

APPLYING SUPPLY CHAIN METHODOLOGY TO A CENTRALIZED SOFTWARE LICENSING STRATEGY

by

Rachel Felice Sheinbein

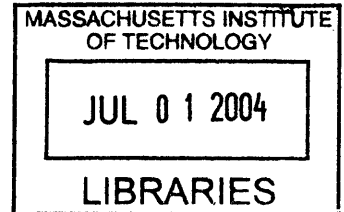
B.S. Chemical Engineering, University of Pennsylvania, 1997

Submitted to the Sloan School of Management and the
Department of Civil and Environmental Engineering
In Partial Fulfillment of the Requirements for the Degrees of

Master of Business Administration
Master of Science in Civil and Environmental Engineering

In conjunction with the Leaders for Manufacturing Program at the
Massachusetts Institute of Technology

May 2004 [June 2004]




© Massachusetts Institute of Technology, 2004. All rights reserved.

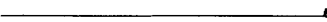
Signature of Author _____

MIT Sloan School of Management & MIT Department of Civil and Environmental Engineering
May 7, 2004


Certified by _____


Dr. Sara L. Beckman, Thesis Advisor
Sr. Lecturer, University of California Berkeley, Haas School of Business

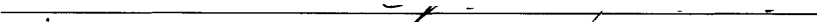
Certified by _____


Dr. Abbott Weiss, Thesis Advisor
Sr. Lecturer, Engineering Systems Division, MIT

Certified by _____


Dr. Donald B. Rosenfield, Thesis Reader
Sr. Lecturer, Sloan School of Management, MIT

Certified by _____


Dr. Cynthia Barnhart, Thesis Reader
Professor of Civil and Environmental Engineering, MIT

Accepted by _____


Margaret Andrews
Director of Master's Program, Sloan School of Management, MIT

Accepted by _____


Dr. Heidi Nepf
Chair, Graduate Committee, Civil and Environmental Engineering, MIT

APPLYING SUPPLY CHAIN METHODOLOGY TO A CENTRALIZED SOFTWARE LICENSING STRATEGY

by

Rachel Felice Sheinbein

Submitted to the Sloan School of Management and the
Department of Civil and Environmental Engineering on May 7, 2004
In Partial Fulfillment of the Requirements for the Degrees of

Master of Business Administration and
Master of Science in Civil and Environmental Engineering

ABSTRACT

Eleven percent of companies spend between \$150K and \$200K per year per engineer on software development tools and nine percent spend more than \$200K, according to a Silicon Integration Initiative/Gartner/EE Times study from 2002. For Agilent Technologies, these costs result in spending tens of millions of dollars each year on software, and for Motorola, the costs are more than \$100M each year. From the current trends in software spending, one can infer that companies will pay even more for software in the future, because the cost of the software itself is rising and because of the complexity of the technology needed for innovation.

In order to understand whether the total spending on software is appropriate and necessary, Agilent sponsored this project to create a model that analyzes the trade-offs between the cost of software and the cost of software unavailability. The model treats software licenses as supplies to the development of a product, and thus, supply chain methodologies such as inventory (cost of licenses), stock outs (cost of unavailability) and service level are applied. The goal of the model is to minimize software costs while maintaining a satisfactory level of service. The thesis explains the model and then shows the results from applying it to four software products that Agilent currently uses.

The results show that in the absence of this type of analysis, Agilent spends more than necessary for software licenses. In fact, Agilent can reduce costs by at least 5%. This model can be used by Agilent and other companies to optimize software purchases.

Thesis Supervisor: Sara Beckman, Sr. Lecturer, Haas School of Business, University of California Berkeley

Thesis Supervisor: Abbott Weiss, Sr. Lecturer, Engineering Systems Division, MIT

Thesis Reader: Donald Rosenfield, Sr. Lecturer, Sloan School of Management, MIT

Thesis Reader: Cynthia Barnhart, Professor, Department of Civil and Environmental Engineering, MIT

ACKNOWLEDGEMENTS

Without Agilent Technologies, this thesis would not have been produced. Thank you to Ted Lancaster and Pete Woodhouse for creating a rich problem statement and giving me the opportunity to explore the solutions. Scott Scaramastro served as an excellent mentor and reminded me to relax when I thought the answers would never be realized. I could have never asked for a better project champion than Janet Wolff. Champion is the perfect word for her role. I would also like to thank Patrick Pulis and Charlie Rothschild for always being available to bounce off ideas, and Marie Shrodes and Gene Wright for teaching me to use an excellent project management process.

My advisors, Drs. Sara Beckman and Abbott Weiss, helped me to produce a powerful model that is still simple. They also patiently edited this thesis so that the model could be understood by others. I appreciate their wisdom and steadfastness.

I wish to gratefully acknowledge the Leaders For Manufacturing program, a partnership between MIT and major U.S. manufacturing companies, who sponsored this research. I feel honored to be a part of the incredible LFM program that provides a challenging and nurturing environment for students to explore and grow. The supportive staff opens the doors to learning by removing obstacles. Specifically, Bill Hanson always reminded me of my strengths, gave me thoughtful advice, and created opportunities for me to lead.

Furthermore, my LFM classmates stimulated new thoughts and taught me more than I considered possible. I would particularly like to thank A-P Hurd for her keen insight, her poignant questions and her ability to produce such eloquent sentences. I would also like to thank Michelle Bernson for always having time to create opportunities for our class to join together and for serving as a role-model for community involvement. In addition, my classmates from all my previous years of education were excellent teachers and they continue to challenge me.

Finally, I cannot say enough about my family. I feel so lucky to be part of a family that continuously supports me, brags about me, and makes it possible for me to succeed. My grandparents, who did not have a chance to have a college education, still emphasized the importance for their children and grandchildren. Furthermore, my Grandma Sally reminds me that small things can be so wonderful. My siblings have varied interests and each contributes to the community in meaningful ways. I learn from their achievements daily. My dad spent many hours reminding me that I should not give up on my math homework, even when I was frustrated. His enthusiasm and guidance served as the necessary foundation for my education and career. I wish he could have read this thesis to see some of the results of his encouragement. And, lastly, and most important, I would like to acknowledge my mother. My mom is an incredible, independent and brilliant woman. She is also an unbelievable mom and friend. Her relentless support of all of my endeavors as well as her willingness to jump in and help out at any time, made it possible for me to complete my studies. Thank you so much for all that you have done for me.

ABSTRACT	3
ACKNOWLEDGEMENTS.....	4
Chapter 1: Introduction.....	7
1.1 Project Motivation:	7
1.2 Chapter Overview	8
1.3 Project Overview:	8
1.4 Agile Overview	9
1.5 Agile Engineering Services (AES) Overview	11
1.6 Summary of Findings.....	12
1.7 Project Management (PM): The Construct.....	13
1.7.1 Overview of PM tools for AES.....	13
1.7.2 Assessment of the PM tools as applied to this project.....	15
1.8 Organization of Thesis.....	17
Chapter 2: Project Description: Applying Supply Chain Concepts to Software License Acquisition.....	18
2.1 Introduction.....	18
2.2 Comparing Cost of Licenses (inventory) and Cost of Unavailability (stock out)	19
2.3 Definition of Service Levels	19
2.4 Model Inputs: Quantifying Costs of Licenses and Inventory	21
2.4.1 Cost of Licenses (inventory).....	21
2.4.1.1 License Usage Data Integrity Issues	24
2.4.1.2 Cost of Licenses Summary	25
2.4.2 Cost of Unavailability (stock out).....	26
2.4.2.1 Hypothesis: Profit Delay and Overtime Frustration	26
2.4.2.2 Engineers' Perspective.....	26
2.4.2.2.1 Conclusions from Interviews	27
2.4.2.3 Product launch delay: Literature Review.....	28
2.4.2.4 Overtime/Frustration.....	29
2.4.2.5 Revenue Loss from Product Launch Delay	29
2.4.2.6 Profit Loss: Ranges Based on Literature Review	31
2.4.2.7 Cost of Unavailability: Summary of Profit Loss Scenarios.....	32
2.4.2.8 Graphical Summary of Cost of Unavailability	32
2.5 Model Output: Determining Total Costs	33
2.6 Conclusion	36
Chapter 3: Case Studies	37
3.1 Introduction.....	37
3.2 Case Studies.....	37
3.2.1 Case Study 1: Printed Circuit Board (PCB) Design Tool 1.....	38
3.2.1.1 Cost of Licenses.....	38
3.2.1.2 Total Costs	42
3.2.1.3 Cost Implications	45
3.2.2 Cost Implications for Case Studies 2-4.....	46
3.3 Key Learning from Case Studies 2, 3, 4	47
3.3.1 License usage time range	47
3.3.2 The normal distribution assumption	48

3.3.3 Contract implications	49
3.4 Conclusion	51
Chapter 4: Future Use of Models and Benchmarking	52
4.1 Introduction.....	52
4.2 Forecasting.....	52
4.3 Review Process	53
4.4 Benchmarking.....	54
4.5 Conclusion	56
Chapter 5: The Organizational Environment and its Impact	57
5.1 Introduction.....	57
5.2 Economic Climate.....	57
5.3 Culture: Collegial and Loyal.....	58
5.4 Politics: The Powerful Development Engineers	60
5.5 Structure: Decentralized Groups.....	63
5.6 Inventing: the common denominator	66
5.7 Interning in this organization.....	66
5.8 Reflection.....	68
5.9 Conclusion	69
Chapter 6: Conclusion.....	70
6.1 Introduction.....	70
6.2 Summary of the model.....	70
6.3 Recommendations.....	71
6.4 Lessons Learned.....	72
6.5 Summary	73
Appendix A:.....	74
BIBLIOGRAPHY	76

Chapter 1: Introduction

1.1 Project Motivation:

More than forty percent of the 166 companies studied in 2002 by Silicon Integration Initiative/Gartner/EE Times spend more than one hundred thousand dollars per engineer on design automation tools.¹ This money is spent to acquire licenses or copies of the software to perform various engineering tasks ranging from printed circuit board assembly layout to integrated circuit simulation to testing software code. Historically, companies have purchased local licenses for each group within the company. In the past five years, however, they have started to acquire the licenses at the corporate level, and to share them globally across the engineering functions in the company².

These companies have realized major cost savings as a result, as they can buy far fewer licenses to serve the overall demand when those licenses are shared. A small consortium of companies that have implemented global licensing (e.g., Agilent, Intel, Ford, Honeywell, Motorola, Texas Instruments) share best-known practices by meeting twice a month. However, the process of forecasting demand for licenses, deciding how many to acquire and administering their use within the company still needs improvement. This drive for improving the global licensing program motivated Agilent to develop the project described in this thesis.

Agilent Technologies, in particular, spends tens of millions of dollars each year on software used by engineers to develop products, but it does not have a rigorous approach to determine whether the amount spent is necessary to support the needs of the organization. The goal of this project is to provide Agilent a more analytic method for minimizing centralized license software costs while maintaining a satisfactory level of service for the design engineers. This project employs a quantitative method based on

¹ Silicon Integrated Initiative Inc., Gartner Dataquest, EE Times. *Engineering Design Automation Study*. May 2002.

² Griffith, Dan Chairperson, Central Enterprise Licensing User Group (22 company consortium) and Manager, Comprehensive Software Asset Management at Motorola. E-mail dated 12-2-03.

those used in supply chain management to evaluate the tradeoffs between the cost of licenses (i.e., having an inventory of licenses) and the cost of license unavailability (i.e., having engineers sit idle, or project launch delayed). The value of this project is to have better information than is currently provided by heuristics, thus saving Agilent money.

1.2 Chapter Overview

This chapter introduces the project and outlines the results of an internship conducted at Agilent Technologies in the summer and fall of 2003. It also gives a high-level overview of the company and the group that hosted the internship. Finally, it explains the organization of this thesis.

1.3 Project Overview:

Based on the literature on supply chain management, specifically related to balancing inventory carrying costs and customer service levels, I developed a model for examining the tradeoffs between software license costs and license unavailability. I then conducted case studies on four different software products that varied in terms of cost, complexity and usage patterns. In each case, I studied the usage patterns for those products over the past year, collected data on the costs of the licenses, and assessed the effects of unavailability of the products to the engineering organization and the company at large. From this information, I was able to determine an optimal number (inventory) of licenses to have based on the costs of licenses and the cost of unavailability (stock outs) at different service levels.

In the current business environment, the average of the results of the case studies showed that Agilent can save more than 5% of what it currently spends on software licenses by reducing the number of licenses it owns. The general results from the case studies will aid in making determinations for other current and future software products.

Furthermore, the model itself can be used to predict license requirements as the economy recovers and business conditions change.

1.4 Agilent Overview

Agilent Technologies' mission is: "to provide solutions and technologies that revolutionize the way people live and work." Agilent is a \$6 billion company that has been unprofitable from August 2001 until October 2003 (effectively the 9 quarters preceding and including this internship). Agilent was originally made up of three groups from Hewlett Packard (HP). HP (now Hewlett Packard Compaq), founded in 1939, decided to spin-off these groups on November 1, 1999 in order to form Agilent Technologies. See 1.4.1.

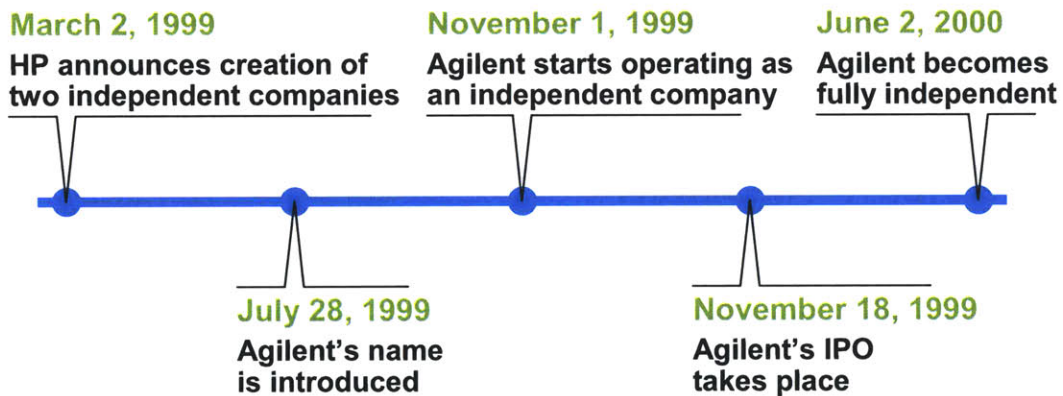


Figure 1.4.1 Timeline of Agilent's spin-off from HP³

Agilent now has four main business groups: Test and Measurement (TMG), Automated Test Equipment (ATG), Semiconductor Products (SPG) and Life Sciences and Chemical Analysis (LSCA).

The Test and Measurement Group (TMG) is the largest and accounts for about 43% of Agilent's revenue and includes the Automated Test, Communications Solutions and

³ Figure taken from Agilent Profile Slides, Agilent Technologies, Inc., 2003

Electronic Product and Solutions divisions. The core businesses within the group are communications test, electronics test and service⁴. TMG considers broadband, IP/data networking, wireless, network management software and aerospace/defense areas of opportunity. This group produces products such as oscilloscopes, RF and Microwave instruments and systems, and lightwave and photonic measurement solutions.

The next largest group, which generates about 26% of Agilent's revenue, is the Semiconductor Products Group (SPG). SPG lists its core businesses as fiber optic communications, infrared components, ASICs and optoelectronics. Future opportunities include gigabit networking, cellular chipsets, wireless appliance, and networking ASICs. One of the blockbuster products that came out of this group in the last few years is the optical device that is used in computer mouse devices.

Life Sciences and Chemical Analysis (LSCA), which makes up about 20% of Agilent's revenue, has maintained profitability even during the times when Agilent as a whole was not realizing a profit. This group has products in the exciting biotechnology space, including micro arrays, gene expression and proteomics. Support services and training bring in about 20% of the revenue for this group. Traditional products include chemical analysis equipment such as general and liquid chromatography and mass spectrometry tools. Areas of opportunity include lab-on-a-chip, bridging informatics, as well as homeland defense, food safety and environmental work in Asia.

The Assembly Test Group (ATG) brings in about 11% of Agilent's revenue. This group has equipment for testing parametric diagnostics, non-volatile memory, mixed-signal, analogs, systems-on-a-chip, automated x-rays etc. It sees future opportunities in areas such as digital consumer and wireless products and enterprise networking and storage. An organization chart with all these groups is shown in Figure 1.4.2 below.

⁴ Product information from Agilent Profile Slides, Agilent Technologies Inc, 2003

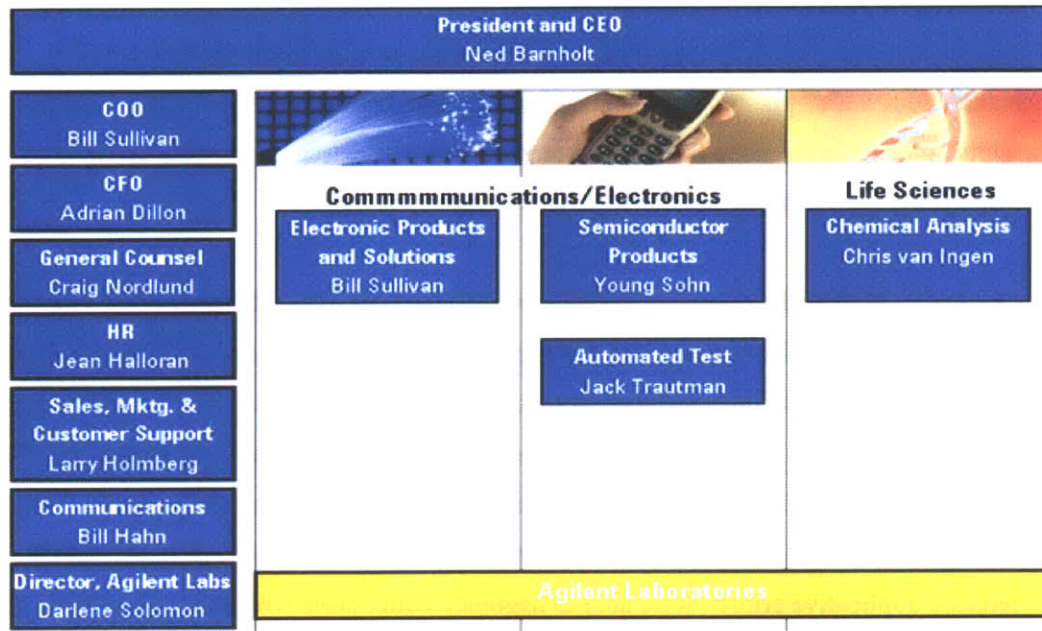


Figure 1.4.2 Organization Chart for Agilent Technologies

As is clear from the lists of inventive technologies and product lines, new product development is core to Agilent’s future success. All of these groups use software in the development of their products. Some software tools are used solely in certain groups; ASIC design tools, for example, are used only in SPG. Other software products, such as software bug detection tools, are used in every business. Thus, the research represented in this thesis is germane to the entire organization and critical in supporting the organization’s future.

1.5 Agilent Engineering Services (AES) Overview

In addition to these business groups, Agilent has centralized support groups that it refers to as the Global Infrastructure (GI). Groups such as finance and legal are part of the GI as well as the group that hosted this project, Agilent Engineering Services (AES). AES plays two critical roles: it provides central engineering support and owns the centralized

systems for product generation. Central engineering support includes the role of coordinating knowledge transfer and process leadership among engineers in different organizations and geographies. The group's projects include implementing a global enterprise resource planning system and holding monthly knowledge-sharing meetings with engineers in the same discipline from different business groups. This group is also responsible for managing central software licensing.

Originally, each of the four business groups negotiated its own software licenses for the software to support its new product development, but a coordination initiative launched by AES resulted in sharing software licenses across the corporation. For the past 2 years, AES has negotiated and managed the contracts for twelve software vendors that provide software for such tasks as ASIC development, printed circuit board design and software coding. This consolidation effort alone saved Agilent a large amount of money as they reduced duplicative efforts, were able to negotiate better deals with some of the software vendors, and could carry fewer licenses while still providing the same service level. AES is now ready to take the next step, however, to fine tune the centralized process and see if there is the potential for further savings. AES thus developed the problem statement for the project for this thesis:

Agilent would like to allow our engineers to have access to the best tools, whenever they need them. However, this is prohibitively expensive, so a balance between supply and demand needs to be struck. The problem we face is to know where that balance point is, predict it and be able to supply our engineers at that level.

1.6 Summary of Findings

The research project results indicate that in the absence of a cost analysis, Agilent buys at least 5% more software licenses than it needs. Without the cost data, Agilent can only rely on past usage charts and informed intuition for making decisions on quantities of

licenses to buy. The cost information, coupled with usage information, gives Agilent a powerful tool to determine whether buying more licenses is really worth the expense. Cost pressures at Agilent are intense, particularly after two years without profits, so this analysis is very timely. There is the possibility of saving more than \$1M per year using this model. Further, as Agilent starts to make money and grow, this model will allow the company to appropriately increase the number of licenses purchased to meet the growing needs of the organization.

1.7 Project Management (PM): The Construct

In order to conduct the project, I used Agilent Engineering Services' (AES) tailored project management process. The PM tools forced me to operate to a schedule. They also created a direction for the project and ensured that management was aware of my plans.

I include this section in my thesis, in part, because of anecdotal evidence I've collected from my classmates about the lack of project management on many of their projects. Overall, I found these tools useful, and I believe that my classmates would have benefited from the use of this type of tool.

1.7.1 Overview of PM tools for AES

AES employs a project management lifecycle tool for each of its projects. There are five main sections: concept, proposal, execute/monitor/control, close and lessons learned. The execute/monitor/control portion is made up of four phases: analysis, design, build and validate. (See figure below 1.7.1.1)



Figure 1.7.1.1 Project Management Lifecycle

A project starts at the conceptual phase. At this point, someone has an idea for a project that will aid the company, but the details are unspecified. The employee writes up a concept sheet and receives approval from management to go to the proposal phase. During the proposal phase, the employee writes a clear project plan with a problem statement, the business goals, the scope, the resources needed etc. This is all presented to management along with a risk matrix, a net present value analysis, and a work breakdown structure (WBS). If this is approved, the project enters the analysis phase and actual work begins on the project itself.

A standard project would continue through the design, build and validate phases with management review after each phase. However, if it makes sense to combine phases because iteration is needed or design and build can be one effort, then this is possible. In fact, the lead for this process in AES said about the project management tools, “It’s not the templates and it’s not even the process, it’s really about the thinking,”⁵ so there is room for flexibility when it makes sense.

AES uses one lifecycle to make the process straightforward to the users, but there are many different lifecycles that can be used depending on the focus of the project.⁶ These include control-oriented project lifecycles, quality-oriented lifecycles and risk-oriented

⁵ Wright, Gene, Process Lead for AES Program Management tools at Agilent. Phone conversation 9-03.

⁶ Bonnal, Pierre, Didier, Gourc, and Lacoste, Germain “The life cycle of technical projects” Project Management Journal. Vol 3 Mar 2002 p.1

lifecycles.⁷ A control-oriented project lifecycle, for example, would focus on ensuring that the project meets its specifications throughout the implementation phase and meets the schedule within budget; a quality-oriented lifecycle would spend more time on validating that the results prove, without any ambiguity, the original intent of the project.⁸ AES' lifecycle is more general. It includes components of control, quality and risk, but it focuses on each of these at a higher level than the lifecycles described above. This is so that it can apply to all of the projects for the group.

In addition to using a standard lifecycle, AES also makes the process simple by offering templates for many of the tasks. There are templates for all of the documentation for the proposal phase, and phase reviews have a power point template. There is also a similar power point template for program reviews, which occur between phase reviews in order to give management an update on the project's progress. These program reviews provide monthly updates on the project, and project managers are also encouraged to publish weekly updates. This provides an opportunity for stakeholders to give feedback throughout the process.

1.7.2 Assessment of the PM tools as applied to this project

The PM process was easy to learn, and after a week at Agilent, I knew the basic components. Most projects would have started with the concept phase. However, that phase was completed before I arrived: management had created the project and had already bought into it. Therefore, as soon as I talked to a couple of people and gathered ideas about my project, I began writing the proposal documentation. Even though by this point I had some idea about my project, I did struggle to convey the goals, objectives, and the risks in the plan. I also struggled to compute the expected net present value of the project because it was based on huge assumptions, so the analysis seemed questionable to me.

⁷ Ibid.

⁸ Ibid.

I had a lot of apprehension in the beginning, but as I started the analysis phase, I kept going back to the original proposal to help with scope and to remember the original intent. Although I thought the original plan would not reflect the actual execution, I found that it did. The proposal served as the foundation for the next steps and is still applicable at the end of the project. It helped me decide when to take one path versus another, but it was not too restrictive. Later the words I used in the program plan took a quantitative form and had a deeper layer of meaning. Some of the statements became inaccurate, but nonetheless the proposal documents were very powerful and useful.

The PM tools gave structure to the project and provided a built-in process for managing the project. Even though the actual work to be done was not always clear, there was never any question about the process to use. I talked with other MIT interns working at different companies, and many of them did not set up formal plans in the beginning. This often resulted in frustration and unclear goals, and even a struggle to come up with a coherent thesis topic towards the end of the internship. However, I did not encounter these problems because of the PM tools, which provided clear guidance throughout the internship.

In addition, I received instruction on the tools and support and encouragement for utilizing them. The PM tools provided the most impact at the beginning of the project and in the getting-started phase. Later in the project lifecycle, the communication and review tools were used, but they required a lot less thought and did not aid in shaping the project.

One point of note in the process is that preparing for the presentations often took a significant amount of time. The purpose of the presentations is to communicate the results to date and receive feedback. AES conveys that the presentations can and should be created quickly, but I did not find this to be the case. However, it was beneficial to put in this time because it resulted in more substantial feedback; AES should not bill the presentations as a quick endeavor.

1.8 Organization of Thesis

The remainder of this thesis provides more detail about the development of the model and the results achieved in the four case studies. Specifically, Chapter 2 describes the application of supply chain methodology to software licenses and the particular model developed as part of this project. Chapter 3 then reports the findings of the models for four different software products. The results in Chapter 3 are based on historical data, so Chapter 4 explores the effects that various future events might have on the integrity of the findings and also recommends points at which to reexamine the results. Chapter 5 describes the organizational environment and its effects on the project. Finally, Chapter 6 gives the recommendations and lessons learned.

Chapter 2: Project Description: Applying Supply Chain Concepts to Software License Acquisition

2.1 Introduction

Agilent Engineering Services has a team of people responsible for the Engineering Design Automation (EDA) global-software-tool licensing program. Three of the members of this team manage the contracts with the suppliers and determine the license needs for each software tool. No standard process for making decisions on number of licenses exists, but the negotiators look at usage graphs that include usage utilization and periods of peak usage when making their decisions. Each negotiator's intuition also plays a role in the decision making process.

AES management wanted to know whether its investment in EDA software tools was appropriate. With the current process, the negotiators did not take detailed cost trade-off information into consideration when making decisions and did not know whether they were purchasing the optimal number of licenses to adequately serve the engineering community at reasonable cost. As a result, AES created this project to construct a cost-based model.

After reviewing different modeling techniques, I chose a supply chain model that is commonly applied to physical products. The methodology relies on basic supply chain concepts such as inventory, stock out and service levels. Specifically, the methodology calculates the cost of having inventory (in this case the cost to purchase software licenses) and the cost of having stock outs (in this case the cost of not providing an engineer with a software tool when needed) and allows the user to identify a cost effective level of service (acceptable number of stock outs). This chapter describes the application of supply chain concepts for software licenses and the definition of the specific terminology for this project.

2.2 Comparing Cost of Licenses (inventory) and Cost of Unavailability (stock out)

AES is responsible for contracting with suppliers for software licenses. In more than 60% of the contracts⁹, it tells the software supplier how many licenses it needs, based on the best estimates the negotiators can make about what demand will be over the life of the contract, and then pays for that many licenses over the period of the contract. The contract length varies from 3-5 years, and usage can fluctuate during that time. The supplier makes the licenses available on a Central Licensing Server (CLS), supported by Agilent, from which members of the worldwide Agilent engineering community can check out the licenses. When all of the licenses are checked out, engineers must wait for one to become available on the CLS. Agilent does not want to have so many licenses that it is paying for a large number of unused licenses, but on the other hand, it wants to have enough so that licenses are available when engineers need them. Assessing this tradeoff is similar to determining how much inventory to carry of physical products: there are costs of carrying unused or unsold inventory, and there are costs of not having that inventory available when needed.

In making inventory trade-offs, a company will establish the amount of inventory such that it can satisfy a specified demand. In doing so, it will consider the cost of holding this inventory as well as the cost of unavailability or stock out. (This is the cost incurred when a customer requests a product, and it is unavailable.) With this information, the company will establish an optimal service level (number of times the product is available as a percent of time asked for). In this project, I applied these same concepts of inventory, stock outs and service levels to software tools.

2.3 Definition of Service Levels

⁹ On the other 40% or so of Agilent's contracts, the supplier will provide unlimited licenses and charge a fixed amount or charge a pay per use fee. Under such circumstances, the model developed in this thesis is less applicable but still provides some insight (See Section 3.3.2.3). When Agilent pays only for its usage, then there need be no stock outs, and Agilent never pays for excess inventory of licenses.

Service level is a common supply chain term used to explain what percentage of the time the orders for supplies or products will be filled or available from stock.¹⁰ For this project, the service level translates into the number of hours that a license is available each year when requested. In this model, I assumed that licenses may be requested 24 hours per day, 365 days per year. This is a reasonable assumption given that Agilent is a global company, and that engineers are working somewhere in the world at any given time during the day. In actual fact, for most licenses, there are regular requests throughout the 24-hour day, but weekends and holidays show lower demand. As Agilent increases its operations outside of the US there will be more consistent use throughout the days and weeks. For now, my assumption of 24-hour use 365 days a year is conservative; minimally, it ensures that peak (weekday) use is covered.

If Agilent chooses to offer a 99% service level, licenses are unavailable 1% of the time, which is 88 (365 days/year * 24 hours/day * 0.01) hours per year. Other service levels and their expected hours of unavailability are shown in Figure 2.3.1

Service Level	Hours of unavailability each year
50%	4380
75%	2190
99%	88
99.9%	8.8
99.95%	4.4
99.99%	< 1
99.995%	< 0.5

Figure 2.3.1 Hours of Unavailability at Various Service Levels

These hours of unavailability are spread out over all of the users and over the entire year. The probability that all the hours will occur in a row (meaning that one user would suffer all 4.4 hours of unavailability at the 99.95% service level) is infinitesimal, and the

¹⁰ Hopp, Wallace J., Spearman, Mark L., Factory Physics: 2nd Edition. McGraw-Hill Higher Education 2001 p. 70, 489.

probability that the same user will experience more than one outage depends on the total number of users. (If there are more users, the probability that one user will experience more than one outage decreases)

2.4 Model Inputs: Quantifying Costs of Licenses and Inventory

Holding unused software licenses results in an inventory cost (cost of overage) and unavailable licenses result in stock out costs (cost of underage). In order to optimize license numbers, I needed to quantify the cost of each of these for different service levels for each software tool. I will explain how I determined the costs, including the data collection process and the calculations used. I will also relate the assumptions and data integrity issues, which will help to determine the model's applicability for varying circumstances.

2.4.1 Cost of Licenses (inventory)

This section describes how I determined the cost of licenses for each of the service levels of interest. It starts by describing the usage data I collected for each of the four licenses I studied. It then describes how I used this data to determine the mean and standard deviation of demand. Finally, it describes how I calculated the required number of licenses for each service level.

The Central License Server (CLS) stores usage data for the software licenses it distributes. For each of the four software tools I studied, I extracted hourly usage data for periods of 3, (ex: Jul '02 – Sep '02) 6, (ex: Jul '02 – Dec '02), 9 (ex: Jul' 02 – Feb '03) and 12 (ex: Jul '02 – Jun '03) months. I did the analysis at these different time intervals in order to assess the usage data's sensitivity to time. If the data did change significantly over time, then it would make sense to compare the results from the different time intervals. However, at Agilent, there was not a meaningful difference when using 3 months of data versus 6 or 9 or 12. (This will be discussed again in Chapter 3.)

Because the model I used to determine license needs at the various service levels requires that the demand data be normally distributed (or that it be transformed to normally distributed data), I first tested the data to see whether it was normally distributed. For the demand data that was normally distributed, I calculated average demand and the variability of the demand for each of the time periods examined (3-months of data, 6 months of data, 9 months of data and 12 months of data).

For the data that was not normally distributed, I attempted a transformation of the data to make it so. Normally distributed data has skew and kurtosis values close to zero. It is most important that skew be close to zero, so the goal when transforming data is to first try to achieve a skew close to zero and then work on the kurtosis. Transformation is done by applying an exponent (referred to as theta: θ) between 0 and 1. One iteratively applies different exponents to the data set until the skew and kurtosis approach zero. Once the optimal θ is found, the average and standard deviation can be calculated. Later, the estimated numbers of licenses can be untransformed by raising them to the power of $1/\theta$.

The model works best if the data is normally distributed or transformable to normal. However, if the data is not normal, the model can still provide useful insight. This will be shown when reviewing the case study results in Chapter 3.

For a given service level, the necessary number of licenses can be calculated as the expected or average number of licenses demanded plus a safety stock of licenses to cover the variability in the demand. This safety stock is calculated as a “z” value times the standard deviation of demand, where the “z” value is determined by the required service level. The “z” value represents the point on the standard normal distribution curve for which the area under the curve to the left of that point is equal to the service level. This point is called the “inverse of the normal cumulative distribution” and can easily be calculated using the NORMSINV function in Excel. So, for each service level of interest, and each software tool I studied, I calculated the required number of licenses as:

$$\text{Number of licenses required} = E(D) + z * \sigma_D \quad (\text{Formula 2.4.1.1})$$

where $E(D)$ = Expected Demand for licenses

z = the inverse of the normal cumulative distribution for the given service level
= NORMSINV (service level)

σ_D = standard deviation of demand

Once I knew the number of licenses required, I multiplied the number by a cost per license to determine the cost of the licenses at different service levels. At Agilent, the price of each license ranges from \$1000 to \$100,000. (This model assumes that there are no additional overhead costs for storing the licenses or managing the licenses.) See

Figure 2.4.1.1 as an example of cost of licenses data:

Service Level	Number of Licenses (based on 6 month usage data)	Cost of Licenses (price is \$15,000)
0.5	16	\$240,000
0.7	20	\$300,000
0.75	20	\$300,000
0.8	21	\$315,000
0.9	23	\$345,000
0.95	26	\$390,000
0.99	31	\$465,000
0.999	35	\$525,000
0.9999	40	\$600,000

Figure 2.4.1.1 Example of License Numbers and Cost Data for 6 month Usage Data

Graphically, this data looks like Figure 2.4.1.2:

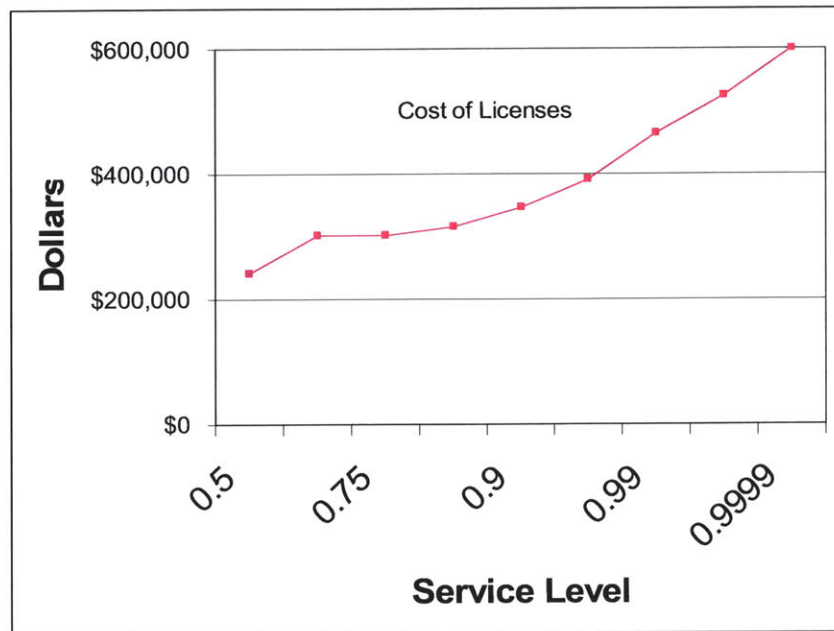


Figure 2.4.1.2 Graph of Cost of Licenses versus Service Level

These same types of results can be computed based on 3, 9 and 12 months of usage data. (See results of the Case Studies in Chapter 3)

2.4.1.1 License Usage Data Integrity Issues

Although I pulled hourly usage data for the same 9-12 months (depending on availability) -- usually from July 2002 to July 2003-- for each software product, the CLS returned a different number of data points for each. None of the data returned included every hour requested, and it is possible that, as a result, the data does not represent the true usage. A rough calculation suggests that for 6 months with 30 days in each month and 24 hours per day, there should be 4320 data points. For the four case study products I received 3178 data points (74%) for one, 2641 data points (60% of the potential hours available) for another, 2374 points (55%) for the third and only 10% for the fourth (This one also did not have normally distributed data, so I did not make any cost conclusions for this product. This will be discussed in Chapter 3.)

Furthermore, the available data only includes data up to the peak number of licenses. If licenses were unavailable (i.e. there is a denial of service for the user), and there were one or more engineers who needed a license at the time of a peak, this data was not recorded. In other words, demand for licenses above the total number of licenses available is not recorded. This could cause a skew in the data towards lower demand than was actually experienced, because actual demand may have been more than actual usage during periods of peak demand when all licenses were checked out. Fortunately, there were very few times when all licenses were checked out, so this issue likely does not have an impact on the data. I recommend, however, that Agilent commence collecting “stock out” data to more accurately reflect demand in future applications of the model.

Finally, all of the usage estimates are based on historical data. There is no forecasting. For the most part, this did not seem to matter, because there was not a major change in average usage when comparing 3 months, 6 months or even a year of data. However, this could be a problem in the future if there are large increases or decreases in license use. Although Agilent knows that it needs to improve its forecasts of software tool usage, we determined that this was outside the scope of my internship project. I nonetheless recommend that Agilent consider developing a tool to forecast software tool usage (see Chapter 4) that can be used as input to this model in the future.

2.4.1.2 Cost of Licenses Summary

There are five steps in determining cost of licenses at different service levels.

1. Collect historical usage data
2. Check for normality (if necessary, transform to normal)
3. Calculate the mean and standard deviation for the data set
4. Calculate the license needs at different service levels (Formula 2.4.1.1)
5. Multiply the license number by the cost per license to determine cost at different service levels

2.4.2 Cost of Unavailability (stock out)

For inventory control, a stock out is defined as a demand unit not being filled because the stock is not available.¹¹ This cost could include the cost of a lost sale, the cost of being late in filling the demand and/or the cost to meet the demand in some other way. In this case, the cost of a stock out is the cost to Agilent of the unavailability of a license when the engineer requests it. This cost could theoretically range from nothing to the cost of shipping a product late if its introduction were delayed by the unavailability of the software. To better understand the cost of unavailability at Agilent, I came up with a hypothesis about the components of “unavailability” cost, and tested my hypothesis by interviewing Agilent managers and engineers. I then reviewed relevant literature to further understand the cost implications.

2.4.2.1 Hypothesis: Profit Delay and Overtime Frustration

I hypothesized two extreme situations: in the most severe case, a lack of license could prevent an engineer from completing his or her work, and result in a product launch delay. This could potentially be quite expensive to the organization. On the other end of the spectrum, the lack of a license could simply cause the engineer to have to work overtime to make up for the lost time, and would be frustrating, but cost the organization very little. I approximate this cost as if overtime had some negative value, even though 90% of the license users at Agilent are salaried, and thus don't receive overtime pay,

2.4.2.2 Engineers' Perspective

To learn about the real-life effects of license unavailability, I interviewed 30 engineers from different business groups with electrical, mechanical and software backgrounds. I created a questionnaire (see Appendix A) and set up phone meetings. I asked the

¹¹ Hopp, Wallace J., Spearman, Mark L, Factory Physics: 2nd Edition. McGraw-Hill Higher Education 2001 p. 78.

engineers about their experiences with license unavailability, and what the effects of unavailability had been or might be.

Only half of the engineers had actually experienced a stock out and most of these engineers had outages before the Central Licensing System (CLS) was put in place. The engineers who had experienced outages said that they never lasted more than a day, and they never resulted in a product launch delay. However, the outages often resulted in frustration and/or having to rearrange work schedules.

Even though half of the engineers had not actually experienced license unavailability, I needed to understand its impact for analysis purposes. Therefore, I asked the engineers to imagine the impact hypothetically. Their responses indicated that at some points in the project, such as close to a product release, an unavailable license would directly impact the product release date. However, it was possible to conclude, based upon their input, that a couple of hours of outage would usually not have much of an effect on a product launch. It would cause some frustration and rearranging of schedules, but 60% thought that it would probably take an outage of two days to a week for license outages to impact the schedule. Twenty percent believed that in the worst case half a day (four hours) to a day of license outages could actually impact the schedule.

2.4.2.2.1 Conclusions from Interviews

Based on the input from the engineers, I made the following assumptions for the form of the cost of unavailability curve. When licenses are unavailable 4 hours or more per outage, I assume that the cost of unavailability includes both overtime (representing engineer frustration and work rescheduling costs) and product launch delay costs. When licenses are unavailable 4 hours or less per outage, I assume that the cost of unavailability is only the cost of overtime, and that product launches will not be delayed. Note that this is a conservative assumption, as only 20% of the engineers interviewed thought that product launch delay could occur at the 4-hour delay level. A less conservative, yet still

practical assumption would be that product launch delay costs kick in at two consecutive days of unavailability per year. Although I do not show the results of such an analysis in this thesis, this would cause Agilent to carry even fewer licenses.

2.4.2.3 Product launch delay: Literature Review

The engineers did relate in their interviews that when a task is on the critical path, and the software that is necessary to complete the task is unavailable for 4 hours or more, the task is delayed. This results in a slip in schedule. In order to understand the cost implication of this delay in schedule, I consulted literature on the subject.

Many articles stress the negative implications of product launch delays. One McKinsey study, from the early 1980s, that is quoted in almost every subsequent article on this subject states, “high-tech products that came to market even 6 months late earned 33% less profit over 5 years.”¹² The article indicates that it makes more business sense to go over budget on development costs because, according to the McKinsey study, “profits were only reduced 4% when they came out on time but 50% over budget”¹³. The point is further emphasized in a more recent Arthur D. Little study that said, “Exceeding the launch time by 10% (which for a development period of 2 years is less than 3 months) reduced total revenues by 25-30%”¹⁴.

This often-quoted McKinsey study, originally written by Donald Reinertsen, summarizes the effects based on studying the high-tech industry. Reinertsen becomes more specific in later articles and books in which he states that for each product’s market scenario, such as a monopoly, or products with high switching costs, there are different costs of delays.¹⁵ In the same book, as well as an article he published in 1999, he says that knowing the cost of delay is extremely important, because it helps a company evaluate

¹² Edward, Katherine, “Achieving Cycle-Time Excellence in the Global Economy: The Impact of Speed”, 2001 AACE International Transaction, PM.12.2

¹³ Ibid

¹⁴ Ibid

¹⁵ Reinertsen, Donald, Managing the Design Factory: A Product Developer’s Toolkit (Free Press: October 1997) p. 28

trade-offs of speed, product performance etc.¹⁶ Reinertsen specifically mentions that companies need a business level cost of delay in order to decide when to add another computer-aided design license.¹⁷

Overall, it is clear that there are costs associated with delaying a product launch. Different authors might offer varying costs, but there is no debate that delays are costly. Furthermore, delaying a project results in more financial losses than when a company exceeds research and development or production spending in an effort to stay on schedule.

2.4.2.4 Overtime/Frustration

As mentioned earlier, more than 90% of the users of the software tools at Agilent do not receive overtime compensation. However, there are cost implications of employees working extra hours and, as a result, often becoming frustrated. Hourly workers typically receive one and half times their pay (1.5X) for overtime. In order to also incorporate frustration, I added an addition factor of 0.3X, for a total overtime/frustration factor of 1.8X (where X is the average U.S. engineer's salary). However, 30% of the engineers live outside of the US and, for the most part, are paid less than the US wage. Overall, this multiplier only serves as an estimate, but unless this factor was off by a factor of 5 (i.e., the right estimate was actually 9X), the outcome of the model is not sensitive to these assumptions.

2.4.2.5 Revenue Loss from Product Launch Delay

Based on the literature review, and inputs from the engineering community at Agilent, I assumed that more than four hours of software license unavailability causes product

¹⁶ Reinertsen, Donald, "How Much is Speed Really Worth?" *Electronic Design*, May 3, 1999 v47 p. 64F

¹⁷ Reinertsen, Don, *Managing the Design Factory: A Product Developer's Toolkit* (Free Press: October 1997) p. 190

launch delays that result in profit losses. However, there is no proof that this cause and effect relationship actually exists. There are many possible causes of product launch delays such as time to overcome a technical challenge or a problem during manufacturing. The cause and effect relationship between license unavailability and product launch delays is more tenuous, even though the engineers and R&D management believe it could exist. Engineers also mentioned that for certain products, if the launch is delayed beyond the market window, the delay could result in missing the market opportunity completely or losing a customer. Licenses would have to be unavailable for a much longer amount of time than the service levels that the model incorporates, so the model ignores this possibility.

If a license is unavailable and does cause a product delay, it would be difficult to determine which product was actually delayed. Therefore, the model assumes that, in the worst case, the license unavailability causes a delay for Agilent's highest revenue generating product or, in the more likely situation, the average revenue-generating product. I received the revenue per quarter for the highest and average product for Agilent's largest product group, EPSG, but obviously these are just ballpark numbers as the revenue for products changes each quarter.

The only available product information came in the form of revenue generated per quarter. However, the cost of unavailability impacts profits, so for each scenario I estimated a range of potential profits from 10-20% of revenue. Obviously, different products have different goals, but this is a good estimate because it is the target for Agilent overall and managers from the business groups concurred that it was appropriate for most products. However, some products might generate more or less profit. Figures 2.4.2.5.1 and 2.4.2.5.2 show the range of profit loss for the highest revenue-generating product and an average revenue-generating product.

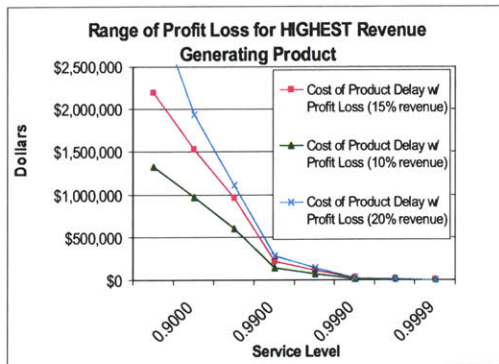


Figure 2.4.2.5.1 Profit Loss Range for Highest Revenue Generating Product

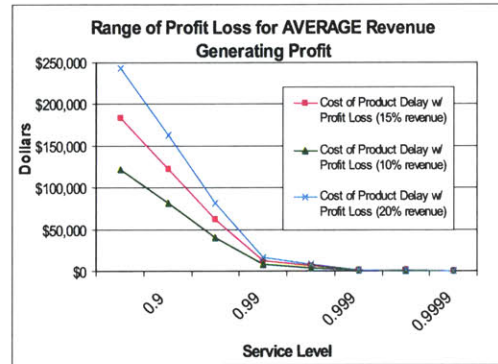


Figure 2.4.2.5.2 Profit Loss Range for Average Revenue Generating Product

2.4.2.6 Profit Loss: Ranges Based on Literature Review

The figures above assume that every hour of unavailability results in an hour of profit loss and do not incorporate overtime/frustration costs. However, the literature study in Section 2.4.2.3 does not state that this is the case. For example, the ADL study explains that a 10% delay results in a 30% revenue loss, not a 100% profit loss.

A limitation of the literature study is that it relies on knowing the length of time that a product is delayed beyond the original development cycle. The actual length of delay will not be known until the end of the project, which is too late to determine license need. In order to apply the study, in one version of the model, I just assume that the license unavailability results in a 30% reduction in revenue, no matter what the delay actually is. (See Figure 2.4.2.6.1 for the range of profit loss for the 30% of the highest revenue-generating product.) Another version of the model assumes an immediate loss of profit, so that every hour of delay results in an hour of profit loss. (See Figures 2.4.2.5.1 and 2.4.2.5.2) In the latter scenario, the cost of stock out is higher so would cause the model to suggest that Agilent buy more licenses and carry more inventory and have a higher service level than in the case in which the ADL hypothesis is applied.

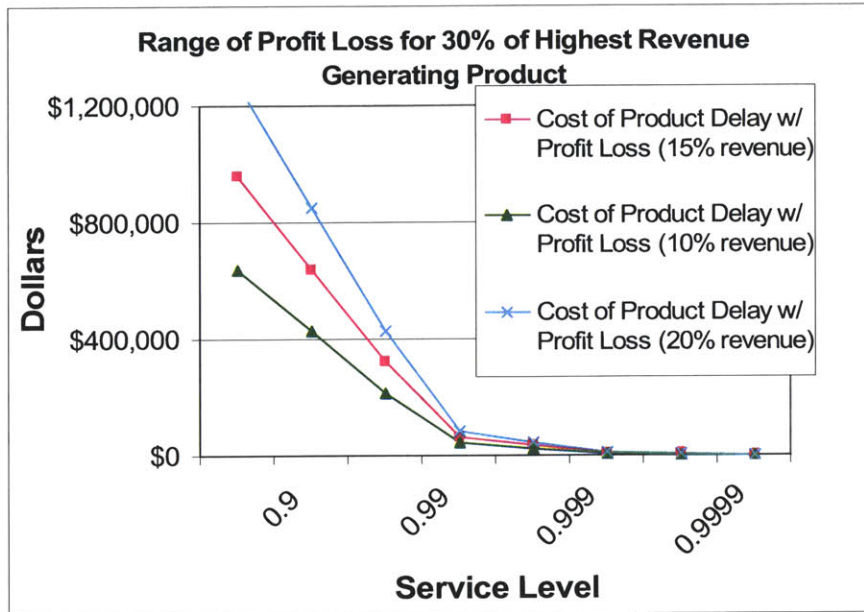


Figure 2.4.2.6.1 Profit Loss Range for 30% of Highest Revenue Generating Product

2.4.2.7 Cost of Unavailability: Summary of Profit Loss Scenarios

For each software tool, I examined 3 profit-losing cost of unavailability situations:

1. Loss of one hour of profit from Agilent's *highest* revenue generating product for each hour of license unavailability. (Figure 2.4.2.5.1)
2. Loss of one hour of profit from Agilent's *average* revenue generating product for each hour of license unavailability. (Figure 2.4.2.5.2)
3. Loss of 30% of hourly revenue from Agilent's *highest* revenue generating product for each hour of license unavailability. (Figure 2.4.2.6.1)

2.4.2.8 Graphical Summary of Cost of Unavailability

Figure 2.4.2.8.1 summarizes the cost of unavailability graphically. The lowest line is the stock out cost if there is only overtime/frustration and no revenue losses from product launch delay. The top line shows what the costs would be if there was only revenue loss and no overtime/frustration. In actuality, as explained in section 2.4.2.2.1, the expected

cost of unavailability is a combination of overtime/frustration and revenue loss up until a certain number of hours of outages per year. On this graph this transition point is shown at 4 hours of unavailability per year. When the unavailability is less than 4 hours per outage, the engineers can definitely make up lost time, and the license unavailability only causes overtime and frustration. In other words, the expected cost is the same as the overtime/frustration cost.

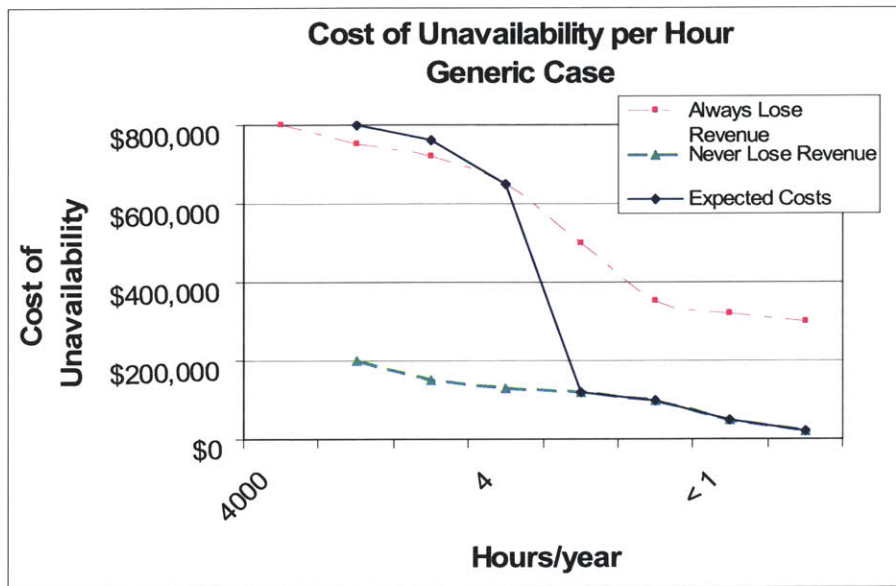


Figure 2.4.2.8.1 Cost of Unavailability: Generic Construct

2.5 Model Output: Determining Total Costs

In order to determine the optimal number of licenses, the cost of licenses is added to the cost of unavailability to calculate the total cost. Total cost is then calculated for various service levels. The goal is to operate at the service level that has the lowest total cost.

I applied this approach to four development software tools; the results of which will be discussed in detail in Chapter 3. For each of those, I determined the cost of licenses at different service levels, using 3, 6, 9 and 12 (when available) months of data. I then calculated the cost of unavailability under the three scenarios described above. Finally, I

generated for each software tool three total cost curves, representing the three different cost of unavailability assumptions. From this set of curves, for each software tool, I was able to suggest an optimal number of licenses that Agilent should buy.

Figure 2.5.1 is an example of the set of cost curves generated for each tool. It uses 6 months of license usage data and the cost of unavailability scenario is the first one in section 2.4.2.7. The graphs show costs in dollars on the y-axis and service levels and corresponding license needs on the x-axis. The graph also displays three curves. These incorporate the cost of unavailability for ranges of profit as percentages of revenue (as shown, for example, in Figure 2.4.2.5.1).

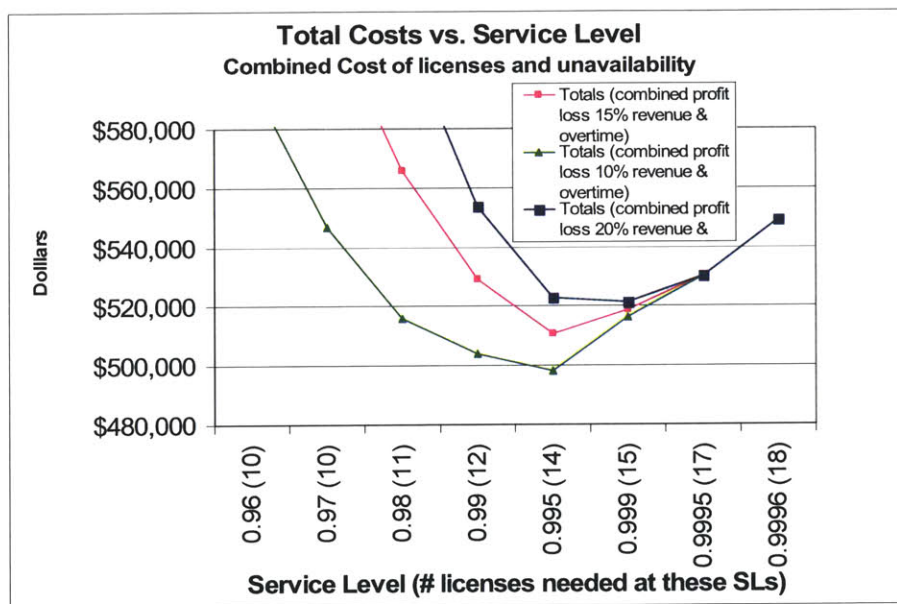


Figure 2.5.1 Total Cost vs. Service Levels

Agilent’s goal is to operate at the lowest total cost. In this case, note that the three curves have their lowest point at the same level – or close to the same level. Assuming that the potential profit loss is 10% or 15% of revenue, then the lowest point on Figure 2.5.1 shows that Agilent should operate with 14 licenses at a 99.5% service level. If Agilent assumes that it is making a 20% profit, then, according to that curve, it should operate with 15 licenses at a 99.9% service level. The absolute total dollar difference between 14

and 15 licenses on the 20% profit curve is less than \$2000. The operating manager can decide whether it's worthwhile to spend the extra money or whether the probability of being on that curve is so small that it's not worthwhile.

One other curve that examines the worst case scenario for Agilent, meaning Agilent would have to buy the most number of licenses in this case, is when license unavailability always translates into profit loss and never into overtime frustration. The final curve, with the sub-title “(HIGHEST PROFIT LOSS)” assumes that the potential profit loss is the highest (20%) and that at every service level, the cost of unavailability translates into product loss. At no point on this specific curve is overtime/frustration considered. This curve is shown in Figure 2.5.2. In this case, Agilent should operate with 15 licenses at a 99.9% service level.

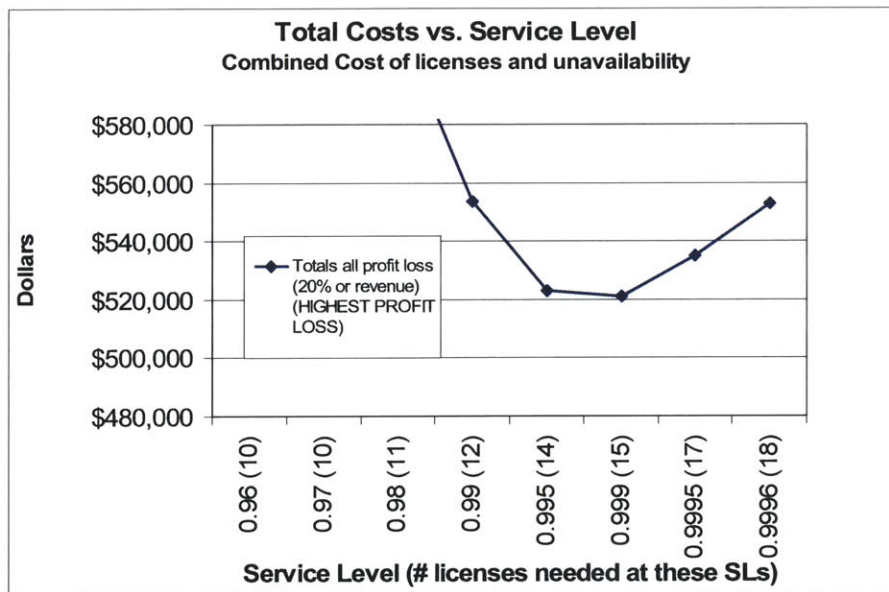


Figure 2.5.2 Total Cost vs. Service Levels

Figures 2.5.1 and 2.5.2 are total cost graphs just based on 6 months of license usage and scenario one from section 2.4.2.7. But, there are similar graphs for different periods of license usage and the two other scenarios summarized in section 2.4.2.7.

All of these different situations result in different total costs, and therefore, have different implications for Agilent. The company can use all of this information to make a business decision. For example, if the costs do not vary much between the cases, then the cost of carrying extra licenses is relatively minor. Thus, Agilent might choose to carry more licenses. On the other hand, if the cost of carrying more licenses is more significant, then Agilent would err on the side of risking a stock out and carry fewer licenses.

For example, in the case above, the difference between the lowest point in all of the cost curves is one license. Agilent might decide that it does not know which curve to operate on, so it will buy an extra license in order to cover the worst case. Therefore, by determining the cost of unavailability, and by knowing the cost of each license, Agilent now has the information to determine whether it's fiscally worthwhile to add another license to the pool or when it makes sense to take a license away. The results of the model give all of this information in order to make the best decision.

2.6 Conclusion

This chapter outlined the project and the method for applying supply chain methodology to software licenses. It also explained the terminology used in this application. The next chapter will apply this methodology to four different software products.

Chapter 3: Case Studies

3.1 Introduction

I tested the methodology described in Chapter 2, by applying it to four different software development tools that Agilent uses. The software tools that were chosen for analysis represent a variety of products that engineers utilize for development, and each provides unique insight into the applicability of the model. This chapter describes the assumptions inherent in the model. It then gives detailed results for Case Study 1 and the key learnings provided by Case Studies 2, 3, and 4.

3.2 Case Studies

I applied the methodology described in Chapter 2 to the following four products:

1. A \$25K Printed Circuit Board (PCB) design tool (Case Study 1)
2. A \$125K ASIC chip design tool (Case Study 2)
3. A \$1600 software coding bug detection tool (Case Study 3)
4. A \$11K PCB analysis tool used to do quality analysis at the end of PCB design (Case Study 4)

I chose these products because they represent a broad range of costs and are used for a range of purposes. They cover two electrical engineering products, chips and PCBs, and software engineering. In addition, the first three tools are used during the design phase, but the fourth tool is used to test a design and fix errors at the end of development.

I applied the same methodology to each software product. Therefore, the following explains Case Study 1 in detail, but only describes the differentiating factors for the other case studies. All the data and graphs in this section use hypothetical data to protect Agilent's and the software suppliers' confidentiality. However, the data and graphs still illustrate the important points of the model.

3.2.1 Case Study 1: Printed Circuit Board (PCB) Design Tool 1

Agilent engineers and designers use this PCB design tool to create layouts of printed circuit boards.

3.2.1.1 Cost of Licenses

Figure 3.2.1.1.1 displays a month of hourly license usage data for this tool. Each day should have 24 distinct data points, representing the 24 hours in a day. However, these 24 points are not shown on this graph because license usage from hour to hour might be the same. For example, on July 1, four hours during the day have 15 licenses checked out, and another four hours have 17 licenses checked out etc. Therefore, every hour is not distinct on this graph. In addition, some of the data is unavailable. (as discussed in section 2.4.1.2) The graph also shows that there is lower usage on holidays (e.g., July 4th) and weekends.

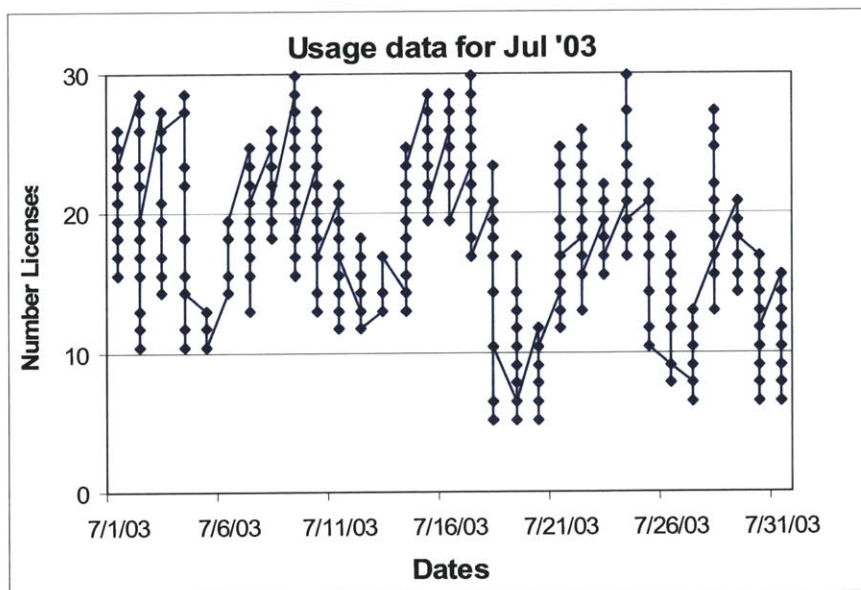


Figure 3.2.1.1.1 One month of license usage data

The usage data collected spans from July 2002 – July 2003. During this time period average hourly usage remained constant, except for higher usage from July – September 2002. Agilent management believes that the decrease in usage after September 2002 was due to the employee reductions that occurred during this time. Figure 3.2.1.1.2 shows that, as a result, the mean and variability and resulting license needs are relatively close when comparing 3, 6 and 9 months of data, but are higher for 12 months.

Period	Mean	Standard Deviation
3-months	15.9	6.7
6-months	15.2	6.9
9-months	16.0	7.0
12-months	18.0	7.8

Figure 3.2.1.1.2 Mean and Standard Deviation for Case Study 1

Even though the average license usage and standard deviation is slightly different for each time interval, the only interval that results in different license requirements is when using 12 months of data. As a result, the cost of having these added licenses is more than for 3, 6 or 9 months. Therefore, I used 12 months of data as an extreme case and 6 months of data to represent the more likely case. I also show results for 6 months of data in order to highlight the difference in costs. Furthermore, for most contracts, Agilent can renegotiate the number of licenses every 6 months, so it would be worthwhile to look at data in this time interval.

As described in Section 2.4.1, the model requires that demand data be normally distributed (or be transformable to a normal distribution.) For Case 1, demand was normally distributed as shown in Figure 3.2.1.1.3. The bars in this graph represent the number of times a given number of licenses were required during a 6-month period. The solid line represents the shape of a normal distribution.

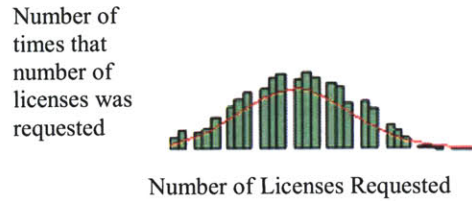


Figure 3.2.1.1.3 Fit of Normal Distribution for 6 months of usage

Given that the demand data is normally distributed, I can calculate the mean and standard deviation of the data as shown in Figure 3.2.1.1.2. Plugging the mean and standard deviation into Formula 2.4.1.1, I can calculate the number of licenses required at each service level as shown in Figure 3.2.1.1.4.

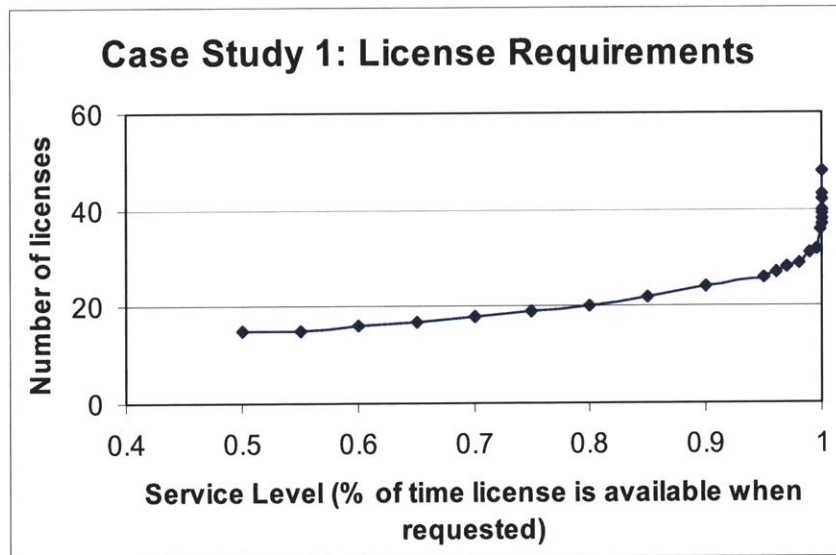


Figure 3.2.1.1.4 Licenses Required versus Service Level for 6-month Demand Data

The cost of each license is a constant (\$25,000 in this case). The curve showing number of licenses required (Figure 3.2.1.1.4) is thus easily transformed into the cost of licenses by service level curve shown in Figure 3.2.1.1.5.

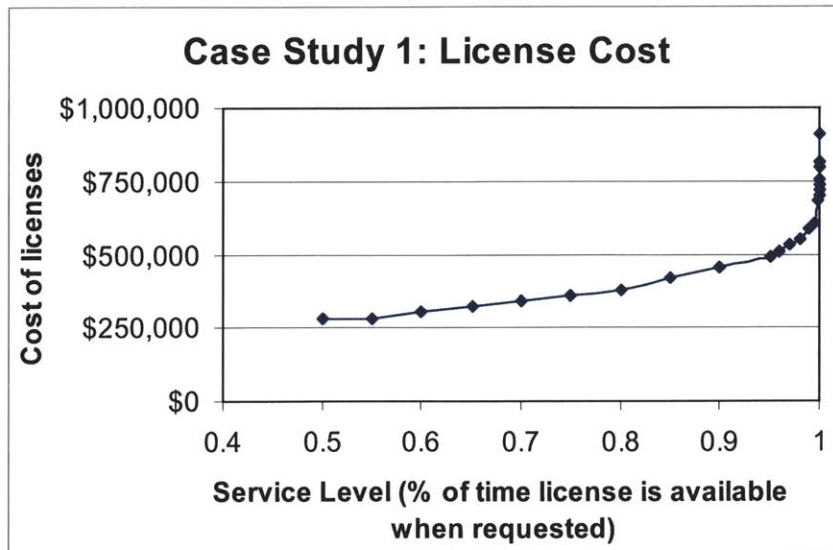


Figure 3.2.1.1.5 License Cost versus Service Level for 6-month Demand Data

The number of licenses, and thus cost of licenses, data are exactly the same for 3-month and 9-month demand data. Because average usage and usage variability were both higher for the 12-month data, there are more licenses and higher license costs required for each service level. The more variable the demand, the greater the number of licenses that will be needed to provide any given service level. As shown in Figure 3.2.1.1.6, for 3, 6 and 9 months of data, a 99.99% SL requires 40 licenses at a cost of \$757,000, but for 12 months of data, a 99.99% SL requires 46 licenses at a cost of \$871,000,

Period	Number of Licenses Required at 99.99% Service Level	Cost of Licenses
3-months	40	\$757,000
6-months	40	\$757,000
9-months	40	\$757,000
12-months	46	\$871,000

Figure 3.2.1.1.6: Number and Cost of Licenses by Length of Demand Data

3.2.1.2 Total Costs

As explained in Chapter 2, the total costs are the summation of the license costs and the costs of unavailability. The first three Total Cost graphs in this section will use 12 months of license usage and different cost of unavailability scenarios. The fourth one will use 6 months of data as a comparison.

The most expensive cost of unavailability scenario is when the engineers are working on the highest revenue generating product and each hour of unavailability results in an hour of profit loss. The following Figure 3.2.1.2.1 represents this situation for PCB Design Tool 1:

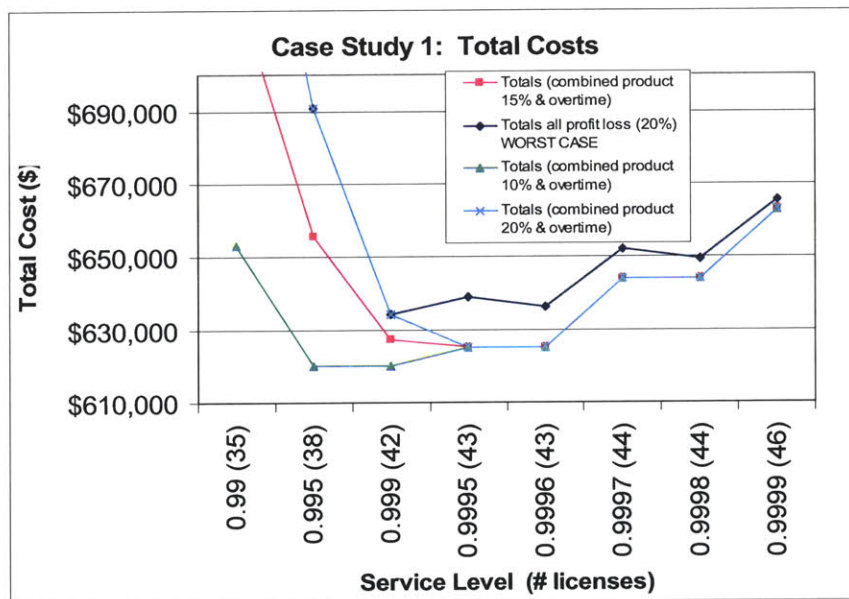


Figure 3.2.1.2.1 Total Cost vs Service Level (Cost of Unavailability based on the highest revenue generating product) using 12 –month usage data

The graph shows that different assumptions yield different results. For example, assuming a 10% profit requires 38 (99.5% SL) licenses while the assumption of 15% or 20% requires 43 (99.96% SL). Finally, assuming that at every service level license unavailability translates into a product launch delay, then the total costs are lowest at a 99.9% SL (42 licenses)

Applying the ADL assumption to the highest generating revenue product (scenario 3 in Section 2.4.2.7), while using the same period of usage 12 months for cost of licenses data, results in the following:

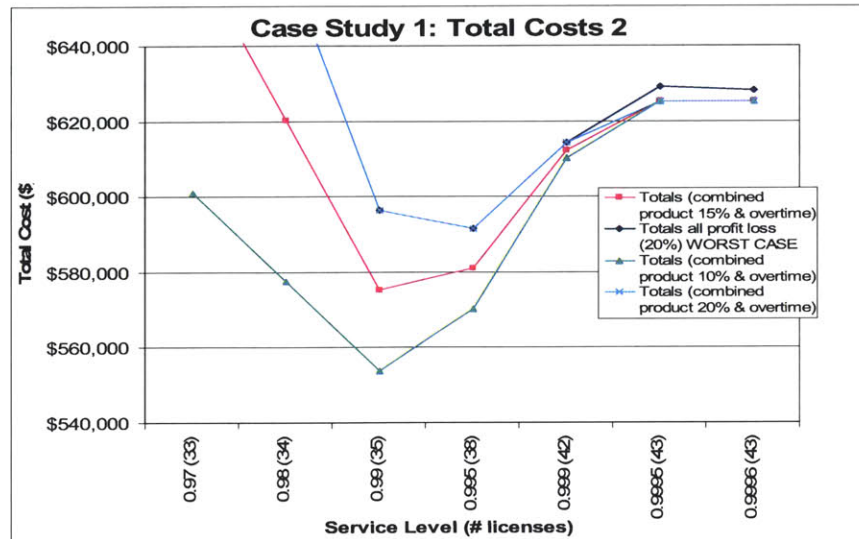


Figure 3.2.1.2.2 Total Cost vs Service Level (Cost of Unavailability: applying ADL assumption to highest revenue generating product) using 12 –month usage data

Applying the ADL assumption requires carrying fewer licenses. For example, assuming a 10% or a 15% profit requires 35 (99% SL) licenses. Assuming a 20% profit or assuming that at every service level license unavailability translates into a product launch delay, both require 38 licenses. (99.95% SL)

The two scenarios above assume that a license is unavailable for an engineer that is working on the highest revenue-generating product. The majority of engineers are working on an average revenue-generating product (scenario 2 in section 2.4.2.7). Therefore, the cost when a license is unavailable is much lower and so is the total cost, as shown in the following graph:

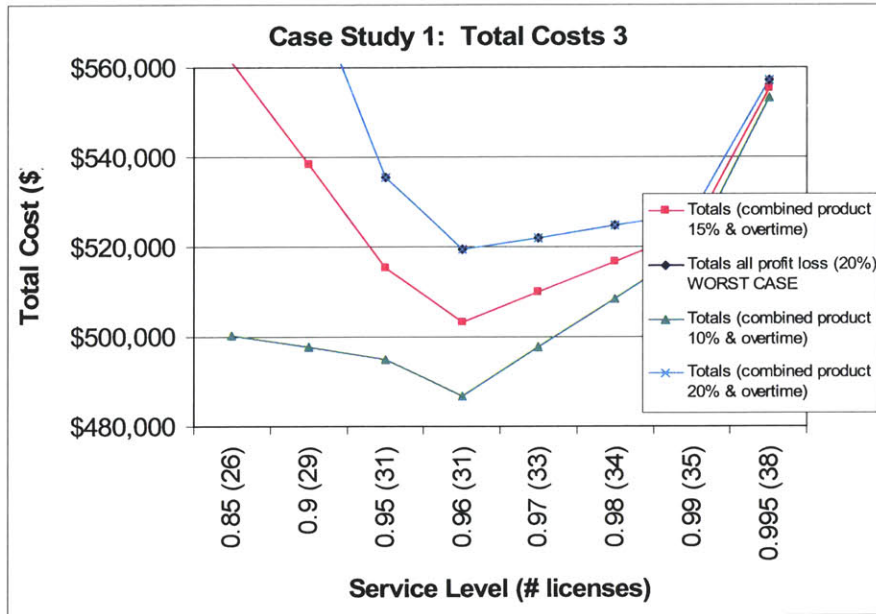


Figure 3.2.1.2.3 Total Cost vs Service Level (Cost of Unavailability based on the average revenue generating product) using 12 –month usage data

In this scenario, all assumptions require 31 licenses (96% SL). One major difference in this graph is that the service levels near the optimum point are lower than 99.95%. All the curves only include profit loss for the cost of unavailability (no overtime/frustration cost). Therefore, the “WORST CASE” lays directly over the “Totals (combined product 20% overtime)” curve; they both have the same values for the range shown here.

These three situations shown above demonstrate the effects of changing the cost of unavailability scenarios. The graphs above all use 12 months of license usage data, but for Case Study 1, there are also different results if I use 3/6/9 months of historical license usage data. The following graph uses 6 months of data (which is the same as using 3 or 9 months) and has the same cost of unavailability assumptions as described in the first case study (Figure 3.2.1.2.1):

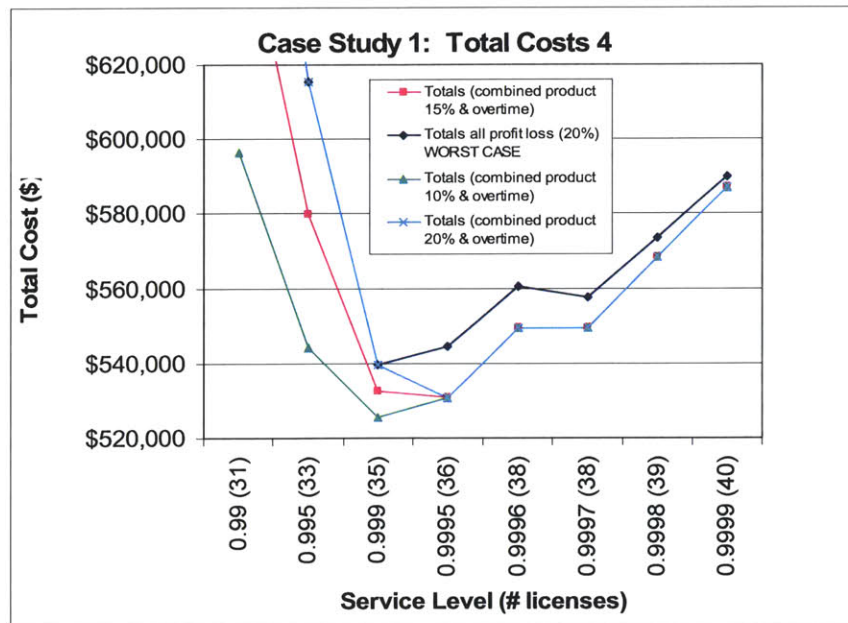


Figure 3.2.1.2.4 Total Cost vs Service Level (Cost of Unavailability based on the highest revenue generating product) using 6 –month usage data

This case, assuming a 10% profit, requires 35 (99.9% SL) licenses instead of the 38 licenses that are needed when the model used 12 months of usage data. Furthermore, assuming that at every service level, license unavailability translates into a product launch delay, then the total costs are also lowest at a 99.9% SL (35 licenses) instead of 42 licenses. The data in the above graph uses more recent historical usage, and thus, seems more indicative of the current situation. However, Agilent would have to decide if the 12 months of data is more likely to represent future usage than 3, 6 or 9 months.

3.2.1.3 Cost Implications

The graphs above illustrate that different assumptions have different implications. Agilent now has the data to make a decision based upon which scenario it thinks most represents reality and by comparing the cost for each situation.

The PCB Design Tool 1 licenses each cost \$25,000. Below is a summary comparing the cost for “Total all profit loss (20%) WORST CASE” for the four scenarios described above:

	License Needs	Cost
Total Cost 1 (Figure 3.2.1.2.1) (12 months usage data, highest revenue generating product)	42	1.05M
Total Cost 2 (Figure 3.2.1.2.2) (12 months usage data, 30% of highest revenue generating product)	38	0.95M
Total Cost 3 (Figure 3.2.1.2.3) (12 months usage data, average revenue generating product)	31	0.53M
Total Cost 3 (Figure 3.2.1.2.4) (6 months usage data, highest revenue generating product)	35	0.88M

Figure 3.2.1.2.5 Comparison of Cost of Licenses for different Total Cost assumptions

These four examples have a cost range of \$500,000. Thus, choosing which assumption to use has substantial cost implications.

Agilent currently has 49 of these licenses at a cost of \$1,225,000. Therefore, if Agilent decides to use the assumptions in Total Cost 1 and the curve for “Total All profit loss (20%) WORST CASE”, the cost savings are \$175,000 per each year of the contract.

For all the Total Cost curves presented in Figures 3.2.1.2.1 – 3.2.1.2.4, the number of licenses needed range from 31 (Figure 3.2.1.2.3 all curves) to 43 (Figure 3.2.1.2.1 20% combined profit and overtime curve) The license cost for this range is \$775,000 to \$1,075,000. Therefore, Agilent can save between \$150,000 and \$450,000. For Case Study 1, it is clear that Agilent buys more licenses than it needs.

3.2.2 Cost Implications for Case Studies 2-4

As mentioned earlier, the same methodology was used for the other three case studies. Applying the same assumptions as used for Total Cost 1 (Figure 3.2.1.2.1) and looking at the “Total all profit loss (20%) WORST CASE”, the results of Case Study 2 show that Agilent needs 25 licenses, but has 38 licenses, for a total savings of \$1.6M. For Case Study 3, Agilent needs 51, but has 52, for a savings of \$1600. The results of Case Study 4 are inconclusive (as will be discussed in section 3.3.2.2), but it seems that Agilent does not need to reduce licenses for this software product.

3.3 Key Learning from Case Studies 2, 3, 4

Case Study 1 served as a template for the other three case studies. I chose the other case studies to serve as a contrast to Case Study 1 and provide further insight into the model. There are three major points that were learned by applying the model to these case studies:

1. The model shows that there is an insignificant difference in usage data among the different time intervals (i.e. 3, 6, 9 or 12 months of data)
2. The model provides limited insight for usage data that is not normal or transformable to normal
3. The model aligns best with contracts in which Agilent determines how many licenses to purchase and pays for that number, regardless of use.

3.3.1 License usage time range

Before creating the model, the intuitive approach led me to look at different time intervals, because typically different time lengths have wider changes in demand and increased variability. However, this did not prove true. As described in Case Study 1, historical license usage data resulted in the same cost of license curves when using 3, 6, or 9 months of data. Furthermore, for the other 3 case studies, there was no difference in average hourly usage when comparing 3, 6, 9 or 12 months of data.

Although during the year that I extracted the data Agilent had large employee reductions, overall, the usage data was still not sensitive to time, or, except in case study 1, to fluctuations in employee headcount. (See further discussion in Chapter 4) Thus, at this point, because the license usage data is difficult to obtain, in order to save time, the employee responsible for collecting the data could only extract three months of data. (This time interval proved to be the same as collecting usage data for 6 or 9 or 12 months for three out of four case studies) Furthermore, Agilent can only change the number of licenses it carries every 4 to 6 months. Therefore, it is not worthwhile to recalculate cost of licenses versus service level more than every 6 months.

3.3.2 The normal distribution assumption

As discussed in section 2.4.1, the model applies best when the data is normally distributed or transformable to normal. Case Study 1 was the only product that had normally distributed usage data, which made the calculations straightforward. The usage data for Case Studies 2 and 3 was right-tailed as shown in Figure 3.3.2.2.1.



Figure 3.3.2.2.1 Example of distribution for Case Studies 2 and 3

The solid line, which is the shape of a normal curve, shows that the normal distribution inaccurately fits the usage data. Specifically, for Case Studies 2 and 3 the skew and kurtosis values were not close to zero.

However, for Case Studies 2 and 3, it was possible to transform the data to normal, calculate the cost of licenses at different service levels and then transform the results back to the original scale (as described in Section 2.4.1). The picture serves as a visual check.

After applying the optimal exponent to the original usage data, the transformed data distribution looked like:

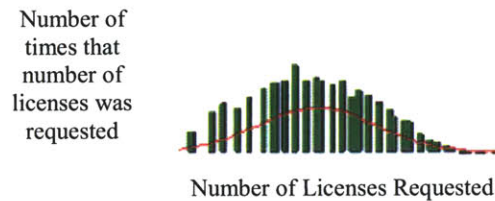


Figure 3.3.2.2.2 Example of transformed distribution for Case Studies 2 and 3

The usage data for Case Study 4, on the other hand, was not normally distributed and cannot be transformed to normal. This is the case because Agilent only owns five licenses of the software product. In addition, the engineers only use the licenses at the end of each PCB design, and therefore, the usage pattern is cyclical. There are certain months when only 1 or 2 licenses are used and other months when all 5 are used. Therefore, any assumption of a continuous distribution is tenuous. The raw data does appear to be similar to a Poisson distribution, so I compared the Poisson probabilities to the ones observed in the usage data.

Due to the Poisson distribution, I could not calculate the service level for any number of licenses, but I could determine that Agilent currently has enough licenses to operate at a 99% service level. The distribution also made it impossible to calculate total costs curves.

The analysis for Case Study 4 revealed that both the usage pattern and the number of licenses contribute to the resulting distribution. It is not clear how many licenses are needed in order to assume a normal or transformable to normal distribution. However, as a rule of thumb, Agilent should not apply the model to software products that have less than 10 licenses.

3.3.3 Contract implications

The contract between Agilent and the software supplier influences the applicability of the model. The model was built to directly correlate with the contract terms in Case Study 1 (which has the same terms as Case Study 4). For Case Study 1, Agilent estimates the number of licenses needed and then pays for each one over the period of the contract. As explained earlier, unused licenses cost Agilent money. Therefore, the model optimized the license needs and, in this case, determined that Agilent has too many licenses. At the next opportunity, Agilent can stop spending extra money for licenses it does not need.

Case Study 2 has similar contract terms to Case Study 1. As mentioned in section 3.2.2, following the same reasoning as in Case Study 1, Agilent has 13 extra licenses (\$1.6M). However, there is one slight modification in the contract for Case Study 2. The provider of this software gave Agilent 13 extra licenses for a period of time as a bonus for the business and in hopes that Agilent would end up paying for these licenses after the free-trial period ended. Therefore, according to the model, at this time, Agilent is really not paying for extra licenses. The results do show Agilent that after the free-trial period, if the business is the same; Agilent does not need to buy extra licenses.

The software code bug detection tool, analyzed in Case Study 3, has different contract terms than the ones for Case Study 1 and 2. In this case, Agilent pays for the peak number of licenses used over the last three month period. Once again in section 3.2.2, it states that if the contract were the same as Case Study 1, then Agilent would be paying for one extra license at a cost of \$1600.

However, Agilent paid for peak licenses over the three month period, and the peak usage was only 35 licenses. The model shows that for this software product, the terms of the license agreement save Agilent money. If Agilent paid for 51 licenses, the optimum number, instead of 35, the peak usage, it would pay an extra \$26,000 per year over the length of the contract.

It seems that Agilent receives a good deal for Case Study 3. If peak usage is less than the optimal number of licenses, then Agilent might want to renegotiate contracts for these

cases. For Case Study 2, the software supplier convinced Agilent that it receives a bonus during the free-trial period. In actuality, Agilent does not need the free extra licenses. Overall, the model provides useful insight even if the contract terms differ from Case Studies 1 and 4.

3.4 Conclusion

This chapter provided the results of the model for the four case studies. It showed that different assumptions are worth examining because they produce a large range of cost implications. (ex: Figure 3.2.1.2.5) It also explained that the results do not change by modifying the time interval, that the model relies on using normally-distributed or transformable-to-normal data, and that the contract terms that require holding inventory provide the most insight. The next chapter will examine the future use of the model.

Chapter 4: Future Use of Models and Benchmarking

4.1 Introduction

As mentioned earlier, the model does not include forecasting in order to determine future license usage. This chapter will briefly describe potential usage indicators and methods for forecasting. It will also provide examples of processes that other companies use to manage their central licensing.

4.2 Forecasting

There are different events that can affect software license usage such as a change in the number of engineers, an increase or decrease in new product introductions (NPI), the degree of complexity of the new products, or a shift in market segment focus. Ideally, Agilent could correlate the impact of each of these on license usage and create a forecast model based on this. The difficulty lies in collecting the appropriate data.

For example, retrieving employment data seemed relatively easy and AES management believed it would have the strongest correlation to license usage.¹⁸ The total engineering headcount and some information on the number of engineers in each discipline was available. However, correlating the correct employees to the correct software tools posed a problem. Some employees fall under the heading of electrical engineering but use the mechanical engineering software tools. In fact, there are about 500 employees categorized as mechanical engineers, but there are more than 1000 users of the mechanical engineering software.¹⁹

¹⁸ The results from the case studies presented in Chapter 3, showed that even though there were more than 15% reductions in headcount during the data collection period, it did not affect the license usage. The exception is for Case Study 1, in which the results using 12 months of data differed from 3, 6, and 9 months of historical data. Overall, employee numbers might correlate, but it might take larger than 15% changes to affect the results of the optimization model. In any case, the point of section 4.2 is not to say that these correlations definitely exist, but that these are some of the components that might be correlated to data usage. Furthermore, this section is explaining some of the challenges in building a statistical model.

¹⁹ Keeley, Ron, Centralized Representative for Mechanical Engineering at Agilent. E-mail dated 8-19-03.

The other challenge is that, in the best case, the license usage data only exists for the last 18 months. So, even if the engineering data could be categorized appropriately, it is difficult to see trends over a short period. Furthermore, the headcount data is rolled up by month, but the usage data for most software fluctuates greatly over a month. An average for the month would not truly represent the data. Another approach could be to correlate engineer headcount with maximum usage in a month or another more representative statistic.

Even though the above demonstrates the challenges of forecasting based on statistical correlations, it would not be an impossible task. Agilent could create a process to ensure that the correct data is collected so that correlation can be determined. Once the relationships between license usage and different events have been established, Agilent could monitor the relevant indicators. When significant changes in the trigger points occur, AES can run the optimization model, developed for this project, with the future usage data and determine license needs. Forecasting models would serve as a powerful complement to this optimization model.

4.3 Review Process

In the absence of formal forecasting models, AES can still establish a schedule for periodically updating the model. This schedule can be based on known factors. For example, some software contracts have stipulations on when Agilent can remix the types of licenses it owns. Usually, the contract allows Agilent to change the license mix every quarter or every six months. Therefore, if Agilent cannot make any changes until the established time has passed, it is not worthwhile to optimize the model on a more frequent basis.

Furthermore, as discussed in section 3.3.2.1, even though there were employee reductions and business changes during the last year, there were not large changes in the usage data. Only Case Study 1 had different results when analyzing 12 months of usage data instead of 3, 6 or 9 months. Additionally, none of the case studies showed changes when

comparing 3 months to 6 months of usage data. Given that usage patterns were consistent across the sampled time intervals, it still seems reasonable for AES not to run the optimization model more than every 6 months. This cycle also fits in with the contract conditions.

4.4 Benchmarking

Other companies have also struggled with forecasting and determining license needs. As part of the project, I discussed methods that other companies use. I met with members of the Central Enterprise Licensing User Group that work for Intel, Honeywell and Motorola. None of these companies developed processes based on cost trade-offs, but they did have other methods that provide learning.

Honeywell has a program called Centralized Engineering Application Licenses (CEAL) to manage software licenses across the corporation. One metric this group tracks is license utilization, which Honeywell defines as used divided by used plus queued. (Honeywell has a system set up so that if a license is not available when requested, the engineer can put his/her name in a queue and will be given a license when it is available)

Honeywell management established a goal that license usage should be at 98% utilization. The company tracks this metric on a monthly basis and buys enough licenses to meet this goal. According to a negotiator in the CEAL group, this goal was set arbitrarily.²⁰

Motorola also set an arbitrary goal of ensuring that users do not encounter denials on more than 9% of the period days.²¹ For example, if during a three month period license usage for a software product hit the peak number on eleven days (equals approximately 12%), Motorola would investigate whether it needed to buy more licenses. Employees

²⁰ Stanley, Rachel, License Administrator Specialist. Honeywell. E-mail dated 8-19-03.

²¹ Griffith, Dan, Manager, Comprehensive Software Asset Management at Motorola. E-mail dated 12-2-03.

on the Comprehensive Software Asset Management (CSAM) team would look at this peak usage, and would base the decision to buy more on trends such as whether the peak usage all occurred recently or all happened in the first month.

They would also check if there are opportunities to improve utilization of the available licenses. In order to do this, Motorola looks at metrics such as the global locations where usage occurred. If there were times of low peak usage, Motorola would also evaluate whether engineering jobs could be scheduled during these times.

Overall, Motorola has a strategy to collect multiple metrics in order to plan for the future and manage software licenses. Each month, CSAM reviews peak usage, effective peak usage, and usage by locations. The CSAM group also meets with the users in order to gauge engineering direction. In addition, the group makes this data available to the software users so that they can make decisions based on metrics.

Intel also looks at past data to determine license needs, but Intel stressed the formal review process that it had with the user community. Most companies meet with the engineering management to ask about future usage, but Intel actually has the managers create documented forecasts. Putting the numbers in writing creates accountability and forces management to thoughtfully determine future needs.

This significant management involvement in the license forecasting process makes the engineers more willing to accept license unavailability. None of the companies intentionally want to delay a project based on license unavailability, but Intel seemed more willing to have temporary outages than did Agilent, Honeywell, or Motorola. Denials seemed to be more of an accepted part of the system than at other companies.

The attitudes of the companies showed that the license strategy is not just based on numbers, but is also a result of the organizational environment and culture. All the companies that I met with had a technology focus and wanted to use software tools to enhance the innovative process. The companies did want to manage the license program

and reduce costs; however, due to the importance of continual technical achievement, cost was certainly mentioned less than availability.

Each company also had specific processes for the licensing program that coincided with the company strategy. For example, Motorola's focus on metrics fits right in with the company's adherence to the principles of Six Sigma. As mentioned earlier, Intel seemed a little bit more accepting of license outages. Although Intel also relies on technical advancement, its core competency is manufacturing. Thus, the company might cater more to the employees in manufacturing than to the ones in the design community.

It is clear that many companies are struggling with similar issues. In addition, companies focus on different issues depending on such aspects as the organizational environment. There is much that each company can learn from other companies' approaches.

4.5 Conclusion

This chapter showed the challenges and benefits of eventually creating a formal forecasting method. In the meanwhile, a review process can be initiated to have a regular update of the model. The chapter also described aspects of Honeywell's, Motorola's and Intel's software licensing programs. The next chapter will examine the impact of Agilent's organizational environment on the project.

Chapter 5: The Organizational Environment and its Impact

5.1 Introduction

As described in Chapter 4, a company's strategy and culture influence its software licensing program. This can be shown in more detail by outlining the organizational situation at Agilent and explaining its impacts on the development of this project and on the future use of the model's results. During my internship, I was able to assess how the economic environment, the structure of the organization, the political power and the culture all affected the project. This chapter will describe these observations as well as reflections on the use of the tools described in the Sloan Leadership Model.

5.2 Economic Climate

By the end of the summer of 2003 (the start of this internship), Agilent had endured nine consecutive unprofitable quarters. Consequently, the number one objective of the company was to return to profitability. This involved difficult measures such as major reductions in traveling and lay-offs. Up until 2001, Agilent employees had never experienced lay-offs. When the CEO, Ned Barnholdt, announced the first round to employees over the PA system, he said, "This is the toughest decision of my career, but we've run out of alternatives." The first two rounds cut more than 8000 employees, and to date, through this program, the workforce has been reduced by more than 30%.

Agilent's economic situation and associated actions had a negative impact on morale. Although the employees knew lay-offs were necessary and even supported them, it was still difficult to remain motivated in this environment. Furthermore, employees felt sympathy towards ones who had lost their jobs and also felt uncertainty towards their own future careers.

This economic climate led to creating projects in order to examine the necessity of current spending. The idea to analyze software expenditures by applying supply chain

methodology was one of these projects. At the onset of the project, AES assumed the current approach adequately determined license needs. However, the group thought that a quantitative method could improve the accuracy of license purchases. It was possible that the results would show that Agilent actually needed more licenses, but at least this conclusion would have been determined by a more rigorous analysis than the current method. In the end, the results proved that Agilent can actually reduce the number of licenses, which aligns well with the current economic pressures.

Although Agilent's effort to become profitable might create necessary and cost effective projects, the current economic situation does pose a challenge to implementation. For example, the optimization models developed for this project will require updates and will be more beneficial if they are applied to additional software products. However, with the employee reductions, people are feeling strained to sustain existing programs, let alone take on new ones. Management will need to help employees establish appropriate priorities during these challenging times.

5.3 Culture: Collegial and Loyal

As mentioned above, Agilent had never imposed lay-offs since its creation (Originally, Agilent was part of Hewlett-Packard, which did not have layoffs either.) Many employees of HP and Agilent believed they had a non-verbal contract with the company for employment for life.²² This created a culture of loyalty and trust; it also allowed people to work together for long periods of time and develop friendships.

Employees remain at Agilent for large portions of their career, and people are proud of their employment length. I had many conversations that started with comments such as "10 years ago..." or "I have only been here 10 years, so I don't know as much as he does..." A long length of employment results in automatic credibility.

²² Gibbons, Dr. Robert, Sloan School at Massachusetts Institute of Technology. Course: Strategy and Organization, Spring 2003.

Due to this substantial period of employment at Agilent, deep relationships and personal networks develop. Consequently, as a new intern, I had to tap into others' networks to get needed information. Furthermore, people spent time protecting the co-workers in their network. Co-workers cautioned before giving me names by saying, "That person is really busy..." or "I'm not sure whether I should put you in touch with him..." and even, slightly jokingly, "Don't tell him that I gave you his name." I believe that some of these excuses were because people are actually busy, but they are also because employees value their networks and are more successful because of them. Therefore, they do not want to jeopardize their relationships.

Another symbol of this relationship-laden culture was the guidance I always received to contact people by phone first instead of by e-mail. One co-worker said, "Call them first, because then they'll warm up to you". Agilent, like many large technology companies, has employees spread throughout the world. Even so, they make an effort to develop relationships at least through the phone. Employees perceived e-mail as too impersonal for initial contact.

This dispersed employee base and the desire to maintain relationships caused Agilent employees to establish unique venues to promote more intimate interactions. At a staff meeting when employees were asked to give updates on their work, some would make the verbal report and then share a personal anecdote. They would tell stories about a recent illness or a new baby. With the ability to share media through the web, one co-worker even showed a picture from her last vacation and thanked people for covering for her while she was away.

Another venue was a creative holiday party. The current economic climate restricted travel, but AES employees still wanted to have the party. So, they created a virtual one. Employees sat at their own work location, dialed into a conference call and talked while they drank their favorite beverage.

These strong relationships and the maturity of the workforce also produced a high-trust environment. This made managing a global group easier, especially without the ability to travel. In addition, management had no problem with telecommuting. More than half the AES team worked from home all of the time or sporadically. One manager said to me, “All that matters is that the work gets done.”

I also noticed that the managers in AES trusted their employees to make decisions. For example, the software license negotiators were responsible for major contracts. Their manager let them conduct all the negotiations, determine license needs, and secure the contracts. The manager of this group told me that the negotiators were so knowledgeable, she would only get involved if they needed help and to reduce any bureaucratic obstacles.

For the most part, these already-established networks and global employee base did not negatively impact my ability to get the information needed. However, it did make it more challenging. For example, when I interviewed engineers about the implications of license unavailability, half refused to meet with me or would accept a meeting and not attend. When I asked co-workers to send out a note to their contacts to say that it is worthwhile to meet with me, more engineers were willing. This was less efficient, but it seemed to be the required method.

New employees do not have the built-in respect from length of employment nor the personal networks, so they have to learn how to work through other people’s networks. It could become frustrating if employees feel that they have to put in their time before they really can advance or be respected. On the other hand, this collegial environment produced positive results and made Agilent a great place to work.

5.4 Politics: The Powerful Development Engineers

As mentioned earlier, Agilent’s mission statement is, “to provide solutions and technologies that revolutionize the way people live and work.” Agilent also lists

innovation as one of its values.²³ To accomplish these goals, Agilent relies on its development engineers. Thus, it is no surprise that these engineers hold a lot of power within the company.

This power structure was obvious even from the half-day plant tour that my classmates and I attended six months before this internship. During the tour, we heard a talk from an employee who managed supply chain programs at Agilent. He talked about the challenges of reducing inventory and the number of different supplies in each product. He said that many products even had customized screws; this was taking the value of innovation to an extreme. When I jokingly asked whether the company emphasized technological achievements so much that it almost did not care whether it could actually make money on the product, he smiled and nodded his head affirmatively.

Of course, the economic climate highlighted the need for more focus on operations and costs, but this employee still described programs in this area as a struggle. He also said that employees with MBAs received less respect than employees with PhDs. Whether this was just his perception or truth, these types of comments definitely highlighted the tension between business and engineering.

This tension showed itself when I was interviewing employees to determine the cost of unavailability for my model. There was a clear difference in opinion between managers and engineers involved in finance versus ones in development. Employees in finance would explain how it is not possible for license unavailability to cause product delays. They told me that there are so many events that could cause delays, such as technological obstacles, manufacturing challenges, or parts not being available. Licenses would have to be unavailable for weeks for it to affect a product launch. Otherwise, engineers could definitely make up the time. Even after I interviewed more than 30 people in different areas of the company, a co-worker, who had a Finance background, told me that he could not believe the model because the cost of unavailability could not be equated with lost revenue due to a product launch delay.

²³ Value Statement from www.agilent.com

On the other hand, managers and engineers in the development groups would tell me that even an hour of license unavailability could jeopardize the launch date. One manager said that license unavailability is totally unacceptable; engineers must have these software tools at all times. These comments sharply contrasted with the ones from the finance employees, and I had to assess all of these opinions when making final determinations on cost of unavailability.

This power struggle created other challenges during the project. It quickly became apparent that there is a power structure even within the engineering ranks, and the top position is held by the electrical engineers (EEs). When I was setting up interviews, the mechanical (MEs) and software engineers (SWEs) were more willing to meet than the EEs. At first, I thought this was due to the relationship between my contacts who introduced me to the MEs and the SWEs. However, my conduit to the EEs had just as good a relationship.

One of the my co-workers said, "Of course the MEs are more willing to meet you and are nicer, because MEs were originally EEs who couldn't make it and so they're just regular guys." I hardly agree, but I realize that this represents some of the attitudes of people in the company. And, many of the EEs believed this about themselves. One in particular, answered the phone when I called and seemed interested in contributing his ideas to my project. About two minutes into the conversation, he started yelling at me that he did not have the time to talk about software licenses; he has more important things to do. Then he hung up on me.

The attitude was shocking. I interpret it as being a result of the superiority complex that the company fosters for EEs, but also due to the added stress from the economic climate and workforce reductions. When I relayed the story to my co-workers, most found it unacceptable, and somewhat surprising. However, they did concur that EEs have the most power, because they are responsible for many of Agilent's products.

This acceptance of the engineer as king will definitely impact the potential implementation of the results of the model. Two of the case studies showed that Agilent has more licenses than needed. But, reducing the licenses would mean that there would be more license unavailability. Engineers might have to wait for licenses or rearrange their work hours. This would inconvenience the employees that have the political power.

And, this political power does not just exist because of pure politics; Agilent relies on the engineers for technical innovation. Furthermore, as pointed out by the manager of Quality and Engineering Services (the group that AES reports to), creative ideas do not happen on demand. He rhetorically asked, “what if an engineer thinks of something and the license is unavailable, and s/he cannot get in the same mindset when it does become available?”

If AES does decide to implement the methodology described in this thesis, it should gain buy-in from the engineering community before implementation. I do think that the business groups are looking for opportunities to save money, and this might be a way to accomplish this goal. However, AES treats these groups as customers, and in order to maintain a good relationship, ensuring (and believing) that the benefits of license reduction are worthwhile is imperative.

5.5 Structure: Decentralized Groups

The organizational structure at Agilent facilitates the innovative process. Agilent has decentralized groups that are each responsible for their own profit and loss. Decentralization fosters speed and flexibility, so groups can move faster with technological developments. On the other hand, it does reduce opportunities for shared learning and copying best-practiced methods across organizations. Agilent deals with both of these components of decentralization, but its goals are output and ingenuity, not collaboration. Therefore, a decentralized structure is the best way to accomplish this.

The McKinsey Quarterly²⁴ explained that decentralization not only creates a faster-moving organization, but it also allows for “atomization”. Atomization helps diverse businesses with different products and in various industries to identify and pursue opportunities in a unique way. In order to gain the benefits of decentralization, a company requires a certain culture, talent, and long-term commitment. Agilent has committed itself to this organizational structure and aligns the culture, goals and politics appropriately.

Decentralization at Agilent allows each group to operate independently and to decide on which technology and products to pursue. This structure creates autonomous groups, and I believe that this independence could even be seen at the individual level. For instance, as described in the sections 5.3 and 5.4, the development engineers and the program managers are given a lot of responsibility and can make important decisions. Workers in centralized organizations, on the other hand, often have to receive multiple layers of approval before an idea is accepted. Agilent has a focus on innovation, so this structure empowers the individual’s creative process.

On the other hand, Agilent does realize that there are some processes that should be managed centrally. Therefore, as described in Section 1.4, Agilent has centralized support groups, such as Finance, IT, and AES that make up the Global Infrastructure (GI). These groups report to the Chief Operations Officer.

Centralizing these groups reduces redundant efforts and saves Agilent money. Coordinating services, according to a 2001 Andersen/akris.com study, return a payback in 1-2 years for 29% studied and in 3-4 years for 77% of those interviewed.²⁵ The study also showed that 34% of the companies in the study had a 20-40% headcount savings.²⁶

²⁴ Barnett, F. William, Jr., Berland, Terrace P. “Strategic Thinking on the Front Lines”, *The McKinsey Quarterly*, Spring 1999.

²⁵ McReynolds, Scott; O’Brien, Brian, “Earning Pressures Boost Shared Services”, *Financial Executives* Jan-Feb 2002 v18 i1.

²⁶ Ibid.

However, centralized groups within a decentralized company encounter challenges. For example, the members of the centralized groups try to coordinate efforts of employees within different business groups. The employees in the business groups are accustomed to their independence; they also do not have the incentive to compromise for people in another business group because they are mostly judged on only their own group's performance.

As a result, the employees in the GI groups need to utilize their sphere of influence and networks when convincing people to coordinate for the good of the whole company. Originally, AES had to use these skills when creating the centralized licensing program. Managers and engineers in the business group did not want to share licenses across the organization, because then they would have less control and possibly have unavailability when they needed a license. However, AES ensured that it would not compromise service and that the cost savings were too tremendous not to pursue a centralized approach.

My interviews confirmed that AES delivered on its promise. The engineers said that the centralized program actually reduced licenses, because the infrastructure was better. It also eliminated the need for individuals in the group to negotiate contracts, which saved the group time. And, most importantly, it saved more than 30% of the total license costs.

The promises made at the onset of the program were a necessary action of a centralized group trying to receive buy-in from decentralized organizations. These promises, however, make it unpopular for AES to go back to these groups to tell them that the service level might actually be too high. As described in Section 5.4, I even found that talking about potential license unavailability often was not received well. Groups do threaten that if the service declines, they can always go back to managing licenses for themselves. This might be a local optimum, but is certainly not a global one.

At this point it is probably too expensive for each group to replicate the central function, but the threat mentioned above puts the onus on AES to have a strong sales pitch in order

to make the groups follow the results of the model. AES might want to sell the program by mentioning that for some products Agilent has more licenses than it will ever use. The amount of waste might resonate with the engineers. AES can also propose that it will pass on the cost savings to the groups' bottom-line as it did when it initiated central licensing.

5.6 Inventing: the common denominator

As I observed on the plant tour, Agilent thrives and succeeds because of great technical inventions. Its organizational processes foster this goal: the loyal and collegial culture empowers people to take responsibility and invent, the power lies with the engineers who create future products, and finally, a decentralized organizational structure increases speed and allows for autonomous organizations to make the best decisions about which future technology path to pursue.

5.7 Interning in this organization

The lenses of culture, politics and structure, used above, provide one way to analyze an organization, and the Sloan Leadership Model gives another. The Sloan Leadership Model provides a framework to achieve change within an organization. This model requires one to use the skills of Sensemaking, Relating, Visioning and Inventing.²⁷ These are described as the following: Sensemaking: learning about an organization, the business climate and culture; Relating: building relationships; Visioning: creating a plan and getting support; and Inventing: change and innovation. The following will describe examples of employing these steps.

As an intern, working at Agilent for six months, I had to immediately use my *sensemaking* skills to become aware of the culture, politics and structure in order to accomplish my project objectives. For example, I quickly learned about the people

²⁷ Ancona, Deborah, Malone, Thomas, Senge, Peter, and Wanda Orlikowski, "The Sloan Leadership Model," Draft manuscript, Sloan School of Management, 2003.

networks, and realized that these were critical to tap into in order to obtain the information needed.

I next had to use my *relating* skills to find co-workers that could serve as allies and help connect me to others. I have a natural inclination to create relationships with co-workers and to utilize these to accomplish my work. Because of Agilent's collegial culture, I made an extra effort to meet with as many co-workers as possible face to face. For example, in the second week I drove to an office two hours away to meet with key customers and stakeholders.

Due to the timeline of my internship, I also developed relationships by offering employees something in return. Many co-workers bought-in just because they were curious to see what my analysis would offer. I also realized that my co-workers believed in the AES project management program, and my personal need to organize and plan fit right in with the objectives of the PM program. If I followed the program, which would just be utilizing a strength that I already had, people would be impressed with this effort, automatically learn about my project through the PM communication process, and be more willing to help me. I would then gain their respect, and they would feel comfortable connecting me with their networks. This plan worked.

These *relating* skills also made me aware early on in the project that there were some co-workers and people within the business groups who did not support my project. Even though I have a tendency to try to please as many as possible, I decided that in order to finish my project in six months, I could not satisfy everyone. However, I did not want to completely ignore the non-supporters. Instead of relying on my natural tendencies to align all parties from the beginning, I *invented* a different approach. I went back to some of the non-supporters midway through the project to review the results to date. Some of them then realized that even if they did not agree with all the components of the project, they could still find something useful in one of the parts.

This strategy proved successful. I did not get waylaid from the beginning or become frustrated because I could not please everyone. On the other hand, mid-way through the internship, most of the key stakeholders believed the project would deliver valuable information.

The above example did not rely on my natural talents, but the one described below did utilize my strengths as well as the idea of visioning from the leadership model. As mentioned earlier, I operate best with a plan in mind and seeing the end objective. Less than two months into the project, I presented to my stakeholders graphical representations, with hypothetical data, of the *envisioned* results of the project. Therefore, very early on in the project the key players knew what information the model would give them. Furthermore, the framework was now established, and I could spend the rest of my time filling in the details of the model.

The above gave examples of the ways I used the insights gained from the Sloan Leadership Model to approach the project. The process is not always linear and requires going back and forth between the different steps in order to have the best results.

5.8 Reflection

Throughout the project, I received positive feedback on the work I was doing. At the end of the internship, I had completed all of the objectives that I described in the initial project plan. Furthermore, my model had produced results that could save Agilent money. AES was satisfied with the information received and looked forward to implementing the recommendations.

However, the actual implementation will be mostly determined by the organizational environment at Agilent. The model results, which show that Agilent should reduce licenses, contrast with the focus on technology, and the resulting culture, politics and structure. The analysis in this chapter demonstrates that although a quantitative

perspective is useful, the creation of it and the implementation completely relies on people and working within an organization.

5.9 Conclusion

The organizational lenses and the Sloan Leadership Model provide a framework to analyze a company. These insights help with the implementation of a project such as the one completed for this thesis. I found these tools particularly useful because my timeline was short and I needed the information gained from them to be successful. The next chapter will give final recommendations and lessons learned.

Chapter 6: Conclusion

6.1 Introduction

This thesis explained the supply chain model that was developed for software licenses and showed the results of four case studies. It also discussed the organizational behavior and the impact of this on the project. This chapter will review the model and give recommendations for next steps. It will then outline take-away lessons from the overall project.

6.2 Summary of the model

Applying supply chain principles of service level, inventory and stock outs to software licenses provide a quantitative method for determining license needs. The methodology analyzes the cost trade-offs of holding inventory versus having license unavailability. It then allows the user to identify the optimal level of service.

The model has a few key characteristics. It works best in situations in which the data is normally distributed or transformable to normal and the software contract involves paying for a certain number of licenses regardless of actual use. It assumes that license denials translate into overtime/frustration and/or profit loss due to product launch delays, depending on the hours of unavailability. The model can still be applied without the assumption or scenario described above, but the results are less powerful.

In order to show the cost implications of various scenarios, I tested the model using different assumptions. For example, I assumed a range of potential profit losses for license unavailability. By doing so, management will be able to decide which assumptions are the most relevant and use the results of that scenario. The cost implications varied largely between the situations, but overall, they all showed that potential exists for significant cost savings.

6.3 Recommendations

Agilent now has the results of applying the model to four of its software products. As discussed, this information provides useful insight, and, if applied, can save Agilent money. Furthermore, Agilent has options to gain further benefit from the model. Below are a couple of suggestions on how to proceed. The recommendations are not exclusive and can stand-alone or be done concurrently.

1. Implement the results from the four case studies in the next negotiation period. This would only require reducing the licenses for Case Study 1 and not adding any more licenses for Case Study 2 when the bonus period ends. In this way, Agilent can monitor the effects of having fewer licenses and, possibly, gain more confidence in this approach.
2. Determine the cost implications of a company-wide or product-specific service level and discuss with internal business partners. This approach just utilizes the part of the model that evaluates cost of licenses, so it ignores any disagreement surrounding what occurs when a license is unavailable. It would help AES create service level agreements with its internal customers.
3. Continue applying the model to more case studies. AES could do the calculations on the top 10 most expensive software products (of which it owns more than 10 licenses). If the potential savings seem worthwhile, Agilent could then implement the results.
4. Refine the model and review it with more employees. Although the model was created with the input of many, it still had limited exposure relative to the size of the company. (<1% of Agilent's employees were involved) AES might want to receive more buy-in from different people before it proceeds. Furthermore, there are data integrity issues, such as discussed in Section

2.4.14, and AES management might feel more comfortable with implementing the results after it has more confidence in the data used.

6.4 Lessons Learned

Although I learned many important lessons during my six month internship at Agilent, I'd like to focus on three: 1. The relevance of a company's organizational behavior, 2. the insight that cost analysis gives, and 3. the application of existing models to new areas. I found these particularly striking and an awareness of them will help me in future endeavors.

The model created for this project does (and did and will) not exist in a vacuum. It is filled with assumptions influenced by people, and the results have implications on employees' activities. As described in Chapter 5, the power at Agilent resides with the development engineers. I took this into account, for example, by presenting the model with the assumption that engineers can only tolerate four hours of license unavailability before it impacts the product schedule. I suspect that they could probably endure more, but it would not be as politically accepted. The future use of this model will also be influenced by the culture. The cost savings might not be significant enough to risk countering the culture.

These observations are not made in an attempt to judge whether this is correct or not. They are stated to show how much influence the organizational environment has. At another company, there would have been other issues that would affect the assumptions and the potential implementation. All of this has to be taken into consideration when developing a model, or working on any project, and when marketing the ideas to different organizations. It is a key component in making a project successful.

The second key learning is about the importance of quantifying costs. Companies often use heuristics and rely on people processes to make decisions. This approach can work in many cases, but particularly when the costs involved are substantial, quantifying trade-off

costs can result in significant insight into the business. In this case, Agilent had attempted the project earlier, but had doubts about issues such as the quality of the data. These types of concerns are valid, but I think the cliché, “you have to start somewhere” is valid too. Even if all the components are not perfect, evaluating dollars spent with quantitative cost analysis can still provide excellent information; and now, there is a place from which to improve.

Finally, the basis for this model is not new; it relies on proven supply chain principles. Only the application to software licenses is unique. There is often pressure to invent and generate fresh ideas, but this internship reinforced that there can be useful learning gained by applying old ideas to different areas.

When reviewing the model with classmates and professors at MIT, many mentioned that the project made them think of even other ways that the concept of inventory and stock outs can be applied. There are many models that have been categorized for one specific use, but there might be new, and relatively straightforward, applications that have not been investigated, but can provide important learning. Being aware and open to these types of opportunities could provide breakthrough information for a company.

6.5 Summary

Evaluating software licenses as one would evaluate physical inventory is a new approach that gives companies more quantitative ways to manage their licensing programs. The model described in this thesis is flexible for evaluating a variety of scenarios. It also relies on proven principles and is simple enough to allow for ease of use and further improvements. The challenge is whether organizations will decide that the cost implications of the model are worth the inconvenience to the engineers.

Appendix A:

Engineering Questionnaire:

Describe your experience with EDA license availability:

1. Have you ever been denied an EDA Tool license(s)? Y/N If not, continue to question 8.

Please answer questions 2-7 for a denial situation (or several situations) you've experienced.

2. Which product and application was not available?
 - a. Supplier:
 - b. Application:
3. How long did the outage last?
 - a. less than 1 hr
 - b. less than ½ day
 - c. less than 1 day
 - d. greater than 1 day
4. Did you lose productive time due to the license denial? Y/N
 - a. If yes, what did you do during the time you waited for the license?
5. Did the license unavailability delay the completion of the product development? Y/N
 - a. If yes, by how much?
 - i. Greater than 1 day
 - ii. Greater than a week
 - iii. Greater than a month
 - b. If not, at what point of hours unavailability do you think license unavailability would affect the timing of a product launch?
6. Did the license unavailability create a need for:
 - a. Overtime
 - b. Working after hours
 - c. Change of work schedule
 - d. Creation of a new plan for sharing licenses
 - e. Adding extra resources

If you marked any of the above, please describe this experience in more detail.

7. What were other implications of license unavailability? (i.e. frustration, management complaints, overtime, etc) Please describe.

EDA License Unavailability Scenarios: (if you answered 2-7, and do not want to comment further, please skip to question 11)

8. How would your project be impacted if an EDA tool license was not available?
9. Up to what period of time would have a negligible affect?

- a. Less than 1 hr
 - b. Less than ½ day
 - c. What would you do during the time you waited for the license?
10. At what point do you think license unavailability would affect the timing of a product launch?

Service Levels:

11. If the license was unavailable intermittently (i.e. X hours of outage spread over a week or a month) as opposed to X hours all at one time, would this have a different effect on your work pattern? Y/N
- a. Please describe.
12. Are there times during the development phase that license unavailability has more impact than other times? Y/N
- a. Please describe.
13. Do you have any suggestions for improving the situation if a license is unavailable? (i.e. knowing the cause, having a notification system to know when it is available, have a reserve system, etc)

BIBLIOGRAPHY

- Ancona, Deborah, Malone, Thomas, Senge, Peter, and Wanda Orlikowski, "The Sloan Leadership Model," Draft manuscript, Sloan School of Management, 2003.
- Barnett, F. William, Jr., Berland, Terrace P. "Strategic Thinking on the Front Lines", *The McKinsey Quarterly*, Spring 1999.
- Bonnal, Pierre, Didier, Gourc, and Lacoste, Germain "The life cycle of technical projects" Project Management Journal. Vol 3 Mar 2002 p.1.
- Edward, Katherine, "Achieving Cycle-Time Excellence in the Global Economy: The Impact of Speed", 2001 AACE International Transaction, PM.12.2.
- Gibbons, Dr. Robert, Sloan School at Massachusetts Institute of Technology. Course: Strategy and Organization, Spring 2003.
- Griffith, Dan Chairperson, Central Enterprise Licensing User Group (22 company consortium) and Manager, Comprehensive Software Asset Management at Motorola. E-mail dated 12-2-03.
- Hopp, Wallace J., Spearman, Mark L, Factory Physics: 2nd Edition. McGraw-Hill Higher Education 2001 p. 70, 489.
- Keeley, Ron, Centralized Representative for Mechanical Engineering at Agilent. E-mail dated 8-19-03.
- McReynolds, Scott; O'Brien, Brian, "Earning Pressures Boost Shared Services", *Financial Executives* Jan-Feb 2002 v18 i1.
- Reinertsen, Donald, "How Much is Speed Really Worth?" Electronic Design, May 3, 1999 v47 p. 64F.
- Reinertsen, Donald, Managing the Design Factory: A Product Developer's Toolkit (Free Press: October 1997) p. 28, 190.
- Silicon Integrated Initiative Inc., Gartner Dataquest, EE Times. *Engineering Design Automation Study*. May 2002.
- Stanley, Rachel, License Administrator Specialist. Honeywell. E-mail dated 8-19-03.
- Wright, Gene, Process Lead for AES Program Management tools at Agilent. Phone conversation 9-03.