: since Omega offers a large part of its products using a Software-As-A-Service (SaaS) model, it dedicates a large amount of its resources to maintaining online hosting of websites

This analysis will break outsourcing down into two phases in order to ensure a careful and measured transition. It will begin with strategies for the internal IT infrastructure as a first stage, and then look at how to outsource application hosting for customers.

6.1.1 Server Infrastructure and Cost

The total cost of ownership (TCO) of hosting an IT infrastructure comprises more than licensing costs. Even expensive hardware may not be the largest component, with some arguing that hardware comprises "less than 10% of the cost of a computing service" (Gray, 2003, p. 3). The total cost of ownership (TCO) includes the support and maintenance costs as well as the opportunity cost lost from technically skilled employees having to focus on server support rather than other tasks such as new application development.

As an illustration of some of these hidden costs, a recent failure of a cluster component on a hard drive cost Omega a week of unexpected down time for a large part of the company. Since this was the server for the development and staging of applications, as well as for the company's document management system, it thankfully did not affect production customers. If it had, this would have been a catastrophic failure costing the company a large percentage of its business. For instance, due to a cascading set of failures, there was the possibility that this crash could have resulted in the loss of documents yielding an inestimable cost to the company. This single event tied up the resources of four of the primary technical team for the entire week, as well as redirecting a significant focus of the executives in restoring functionality of this system. At a loaded hourly rate of $40 per hour per employee for 30 hours of work, this resulted in a cost of about $4800 in direct labor. During this downtime, the efficiency of the entire company decreased by at least 70%, with no access to project documentation or development environments. Project timelines were pushed back a week with a large number of opportunity costs associated with them. The potential risk that arose from this event had it been a primary component on a production system could have increased the magnitude of this failure.

6.1.2 Cloud and Exit Strategies

An added benefit of having the entire corporate infrastructure located in various cloud systems is that it makes the company easier to sell when the opportunity arises. Rather than having to move a data centre and ship actual servers to the location of the acquiring company, Omega would simply pass on the login information to the cloud hosting accounts with little cost of migration. As well, company valuation becomes easier with the reliable price structure of cloud computing services.

6.2 Migrating Internal IT Infrastructure

The internal IT infrastructure comprises all of the business productivity software that is required for employees to complete their work. Since the support of this functionality is not part of the company's core competency and focus, outsourcing the hosting of these services will allow the company to focus on tasks that are more important. The two primary expenses for internal server infrastructure are the hosting of Microsoft Exchange for email and messaging, and the company's electronic document management system, Microsoft SharePoint. These two servers have highly complex configurations requiring specialized employees with knowledge of support and maintenance. Without these two key systems running smoothly, company productivity drops completely. Since Omega is small, maintaining full installations of these servers is not economically efficient. This enterprise software can handle thousands of users, with licensing prices matching this scale. Omega is far from reaching the point where hosting this software achieves economies of scale. This underutilization is a source of waste and excess cost within the organization.

6.2.1 Comparison of Available Outsourcing Services

A comparison of available options for outsourcing the hosting of email servers and document management systems resulted in a shortlist two primary services: Google Apps

and Microsoft Online. Omega evaluated Google Apps as a possible service to replace high Microsoft licensing costs for both mail and messaging and office tools. At the time of this analysis, Google's offering had not evolved to a point where it could be a replacement for MS Office. Microsoft has entered into direct competition for Google with its Microsoft Business Productivity Online hosting services, which makes available a cloud-based Exchange server for outsourced email hosting, as well as the hosting of the company's document management system, Microsoft SharePoint. Microsoft's solution seems to be the primary candidate for Omega outsourcing.

6.3 Migrating Application Hosting

Omega offers a large part of its product offerings in a Software-As-A-Service model. The company maintains an array of database and web servers to provide functionality to its customers. Outsourcing this domain of the company requires careful analysis. The hosting of applications is the primary way the company delivers value to its customers.

6.3.1 Security, Compliance and Healthcare

One primary decision factor for outsourcing application hosting is the legal constraints in the health care industry. Security and the protection of patient health information are of utmost importance to the well being of the company. Due diligence must be taken before selecting a third party hosting provider to be sure that the third party follows certain protocols for security and data protection.

Moving to a major cloud service such as Microsoft Online should not result in a decrease in security. Large companies such as Amazon or Microsoft have more resources to dedicate to necessary maintenance tasks such as backups, hardware support and redundancy. Because they can take advantage of economies of scale, they are able to

specialize and bring on dedicated personnel with a focus in these areas. By having dedicated support personnel and 24-hour emergency procedures, an outsourced hosting company has the ability to focus and specialize on security and make sure that the latest updates and virus definitions are always applied. As a small company, Omega is unable to specialize and focus on all of the intricacies of security and hardware support.

Omega must take into account the legal risk and liability of using an outsourced infrastructure. If the third party hosting service goes offline for a long period, or in the worst case, if the service loses data, is there any recourse for legal action? With medical software, the company must follow strict rules to protect patient privacy. There are significant legal risks if the company does not protect patient health information.

Omega must balance the benefit of hosting data in the cloud with the requirements for patient privacy. Amazon Web Services has made available a white paper on creating HIPAA-compliant Medical Data Applications using its cloud hosting service (Amazon, 2009). Microsoft also "undergoes annual audits for HIPAA compliance" (Shinder, 2009). Any hosting service that Omega decides to use should have some sort of documentation around compliance and published strategies for risk management.

For example, HIPAA requirements condone storing private data in an encrypted state. It adds one more layer of protection as long as one can protect the encryption key. The ability to encrypt the database at a low level without compromising performance should be a factor in deciding what cloud provider Omega chooses. The company will need to upgrade its current database software, Microsoft SQL Server, to the latest version in order to have database encryption that does not adversely affect performance. This upgrade is also required to enable in-depth auditing of all transactions within the database.

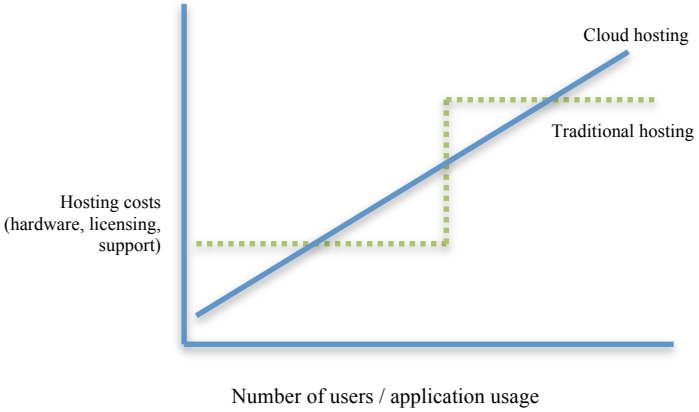
Some jurisdictions and hospital customers require that Omega host patient data on US soil. The same goes for Canadian healthcare laws, which dictate that data cannot be stored on servers residing outside the Canadian border. The healthcare industry places great weight on privacy and the protection of consumers. When selecting cloud services storing patient data, Omega will need to take into account these laws.

6.3.2 The Economics of Cloud Computing

By outsourcing to the cloud, one replaces the stepped and fixed capital costs of an internally hosted infrastructure with the standardized operational costs of using an outsourced service. With running a local server infrastructure, one must face the possibility of unexpected blocks of expenses arising from having to replace a server with a seized hard drive, or having to purchase a large block of new software licenses all at the same time. Rather than having a set and reliable cost every month, the company needs to prepare for the possibility of large blocks of expenses.

There is a greater reliability and standardization of cost structure with outsourcing IT infrastructure. With a cloud-based infrastructure, the company can amortize costs over a longer period through subscriptions that do not include sudden surprises. This decreases the risk for a small company and ensures reliability in terms of cash flow.

Figure 7: Operational Costs of Cloud Hosting vs. Traditional Stepped Costs



Source: Author

Rather than paying large amounts for hosting, cloud hosting allows a pay-as-you-go structure where costs increase in parallel with usage. The figure above illustrates how the cost of cloud hosting increases in a direct relation to the amount the application is used. This allows a small company such as Omega with limited cash to match their infrastructure costs with revenue.

Hosting costs increase directly in relation to the level of traffic. This makes it difficult for the company to offer free services to customers using a penetration pricing or freemium model in order to break into new markets. All usage of a site results in a cost. With traditional hosted servers, the company can afford to offer free services to make use of idle server resources without affecting the bottom line.

This drawback may only be a factor in certain situations however. Typically, setting up traditional server hosting requires accounting and planning for the maximum possible traffic. As an illustration, consider if 360 days of the year server usage runs an average of 1000 users at a time, but for 5 days a year, this balloons to 50,000 users. The server setup thus requires the ability to handle this peak traffic load, even though most of the time this computing power sits idle. Rather than only needing a single server to handle the average

load, it requires five load-balanced servers to account for the peak load. With cloud computing, the server architecture can elastically meet the peak load while charging for only the average load for 98% of the year.

Regardless of this question of idle computing resources, using a cloud-based hosting service for software-as-a-service applications closely ties application use to revenue, since every piece of traffic results in a cost. If Omega can develop billing models that charge the user based on how much an application is used, then the cost of goods sold will more closely relate to revenue.

The company has been moving towards using operational costing models for client contracts. For example, Omega might bill a customer based on the number of patients entered into the system. Customers often prefer this model as well, since it minimizes their own up-front capital costs. The use of cloud hosting for applications could be conditional on what type of billing model has been set with the customer. For existing customer applications that use a yearly support billing model, the pricing would need to be set in a way that would protect against data usage exceeding the actual amount of revenue from the customer.

Before moving the entire hosting infrastructure, management should select a pilot program for a particular product with an operational cost model. This will test out performance and determine ultimate costs for using cloud hosting. Once the company realizes the technical viability and stability of the pilot project, management can look to expanding the migration to the rest of its portfolio.

7: CONCLUSIONS AND RECOMMENDATIONS

Omega is balancing intense growth with the ability to effectively restructure and refine processes. As the company expands in size, using more structured project management methodologies will help teams deliver solutions on time and in quick response to market signals. Outsourcing the hardware infrastructure will help costs remain low with the ability to scale when required.

At the writing of this report, Omega does not currently meet requirements for FDA *21 CFR Part 11*, although it is certainly on the path to achieving this. The necessary project management and document management systems are in place that can act as repositories for audit documentation. Management needs to place additional focus on defined disaster recovery planning, standard procedures for hardware maintenance, and regular internal audits. Adding an additional employee as a Quality Assurance expert whose sole responsibility is to enforce compliance would be a valuable approach to promote due diligence.

Based on discussions with peers and scholarship on the subject, Omega is taking on significant risk and the possibility of high technical debt by using an offshore team. Both the difficulties of coordinating and training in opposite time zones as well as the lack of context for the remote team could result in ballooning development costs and an increase in support difficulties. The recommendations in this paper will help to minimize these issues, such as close communication; code reviews; short iterations followed by releases; and using a structured project management methodology.

APPENDIX A: NEW DEVELOPER DOCUMENTATION

Checklist of steps to set up a new employee:

1. Setting up domain accounts and permissions
2. Software licensing & purchases
3. Legal documentation
4. Account configuration with:
    a. Source Control Server
    b. Project Management Software

Basic set of information for all employees:

1. History of company
2. About company today
3. About products
4. About clients
5. About team and org chart
6. About knowledgebase, where do I go for help
7. My responsibilities and my workflow
8. My applications to do my work
9. My desktop and team collaboration
10. Privacy and security considerations
11. My pay, my days off

Information specifically for development team:

12. How to set up a new development site
13. Basic structure of the platform solution
14. Upgrading sites and preparing upgrade installation files
15. Adding new modules to the platform
16. Documentation of key libraries such as database access and form engine

8: WORKS CITED

ilemanifesto.org. (2001, 02 01). *Principles behind the Agile Manifesto.* Retrieved 06 15, 2010, from Manifesto for Agile Software Development: http://agilemanifesto.org/principles.html

Amazon. (2009, 04 01). *Creating HIPAA-Compliant Medical Data Applications with Amazon Web Services.* Retrieved 06 12, 2010, from Amazon Web Services White Papers: http://koti.mbnet.fi/ideasoft/meh.php?addr=http://awsmedia.s3.amazonaws.com/AWS_HIPAA_Whitepaper_Final.pdf

Atwood, J. (2009, 02 27). *Paying Down Your Technical Debt.* Retrieved 07 10, 2010, from Coding Horror: Programming and Human Factors: http://www.codinghorror.com/blog/2009/02/paying-down-your-technical-debt.html

Axelrod, R. (1987). The Evolution of Strategies in the Iterated Prisoner's Dilemma. In L. D. (ed), *Genetic Algorithms and Simulated Annealing* (pp. 32-41). London: Morgan Kaufman.

Balfour, F. (2006 йил 11-12). *Vietnam's Growing Role in Outsourcing*. Retrieved 2010 йил 27-05 from Bloomberg Business Week: http://www.businessweek.com/technology/content/dec2006/tc20061211_099877.htm

Bartholemy, J. (2001). The Hidden Costs of IT Outsourcing. *MIT Sloan Management Review , 42* (3), 60-69.

Brown, R. (2010, 06 29). *Multitasking Gets You There Later.* Retrieved 07 20, 2010, from InfoQ: http://www.infoq.com/articles/multitasking-problems

Carroll, J. (2003, 03 17). *The commoditization of software.* Retrieved 07 10, 2010, from ZDNet: http://www.zdnet.com/news/the-commoditization-of-software/128144

CIA. (2010, 05 20). *The World Factbook*. Retrieved 05 20, 2010, from United States: Central Intelligence Agency: https://www.cia.gov/library/publications/the-world-factbook/geos/vm.html

CMC Software. (2008 йил 01-01). *Why Vietnam: Intellectual Property Considerations* . Retrieved 2010 йил 27-05 from CMC Software: http://www.cmc-outsource.com/index.php?option=com_content&view=category&layout=blog&id=47&Itemid=55&lang=en

Cockburn, A. (2009, 08 10). *Agile Software Development: Communicating, Cooperating Teams.* Retrieved 07 10, 2010, from InformIT: http://www.informit.com/articles/article.aspx?p=1374901&seqNum=2

Cockburn, A., & Williams, L. (2000). The Costs and Benefits of Pair Programming. *eXtreme Programming and Flexible Processes in Software Engineering - XP2000* (pp. 1-11). Sardinia: University of Utah .

Cusumano, M. A. (2008). The Changing Software Business: Moving from Products to Services. *Computer , 41* (1), 20-27.

David, G. C.-S. (2008). Integrated collaboration across distributed sites: the perils of process and the promise of practice. In J. Ohsri I. Kotlarsky, *Outsourcing Global Services* (pp. 127-150). London: Palgrave Macmillan.

Davison, D. (2003, 01 9). *Top 10 Risks of Offshore Outsourcing*. Retrieved 06 15, 2010, from ZDNet: http://www.zdnet.com/news/top-10-risks-of-offshore-outsourcing/299274

De Baar, B. (2007 йил 05-11). *Dealing With Cultural Differences In Projects*. Retrieved 2010 йил 27-05 from Bas de Baar – Project Shrink: http://www.basdebaar.com/dealing-with-cultural-differences-in-projects-109.html

Derby, E. (2010, 06 12). *Seven Ways to Revitalize Your Sprint Retrospectives.* Retrieved 06 15, 2010, from Esther Derby Associates, inc: http://www.estherderby.com/2010/06/seven-ways-to-revitalize-your-sprint-retrospectives.html

EHR Scope. (2010, 06 21). *List of Electronic Medical Records 2010*. Retrieved 06 21, 2010, from EHR Scope: Connecting Healthcare and Technology: http://www.ehrscope.com/emr-comparison

Ehsan, N. e. (2008). Impact of Computer-Mediated Communication on Virtual Teams' Performance: An Empirical Study. *World Academy of Science, Engineering and Technology , 42*, 1-10.

Fagan, M. (1976). Advances in software inspections to reduce errors in program development. *IBM Systems Journal* , 182-211.

FDA. (1997). *Code of Federal Regulations, Title 21, Food and Drugs, Part 11. "Electronic Records; Electronic Signatures: Final Rule."*. US Food and Drug Administration. Washington: fda.gov.

FDA. (2002). *General Principles of Software Validation; Final Guidance for Industry and FDA Staff.* Food and Drug Administration, U.S. Department Of Health and Human Services.

Fisher, K. F. (2001). *The Distance Manager.* New York: Mcgaw-Hill.

Ford, R., & Randolph, W. (1992). Cross-Functional Structures: A Review and Integration of Matrix Organization and Project Management. *Journal Of Management , 18*, 267-294.

Fowler, M. (2005, 01 13). *The New Methodology.* Retrieved 06 01, 2010, from MartinFowler.com: http://www.martinfowler.com/articles/newMethodology.html

Fowler, M. (2006 йил 18-07). *Using an Agile Software Process with Offshore Development.* Retrieved 2010 йил 27-05 from martinfowler.com: http://www.martinfowler.com/articles/agileOffshore.html

Fowler, M. (2006 йил 18-07). *Using an Agile Software Process with Offshore Development*. Retrieved 2010 йил 27-05 from martinfowler.com: http://www.martinfowler.com/articles/agileOffshore.html

Fowler, M., & Beck, K. (1999). *Refactoring: improving the design of existing code.* New York: Addison-Wesley.

Fox, K. F. (2007, 07 27). *Multi-Tasking: Why projects take so long and still go late.* Retrieved 07 17, 2010, from Theory of Constraints: http://theoryofconstraints.blogspot.com/2007/07/multi-tasking-why-projects-take-so-long.html

Gibson, S. (2006, 03 06). *Bridge Cultural Differences for Better Business in India.* Retrieved 06 15, 2010, from IT Management from eWeek: http://www.eweek.com/c/a/IT-Management/Bridge-Cultural-Differences-for-Better-Business-in-India/

Goldman, D. (2009, 01 12). *Obama's big idea: Digital health records*. Retrieved 06 17, 2010, from http://money.cnn.com/2009/01/12/technology/stimulus_health_care/.

Gopal, A., & Sivaramaktrishnan, K. (2003). Contracts in Offshore Software Development: An Empirical Analysis. *Management Science , 49* (12), 1671-1683.

Gray, J. (2003). *Distributed Computing Economics.* Microsoft Research. Microsoft Research.

Grunbaum, L. (2002). Remaining in a 21 CFR Part 11 Compliant State. *Journal of GXP Compliance , 6* (3), 2.

HHS.gov. (2009, 08 24). *HIPAA: Breach Notification Rule.* Retrieved 06 12, 2010, from U.S Department of Health & Human Services: Health Information Privacy: http://www.hhs.gov/ocr/privacy/hipaa/administrative/Breach%20Notification%20Rule/index.html

Jacquette, F. (2001, 03 01). *Faster, Better, Cheaper... Validated?!* Retrieved 07 04, 2010, from Jacquette Consulting: http://www.jacquette.com/index.php/component/content/article/38-papers-and-presentations/61-faster-better-cheaper-validated

JC Panzar, R. W. (1981). Economies of Scope. *The American Economic Review , 71* (2), 268-272.

Kakabadse, A. K. (2002). Trends in Outsource: Contrasting USA and Europe. *European Management Journal , 20* (2), 189-198.

Knorr, E. (2009, 05 2). *What cloud computing really means.* Retrieved 06 20, 2010, from InfoWorld: http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031

Kogut, B. (2001). Open Source Software and Distributed Innovation. *Oxford Review of Economic Policy , 17* (2), 248-264.

Kostukova, N. (2009 йил 10-12). *Fixed price and dedicated team (offshore development center) models: comparative analysis*. Retrieved 2010 йил 27-05 from Binary Studio: http://binary-studio.com/blog/management-blog/fixed-price-and-dedicated-team-offshore-development-center-models-comparative-analysis/

Krafcik, J. F. (1988). Triumph of the Lean Production System. *Sloan Management Review , 30* (1), 41-53.

Langr, J. (2010, 06 18). *The Only Agile Tools You'll Ever Need.* Retrieved 07 17, 2010, from sfp101: Stress-Free Productivity : http://sfp101.com/?p=4216

Laribee, D. (2009, 12 01). *Using Agile Techniques to Pay Back Technical Debt*. Retrieved 06 25, 2010, from MSDN Magazine: http://msdn.microsoft.com/en-us/magazine/ee819135.aspx

Luehrman, T. (1998, 07). Investment Opportunities as Real Options: Getting Started on the Numbers. *Harvard Business Review* , 1-15.

Mahnke, V. W.-A. (2008). Offshore middlemen: transnational intermediation in technology sourcing. In I. K. Oshri, *Outsourcing Global Service* (pp. 44-70). London: Palgrave Macmillan.

Matts, C. M. (2007, 06 8). *"Real Options" Underlie Agile Practices*. Retrieved 07 3, 2010, from InfoQ: http://www.infoq.com/articles/real-options-enhance-agility

McConnell, S. (2007, 11 1). *Technical Debt*. Retrieved 07 05, 2010, from Construx: Software Best Practices: http://blogs.construx.com/blogs/stevemcc/archive/2007/11/01/technical-debt-2.aspx

Miller, J. (2006, 08 06). *Kaizen in Software Development: Start by Seeing the 7 Wastes.* Retrieved 07 19, 2010, from Panta Rei: A Weblog About Better Ways to Make Things Better: http://www.gembapantarei.com/2006/08/kaizen_in_software_development_start_by_seeing_the_7_wastes.html

Moore, G. (1999). *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers* (2 ed.). New York: HarperBusiness.

Murdock, I. (2009, 02 20). *Open source and the commoditization of software.* Retrieved 07 18, 2010, from Ian Murdock's Weblog on emerging platforms and the power of aggregation and integration : http://ianmurdock.com/open-source-and-the-commoditization-of-software/

Nadler, R. (2007, 07 22). *Agile development in a FDA regulated setting.* Retrieved 07 10, 2010, from Bob on Medical Device Software: http://rdn-consulting.com/blog/2007/07/22/agile-development-in-a-fda-regulated-setting/

Number of Hospitals and Medical Clinics in North America. (2008). Retrieved 2009 йил 27-June from UClue: http://uclue.com/?xq=1878

Number of Hospitals In North America. (2005). Retrieved 2009 йил 27-June from Wiki Answers: http://wiki.answers.com/Q/Number_of_hospitals_in_North_America

Piper, A. (2005, 01 10). *Thrashing: The Productivity and Company Killer.* Retrieved 07 20, 2010, from evancarmichael.com: http://www.evancarmichael.com/Productivity/3235/Thrashing--The-Productivity-and-Company-Killer.html

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* New York: Addison-Wesley.

Porter, M. E. (1998). *Competitive strategy: techniques for analyzing industries and competitors.* Boston: Simon and Schuster.

Ries, E. (2009, 06 29). *Embrace technical debt*. Retrieved 06 28, 2010, from Lessons Learned: http://www.startuplessonslearned.com/2009/07/embrace-technical-debt.html

Schach, R. (1999). *Software Engineering* (Fourth Edition ed.). Boston: McGraw-Hill.

Schwaber, K. (2004). *Agile Software Development with Scrum.* Redmond, WA: Microsoft Press.

Shapiro, C., & Varian, H. (1999). *Information rules: a strategic guide to the network economy.* Boston: Harvard Business School Press.

Shinder, D. (2009, 11 11). *Microsoft Azure: Security in the Cloud.* Retrieved 06 12, 2010, from WindowsSecurity.com: http://www.windowsecurity.com/articles/Microsoft-Azure-Security-Cloud.html

Sivers, D. (2010, 06 19). *How to hire a programmer to make your ideas happen*. Retrieved 07 20, 2010, from DerekSivers.org: http://sivers.org/how2hire

Snir, E. a. (2000). The Emerging Knowledge Economy: Exchange in Internet Spot Markets for IT Services. *12th Workshop on Information Systems and Economics (WISE 2000)* (pp. 1-32). Pennsylvania: Wharton School of Business.

Spagnuolo, C. (2008, 03 10). *Bonuses in an agile organization?* Retrieved 07 10, 2010, from Chris Spagnuolo's EdgeHopper: http://edgehopper.com/bonuses-in-an-agile-organization/

Spolsky, J. (2010, 08 09). *The Econ 101 Management Method.* Retrieved 07 11, 2010, from Joel on Software: http://www.joelonsoftware.com/items/2006/08/09.html

Stetson-Rodriguez, M. (2009 йил 01-01). *Vietnam: Business relations and negotiations*. Retrieved 2010 йил 27-05 from Venture Outsource: http://www.ventureoutsource.com/contract-manufacturing/outsourcing-offshoring/vietnam-manufacturing/vietnam-business-relations-and-negotiations

Stonich, P. J. (1981). Using Rewards in Implementing Strategy. *Strategic Management Journal , 2* (4), 345-352.

Sullivan, K. (1999). Software Design as an Investment Activity: A Real Options Perspective. In P. C. K. Sullivan, *Real Options and Business Strategy: Applications to Decision Making.* (pp. 215-262). Virginia: Risk Books.

Sutherland, J., & Viktorov, A. B. (2006). Distributed Scrum: Agile Project Management with Outsourced Development Teams. *Agile 2006 International Conference, Minnesota* (pp. 1-8). Minneapolis: Scrum Alliance.

Thomas, D. C. (2009). *Cultural Intelligence, Second Edition: Living and Working Globally.* San Francisco, CA: Berret-Koehler Publishers, Inc.

Varian, H. (1995). Pricing Information Goods. *Research Libraries Group Symposium on "Scholarship in the New Information Environment"* (pp. 1-7). Boston: Harvard Law School.

Vining, A., & Globerman, S. (1999). A Conceptual Framework for Understanding the Outsourcing Decision. *European Management Journal , 17* (6), 645-654.

Voss, C. (1984). Technology push and need pull: A new perspective. *R&D Management , 14* (3), 147-151.

Watts, G. (2008, 03 28). *Production Support and Scrum: How should Scrum teams plan for support?* Retrieved 06 17, 2010, from Scrum Alliance: http://www.scrumalliance.org/articles/91-production-support-and-scrum

Wenger, A. (2008, 11 25). *Kaizen for Developers: No Inventory.* Retrieved 07 19, 2010, from Continuations: http://continuations.com/post/61496327/kaizen-for-developers-no-inventory

Williamson, O. E. (2008). Outsourcing: Transaction Cost Economic and Supply Chain Management. *Journal of Supply Chain Management , 44* (2), 5-16.

Williamson, O. E. (1979). Transaction-Cost Economics: The Governance of Contractual Relations. *Journal of Law and Economics , 22* (2), 233-261.

Winter, S. S. (2001). Replication as Strategy. *Organization Science , 12* (6).

Woodward, E., & Surdek, S. (2010). *A Practical Guide to Distributed Scrum* (1 ed.). Boston: IBM Press.