Symbolics, Inc.:

A failure of heterogeneous engineering

Alvin Graylin
Kari Anne Hoier Kjolaas
Jonathan Loflin
Jimmie D. Walker III

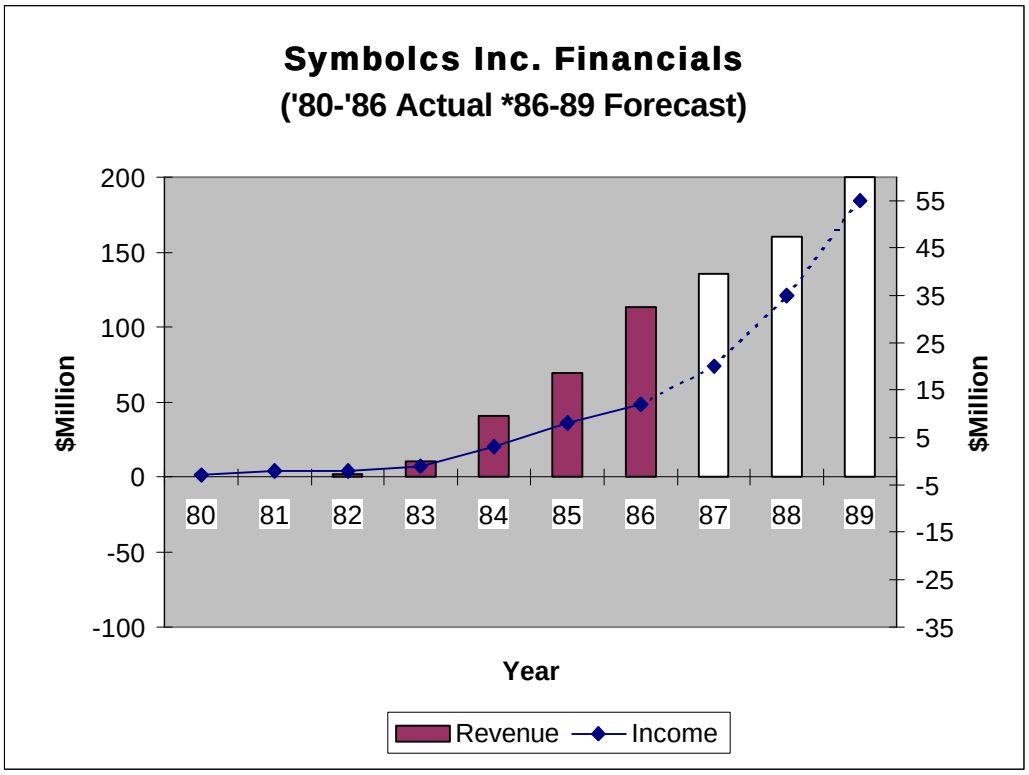
Table of Contents

1 Introduction

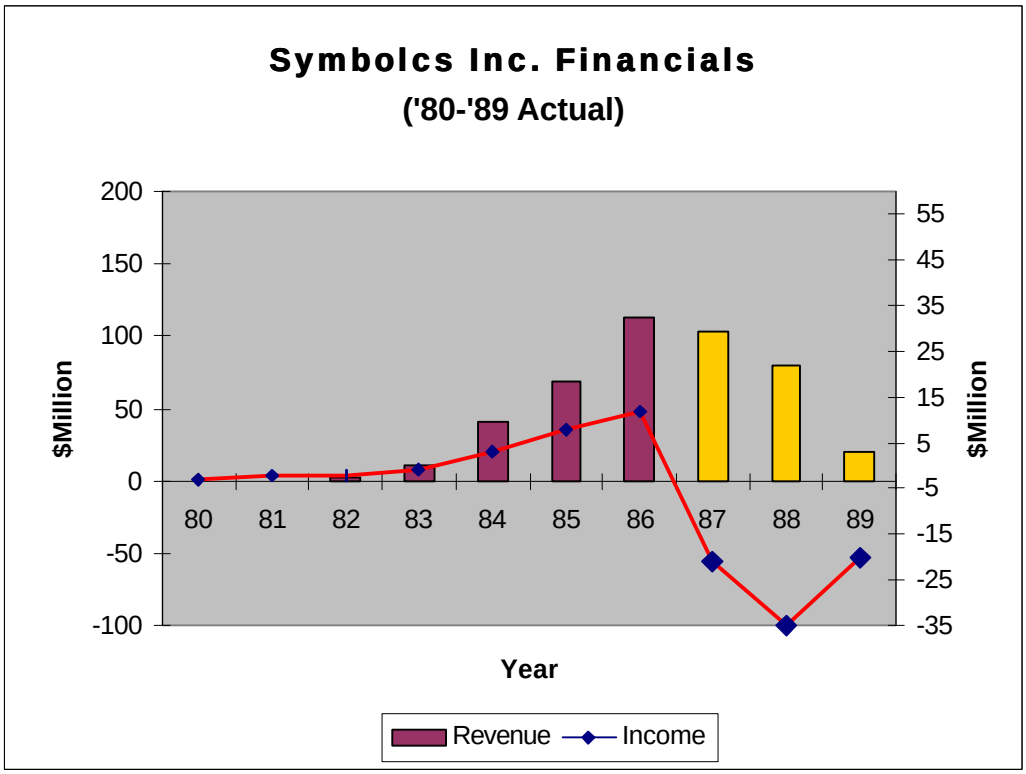
Thomas F. Knight, Jack Holloway and Richard Greenblatt developed the first LISP at the MIT Artificial Intelligence Laboratory in the late 1970s. Many saw the commercial potential for the new machine. Symbolics was formed in 1980 by 21 founders, the majority of which came from the MIT AI Lab. Greenblatt had decided to found his own company LISP Machines Inc. and was not involved. Knight was involved with the company form the beginning, but chose to stay at MIT to finish his PhD thesis. Holloway was instrumental in getting the company off the ground. An excellent programmer, he helped recruit top people to the new company and write the first Symbolics business plan. Another central figure was Russel Noftsker. He had been the director of the AI Lab for 9 years and had previously helped co-found another small company. Noftsker was central in raising money for the new company and later became its first CEO and Chairman.

At first most Symbolics employees kept working at MIT. Symbolics paid their salaries, but had no equipment for them to work on. Symbolics therefore made and agreement with Patrick Winston, the current director of the AI Lab, to allow the employees to keep working in the lab and using lab machines. In return, the AI Lab was given free internal use of all Symbolics software.

Symbolics managed to launch their first product, a re-engineered version of the first LISP machine in 1981. They incorporated in 1985 and revenues grew rapidly. However, by 1987 a crisis had hit the



company. Consolidated revenues form products for fiscal years 1986, 1987 and 1988 were $101.6 million, $82.1 million and $55.6 million, respectively. The only area of growth was that of training and consulting. Consolidated revenues form services in 1986, 1987 and 1988 rose, going from $12.6 million to $21.6 million and then to $25.5 million. However, this was not nearly enough to compensate for the drop in sales. Part of this drop coincided with the end of the federally funded Star Wars program, aimed at making the US able to shoot Russian missiles out of the sky with lasers fired from satellites. This project had led DARPA to fund many expert systems projects, many of which used Symbolics machines.



During the years of growth Symbolics had been searching for a CEO. Over 40 candidates had been interviewed, but none hired. Faced with the crisis, the board hired Brian Sear as COO in December of 1986. He was a former GenRad executive from the west coast, and had been connected with Symbolics as a consultant. He proceeded to cut cost and staff and restrain spending in R&D. This made him unpopular inside the company. Furthermore, his plan to match Sun's broad marketing strategy and unbundle hardware and software brought him into conflict with Noftsker. Noftsker wanted to focus more on the market of high-end machines for symbolic processing. He was close to R&D personnel, many of which he had worked with before Symbolics. They rallied around him as an opposition to Sear. The conflict came to a head in January of 1998, and both Sear and Noftsker were forced out of the company by the board, tired of their arguing. A new CEO was hired, but the company never recovered and filed for bankruptcy a few years later.

2 Theoretical Framework

It is difficult to conduct any form of analysis without a theoretical framework to provide a structure. The first step in this analysis of Symbolics will be to outline such a framework, using ideas from history and sociology to create a model that can be used to analyze our findings.

In his book Inventing Accuracy MacKenzie defines a technical trajectory as

> "*a direction of technical development that is simply natural, not created by social interest but corresponding to the inherent possibilities of the technology*"
>
> Donald MacKenzie, *Inventing Accuracy*, p. 167

Throughout the following discussion he rejects this argument, claiming that technological change is not self-sustaining. Its direction and form can not be explained in isolation from the social circumstances under which it takes place. What in hindsight appears as a "natural" trajectory is in fact a path created by circumstances and the efforts of proponents of the technology to manipulate these to their advantage.

In order for a technology to be successful its proponents must create interest for it in order to obtain resources. They must create an institutional framework in which progress can be made and train employees and the public. MacKenzie coins the concept of heterogeneous engineering in order to describe the complete set of skills necessary to succeed in promoting a specific technology:

> *"People had to be engineered, too - persuaded to suspend their doubts, induced to provide resources, trained and motivated to play their parts in a production process unprecedented in its demands. Successfully inventing the technology, turned out to be heterogeneous engineering, the engineering of the social as well as the physical world."*
>
> Donald MacKenzie, *Inventing Accuracy*, p. 28

MacKenzie's theoretical model is created in the context of discussing nuclear missile guidance. He focuses on the MIT Instrumentation Lab (Draper Lab Inc.) as a key center for missile development. Charles Draper, the head of the lab and one of the main proponents of inertial missile guidance, is MacKenzie's prime example of successful heterogeneous engineering.

In this paper we will apply MacKenzie's theories to the subject of our study, Symbolics Inc., a leading Cambridge software company in the 1980s. MacKenzie's theory of heterogeneous engineering easily lends itself to a study of technology companies. As in a research lab, funding must be obtained (initially through financing, later through sales of products and/or services), a framework/company

structure must be established and maintained, the employees must be trained and induced to produce top work, customers persuaded etc.

Of course, just hiring people to execute the different aspects of heterogeneous engineering is not enough. All must be given enough influence and resources to complete assigned tasks. Furthermore, conflicts between different functions will inevitably arise. These must be resolved. As with Draper, much depends on senior management. They set the agenda of the company, keeping the overall goals in mind. They need to be successful heterogeneous engineers in order for a company to succeed.

While struggling with similar tasks, Draper and Symbolics' leadership worked in different environments. Draper's goal was to attain public funds, while Symbolics was competing against other innovators such as DEC, Sun, and IBM in a commercial marketplace. Hence, the framework we have developed for exploring the course of heterogeneous engineering at Symbolics departs from that used in the case of missile guidance.

The framework we have developed (see illustration below) to describe and evaluate heterogeneous engineering in a commercial company is based on a framework expounded by Michael Porter in his book *Competitive Strategy*. This model proposes that the leaders of an engineering firm need to clearly define the goals of the firm, and must carefully align the operations of the firm with those goals. The operations of the firm are broken down into the following units, each of which supports the firm's common goals.

It is a primary function of the company leadership to insure that the goals are clear and focused, and that each group within the company is engineered to best achieve those goals. Porter compares each group to the spoke of a wheel. In order for the firm to compete effectively (and thus be successful) on a long-term basis, the wheel should be in balance and should have every spoke in place.

Our model of heterogeneous engineering makes it evident that while technical superiority (however defined) only one piece of competitive advantage. Engineering innovation requires that the innovators engineer their environment. An innovation, particularly in a highly competitive commercial setting, can only succeed if all the pieces of a firm are working together to promote the innovation. Acceptance of an engineering innovation requires a strong alignment of heterogeneous practices.

It is clear that Symbolics' technology is innovative. Symbolics set high goals for its designs and usually delivered on those design goals (often adding additional features to the product in the process). Symbolics was founded and staffed by some of the brightest technical academics in the country. The technology was in many ways superior to anything that was available at the time.

While producing innovative technology, Symbolics failed in many of the other areas defined by our model. It is the thesis of our paper that Symbolics, as a firm, did not successfully engineer its environment for successful and lasting innovation. The fundamental failure of the organization was the lack of heterogeneous engineering. First and foremost, the goals of the company were not well defined. Furthermore, there was a lack of balance along the various operations of the company. The focus was clearly on research and development. The goals of the firm were shaped by the whim of the research and development department, and as a result many of the supporting functions of the firm were unsuccessful.

At Symbolics, Goals varied in order to support the current direction of Research and Development. Hence, we put Research and Development at the center of the "competitive wheel," because every other units were essentially to support research.

3 Heterogeneous Engineering

In order to examine the failure of Symbolics in the context of heterogeneous engineering, one has to consider each of the individual units of the firm and how these units interacted with each other and the external environment. This section will discuss the different sections of the company in the context of our model.

3.1 Company Goals

The long-term goals or vision of a company is analogous to the compass used by a navigator to guide a ship safely to its destination through any atmospheric conditions. For any company to sustain success in a competitive market, it must have a clear vision that is communicated to and internalized by all the employees of the company. The vision should help the company focus its resources and attention on the right target, while providing a consistent set of principles that would guide it in making its decision through good and bad times.

From our research, it was clear that Symbolics did not possess the type of long-term vision that we described above. This is not to say that the company did not have any goals at all. Unfortunately, it may have had too many goals. Just about every person we interviewed had a different idea of what the Symbolics' long-term vision was. To make the situation worse, the public image being presented was also inconsistent with the internal belief held by the employees and founders. Depending who you asked (and when), the vision for the company includes all of the following:

- Our goal is to "be the leading developer, manufacturer and marketer of advanced computer systems that facilitate the use of artificial intelligence and other symbolic processing techniques" [1]

- "We were always a workstation company… we were never an AI company"[2]

- Symbolics was "*the* preeminent AI company"[3]

- "The hardware was always there just to support the software"[4]

- "The company was founded to commercialize symbolic processing… which basically meant Object-Oriented"[5]

[1] Symbolics, Inc. 1985 Annual Report, p. 1
[2] Russell Noftsker, CEO and chairman of the board, interview
[3] Harvey P. Newquist, editor of *AI Trends*, interview
[4] Russell Noftsker, CEO and chairman of the board, interview

• “The real overriding goal was to enable rapid development of highly complex software, which could safely be assumed to be written in Lisp.”[6]

• “To be honest, a goal that was very influential was to have fun trying out all kinds of relatively far-out ideas. A lot of software got written for its own sake more than to satisfy and business need.”[7]

• “Other people probably had other goals, such as marketing goals”[8]

This is just a sampling of the different viewpoints that existed at the time about Symbolics. It was a LISP machine company, an AI company, a software company, a general purpose workstation company, a place to develop fun technology without business-related distractions, and at one point, a chip and add-in board company. It was trying to be everything for everyone. As a result, the company did not have the focus or vision needed to drive a company through troubled times. David A. Moon, a prominent software and hardware engineer at Symbolics, poignantly asserts that “Another reason the company foundered was a certain element of sheer incompetence, or indecision about what the company’s real goal was, of course.”[9]

There was one thing that it seemed everyone did agree on: Symbolics wanted to be a technology leader. That was how they would compete. In most cases, this would have been a good place for a corporate strategy to start, but in this case, this strategy was one of the contributors to the company’s failure. Because there was no central goal or direction for the company to move towards, there was very little synergy between the work of the different groups in the company. Symbolics operated like an academic research institute, where unrelated research was happening in different groups, depending only on the different research interests of the staff. Costs were not taken into account, and the company tried to do every piece of the system in-house. Symbolics designed and implemented the chips, the boards, the chassis, the peripherals, the operating system, the development tools, the email system, the networking software, and even the end-user applications. Symbolics even ran its own manufacturing facilities. For a relatively small company to try to accomplish all this was certainly no small feat.

Symbolics' technology driven culture resulted in the invention and application of many innovative ideas, but without clear direction, innovation by itself led the company nowhere. With all the talented minds in the company at the time, it's truly a pity that Symbolics failed. If it had only focused its people's

---

[5] Russell Noftsker, CEO and chairman of the board, interview
[6] David A. Moon, software engineer, email
[7] David A. Moon, software engineer, email
[8] David A. Moon, software engineer, email
[9] David A. Moon, software engineer, email

efforts in a unified way and set a clear long-term direction towards a more durable market, there might have been a way for it to survive.

3.2 Research and Development

It is impossible to discuss research and development at Symbolics without any technical background knowledge. In this section we will discuss the LISP programming language, some of the reasons why the LISP machines were developed and finally how the research culture at Symbolics came out of this work and helped shape the company as a whole.

Background of the Technology

LISP ("LISt Processing language") is a programming language designed in the AI community in the early 1950s. Specifically designed to symbolically represent objects and the relations between them it is the oldest (and for many the only true) object oriented programming language. On the lowest level, LISP represent all objects, even the expressions of the language itself as lists. This uniformity is one of the major strengths of the language and makes it easily extendible. LISP lists are dynamic and can grow and shrink without bounds during the execution of a program (this is not the case in most programming languages. In C/C++, the dimensions of an array must be declared at the beginning of a function). This makes LISP ideal for representing complex objects about which little is known in advance. Furthermore, since data and programs are represented in the same manner, programs can generate other programs and execute them. Again, this makes it possible to model more complex processes.

Lisp offers even further advantages over other programming languages. For example, Lisp is easy to learn because the syntax is very uniform due to the elegant parenthetical structure of every Lisp expression. The following code demonstrates this elegant parenthetical structure (and is also an example of the ubiquitous use of recursion in Lisp programs):

```
; This function returns the factorial of ‘n’.
; For example, calling (factorial 4)
; will return the number 24.
(defun factorial (n)
        (cond ((= n 1) 1)                       ; if n=1, return 1.
                (t (* n (factorial (1- n))))))  ; otherwise, return
                                                ; n times the factorial
                                                ; of n-1.
```

The versatility of LISP comes with a price. Allowing lists to grow dynamically requires a lot of overhead (memory needs to be allocated/deallocated during execution) making LISP require lots of space. Furthermore, LISP keeps more of the programming environment in memory compared to other programs in

order to efficiently pass objects back and forth. In the 1970s machines had very limited virtual address spaces (some IBM machines only had 512K). Each address was only half a word long. While saving space, this made it impossible to easily upgrade to a larger address space. Running LISP on these machines was therefore slow, since the program was forced to constantly swap objects in and out of memory.

LISP is a compiled language (although the introduction to a LISP interpreter is an important part of 6.001, the introductory programming class at MIT). LISP could be compiled on the PDP-10 and other time-shared systems in the 1970s, but only inefficiently. A serious impediment was that there was not machine instruction to facilitate garbage collection. All of this made the AI community eager for a new machine, optimized to run LISP.

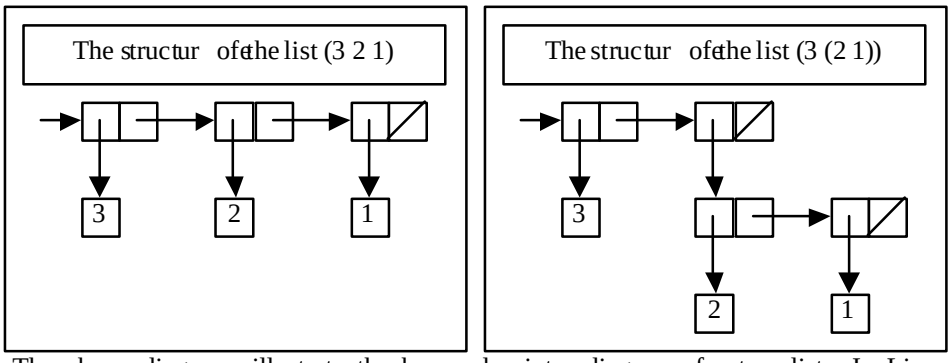
Why the Need for Machines Optimized for LISP?

In Tom Knight's thesis, he points out three primary reasons that motivated his efforts, starting in 1974, to develop the original Lisp machines at the MIT AI Lab. He asserts in his introduction that the main difficulties are:

- Inadequate virtual address space for large user programs.
- Inadequate computing power for development if intelligent programming tools.
- Inefficient information coding of compiled instructions.

Inadequate virtual address space for large user programs

The first point is illustrated by examining the addressing scheme for the PDP-10, a computer that would be the second best option for running Lisp programs in 1979. In order to demonstrate this idea, it is useful to illustrate the box-and-pointer diagram familiar to any novice Lisp hacker:



The above diagrams illustrate the box-and-pointer diagrams for two lists. In Lisp, every expression (including data and the program itself) is a list. Hence, the efficiency of list storage and list manipulation is of paramount importance.

Each of the rectangles above (formed by putting two boxes together) is called a “node.” A node consists of two pointers. In a list, the left pointer points to an element of the list and the right pointer points to the *next* element of the list. In the more complicated structure of the list ’(3 (2 1)), the second element of the list is itself a recursively embedded list ’(2 1).

In a PDP-10, a node is represented as a 36-bit structure with each pointer using 18 bits. This implies that the address space for memory can only reference 2^{18} bits, which means 256 Kilobytes is an upper bound for the amount of memory available to a Lisp program. This limitation is severe when one considers writing huge expert systems that may require thousands of kilobytes of memory. According to Knight, “Expansion of the virtual address space in current PDP-10 Lisp implementations requires use of two words per Lisp node and drastic changes to both the interpreter and the compiler [Knight, 4].”

Inadequate computing power for development if intelligent programming tools

Knight envisions a development environment that is highly interactive with the user. For example, Knight envisioned that when you make a change to a procedure, the computer will issue a warning citing all the other procedures that may be affected by the change. This would facilitate development. In addition, Symbolics machines would allow for interactive debugging (allowing a pause in the execution of a program, allowing the programmer to make changes to the program in mid-execution, and allowing the programmer to resume execution of the program after the changes).

The heavy interaction between the computer and the programmer would require significant processing power. A PDP-10 could provide this power to one user but not to multiple users simultaneously using the timesharing system. The system would not be sufficiently reactive due to the fundamental design assumption of timesharing by the makers of the PDP-10.

Inefficient information coding of compiled instructions

The compiled code for Lisp instructions on the PDP-10 is much larger the compiled code for Lisp instructions on the CADR machine that Knight built for his 1979 thesis. This is best illustrated by example, as Knight did in his thesis when he shows the assembled instructions for both the PDP-10 and the CADR when compiling this factorial program:

```
(defun factorial (n)
        (cond ((= n 1) 1)
                (t (* n (factorial (1- n))))))
```

The CADR uses 10 instructions taking 160 bits of memory. The PDP-10 uses 16 instructions taking 576 bits of memory. The compiled code is reproduced in this paper from Knight's thesis for comparison.

Other problems running Lisp with existing platforms

Other problems with the PDP-10 included a lack of optimizations for garbage collection and for Object-Oriented procedure calls. In addition, computers such as the PDP-10 were bulky and noisy, which made a PDP-10 "an unwelcome office mate" according to Knight and Greenblatt [7]. This annoyance in particular influenced the design of the network architecture of the original Lisp machine in an interesting manner (see the section on the network architecture).

Design of the LISP Machine

The primary design criterion for the Lisp machine was to run Lisp quickly enough to allow efficient development of AI programs. Russell Noftsker asserts that a developer would be 10 times as productive on the Symbolics platform as he or she would be on an alternative platform. An extended list of the design criteria, according to the Symbolics Technical Overview, includes each of the following tenets:

- Fast Lisp execution
- Dedicated personal computer and console
- Tagged architecture (run-time data-type checking and generic instructions)
- Virtual memory
- Integrated local area network
- Interactive, high-resolution, bit-mapped graphics

Conspicuously absent from this list is time-sharing, which was considered at the time to be a necessity for any successful computing environment. Tom Knight and Richard Greenblatt did not believe in making a time-sharing system (although the system did support an interesting alternative scheme of multiplexing between users that will be discussed later).

This section will primarily document the features and advantages of using Lisp for development, the inadequacies of existing computing environments (such as the PDP-10) to develop Lisp, and the features (both in hardware and in software) that made Lisp machines ideal for Lisp development. Finally, the product line of Symbolics, Incorporated and its evolution will be examined.

What made Lisp machines better for running Lisp from a Hardware perspective?

Lisp machines were designed from the ground up in order to run Lisp. This principle manifests in a variety of technical details of the hardware and software design.

Core Instruction Set and Microcoded Extensions

From a hardware perspective, the instructions of the Lisp processor architecture are chosen carefully to make certain functionalities easy for the compiler to implement. The core instruction set of the Lisp processor is augmented by a microcoding capability that allows macros for commonly used Lisp functions to run extraordinarily efficiently by downloading macros for the commonly used Lisp functions into a cache-like structure on the processor. Together, the core instruction set and the microcoded functions effectively implement Lisp in hardware.

One example of a functionality that uses these codes effectively is the ephemeral-object garbage collector written by David A. Moon at Symbolics (well after the original Lisp machines were made). While most garbage collectors cause noticeable pauses in program execution, Moon's garbage collector was so cleverly written (according to Russell Noftsker) that it would actually cause the system to run faster with garbage collection than it would without garbage collection. This garbage collection scheme was facilitated by the design of the instruction set, which included instructions that would only be used by a garbage collector. (Interestingly, Symbolics at one time was working with Intel to build a development platform based on the 386, which led to the inclusion of an extra instruction in the final 386 architecture to facilitate garbage collection).

Data Tags

The Lisp machines dedicated certain bits of the architecture as data-type bits. In the original 32-bit architecture, 4 bits of every data chunk would indicate the type of the data in the other 28 bits. This is ideal for Lisp because Lisp does not have strong typecasting like C, Fortran, or Cobol do. In a list representation, a pointer to data can point to any type of data (either numerical, ascii, or perhaps another list). This eliminates the need for data type declarations in programs and also catches bugs at runtime, which *dramatically* improves system reliability. Data tags represent a significant advance beyond the state of the art.

Virtual Memory

Virtual memory addressing allows the machine to run larger programs while abstracting memory issues from the programmer. This feature is essential to writing large programs such as expert systems.

Error Correction Codes

With error correction codes (ECC), memory or disk corruption could be detected and fixed at runtime. This is essential for building a large, reliable program in an environment where physical matters (such as floods or flying bullets) could affect the hardware. In addition, memory and disks were inherently unreliable at that time and therefore ECC was highly desirable even at the cost of 4 or 8 extra bits for every 32, 36, or 40 bit word (depending on the generation of Lisp machine).

What made Lisp machines better for running Lisp from a Software perspective?

From a software perspective, the Lisp machines offered tremendous benefits to developers of Lisp programs.

The User Interface was very User Friendly

The UI used mice, windows, and familiar GUI concepts for ease of use. This enhanced developer productivity.

The entire Operating System was written in Lisp

The Operating System was written 100% in Lisp. All levels of the software were written in Lisp. This allowed for a high level of compatibility with Lisp programs.

Garbage Collector

The garbage collector was built into the system. The ephemeral-object garbage collector was extremely efficient due to synergies between it and the processor-level instruction set.

Great Documentation

"Outstanding documentation was a hallmark of Symbolics systems"[10]. Documentation existed in electronic form (using innovative hypertext much like that of a modern web-browser) and also in printed form. Symbolics won awards for its documentation. In addition, Symbolics' user training programs were considered to be quite good.

Superb Development/Debugging tools are built into the OS

Most importantly, outstanding development tools were built into the operating system. "Symbolics Lisp" is the dialect of Lisp that is supported on the Symbolics platform. The Symbolics Lisp dialect offered numerous benefits over other Lisp dialects. According to the Symbolics Technical Overview, these include:

- The Flavor system for object-oriented programming with message passing.
- Incremental debugging
- Flexible function calling and multiple-value returns.
- Multiple namespaces (packages).
- Stream-oriented input and output.

[10] Symbolics Technical Overview, http://home.brightware.com/~rwk/symbolics

• Modern control constructs, including a very general loop iteration facility, asynchronous non-local exists, co-routines, and processes.
• Macros for extending the Symbolics-Lisp syntax.
• Predefined functions that support such operations as sorting, hash tables, linear equations, and matrix operations.
• A full range of data types, including many numerical types, lists, strings, arrays, planes, and user-defined structures.

Perhaps the two most important of these are the Flavor system and the incremental debugging facilities.

The Flavor System offers Object-oriented programming, permitting modularity in programs by encapsulating procedures and data into objects. New flavor types can be cloned by combining existing flavor types [Symbolics Technical Overview]. Inheritance can be bi-directional (non-hierarchical). Noftsker points out that message passing would take about 3.5 microseconds on a 3600 series Symbolics machine compared to 1000 microseconds on a traditional architecture.

Network Architecture

The computers of that era were large and noisy. For this reason, Knight envisioned an environment where several Lisp machines would be networked to each other, a central file server and to the Arpanet (which became the Internet). All of the Lisp machines would be connected to a “video switch” that could redirect the I/O (display, mouse, and keyboard) into an office where the I/O peripherals were located. This would allow, for example, 4 Lisp machines to be switched between 20 offices so long as no more than 4 of the office-dwellers were logged on at one time. By using the video switch, the noisy computers were kept out of the users’ offices. A diagram of the network architecture from Knight’s thesis is reproduced in this paper.

Weaknesses of the LISP Machines?

For non-Lisp programs such as a Fortran program, a Symbolics machine would not necessarily have a significant performance advantage. However, the Symbolics incremental debugging environment would still allow for easier debugging of a Fortran program. However, Russell Noftsker points out that a well-written database would run faster on a VAX than on a Symbolics machine.

Nevertheless, although the Symbolics machines may have been comparable or even somewhat superior to a DEC machine for non-Lisp applications, the Symbolics machines were perhaps 5 times as expensive as a DEC.

Research Culture Within Symbolics

Symbolics was driven by research and development. In 1987, Symbolics spent twice the industry average on research and development (Bottorff, 37). [11]The influence of the research and development goes far beyond financial domination, however. The research and development department set its own agenda, set the agenda for top management, and therefore set the agenda for the entire firm. Symbolics exhibited all the problems that one might imagine when a group of highly academic researchers come together to run a $100 million dollar company.

There was a Lack of focus within Research and Development

The research and development department was full of strong individuals with academic backgrounds. The senior engineers were used to the freedom to pursue own research interest they had enjoyed in academia. When asked whether the general management had problems controlling the focus of the research and development department, Howard Shrobe (then a Symbolics engineer and now an MIT professor) responded, "I always felt like it was research and development that had a hard time controlling the general management." Regardless, more often than not research and development would have the final say in what research and development would focus on, and in addition they seemed to set the tone for every other department as well. David A. Moon agrees when he says, "A lot of software got written for its own sake more than to satisfy any business need… I don't think changes in the market or customer base had much effect [on the technology]"[12]. Central management was not able to harness the brilliance of Symbolics' research and development into any single direction. Rather, the leadership of the company (particularly Russell Noftsker, the CEO) was himself highly academic and allowed the academics in research and development follow their own course.

One example of the random focus of the technological employees is the use of high-resolution montiors. Since the type of monitors Symbolics wanted to use were not available from any vendor, they decided it would be necessary to build their own. While entry into the monitor business at most companies would be a strategic decision made by top management (who might employ outside management consultants to study the market and strategic implications), at Symbolics the researchers drove this decision through. David A. Moon observes, "High-resolution monitors were nice for doing the kind of software we wanted to do, but really secondary to the main goal. Maybe they made us waste a lot of time experimenting with window systems!"[13]

It is frightening to consider that Symbolics, a small company, would decide to enter the market to manufacture monitors even when it was secondary to "the main goal" (which was either software,

[11] *Mission Impossible?*, Dana Bottorff, New England Business, No. 6, 1998, p. 27.
[12] David A. Moon, software engineer, email
[13] David A. Moon, software engineer, email

workstations, operating systems, expert-system compilers, general symbolic processing, or add-in boards, depending on who you asked and what time of day). Futhermore, the decision to enter the display hardware market led to a loss of focus of the software developers who then decided to building window systems, and then went on to build hypertext and other nice, but non-essential, features into the system. Moon recollects "We wasted a lot of time doing early versions of technologies (high resolution displays, laser printers, ethernet-type networking) that were done better and cheaper a few years later by others."

Essentially, the research and development department followed a classic "random walk." They would get heavily involved in a tangent to the "main goal," develop that tangent, and in the process get lost on further tangents. For example, display hardware is one tangent and lead to windowing systems, another tangent. Laser printers seem equally irrelevant to the core mission of Symbolics. Furthermore, the extreme lack of focus of the research and development is a likely reason that no two employees will offer the same definition of Symbolics' "main goal," although it is common that employees will refer to the notion of the "main goal" in the abstract (without explicitly articulating it).

Management's lack of control over the research and development department (and therefore their lack of control over the technology) is a primary symptom of a lack of heterogeneous engineering. Strong central management should clearly articulate the main goals of the firm and furthermore should communicate it clearly to researchers and developers to the extent that each technical employee can formulate *exactly* what the main goal of the firm is. This process of defining and articulating goals is a critical component of heterogeneous engineering.

A Highly Academic Notion of Technical Superiority lead to Product Difficulties

Symboics research and development department was comprised of people with academic backgrounds. Many engineers retained a highly academic notion of technical superiority and this helped shape the company culture. At Symbolics, *technical superiority was measured in terms of clean design*. Other important goals were sacrificed in the name of clean design, including:

- Low cost
- Compatibility
- On-time delivery

The significance of these three failures will be discussed in detail in the following sections.

Clean design was always a primary goal for Symbolics' engineers, resulting in a culture where to goal was to get things right rather than getting things done. Clean design is often cited as the reason that Symbolics never hesitated to expand into new fields (such as laser printing, display hardware, et cetera). It was critical for to their engineers to produce an elegant solution, and they did not consider other vendors products to be elegant. David A. Moon explains, "We did everything, we used no software from anyone

else. That means a whole operating system, file system, many network protocols, email system both client and server, language development environments, documentation production and viewing system, color graphics and animation, window system, presentation system (somewhat of a lost art these days), inference engines, and a lot more."

One interesting example of Symbolics' preference for clean design relates to the Y2K problems that concern economists and the Department of Defense as the new millineum approaches. "I don't remember ever once thinking about Y2K. However, our 'do it right and don't worry overmuch about the cost' software philosophy probably means that we never even considered putting in Y2K bugs." At that time, making computers Y2K compatible came at considerable cost due to the scarcity of RAM. "You need to bear in mind the environment (computer industry) of the period of 1976-1984 when these things were happening. Computers with substantial capacity were very expensive… The AI Lab ran on a time-shared machine with… 2 megabytes of memory… The system cost something like a million dollars, back when that was a lot of money."

The nobilty of clean design came at the expense of producing low cost systems. Moon recalls, "Lower cost was also a 3600 goal but it was not really achieved." Cost was not a primary concern for Symbolics initially. Its market was largely isolated from inexpensive competition until the mid-1980s (when cheap Sun and DEC workstations first became feasible alternatives to Lisp machines for running Lisp).

Compatibility also suffered as a result of clean design. Symbolics' system designers preferred to build their own peripherals rather than make their system compatible with other vendors' systems. Newquist writes:

> *"If the companies currently selling AI products wish to survive into the 1990s in any way, they will address the needs of DP/MIS [Data Processing and Management Information Systems] departments. Period."*
> Harvey P. Newquist, *AI Trends '88*, No. 13

Symbolics' machines were not even compatible with other machines such as those built by IBM that were very popular with DP/MIS departments for many years before and after Symbolics existed. Also, Symbolics machines were slow at running databases (according to Noftsker), which is critical for Information Technology. Therefore, it is clear that Symbolics was not addressing the needs of IT. Why did Symbolics last so long before this mattered? This will be answered in the section focusing on the sudden loss of sales.

Finally, delivery timing was a problem for Symbolics that resulted largely from the attention to detail and to clean design. Late product launches provided fuel for venture capitalists (who were on the Board of Directors) to cut programs that Noftsker saw as essential. For example, Symbolics was at one time working with Intel to build a cheap delivery platform (that would be suitable to run Lisp programs, but not suitable to develop them). This project involved writing software to run on the 386 and would have allowed Symbolics to enter the delivery market. The project was cut by the Board in the mid-1980s due to an overall program of cost-cutting induced by slipped deadlines.

3.3 Product Line

With such a vast array of products, Symbolics failed to market properly the one product that could have saved the company, their software. During the critical years at Symbolics (1986-1988), many potential customers of Symbolics were interested solely in Symbolics' software. However, customers could not buy the software without purchasing a expensive LISP machine. At that point, most customers would rather buy a substantially less expensive machine from another company that performed almost to par with a LISP machine and still have enough money to buy decent software. A heterogeneous engineeer would have seen the marketing potential of Symbolics software and saw to it that Symbolics software was sold. There is little doubt that Symbolics had heterogeneous engineers on staff. Unfortunately, due to Symbolics step learning curve to realizing the potential of their software, heterogenous engineers were not in charge. Symbolics did not completely switch to selling software until 1993. By that time, Symbolics window of opportunity had already closed and the fate of Symbolics had been set. It is quite paradoxical that a company that considered software to be their main focus did not see the marketablility of their software earlier.

It is virtually natural for engineers to want their products to be perfect with every feature imaginable. That is why it is important for technical companies to have business employees to keep engineers in check. Businees employees have a sense of how many features and how perfect the product needs to be to satisfy the market. By know what the market wants, business employees help minimize product release delays. When one tries to add too many features to a product, this usually causes delays in the release date of the products. This, in turn, leads to customers going else where for their products even if the performance is not up to par.

Part of Symbolics failure was attributed to the excessive features of Symbolics products. The excessive features greatly increased the price of Symbolics products. Therefore, Symbolics almost forced the market to buy from competitors at a cheaper price. However, Symbolics purpose for adding the excess features was to make their products superior and tempt the market to buy their products. Yet, Symbolics failed to realize that "money does not grow on trees" and that money plays an important role in what the market buys.

As if increasing the prices of products with excessive features was not sufficient, Symbolics reinforced their fate of failure by not building industry support. Symbolics built all components for the LISP machine in-house. Therefore, they did not win much support from companies that could have profited from the making of LISP machines. For example, a company that specialized in producing monitors had no vested interest in Symbolics,who made their own monitor. If Symbolics had bought some of the components necessary for building the LISP machines from other companies, perhaps Symbolics' fall would have been less drastic and Symbolics could possibly still be in existence. The only possible downfall of Symbolics building industry support is that an industry profiting off the making of LISP machines, hardware, would not like for Symbolics to focus exclusively on software. The supporting company would lose a profit source once Symbolics converted to being a software company. Yet, since Symbolics did not see the profit of their software, building industry support could have only helped.

3.4 Target Markets

We have chosen a wider definition of target market than the one used in Porter's original model. In our discussion of target market we will include end users (programmers and engineers) as well as large institutional and corporate buyers of Symbolics' machines.

Narrow Company Focus

If there is a single way in which Symbolics was focused, it is that their target market was Lisp programmers who wanted high-end, expensive development machines. Symbolics maintained a focus on this small segment for many years (until they eventually tried alternative strategies when the final bankruptcy of the firm seemed imminent). Symbolics machines always included as many features as possible, and at a price that was many times that of any delivery machine.

It is clear why the first Lisp machines (developed at the MIT AI Lab) were development machines. Those machines were developed by Tom Knight, who explains that the machines were designed and developed *exclusively for use within the Lab*. What is unusual is that the design focus never shifted after the technology was commercialized. When the customers were no longer the designers, the designers still seemed to focus on themselves as the model users. David A. Moon confirms this: "I don't think changes in the market or customer base had much effect on the technical details." Given this, it is interesting to examine the effects of this mentality on the design of the Lisp Machines.

The reason they focused on development machines is partly due to the overbearing role that research and development played in product definition. Moon explains, "The model user was someone doing exploratory programming. At the time we

started there was no standard software worth anything, so being able to run standard software from elsewhere was never a goal." This description of "exploratory programmers" matches exactly the people who were developing the system. They were implicitly imagining the users to be people like themselves, people with MIT degrees or who could program at that level. Noftsker claims that a developer using a Symbolics machine could be 10 times as productive when coding in Lisp than that developer woul be if developing on a DEC Vax or equivalent. However, this gain in productivity can only be acheived after learning a whole new set of commands and menus. The steep initial learning curve made Symbolics suitable only for developers with software experience. Many of these were reluctant to learn a whole new skill set.

The Lisp Hackers: A Rare Breed

The Symbolics organization did not effectively convince the world that Lisp would be the dominant programming paradigm. By 1988, the proliferation of LISP machines may have exceeded the proliferation of LISP programmers. Newquist writes:

> *"With an installed worldwide base of some 7000 LISP machines, there is even the possibility that there are actually more [LISP] machines in the marketplace than there are experienced LISP hackers. Frightening."*
> Harvey P. Newquist, *AI Trends '88*, No. 48

Patrick Winston says that this is the primary reason that Symbolics was "doomed" from its conception (interview). He cites that Symbolics ignored the delivery market for AI programs, and hence probably would have hit a wall whether the AI industry had crashed or not.

A primary component of successful heterogeneous engineering is to convince the world that the future is destined to be dominated by the technology. Effectively executed, this will have a profound psychological impact and will actually create a self-fulfilling prophecy. MacKenzie asserts that "if it comes to be believed that there is only one way to advance a technology, then that one way has at least a chance of becoming a reality. The others do not."[14] This effect manifested itself in the advent of gas bearing, which replaced ball-bearings in the gyros used by Draper's guidance systems. The Air Force mandated to Draper that the gas bearing technology was superior and the lack of funds allocated to the exploratory development of ball-bearings insured that this would be the case.

[14] *Inventing Accuracy*, Donald MacKenzie, p. 391.

The gas-bearing example demonstrates the critical role of heterogenous engineering with respect to the users of the technology. Users of a technology must be engineered. There is significant evidence that the Symbolics employees discounted this critical function as automatic. Robert, who quit as vice-president of sales in March, 1988 claims that "At Symbolics, the attitude was: 'The world will find out' [how superior its systems were]."[15] This is a clear failure of heterogeneous engineering.

Brian Sear, the President and COO in the late 1980s, offers an additional insight into why customers were ignored. Sear claims that Noftsker and other who helped found the company were "very brilliant people, and they were very much enamored with leapfrog technologies. They wanted to repreat what they done in 1983 and 1984 [when the company burst onto the commercial scene], so they tended to downplay taking care of current customers. Russell had a philosophy that all technology was good and would be useful at some point."[16]

It turns out that technology is only useful if the environment is engineered to find it useful. Most engineers had learned other languages such as Cobol or Fortran when they were trained and therefore would find C to be a preferable development environment.

Star Wars was an Indirect Target Market for Lisp Machines

Russell Noftsker points out that many of the Symbolics machines were purchased by researchers funded (directly or by contract) through the Star Wars program, overseen by DARPA. Noftsker now points out that the notion of shooting missiles out of the sky with laser beams from space is inconcievable with the technology available in the 1980s, or in the 1990s for that matter. Regardless, development of expert systems was a key component of the Star Wars program, and Symbolics benefitted because those expert systems could be best developed on a Symbolics system. To some extent, Symbolics' success and failure followed the trajectory defined by Star Wars funding. We found no evidence that this is due to the type of heterogeneous engineering Charles Draper engaged in during the development of missile guidance when he essentially engineered the military decision makers to support his system. Rather, Star Wars funding was largely responsible for supplying a relatively *forgiving* target market to Symbolics in the early years (before it was a public company).

The forgiving nature of this market is perhaps one explanation why the company could be so successful with so little focus during its early years. Symbolics' management perhaps realized that the Star Wars funding would dry up, but did not engineer the company for tougher times. The dominance of the research and development department was never checked, even when the company faced a much tougher

[15] *Symbolics: The Soul of a New Regime*, Keith H. Hammonds, Business Week, June 13, 1988
[16] Mission Impossible, Dana Bottorff, New England Business, 6 1989, p. 27.

marketplace in the late 1980s. Moon observes, “the age of 80% gross margins in the computer industry was vanishing, and so was the age when selling 3000 machines a year was a big success.”

3.5 Manufacturing

A key aspect of heterogeneous engineering for any products based company has to be its ability to manufacture a product to the specified quality at a reasonable cost and deliver it on-time to customers. As Symbolics was generally viewed as a LISP machines company, one would assume that the manufacturing and manufactureability of the product would be an important issue for them. From our research, we found this was not the case. Here is yet another portion of the wheel that was not adequately addressed by the company.

We see this in several aspects of their organizational structure and processes. From the beginning of the company, the manufacturing facility was located in California, all the while, the research, design and management center was in Massachusetts. Although this is common practice in large manufacturing companies where there are cost, tax or distribution issues mandate multiple manufacturing facilities away from the headquarters, it is very unusual for a start-up with minimal sales and personnel to immediately locate their manufacturing capabilities and personnel across the country. This separation can lead to multiple communication, organizational, cost and even production problems that would have wide scale implications on the success of the company.

Apparently, the justification for separating the facilities was that the manufacturing manager at the time was a resident of California and the president of the company had a home there also. This just does not seem to us like a rational business justification for the separation that existed. In fact, this became a major issue for the major investors of the company as well. Certain board members from their investment partners were urging the move of the manufacturing to combine operations, but the management team was highly against that proposal. This issue even became a major source of conflict between the company management and the investors. It was also a reason for difficulty in finding new investors during a later round of fund raising.

Beyond the added cost and complexity associated with separating manufacturing from the rest of the company, we also found that the design of the products were not coordinated well with the needs of manufacturing. The majority of the architects, designers and engineers for Symbolics’ machines were from a software background. There’s no doubt that these were brilliant people with some very innovative ideas, but Symbolics failed to hire people with industry hardware design experience that knew the issues with taking a product from concept to commercialization. Symbolics did some efforts to reduce cost and

increase integration with new board and chip technologies, but the lack of expertise in productization was clear. Designers with little industry experience will tend to added features or set specification that would drive up costs significantly, reduce reliability and cause delays unnecessarily for things that would only bring minimal returns. It is very different to create prototypes that "pretty much" worked under ideal conditions compared to designing a product that would be robust under adverse operating conditions and could be mass produced at a reasonable price. The lack of attention to this area was one of the reasons why they were not able to lower prices and became uncompetitive with companies like Sun and Apollo when their low cost workstations came to market. The inattention to manufacturing also resulted in multiple product delays that cause the company to miss market windows of opportunity when the other workstation companies where just getting off the ground. Paying attention to this aspect may not have save the company, but it certainly would have gave them a better chance to fight against the threats that arose during the troubled years of the company.

3.6 Sales and Marketing

We have chosen to discuss the areas of sales and marketing jointly. The main reason for this is that they were viewed as one function by Symbolics managers. Both were viewed as inconveniences, necessary in order to financially support development.

Sales and marketing played a small role at Symbolics. They were not consulted during the development process, but brought in only when a product was being manufactured. Their job was to take the product and transform it into revenue to support the development of new products. Little, if any, attention was paid to the importance of sales and marketing as a link to the customers. These divisions were the part of the company closest in contact with users, and they had better insight into the concerns and requirements of Symbolics buyers than the Research and Development department. However, by not taking the concerns of sales and marketing into account when planning new development proposals, Symbolics isolated itself from its customers.

There were several reasons for Symbolics reluctance to take sales' and marketing's concern into account when developing products. One was the academic notion of technical superiority held by most of the research staff (see the earlier section on R&D). Clean design was viewed as vital to a "successful" product, and any other criteria conflicting with this goal was easily discounted.

The founders of Symbolics were the original users of LISP machines. Their first product, the CONS, had in fact been developed at MIT in order for the users to be able to do their research. As the product line developed, the Symbolics developers continued to use Symbolics products exclusively. Many Symbolics developers saw themselves as Symbolics users. This made it easy to overlook the need for input

from customers. A sense of having the same needs as the customers, and thus a always knowing what the customer wanted, made it hard for Symbolics developers and managers to see the value of information in given by sales and marketing.

3.7 Finance and Control

The financial success of the a company is based not only on bringing in money, both in the form of revenue and investment, but also on controlling the costs associated with company operations and production of products. From the earlier sections, we see that Symbolics did a poor job of managing both sides of this equation. In this section, we will concentrate mainly on managing the cost aspect of this equation.

When a company is small, there is less of a need to install financial control processes for its operations, since the team is small and the problem is manageable. In fact, when Symbolics was starting out and didn't have a lot of funding, it did a great job in boot strapping the company with very little resources. Unfortunately, when Symbolics became a relatively successful company with over 1000 employees, it still did not institute the financial management processes or controls that is required in running a company of its size. With spending in the tens of millions a year, it is crucial that management is aware of how much the company is spending and how its allocated. It is also important to create controls within the system to make sure that the spending is aligned with the corporate goals and the business environment at the time.

Once funds were available, Symbolics was spending money like a lottery winner with new found riches. They moved into lavish offices, bought expensive furniture, provided company cars, offer extravagant perks, and paid higher than industry salaries. Symbolics was not alone in this type of behavior. From what we saw, all the AI companies of the time acted in the same way. This may have been a reason why that all seemed to end up failing a few years later.

Although the amount of spending was a major issue for the company, the larger problem may have been the allocating of the spending. As we mentioned earlier, the allocation of spending needs to be directly aligned with the business goals of the company. Since Symbolics was such a technology focused company, even in times of financial difficulty, it focused its spending on R&D, and reduced costs on the marketing and sales aspect of the business. And since there were no clear goals in the company, the cost cutting that was done on the technology side was on things that looked to have been strategically critical to the long-term success of the company. When finances got really bad, the investors started to come in and

force cost cutting measures that may have contributed to the ultimate downfall of the company. The investors forced Symbolics to delay or cut projects relating to an integrated VLSI chip which would have allowed the company to compete in the lower cost workstation market. They also forced Symbolics to eliminate software projects with Intel to provide LISP deployment platforms on x86 systems. There were some real conflicts and inconsistencies in the management thinking and the investors beliefs that arose during the course of the companies life that were forced to manifest itself as a result of poor financial management and control.

If there were proper financial control processes in place in the company, they would likely have prioritized their activities much better to avoid getting in financial crunches, and when it them, would know exactly which non-priority projects or personnel to reduce. We believe that part of the reason for the lack of focus on the financial aspects of the business is due to the academic background of the employees and the management. They came from an environment where one requests for funding, gets it and spends it as he chooses with little or no regard for long-term consequences of the spending. In the business environment, a company can not do that an survive.

3.8 Human Resources

In his book *Soul of a New Machine* Tracy Kidder describes Tom West looking at the boards of a new DEC computer. West believed that the bureaucratic organization of DEC was reflected in the board. It is important to realize that Kidder's theory supports MacKenzie's argument that natural trajectories for technology does not exist. Kidder argues that the background of a developer will significantly influence the technology he creates. This is mutually exclusive with technological development corresponding to the inherent possibilities of the technology. Kidder verifies that social factors will play a role in technical innovation.

Like in the case of DEC, the organizational structure and people working at Symbolics were reflected in their products. There are a few salient high-level observations gained from the technical details of the machines:

- The machines were for high-end development
- The machines were optimized for Lisp
- The machines were well-documented and “open”
- The machines included every possible feature
- The design was clean
- The machines (especially the network architecture) were designed to be quiet

Each of these design characteristics can be linked to traits of the technical people in the firm.

Symbolics from the beginning focused on the market of high-end development machines. These were what they designed best. Developers themselves, they knew the design requirements intimately, and using their own tools led to new ideas on how to improve them. Furthermore, development machines are more demanding to design than delivery machines. Symbolics focused on the problem that its developers found most challenging, and therefore interesting. Furthermore, through improving their products they also improved their own working environment. One can argue that from the first machine created at the AI Lab, Symbolics developers were never quite weaned off a belief that since they were users of the technology they were also good guides to what the customers needed.

LISP has always been most adamantly supported by academics, particularly academics in the Artificial Intelligence field. (LISP was also a natural choice if one assumes that the target market was always within the AI field, although this assumption is challenged by Noftsker, a founder and CEO). David A. Moon's assertion that "highly complex software… could safely be assumed to be written in LISP" was not gernerally true given that little, if any, commercial database development was done in LISP. However, for a research and development department with close links to the AI research community, LISP might have seemed mroe prevalent than was actually the case in the rest of the industry. The Symbolics programmers were trained in LISP and they were part of a subculture where LISP was dominating. This gave them a very different world view than for instance IBM developers, who at that time focused on complex business applications not written in LISP.

The Symbolics documentation was outstanding, another characteristic common to academics. The online hypertext documents and printed manuals developed by Symbolics engineers won several awards, and former Symbolics employees still treasure them. An academic is completely dependent on documenting his work. Only through publication can he/she receive credit and acknowledgment. The extensive documentation of Symbolics products, well beyond the standards of the computer industry, shows sign of this academic mind set. Only though documenting their products can these be fully appreciated.

The Lisp machine operating system was open for many years. By "open," we mean that one could view the source code for the operating system, change the code, and recompile it. Symbolics' OS was open for many years until it became clear that Texas Instruments was reverse-engineering the code.[17] In general, commercial software companies do not distribute source code for their products. This is strongly indicative of the background of the technologists in the firm. Sharing and distributing knowledge are fundamental ideas in the academic community. The ideal of a large free and commonly maintained body of source code is part of the academic computer culture.

[17] Russell Noftsker, CEO and chairman, interview

The feature creep evident in the design of the Symbolics machines is also a result of the academic backgrounds of the designers. All of the Symbolics founders were researchers or had spent much time in a research setting. The first machine was in fact the subject of Thomas F. Knight's master's thesis. Many of their first products had several new and revolutionary components such as special software, high quality monitors, and mice (which were manufactured especially for Symbolics because they were not yet in common use). All these new innovations drove up the price of the computers and made several models late for shipping. Symbolics seemed unable to focus on one or two improvements to each machine. It is also not clear that the customers considered the innovations worth the increased price. Symbolics developed new technology for the sake of innovation, because they believed it to be useful or because the problem was an interesting one.

The designers adhered to a philosophy to “do it right and don't worry overmuch about the cost”[18] obeying a principle that is natural given their background. The designs of Symbolics machines are amazingly clean. Everything, from application software to the operating system were written in LISP. Even the microcode contained LISP instructions. There were no major hacks and shortcuts. The overall design was consistent and well layed out, using Symbolics own specially designed dialect of LISP. The price they paid for this elegance was compatibility. The beautifully written software needed specific hardware in order to run. This isolated Symbolics from software vendors and made their products harder to sell.

The machines (especially the network architecture) were designed to be quiet. The prototype of their first product, the LISP machine, was a machine designed to satisfy the designers’ needs. Symbolics was a company founded by programmers and run by programmers. This went further than offering the programmer the necessary address space (see technical discussion). The LISP machine was specifically designed to keep the noise disk drive and LISP processor out of the programmer's office. Only the monitor and keyboard was kept in the programmers workspace, communicating with the other components via Ethernet. This was a design radically different form the Xerox ALTO and other contemporary machines. Only a programmer spending hours in front of a computer every day would have prioritized noise reduction in the design process.

4 Timing Factors

Despite the sudden change in Symbolics fortunes, many of the fundamental problems of heterogeneous engineering at Symbolics, such as a failure to set and communicate clear goals and to engineer developers to use their machines, existed since the founding of the company and in some cases before the founding of the company.

[18] David A. Moon, software engineer, email

Why was the success of Symbolics so dramatic and why was the decline so sudden given that the core problems with the company existed from the beginnining?

- Symbolics saturated the Lisp machine market (and did not engineer growth in the overall segment).
- SDI (or "Star Wars") funding kept Symbolics above water by provided indirect funding.
- Most importantly, Sun Workstations came to be a viable alternative to Symbolics Workstations.

By 1988, the proliferation of LISP machines may have exceeded the proliferation of LISP programmers. Because Symbolics always focused on high-end Lisp development machines, Symbolics' success was bounded above by the number of high-end Lisp programmers. Francis Feeney, assistant general counsel at Symbolics, notes "When we were growing so quickly here internally, I don't think anybody stopped and said, 'Well, wait a minute, once we saturate that fixed marketI is there going to be a [broader] market for our system?'"[19] Symbolics' management put on their blinders and ignored this important issue. They clearly did not engineer a broader need for their systems. Within eight years of the founding of the company, Symbolics had begun to see the effects of selling a Lisp machine to everyone who had $100,000 to pay for one.

One of the reasons that Symbolics was so successful before 1986 is that the expert-systems market was subsidized by funding for the Strategic Defense Initiative. SDI funding did not go directly to Symbolics. However, SDI funds did go to many of SymbolicsU customers who were developing expert-systems. The demand for Lisp machines in the early 1980s was largely fueled by this phenomenon. Symbolics was the first Lisp machine manufacturer to market and they always beat TI, Xerox, and LMI within the Lisp machine segment. Furthermore, in the early 1980s, no Unix workstations were considered viable alternatives as platforms for developing Lisp systems.

The market created by funding from SDI was quite forgiving. The government was interested in creating complex Lisp programs and Symbolics machines were the leading alternative at that time. Officials who allocated funds for SDI did not demand cost-effective results from their research funds and hence the expert-systems companies boomed during this period.

Another reason that Lisp machines were in high demand (despite a general lack of results) is due to a "natural teajectory" phenomenon. AI, and especially expert systems, were seen as the future of computing. This is due to hype generated by Feigenbaum, a Stanford University professor. Tom Knight claims that Artificial Intelligence was oversold primarily because Feigenbaum fueled outrageous hype that

[19] *Misadventures in AI: What Went Wrong at Symbolics*, Andrea Cohen, Electronic Business, September 1, 1988, p. 86.

computers would be able, for example, to replace medical ddoctors within 10 years. Adidtional hype was generated by the Japanese governmentUs sponsoring of the "Fifth Generation" project. This project, essentially a massive effort by the Japanese to develop machines that think, sparked a nationalistic chord in America. SDI funding in some ways was a means to hedge the possible ramifications of the "superior" AI technology of Japan.

The rapid growth of the expert-systems market essentially could mask the problems within Symbolics. Competing in a market with few competitors that is doubling every year is easy. For a few years, Symbolics major barrier to revenues was producing the machines. Even while growing exponentially, Symbolics was plagued by operations problems and could not deliver machines to every customer that wanted them. Due to huge demand, almost no amount of mistakes could prevent Symbolics from succeeding in the early 1980s.

By the late 1980s, it was clear that the Fifth Generation project was no closer to developing strong AI than the Americans were. This bubble based on the hype generated by Feigenbaum and the Japanese project ultimately burst, causing a general decline in the expert systems market and a drop in funding by the government.

Perhaps most harmful to Symbolics was the introduction of Sun workstations as viable alternatives for Lisp development machines. The hard-coded, apparently immutable culture to “do it right and don't worry overmuch about the cost”[20] forced Symbolics to sacrifice not only low cost but also compatibility and on-time delievery. These problems primarily stemmed from the highly academic culture at Symbolics.

Ignoring cost, on-time delivery, and compatibility left room for Sun to add significant value in the Lisp machine marketplace. These are the three strongest reasons that Sun could leverage its general-purpose workstations into Symbolics’ segment.

It is important to note exactly what the costs of the various machines were. The difference in cost could be measured not in percent, but in hundreds of percent. While two machines may be considered competitive if one is 20% more than another, charging 200%-500% extra for the Symbolics machine made the product uncompetive. Newquist stressed this:

> *“Given the cost of individual LISP workstations, it was inevitable that users would look for alternate sources to deliver and develop their AI applications. General-purpose workstation vendors such as Sun Microsystems and Apollo computers have been quick to fill this bill.”*
>
> Harvey P. Newquist, *AI Trends ’88*, No. 48

[20] David A. Moon, software engineer, email

The cost of a Symbolics 3600-series system in 1988 ranged from $36,000 for a low-end system to $125,000 for a high-end system[21]. The Sun-X computer started at only $14,000 and could run LISP well enough for development. In addition this machine could be used for many other applications.

A primary component of addressing needs of IT departments is compatibility. IT departments are based on legacy systems and they will not scrap the current system until it is absolutely mandatory. Symbolics sacrificed compatibility for clean design and therefore lost favor within the IT community. For many years, Symbolics could sell incompatible machines to expert-systems researchers, but incompatibilty was not an option when selling to IT customers.

Finally, late delivery plagued Symbolics. Late delivery is yet another symptom of Symbolics' academic culture. The 3600 series was about one year late, and the I-Machines (40-bit) were about 2-years late. Furthermore, Symbolics could not manufacture machines adequately during the years of high growth. Business customers cannot tolerate late delivery of machines. Furthermore, when the release SymbolicsU major new product was two years late, they had given an irreparably large window of opportunity to their competition.

What allowed Symbolics to sidestep the competitive problems with Sun during the growth years? Primarily, Sun machines were not adequate to do the job of Sun machines until the mid-1980s. Other Lisp machine vendors were no more competitive than Symbolics. LMI was also run by academics. Texas Instruments was primarily in the business to reverse-engineer SymbolicsU machines (which released source code for many years), and Xerox did not consider the Lisp machine market to be its focus. SunUs cheap and adequate Lisp development platform, based on a general-purpose Unix machine, ultimately forced every Lisp machine vendor out of the market.

5 Conclusion

In this paper we have introduced a model of heterogeneous engineering well suited for commercial companies. Analyzing Symbolics in the context of this model, one can clearly see why the company failed. Symbolics failed to set clear goals, and this made it difficult for the different divisions of the company to work together. Furthermore, the academic culture of the company let it to focus almost excusively on research ond development. This led to lack of resources in the other divisions of the company. Furthermore, little attention and care was given non-research functins. As a consequenceof this, the company did not see problems in other divisions, and collapsed when problems in these made it unable to pay for its research.

At Symbolics, Goals varied in order to support the current direction of Research and Development.

Symbolics is a classic example of a company failing at heterogeneous engineering. Focusing exclusively on the technical aspects of engineering led to great technical innovation. However, Symbolics did not successfully engineer its environment, custormers, competitors and the market. This made the company unable to achieve long term success.

[21] Harvey P. Newquist, *AI Trends '88*, No. 51

6 Bibliography

- Bailey, Douglas M. *Symbolics Coming up With New Line, Analysts Believe Artificial-Intelligence Firm Will Broaden Its Market Base*, The Boston Globe, July 29, 1986.
- Bottorff, Dana. *Mission Impossible?*, New England Business, No. 6, 1988.
- Carton, Barbara. *Symbolics Names new Chairman, CEO*, The Boston Globe, May 24, 1988.
- Cohen, Andrea. *Misadventures in AI: What Went Wrong at Symbolics*, Electronic Business, September 1, 1988.
- Goldstein, Mark L. *Russell Noftsker: A Scientist Lost in the World of Marketing*, Industry Week, November 16, 1987.
- Hammonds, Keith H. *Symbolics: The Soul of a New Regime*, Business Week, June 13, 1988.
- Kidder, Tracy. The Soul of a New Machine, Avon Books, New York, New York, 1981.
- Knight, Thomas F., Senoir Research Scientist, MIT's AI Laboratory. Interview.
- Knight, Thomas F. *Implementation of a List Processing Machine*. MIT EECS Master Thesis, January, 1979.
- MacKenzie, Donald. Inventing Accuracy, The MIT Press, Cambridge, Massachusetts, 1990.
- McCain, Nina. *Computers Seen Replacing MDs, Lawyers*, The Boston Globe, November 1, 1984.
- Moon, David A. Software engineer and founder, Symbolics. Email December 2, 1998.
- Moon, David A. Software engineer and founder, Symbolics. Email December 3, 1998.
- Newquist, Harvey P. Editor of *AI Trends*. Interview.
- Newquist, Harvey P. *AI Trends '85: A comprehensive Annual Report of the AI Industry*, DM Data, 1985.
- Newquist, Harvey P. *AI Trends '86: A comprehensive Annual Report of the AI Industry*, DM Data, 1986.
- Newquist, Harvey P. *AI Trends '87: A comprehensive Annual Report of the AI Industry*, DM Data, 1987.
- Newquist, Harvey P. *AI Trends '88: A comprehensive Annual Report of the AI Industry*, DM Data, 1988.
- Newquist, Harvey P. The Brain Makers, Sams, 1994.
- Noftsker, Russell. CEO, chairman of the board and founder, Symbolics. Interview.
- Pitta, Julie. *Where Lisp Slipped*, Forbes, October 16, 1989.
- Rosenberg, Ronald. *AI Alley's Longest Winter*. The Boston Globe, December 18, 1988.

- Rosenberg, Ronald. *Artificial Intelligence Industry Observer Gives Its Days of Glory and Official Record*. The Boston Globe, June 15, 1994.
- Rosenberg, Ronald. *A Simpler Kind of Computer Talk*, The Boston Globe, October 6, 1980.
- Rosenberg, Ronald. *Making Computers User-Friendly When Cobol and Basic Aren't Enough*, The Boston Globe, September 8, 1981.
- Rosenberg, Ronald. *Symbolics Scores an AI Victory*, The Boston Globe, August 6, 1985.
- Rosenberg, Ronald. *Tough Times for Pioneers in Artificial Intelligence*, The Boston Globe, September 16, 1986.
- Shrobe, Howard. Assoc. Director of MIT's AI Laboratory. Interview.
- Symbolics Online Museum. Http://home.brightware.com/~rwk/symbolics/
- Symbolics Technical Overview. Http://www.lavielle.com/~joswig/symbolic-computing.html
- Vannah, Thomas. *Driving Symbolics to market*, New England Business, No. 3, 1989.
- Winston, Patrick. Former director of MIT's AI Laboratory. Interview.