70

The Technology Balance Sheet

C. Gordon Bell with John E. McNamara

SPECIAL REPORT

Just as it is essential to understand an organization's financial health, it is equally necessary to understand and measure its technological health. The first section of the chapter describes the technology balance sheet, a useful approach to measuring a company's technology. The second section presents a number of classic technology-related flaws, ranging from requiring infinite technology (i.e., attempting to develop a product that is predicated on a fundamental discovery or technological breakthrough) to having no sustaining technology. The final section lists the rules for evaluating a new venture's technological position at the end of the concept and seed stages.

A start-up company must examine every facet of the technology that it needs to build a product, and then rank each technology source as objectively as possible.

The Technology Balance Sheet

The technology balance sheet evaluates each of 12 key dimensions of a start-up's technology. Readers may notice that the dimensions used on the technology balance sheet to assess a firm's technology are very similar to the dimensions used throughout the book to assess a firm's overall status.

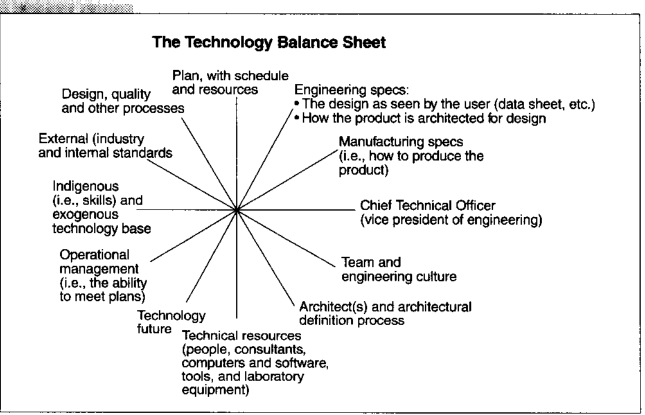
The Technology Balance Sheet
Plan, with schedule and resources
Engineering specs:
• The design as seen by the user (data sheet, etc.)
• How the product is architected for design
Manufacturing specs (i.e., how to produce the product)
Chief Technical Officer (vice president of engineering)
Team and engineering culture
Architect(s) and architectural definition process
Technical resources (people, consultants, computers and software, tools, and laboratory equipment)
Technology future
Operational management (i.e., the ability to meet plans)
Indigenous (i.e., skills) and exogenous technology base
External (industry and internal standards
Design, quality and other processes

Fig 1. Technology balance sheet plotted as a relational graph.

Figure 1 lists the 12 dimensions to be considered and measured:

- Technology base
- Standards
- Design, quality, and other processes
- Plan, with schedule and resources
- Engineering specifications
- Manufacturing specifications
- Chief technical officer
- Team and engineering culture
- Architecture
- Technical resources
- Technology future
- Operational management

These dimensions will be discussed in the following subsections.

Technology

The technology dimension includes internal and external sources of components, plus "know-how," as represented by critical personnel, patents, processes, etc. The company must examine every facet of the technology that it needs to build a product and then rank each technology source as objectively as possible.

Standards

Standards should be regarded as a critical aspect of product design. Establishing uniform ways of doing things (such as having an exact dialect of a language for expressing a program and having programming style con-

C. Gordon Bell with John E. McNamara, "The Technology Balance Sheet," Chapter 6 from "High Tech Ventures: The Guide for Entrepreneurial Success". Copyright © 1991 by Addison Wesley Publishing Co. Reprinted by permission of Addison-Wesley Publishing Co. Inc., Route 128, Reading, MA 01867. Tel: (617) 944-3700; Fax: (617) 944-9338. To order this book, call 1-800-238-9682.

C. Gordon Bell has been involved in more than 20 start-ups in addition to his 20 years with Digital Equipment Corp. John McNamara, currently with Stratus Computer Inc., has worked on R&D teams at MIT's Lincoln Library and at Digital Equipment.

ventions) permits rapid progress because standardized components can be interconnected and built on one another. Although standards are inherently constraining, Dave Nelson (one of Apollo's founders) believes that constraints are what really breed creativity. In designing a product, it is inherently more difficult to start with a blank slate than to start with some restrictions, because in the absence of established criteria, nearly anything is possible. Effort will therefore be squandered exploring an almost infinite number of options rather than channeled and focused in the most productive directions.

A start-up (or a company of any size, for that matter) must understand and implement both external and internal standards. Major aspects of product design are determined by external (industry) standards covering such areas as inputs, outputs, cost (in memory size), and speed (50,000 lines per second). For example, a compiler may be specified as having to accept ANSI (American National Standards Institute) standard C language input, produce code for the Motorola 88000 chipset that is better than the existing compilers, occupy no more than 100 kilobytes of memory, and compile at over 50,000 lines of code per second.

Internal standards are equally important and range from how logic design or programming is done to line width on printed circuit boards. Internal standards must be specified, published, and enforced in a formal manner. For instance, when Digital Equipment Corporation (DEC) first started, the Engineering Committee took responsibility for creating a set of design standards that covered everything from how a physical environment would be specified and tested (power, temperature, humidity, etc.) to how a copyright statement would be placed in memories and programs. Internal standards also include a list of the components that are permissible in new designs.

Upon seeing such standards and component lists, the first reaction of most engineers is that they are bureaucratic and constrain creativity. However, standards are simply a statement of decisions that have been made regarding good practice, which means the designer doesn't have to think about these more mundane aspects of a design (such as the temperature at which the product should be designed to operate) and is therefore free to concentrate on the truly creative aspects.

Design Process

The design process, which specifies what tools engineers use to create and check each part of their product design, must be documented and managed. The design process is intimately tied to the resources a company has to aid designers. Much has been written about software engineering and there are any number of valid models for how programming should be done. The important thing is simply to pick a model that is appropriate for the team and operate according to it.

The Software Engineering Institute (Humphrey, 1989; derived from Deming and Juran) has established a five-level ranking to characterize how effectively a team is functioning in terms of its process capability:

1. *Initial:* There is an ad hoc process. Formal procedures may exist, but there is no management mechanism for tracking results against the procedures. The team rarely makes and meets plans.
2. *Repeatable:* A process exists that deals effectively with routine programs but produces unpredictable results with new programs or new tools.
3. *Defined:* A qualitative description of the process exists.
4. *Managed:* The process contains a minimum set of measurements to define quality and cost; a process baseline exists; etc.
5. *Optimizing:* There exist sufficient quantitative measures for each part of the process to allow the process to be completely understood and fine-tuned.

Humphrey describes a method for evaluating a company's process capabilities and also recommends various processes standards, and methods for attaining software process control. The organization with which he is associated, the Software Engineering Institute, can audit a firm to determine its level of process control, and some members of the institute did so as part of a 1990 trip to Japan sponsored by the Ministry of International Trade and Industry. While there, they found that many large Japanese companies are operating at the highest level in the above ranking (level 5, optimizing), enabling them to achieve quality and productivity levels more than twice that of their U.S. counterparts.

> The resources list must cover both the resources for developing the product itself and those for developing required manufacturing and design processes.

Engineering Plan

The engineering plan includes the schedule and a list of the resources required. The resources list must cover both the resources for developing the product itself and those for developing any of the manufacturing and design processes that the product requires. The important thing about an engineering plan is that the schedule be realistic. Developing a truly realistic schedule is almost impossible if the product has never before been attempted; it is merely very difficult if the product has been attempted previously, but the team has never before worked together. In the latter case, each team member's ability to establish a realistic schedule for his or her portion of the work will probably be untested. In terms of the process-capability levels outlined by Humphrey, it is unlikely that such a software team in a new venture could get above level 2 (repeatable) by the time it ships its first product.

Product-gestation time gets ingrained in the people and the company. Their ideas regarding product-gestation time are often based directly on the lead times at a larger firm, which are guaranteed to be much, much longer. One of the most important aspects of an engineering culture is to establish an accurate but responsive ability to schedule. There are four ways to schedule a project:

Optimistically: Put enormous pressure on the team by preparing an aggressive schedule that the team believes can only be met if everything goes right.

Pessimistically: Build so many delays and contingencies into the schedule that the schedule will certainly be met (an approach unlikely to be used by a start-up).

Realistically: Allow for an appropriate number of contingencies, which will become possible when the team is mature enough and understands the project and each other well enough. However, it often happens that everyone up the chain of command then adds a contingency, and the net result is a bloated, pessimistic schedule (again, not typical of start-ups). With realistic scheduling, the company may end up with two scheduling the optimistic one and the one with the contingencies added.

Running blind: Work on the project until it gets done. The firm that uses this approach had better start with lots of money, be able to raise more money easily, and have plenty of extra resources.

In the final analysis, schedules really don't always work Any critical schedule milestone must coincide with an immovable deadline such as a demonstration to the board, a trade show, a funding event, or a customer shipment. If customer shipment serves as a deadline, quality must always be used to control shipment.

Engineering and Manufacturing Specifications

The engineering and manufacturing specifications describe the product in several ways. First, they describe its external specification, or the product's function, including performance, as seen by a user. Second, they describe its internal specification, or the product's structure and internal function as seen by the engineering team (i.e., a set of components to be designed). Finally, when the product has been fully specified both externally and internally, manufacturing requires process and product specifications describing how the product will actually be built and tested.

Chief Technical Officer

The chief technical officer (CTO), or engineering vice President, is the technical leader in charge of implementing the product. This person is ultimately responsible for all products and is the CEO for the engineering organization. Thus, his or her general qualifications must parallel those of the CEO because the CTO is the "clock" and "standards setter" for engineering.

The company should have selected its CTO by the end of the seed stage, and if it is tackling a technologically difficult product, the CTO must be on board from the start. Funding a high-tech venture without a CTO is extremely risky because he or she is the individual responsible for ensuring that the product is really feasible at the price, quality, schedule, and resource level specified in the business plan.

Engineering Team and Culture

The engineering team and culture are just beginning to form by the end of the seed stage, since at this point, a complete team has yet to be lured and the head of engineering may not even be on board. The organizational structure is quite important because the CTO may have positioned himself or herself as a bottleneck by assuming responsibility for all intergroup problem resolution. As with any organization, theory X, Y, and Z will all work. I do not favor highly top-down engineering organizations because they do not bring out the creativity of the people doing the work. Furthermore, top-down structures eliminate critical intraorganization communication. Worst of all, top-down organizations usually do not engender commitment to schedule, resources, and product on the part of the responsible engineers.

Architecture

The term architecture was coined in 1964 by the IBM System/360 design team to describe a computer's instruction set, or how the computer appeared to a program (or programmer). Architecture is now used in a broader sense that encompasses both "external architecture" and "internal architecture." The external architecture describes the general function of any computer component (i.e., what it does)—such as the instruction-set architecture, operating system, compiler, a network protocol, or spreadsheet—and how this component appears to anyone using it. The internal architecture (or "realization") forms the blueprint for how the components that create the external architecture are implemented; it is what a development team designs and builds.

It is therefore vital to have a product architect who can both define the product externally and be able to play a major role in decomposing it for realization and then engineering it. His or her responsibility for product architecture applies equally to all levels of hardware and software. Thus, the product architect is likely to be the most critical person in the engineering group, including the CTO. [My own background and biases as an architect may account for this belief.]

The architect's key job is to guide the product's implementation and evolution over its lifetime. The lifetime of a good architecture will be considerable, and the company fortunate enough to have chosen a good architect and architecture will profit immeasurably. Much of DEC's success during its first three decades (1957-1987) was based on constant and evolving architectures for its minis, including the VAX. System/360 hardware and software systems and their successors were the basis of most of IBM's revenues and profits for a similar period. More recently, the Apple II and Macintosh architectures have each prospered for over a decade. In 1990, Sun Microsystems has been attempting to repeat the success of these predecessor architectures by establishing the SPARC architecture as the standard for workstation-class computers.

Although most of the examples cited above involve hardware engineering, the same sort of architectural integrity must also be maintained for software. At Microsoft, every product, such as Word or Excel, has a single architect who maintains the product's integrity (and is usually its chief implementer as well). When responsibility for a product is diffused, as in the case of Fortran or UNIX, by placing it in the hands of some amorphous, committee-like group that is pushed around by numerous standards organizations, the product's efficacy declines and its ability to evolve may be stymied.

Faced with the need to cut function in order to meet schedule and resource constraints, it is best to sacrifice some product features rather than performance.

In my view, lack of a good architect, or lack ofa suitable architecture, is the fatal flaw in many high-tech ventures. Although at first, the product architect may be the CTO or even the CEO, ultimately, someone within the engineering organization must assume responsibility for maintaining the efficacy of the product's external specification, especially with respect to how that product is changed as it is implemented in succeeding product generations. In some cases, as in the example of Ardent's Titan workstation described later in the chapter, several architects may be required as a product is broken into various parts.

Not having an architect is quite risky, because it leaves the product's definition to some nebulous process or to a group that gropes with the product design, as I recently saw in a company building a multimedia system. Not having a way to manage the product's design and delegate it to those who must do the design is almost always fatal. One of the biggest dangers is overcommitment. When a technically difficult project begins, and one person functions as CEO, CTO, and architect, the company, engineering group, and product are all likely to be out of control unless the firm has a sufficiently strong staff, including a chief operating officer. In a start-up, such a project must have a full-time architect who will also play a major role in the product's design.

In a recent case, the architectural concept for a product was superb, but the architect had four problems that thwarted effective implementation of the architecture. These problems, which are typical of many architects, were as follows:

- The architect lacked an understanding of the specific benchmarks by which the product would be judged in the marketplace.
- He had trouble finishing detailed specifications for the product, leaving the engineering organization with only a fuzzy idea of what it was supposed to be designing.
- He was not knowledgeable about implementation, which meant that the architecture could not be implemented within a reasonable time and at a reasonable product cost.
- His poor interpersonal skills made it hard to keep the actual implementation in synchronization with an ambiguous specification. (This was both his problem and that of the CTO.)

Although not having any architect can be a problem, having too many architects can also be a problem, as illustrated by the following example.

Venus. On Friday, Aug. 13, 1982, I went to a design review for Venus (VAX 8600). The review, attended by several hundred people, focused on the schedule and the risk involved in not getting the chip layouts to the gate array supplier. I asked whether the design had been simulated or thoroughly reviewed. It hadn't, since the group was in such a hurry to meet the schedule that they wanted to skip the checking stage. On Saturday, I visited the project team to talk with its members and found that the management didn't understand the project and that four individuals each regarded himself as the project's sole architect and wanted the credit. The project had about four design styles, because it consisted of four large subsystems.

By Thursday, no one wanted credit. Within six months, the project was brought under control through many management changes and the introduction of a design process that required the use of design reviews and simulation to ensure the correctness of the design prior to building the hardware. The product ultimately shipped two years later than scheduled, whereas, left on its original course, it would probably never have shipped. The project ended up with an organizational structure consisting of four architects and a lead architect to resolve the conflicts among the group and finish the design.

Despite all these dire warnings about architecture, it is possible for a talented team of architects to work well together and produce an excellent product. The following story about the architecture and development team for the Ardent workstation (Titan) illustrates this point and shows how an architectural task can successfully be broken up.

The Ardent Titan architecture and development team. Given the complexity of a graphics supercomputer, Ardent had to break the definition and responsibility for various elements of its design into independent parts and architectures. The entire project worked well when all the roles were defined. The architecture was the basis for three products, including Titan, which could be evolved through three generations over a five-year period.

The company's chief hardware architect, GSM, was responsible both for defining the instruction-set architecture externally and for defining the internal architecture (how the parts of the entire computer fitted together using a core bus). GSM also defined the processor's internal supercomputer architecture and took on many of the difficult processor-design tasks, although he did not lead the hardware project nor was he directly responsible for implementing the processor. After the first version of the machine was introduced, GSM led benchmarking and observed the machine in real applications. This was critical for the design of the next implementation.

JRA was the architect, leader, and key implementer for the development of the parallelizing vectorizing compiler. Having a single individual be responsible both for the architecture and for leading the implementation thereof was an ideal case. The languages architecture and debugger were the responsibility of SCJ.

WT was the architect, leader, and key implementer for the development of UNIX, which supported the Titan hardware and provided a program environment for the compiler and other applications programs.

TD was the architect, leader, and key implementer for the graphics hardware, while WW designed and built the software pipeline to transform and display 3-D objects. MK was the architect and chief implementer for the graphics library.

All architects/implementers had to cooperate on determining each architectural interface and on the entire design.

Technical Resources

The next dimension on the technology balance sheet is technical resources. This essential category includes people, equipment (both computers and networks), and software to run the engineering enterprise (i.e., operating systems, languages, computer-aided design [CAD] programs, and software licenses). Of all the resources, the technical staff is the most important.

The organization's first hires have to be great, because really good engineers like to be involved with other good (or even better) engineers. Great people hire even better people.

The company's hiring ability determines the quality of the staff. All firms, regardless of how well they may be managed, find that hiring grade-A people takes much longer than anyone had planned. The key to hiring is having the right sources. The most effective approach is to develop a network of contacts with the best people working in each area such that recruiting is by word of mouth. A technical advisory board can be one of the company's greatest hiring assets.

The organization's first hires have to be great because really good engineers like to be involved with other good (or even better) engineers and are intolerant of bozos and turkeys. Great people hire even better people. Poor people hire even poorer people. (This is the pygmy theory of hiring.) Furthermore, because the company can expect to acquire its share of average people merely in the course of making mlnor hiring errors, it must never deliberately hire average people just to fill slots.

The following stories illustrate some interesting approaches to hiring used by start-ups, along with critical observations on hiring the engineering team.

Ardent and Stellar. Both Ardent and Stellar allowed three months to form their team, but it took six months before the companies were fully staffed. Each firm established a technical advisory board to aid in the product definition, and these boards were the key to finding and recruiting the best people.

Wavetracer. Richard Fiorintino, CEO of Wavetracer, recruited engineers by sending a personal mailing to surrounding towns, at a cost of less than $500. Recruiting firms (headhunters) are a last resort, because they will be costly and error-prone. The start-up's leadership team itself is clearly responsible for staffing, no matter what the formal hiring channel may be.

Objectivity. Objectivity, a start-up building an object-oriented database, had a relatively long seed stage, during which it both designed its product and hired its key engineering leaders. When the product development stage started, Objectivity was ready to do full-scale recruiting because it knew what it was going to build and how it was going to build it.

It first created a database of everyone working in the field who had experience in theory, use, and development. It grouped the list according to technology and gave priority to people who had built specialized databases. Using all available sources, Objectivity did a forced ranking of everyone on the list, whether they were available or not. The process was carried out with peers via phone calls and in direct interviews, which were scheduled two nights per week and on weekends.

But the hiring process did not end when the candidates made a commitment and joined the organization. Instead, the process continued for a six-month probationary period, during which each new hire worked with the team and was given an opportunity to discuss his or her design and engineering philosophy at length. Several candidates ultimately left during the probationary period.

Objectivity's scheme had many advantages: the company found a large pool of engineers, got the best people to head the team, formed the team itself gained an in-depth understanding of its potential competitors, and established links with potential buyers. Objectivity's approach did have one flaw, however: by using the grapevine, the organization risked disclosing basic technology and prompting other firms to start up.

Visix Software. Visix built a high-quality, high-performance platform for building graphical-interface, networked applications. Its desktop for UNIX, Looking Glass, extends that of the Apple Macintosh to handle networking. Visix achieved product quality by implementing a rigorous hiring process, by managing to keep a small team together over a five-year implementation period, and by having a single product architect. A key step in the hiring process was to review the code of each potential software engineer. Any engineer reluctant to show his or her work to fellow engineers is a likely loser.

GO Corp. According to Robert Carr, who heads software development at GO Corp. (Carr, 1989), "All good software these days gets done through teamwork." He suggests the following approach:

1. Define the development style. Choices include the collegial model (for people at a more senior level) versus leadership by a chief programmer (where the team has less experience).

2. Hire the best first. Others will be turned off if turkeys are present. Worse yet, the turkeys will want to hire pygmies.

3. Focus on interpersonal skills. Teamwork is number one. Meet with eight to 10 other staff members. Don't push lukewarm people. Listen to what your troops are saying.

4. Don't be afraid to rob the cradle. A 22-year-old may have 10 years of experience.

5. Hire people who have shipped products and been through the cycle, including support and feedback.

6. Don't skimp on salaries. Staff members should receive stock equal to half their salaries. It is better to hire the best people and pay them well than to hire a greater number of poor people and pay them poorly.

Micrografx. Paul Grayson, chairman of Micrografx, recommends that a company create a development environment that fosters excellence and sees three types of reward that can help create such an environment (Grayson, 1989):

1. *Cash:* A royalty can be paid based on 2 percent of sales, with the lead programmer getting 1 percent. Bonuses can be awarded for project completion. .

2. *Recognition:* An outstanding team member can be given celebrity status within the company.

3. *Personal growth:* Although team members should have to prove themselves by working on a low-visibility project during their first six months, top achievers can then be rewarded with the opportunity to "do something new."

Technology Future

The technology future dimension measures the new venture's ability to sustain the competitive viability of its technology. This dimension includes such factors as an assessment of the firm's products and architectures relative to the state of the art, morale, process technologies under development, and ability to hire critical people. Like financeability, the technology future dimension represents an overall look at the company's ability to build competitive products in the future.

For example, assume Company N introduces a Motorola 68000 workstation based on a CISC (complex instruction set computer) microprocessor, perhaps with an attached signal processor, while all the other workstation firms are introducing products based on RISC (reduced instruction set computer) microprocessors (such as Sun Microsystem's SPARC, MIPS, Motorola 88000, or Intel 80860). Because the RISC microprocessors deliver higher performance, Company N's product specifications suffer by comparison, at least superficially. Company N's ability to respond to the ensuing performance race by increasing the workstation's functionality with voice and video, for example, and providing a wide range of applications software in the 1990s will determine its technology future.

Operational Management

Operational management is the engineering organization's ability to manage itself by meeting its product specification, budget, and schedule commitments. Management includes all the techniques of managing design reviews, management by objectives, staff meetings, team building, conflict resolution, etc. Andy Grove, CEO of Intel, has produced some of the best handbooks on this subject (Grove, 1983, 1987).

As a product reaches the final stages of completion, it will become clear that the team must compromise among the following three indigenous variables:

- The schedule, or when the product will be ready
- The complete set of resources that is applied toward meeting the schedule, including computers, consultants, other software, etc.
- The characteristics of the product itself, including performance, product cost, features, etc.

The best approach is for the company to pick two out of three, manage those, and be happy with the outcome. For a start-up, the schedule and resources are really fixed because of the incredible cost of raising additional funds. Furthermore, it is generally inadvisable to attempt to add critical design resources to a project that is already running late, because the firm is then apt to become subject to Brook's law: "Adding resources to a late project makes it later."

Therefore, the function of the company's first product will inevitably be less than perfect. Faced with the need to cut function in order to meet schedule and resource constraints, it is best to sacrifice some the product's features rather than sacrifice performance. Performance equates to quality in many systems and should not be sacrificed. Likewise, reliability is not a "feature"; it is a quality constraint that must never be sacrificed.

Ardent. At Ardent, Tom Bentley, a former Hewlett-Packard engineer who headed mechanical design, said it was hard to find contractors who would meet the company's standards. "We expected a designer to meet both schedule and contract cost [goals], while also meeting the product cost, quality, and features constraints. Steve Jobs expects two [of these], and most companies in the valley are happy with just one."

Technology Balance Sheet for Ardent

While working at Ardent, I used a technology balance sheet to analyze the company's technology capabilities. Table 1 shows the dimensions (and subcategories thereof) that were analyzed.

Table 1. Technology Balance Sheet for Ardent

Technology Base

- Packaging, mechanical design (including thermal and acoustic analysis)
- Industrial design
- Digital systems design
  - Signal propagation, electromagnetic interference, and radio frequency interference
  - General logic design
  - Gate array design
  - Testing
- Architecture/implementation
  - Vector multiprocessing
  - Performance analysis and simulation
  - Graphics
  - Mass storage and input/output
  - Image processing
  - Software
  - Operating and file system
  - Language and compiler design
  - Graphical user interface
  - Database
  - Quality assurance
- Marketing, sales, and product support
  - Benchmarking
  - Mathematics and scientific progress
  - Signal and image processing
  - Visualization
  - Computational chemistry
  - Computational fluid dynamics
  - Mechanical CAD and finite element modeling
  - Seismic processing
- Technical publication

Engineering Specifications

- Reference manuals for all components
- Principles of operation for hardware
- Eight-corners test

Manufacturing Specifications

- Test vectors and specifications for all chips and boards
- Hardware and software release specifications

Chief Technical Officer, Team/Culture, Architect(s)

- Discussed in an earlier section of the chapter

Technical Resources

  - Computing environment
  - Multisegment Ethernet and Appletalk
  - Macintoshes for documentation
  - Sun Workstations (local and windows)
  - MIPS file and computation servers
- Valid logic for logic design
- Verilog for system description/ simulation

Technology Future

- Plan outline for next products

Standards

- See Chapter 8, Figure 8-5 for product standards

Process Definitions

- New products introduction

Plan

- Embodied in master schedule
- Yearly budget with all resources

Operational Management and Control

- Schedule fantasy factor = 1.2 after a major organizational change
- Weekly schedule review at each level
- Staff meetings at each level of management, with minutes and action items
- Management by objectives
- Products committee to track/coordinate all products and future product plans

Technology and Engineering Flaws

Some of the technology and engineering flaws presented in this section are similar to various people and business plan flaws that were discussed in earlier chapters. The flaws range from lack of technology, either because extensive research is needed or because the technology is ubiquitous and trivial, to simply having a poor team. As with other types of flaws, predicating a high-tech venture on technology that is flawed in one or more respects could prove to be fatal.

Tackling a Product That Requires Significant Research to Make It Feasible

A wonderful product that is clearly needed is just waiting to be developed. Designing the product, however, will require an unknown amount of basic and applied research. As of 1990, the estimate of when such a product can be produced ranges from now to 18 months from now to never (although *never* is a word that cannot really be used when it comes to technology). The following example illustrates the slow evolution of a product whose development has required (and will continue to require) a considerable amount of research.

The speech typewriter (speechwriter). Kurzweil AI was formed in 1982 to build a speech typewriter. Its founder, Ray Kurzweil, has produced an impressive array of inventions, including the first machine to read to the blind (1972), which does optical character recognition of variable fonts and is connected to a speech synthesizer. The company developing the reader was sold to Xerox. A second firm, which was formed in the late 1970s to build keyboard-controlled music synthesizers for the professional and home market, is now for sale in 1990.

The aim of speech research, which has been under way since the 1930s, is to understand speech well enough to permit it to be recognized by a machine. In 1980, at least one market research firm published a report estimating the market for voice-activated typewriters at $3.5 billion in 1990. Kurzweil believed that enough was known about speech understanding to finally build a comparatively elementary but nonetheless useful product that would function within certain limited contexts, such as having the machine run by a single, trained operator who would use a large, but limited vocabulary and speak separated words.

Kurzweil's first task was to advance the art on which to base a product. In order to bring himself up to the state of the art in speech recognition, Kurzweil put together a team from the research community at MIT and Harvard to develop technology for speech understanding. In 1985, the firm introduced its first product and tried to sell a recognizer to a number of software companies (whose products included spreadsheets, word processing, databases, CAD, etc.) as a control mechanism, but the product's capability and accuracy were limited and it worked poorly. Furthermore, users had to "train" the recognizer. The Kurzweil AI product predated a product by Articulate Systems (using Dragon's recognizer) to control the Macintosh.

By 1989, the Kurzweil product had evolved into a unique voice editor that runs on a PC and is capable of recognizing keywords and expanding them using a word processor database and report generator. The voice editor is tailored to a particular application by its vocabulary and phrases and is then further timed by the user. In 1990, the product is being successfully sold for writing reports in internal medicine, pathology, radiology, and emergency medicine, since these fields all require reports based on distinct, limited vocabularies.

In contrast to speech-research laboratories such as Bell Labs, IBM, and university laboratories, Kurzweil has advanced toward the goal of a typewriter by building and marketing a product. Other companies have also built and marketed speech recogrrzers for limited use. Unlike other laboratories, NEC has been marketing limited vocabulary recognizers for almost a decade in order to really understand their problems and use. Thus, for a researcher, a start-up is an interesting alternative to the large company or government-funded laboratory, assuming the firm can find investors wuilng to wait for their investments to mature. Dragon Systems, Inc. provides an alternative role model for how a venture requiring a slow-to-emerge technology may be formed.

No doubt the hottest product—the one that absolutely everyone will have, need, and use after 2001—will be the universal speech typewriter! And the next advance will probably be a speech typewriter that does on-the-fly language translation.

Requiring a Trilogy of Breakthroughs

It has been observed that a successful start-up cannot be based on more than two breakthroughs in the state of the art. And for each of the areas requiring a breakthrough, an alternative technology should be available as a backup. Clearly, a risk exists when three or more technologies have to be understood (i.e., researched to the point of being usable) and developed. It is almost assuredly fatal for a start-up to engage in research whose result cannot be known or scheduled, because the company's other functions must all be supported in the meantime, and the funding requirements are uncertain and often open-ended. The schedule for such a project contains loops, parallel and redundant exploratory paths, and conditional branches.

The following example discusses Trilogy, Inc., which attempted to develop a product requiring multiple technologicalbreakthroughs. The "trilogy of breakthroughs" flaw is in fact named after Trilogy, since this flaw contributed greatly to the difficulties the firm encountered.

Trilogy, Inc. Trilogy was started to develop an IBM-compatible line of computers with major subsystems packaged on a single semiconductor wafer. Unisys and Digital invested in the technology as codevelopers. [I made this recommendation. After Trilogy failed, Digital bought rights to all its technology. The power supply, heat sink, wafer-packaging scheme, and facilities were used as the basis for the VAX 9000.] The risks included the following:

1. Interconnecting high-density, high-speed semiconductor circuitry on a single wafer.

2. Devising a scheme to ensure defect-free parts using redundant parts of a wafer.

3. Packaging an entire wafer such that power is input, heat is dissipated, and the wafer is rewired to circumvent inherent wafer defects.

4. Developing a CAD system to manage the redundancy-based logic design and interconnect scheme.

5. Developing a computer design more complex than previous designs.

Some observers felt that Trilogy's pleasant facilities and large staff were fatal flaws. The real culprit, however, was that the requisite technology could not be developed in time to implement a product. The five risks listed above had the following outcomes:

1.The circuits were slower than specified, increasing the design's complexity while decreasing its competitiveness.

2., 3. Not enough redundancy was available to cover wafer faults.

4. The CAD system was quite slow and decreased productivity.

5. The design was so complex as to increase the design time and adversely affect product competitiveness.

One of the most dangerous flaws is arrogance which compels a company to reengineer a hardware or software system because it believes that it can do a better job than its potential suppliers.

Although the preceding problems occurred during the product development stage, the issues were known at the concept or seed stage. In hindsight, an analysis of the situation should have produced an emphatic "no go" until the required breakthroughs were reduced to a manageable number.

When it became clear that Trilogy's technology was inadequate to build the product, the company acquired Elexsi Computer with its remaining capital and attempted to make it succeed. Unfortunately, minisupers from Alliant and Convex were also being brought to market at that time.

Having Little or No Sustaining Technology

Offering just another commodity product of a particular type (i.e., "brand X") in a crowded field is usually a fatal flaw. Starting a company with commodity technology, such as a new chip, is the opposite of the trilogy of breakthroughs flaw. It comes from the belief that the firm has just a slightly better idea about the product or how to sell it. The minicomputer, PC, and workstation industries all began as technology companies to a greater or lesser degree, and the introduction of various components (SSI/MSI, 16-bit microprocessors, and 32-bit microprocessors, respectively) allowed dozens of no-tech companies to enter the market. In early 1990, the smallest PC electronics assembly costs $200, and within five years, just one or two very-high-tech chips (available from Intel and a memory supplier) will form the entire, minimal PC with 2 megabytes to 8 megabytes of memory. Dell Computer is an excellent example of how a company was able to get started and grow with PCs despite the low-tech odds, because Dell considered the whole environment of product, sales, service, and support.

The Not-Invented-Here (NIH) Syndrome

One of the most dangerous flaws is a form of technical arrogance in which a company feels compelled to reengineer every part of a hardware or software system because it believes that it can do a better job than any of its potential suppliers. For a new venture, inventing every possible component in order to make an ultimate product (instead of buying everything possible in order to get to market rapidly with a good product at the lowest development cost) is often fatal.

The other effect of the NIH syndrome is the incompatible-product flaw. A company designs a new interface, such as a programming language or a feature for an existing language, when an old one would have been just fine. In this case, NIH hurts the buyer, who has to change and adapt to something different. Needless innovations and changes that have the effect of rendering hardware, programs, and data incompatible are extremely costly for the whole computing enterprise.

The NIH syndrome is endemic among most engineers, especially in the United States, the United Kingdom, and France. NIH does not necessarily have anything to do with a team's competence, only its lack of business savvy, although the brightest teams are often the most unhappy about using less-than-perfect components. The NIH syndrome's effects on productivity and on profit and loss are devastating, and this syndrome may account for why Japanese engineers are at least twice as productive as American engineers in a field such as automotive engineering. NIH often triggers the formation of multiple companies in one.

Even well-established and well-respected firms have exhibited this flaw. In the early 1960s, IBM found that every computer products group was building a computer based on each group's own logic circuits, requiring redundancy in design, manufacturing, and field spares. Gene Amdahl proposed that any group using components from another group be rewarded and given special recognition. One of his coworkers squelched the idea, claiming that "it's un-American."

The Missing Component

Every day that an organization depends on a risky part or a marginal vendor, it risks its life, because if a critical component (or process) fails to materialize as scheduled, the company may run out of time and, hence, out of money. Selecting poor vendors is a common and hard-to-avoid error. Only through experience will a start-up learn which firms can be trusted to meet their commitments.

Henry Burkhardt, CEO of Kendall Square Research, described the problems of selecting the right vendor by offering what might be called a "tale of three cities." In it, he compared the experience of dealing with vendors in a Texas city, a California city, and a Japanese city:

- Texas: *We have the fastest, biggest, and cheapest parts. If you don't believe it, write me because I'm the president of this new division.* (They don't really have a competitive product. On calling them, the secretary to the president states that you have to write because the letter goes directly to the marketing VP. I wrote to the president and informed them they lie about their parts and even lie about their willingness to listen. The letter does go through the company like wildfire, but the division president is still there, selling the same parts in the same old way.)
- California: *Everyone knows our parts are the fastest and the biggest. We started the industry.* (We ask for a delivery commitment. It reads "We'll make our best effort to deliver." On inspection, the parts fail after a year without special treatment that's not part of the specification. The customers all complain about missed delivery schedules, and manufacturing people scream when they hear the name of the company. Every transaction with the company requires negotiation.)
- Japan: *We have fast, large parts as stated in our specs, and zue are committed to high quality.* (Existing customers agree, and no one can identity a part ever failing. We selected them because the contract simply states that they will meet their specs and deliveries. All specs and delivery dates were met.)

The following story iliustrates the type of havoc that can ensue when a company deals with a poor vendor.

Wavetracer. In building a signal-processing computer, Wavetracer used an unreliable printed circuit board vendor to make its prototype boards. The boards had numerous errors, costing the firm several months over its plan at a critical time when it needed a product and credibility with its first customer. Because of this schedule slip, Wavetracer was forced to seek additional financing earlier and in a greater amount than would otherwise have been necessary. The valuation was decreased and the external ownership increased.

> It is foolhardy to preannounce a product before it has passed its acceptance tests. At the very least, preannouncement is likely to be an embarrassment; at worst, there might be legal repercussions.

New microprocessors have historically had bugs. New complex microprocessors from semiconductor companies—including Intel, Motorola, and National—have all had bugs. The more complex the part, the more error-prone it is; hence, another reason for RISC. The first users are able to help find new flaws and often rediscover flaws that manufacturers forget to address. Apollo, Sequent, and several other companies have war stories to tell in this regard.

Inability to Hire the Engineers

Hiring is absolutely critical, yet every high-tech venture I know of has had more trouble hiring than it ever planned or imagined. This leads to an additional flaw—lowering the standards. By reducing its standards, the firm risks producing both a downward spiral in quality and a bloated staff that generates no meaningful output. A pygmy heading engineering will proceed to hire even smaller pygmies.

Failing to Get Rid of Poor Hires as Soon as Possible

If a person is found to be a poor hire, he or she must be dismissed at the earliest feasible moment. Negative producers should be terminated immediately, placeholders very rapidly, and marginal producers as soon as possible.

[Negative productivity is a principle that I claim is worthy of a Nobel Prize. Normal principles of productivity assume that workers create positive output. Brooks refined the concept of software productivity to express it in terms of the "mythical man-month," and in software engineering, it is understood that different programmers vary in their productivity by several orders of magnitude. According to the principle of negative productivity, it is possible for an individual to produce bad results that others must then redo; hence, someone who is very negatively productive can keep a whole team busy with damage control, preventing the team from producing any output whatsoever.]

Company X. I know of a firm (let's call it "Company X") that was having trouble staffing a new project with good people and made a borderline hire without proper reference checking. When the team discovered that the borderline individual was in fact a poor hire, they felt they could manage him by close supervision and checking. However, he refused to ask for help, chafed at having his work reviewed, and was late—all sure signs of a bad design(er). Simulation revealed continued bugs with no evidence of progress toward a correct design. In essence, bugs were just being moved around. When Company X finally conducted a design walk-through, the engineer quit and went to a competitor, where he may or may not have greater success. Although Company X did nothing to influence its former engineer's selection of a new employer, outplacing negatively productive people with a potential competitor can do wonders for a firm's competitive lead.

Leaking Technology and Product Ideas

If a new venture permits its technology and product ideas to leak, it risks giving both established competitors and other start-ups an opportunity to respond. It is therefore important that the staff say no more than is absolutely necessary in order to sell new recruits. They should try to get recruits to tell more about themselves than the company tells about itself and avoid any mention of costs and schedules.

Preannouncing the Product

It is absolutely foolhardy to preannounce a product before it has been tested internally and passed its acceptance tests. At the very least, preannouncement is likely to be an embarrassment; at worst, there might be legal repercussions.

In *no* case should a product be officially announced before it is operating well enough to pass formal tests that are comparable to actual customer use. Ideally, the product announcement is made at the end of beta testing at customer sites. Anything less conservative is a flaw.

This is one flaw that is even more painful in large companies than in start-ups. In 1966, IBM preannounced a large computer that would compete with Control Data Corporation's 6600 in an attempt to get customers to wait for the IBM product, which, in this particular case, never came. CDC sued IBM and was awarded $600 million in a consent decree that forbade preannouncement.

Technology Balance Sheet Rules

The following is the fundamental rule for evaluating a new venture's technology:

Has the company generated and maintained a complete "technology balance sheet" that is adequate to develop the product and specifies the information listed below?

- "Buy-out" technology (software and hardware), including semis, etc.
- Patentable or unique components that are the basis for the firm's future
- Industry and de facto standards that the start-up must "track" or advance
- The company's own standards or ways of doing things
- Patentable or unique processes, including design
- Plan, with schedule and resources
- Engineering and manufacturing specifications
- Chief technology officer (i.e., the vice president of engineering)
- Team
- Product architects and architectural processes
- People (including consultants) who embody the technology
- Computer-aided design (CAD) and computer-aided software engineering (CASE) tools, computer resources, and network environment
- Ability to acquire future technology

Operational Management Control

The following are some specific rules applicable to the technology balance sheet.

Can the team, at the concept stage, show how all the technology will come together to form a product that will be not only unique but also self-sustaining (i.e., capable of evolving into future generations)?

The technology balance sheet should be used to account for *both* uniqueness and mastery of the technology. Mastering the technology means being able to assemble the "to be acquired" engineering team, consultants, patents, standards, components, design process, CAD tools, etc. This rule tests whether the organization has a way to evolve its product and extend it into future generations or whether it is merely starting on a one-shot basis.

The same rule should be applied again at the seed stage, continually challenging the founders about the uniqueness of their technology. It examines whether the technology remains sufficiently unique, yet implementable, to support a self-sustaining company. The rules in Chapter 8, "The Product," also examine uniqueness.

Can the team, at the concept stage, show how the technology can be developed while requiring fewer than three breakthroughs or significant advancements in the state of the art?

This rule tests whether the technology is too high (sometimes reaching infinity), such that the new venture is engaging in research instead of product development. Applied research or advanced development is being done if a project schedule contains major loops with conditional branches or multiple exploratory paths in its PERT chart. Such a company is likely to be fatally flawed if it has been funded with the goal of developing a product, as opposed to being funded as a research and development partnership. In the latter case, investors are cognizant of the risk, and the goal is to first master the technology before building a product.

Does a simple product development plan, specifying resources and schedule, exist at the concept stage?

This rule tests whether the start-up has a plan outlining the steps and resources that will be required to develop the product.

Does the company have a working product or product prototype and people who understand it?

Ideally, a high-tech venture is based on a working product or product prototype that has been funded by a public institution together with the people who understand and embody the technology, even though such products and people may fail the experience tests required by many financiers.

The next best thing is to base the company on key people who have pioneered in developing technological components. They must have a thorough understanding of the product development process gained through building products for use by others and must be committed to engineering design rather than research.

Probably the worst alternative is to base the firm on the results of military research and development, because it is likely to be fatally flawed, as described in "Augustine's Laws" (Augustine, 1987). Military products are cost- and reliability-insensitive. They don't have to work or are rarely tested to ensure that they work. The development budgets, lead time (measured in decades), and quality of military products are outside commercial bounds.

Has the company's proprietary technology been demonstrated during the seed stage via physical or computer model, breadboard, or some other form of demonstration that would prove its viability, such that the development breakthroughs have been reduced to a level of risk that is acceptable for the product development stage?

This rule verifies that the start-up is in control of its technological destiny by checking whether the seed stage requirements of reducing risk have been satisfied by constructing breadboards, models, or demonstrations of critical technology. Ideally, at this point, the firm has ideas that may result in copyrights and patents in order to protect and enhance its technology.

If the company is depending on a concurrent breakthrough or leading-edge product from another supplier (e.g., a component or system vendor), have the risks been clearly identified and factored into the plan?

This rule determines whether the start-up's risks have been transferred to an outside vendor and then assesses the overall risk in using such a vendor. Information about the vendor's past performance is required, especially evidence of its reliability in meeting delivery schedules. Founding a company predicated on the availability of a component that a manufacturer has never before built is always risky. The new venture gets no points for picking the best technology or engineering the lowest cost if it is then unable to obtain a key part or unable to obtain it on time or in manufacturing quantities.

The evaluation of vendors and components is an excellent position for a seasoned engineer, by the way. Such individuals know which components and suppliers are high-quality and reliable. New engineers, on the other hand, tend to believe specifications.

Does the chief technical officer have the capability and stature to hire, lead, and manage a superb engineering group?

The general qualifications of the CTO must parallel those of the CEO, because he or she is the "clock" and "standards setter" for engineering. The CTO should have a track record of both technical and managerial accomplishment. The CTO's technical background must be solid enough to gain the engineers' respect and confidence in his or her technical decisions. The CTO's managerial skills must be strong enough to deal with conflicting egos, limited resources, and all the other trials and tribulations that a manager faces. This individual should be especially talented at recognizing, selecting, and encouraging topnotch engineers.

Does the product have an architect with proven experience?

As stated previously, the product architect is likely to be the most critical person within the engineering function. His or her key job is to guide the product's introduction and evolution over the course of its lifetime, and a track record of success in past endeavors is the strongest possible recommendation. In some cases, several architects may be required as a product is broken into various parts, but the boundaries of each architect's responsibilities must be clear, and the architects must be capable of functioning as a cohesive team.

Are key technologists, or avenues for hiring them, available?

In one sense, this rule relates to the question of whether the company has the "right tech" (ie., an appropriate level of technology). If the technology upon which the venture is to be based is so "far-out" that only a handful of technologists skilled in that art are available, the firm is likely to have serious staffing problems. On the other hand, if the start-up is to be based on an ingenious use of a recently introduced or established technology, hiring prospects will be much better. Some innovative ways of finding appropriate personnel were discussed earlier in the chapter.

Does the company have hiring criteria, and is there a systematic recruiting process?

This rule checks whether the firm has established hiring criteria, covering both work habits, management ability, and technical skills. Having specifications for each person to be hired is helpful and perhaps essential. In addition, the company needs a first-rate process for initially identifying potential employees and then bringing them in for an interview, screening them, and finally selling them. A critical part of the process is thorough reference checking of all candidates!

Does the job candidates' prior experience show evidence of operational management ability as well as resources- and schedule-planning ability?

This question examines the planning and management history of the engineer/management team. History is likely to be the best predictor of a manager's ability to help people enjoy their work and be productive in it. And with regard to scheduling, if the candidates have historically been on time, then they will most likely continue to meet their commitments in the future.

Has engineering outlined a quality design and product release process together with engineering, manufacturing-engineering, and product-release standards, including, for example, coding practices, design rules, code walk-throughs, and design reviews?

This rule measures the existence and effectiveness of the company's engineering design process. For a software team, it would not be unreasonable to ask whether the process at least satisfies the Software Engineering Institute's process-capability requirements for level 1 and what plans exist to upgrade the process so it will satisfy the requirements of increasingly higher levels (Humphrey, 1989).

It is not uncommon for engineers to react negatively to the establishment of standards and processes. For example, engineers who have just left large firms frequently rebel at anything that might look like bureaucracy or restrictions on their freedom, and engineers coming from a research environment are unlikely to understand the need for any rigor in standards and processes. Object-oriented programming languages and methods promise to make the task of building software substantially easier because they enable modules to be built in a more isolated and independent manner and because more software is likely to be available from other sources and to be reusable.

Is a product development schedule in place, and does it specify gross milestones and resources?

This process question examines whether the start-up has a schedule for the project with enough intermediate milestones. Without such a schedule, it is impossible to make a meaningful business plan. People experienced in high-tech ventures know that it is essential for the company to be operating according to a detailed schedule, even though no schedule can be fully validated until the entire team responsible for the project has been hired and brought on board. An unwillingness to make a detailed schedule at this point is therefore a good early warning indicator that the project will probably be difficult and unpredictable. A start-up can certainly be financed on an open-ended schedule, but this approach can be expected to increase product development spending by at least a factor of 2.

Does the company have a plan for acquiring and operating CAD and CASE tools, computing resources, and its network?

Developing products based on up-to-date technology requires up-to-date engineering tools. Tools represent both a key part of engineering and a large fraction of product development cost. A CAD program for schematic capture or board layout can cost several hundred thousand dollars. A simulator to accelerate the testing of a complex chip may cost half a million dollars. Thus, it is critical for the start-up to prepare a detailed list of all the tools (both computers and the necessary networks) it will require for high-tech hardware and software development. In the early stages, developers often adnunister their own systems, which may include interfaces with various national and international wide area networks, but as a company grows, the expense of system administrators and network administrators must also be assumed.

Conclusion

Chapters 5 and 6 have presented a picture of high information technology and examined how a new venture uses technology to engineer products in a timely and predictable fashion. At each of the development phases described in Chapter 5, the company must have an adequate technology balance sheet covering the following 12 dimensions: its technology base; standards; design, quality, and other processes; plane with schedule and resources; engineering specifications; manufacturing specifications; chief technical officer; team and engineering culture; architecture; technical resources; technology future; and operational management.

Bibliography

Articles

Bell CG: The mini and micro industries. *Computer* 17(10):14-30, Oct 1984.

Carr R: How to build better programming teams. *Soft•letter* 6(4), May 1, 1989.

Gomory RE, Schmitt RW: Science products. *Science* 240:1131-1132, 1203-1204 May 27, 1988.

Grayson P: How to motivate programmers. *Soft•letter* 6(4), May 1, 1989.

Lampson B: Personal distributed computing: the Alto and Ethernet software." In A. Goldberg (ed), *A History of Personal Workstations*, Addison-Wesley, Reading, MA, pp 291-344.

Meindl JD: Chips for advanced computing. *Sci Amer* 255(10):78-88, Oct 1987.

Mendelson H: Economies of scale in computing: Grosch's law revisited. *Communications of the ACM* 30(12):1066-72, Dec 1987.

Rosenstein J, Bruno AV, Bygrave WD, Taylor NT: Do venture capitalists on boards of portfolio companies add value besides money? Working paper for a 1989 study.

Tarter J (ed): Why Goliath usually wins. *Soft•letter* 6(3) June 15, 1989.

References for the Entrepreneur's Bookshelf

One or more of the following symbols appear before a *number of the entries in this section of the bibliography to indicate my recommendations for how these works can most profitably be utilized:*

‡ A reference to own, understand, and use.

•A reference to own and read.

+ A reference to outline, whose expert advice constitutes sound rules by which to operate.

Augustine, Norman R.: *Augustine's Laws*, Penguin Books, New York, 1987. Fifty-two tongue-in-cheek "laws" governing the production of high-technology, expensive, and unreliable military products. For example, "By the year X, only one airplane can be built because it will absorb the entire GNP." Contains many unfortunate, but empirically derived, laws and conjectures explaining the military- industrial complex. Evidence is given to indict the military-industrial establishment for incompetence. An essential work for any company dealing with the military.

‡**Baty, Gordon B.:** *Entrepreneurship of the Nineties*, Prentice Hall, Englewood Cliffs, NJ, 1990. An excellent start-up handbook to supplement White (1977).

Bell, C. Gordon, J. Craig Mudge, John E. McNamara: *Computer Engineering*, Digital Press, Bedford, MA, 1978

‡**Brooks, Frederick P.:** *The Mythical Man-Month*, Addison-Wesley, Reading, MA, 1975. A classic, useful book on programming that's also enjoyable reading. Essential if the start-up's technology is embodied in programs.

Burgelman, Robert A., Modesto A. Maidique: *Strategic Management of Technology and Innovation*, Irwin, Homewood, IL, 1988. See also Roberts (1987).

•**Card, David N., Robert L. Glass:** *Measuring Software Design Quality*, Prentice Hall, Englewood Cliffs, NJ, 1990.

Cooper, Robert G.: *Winning at New Products*, Addison-Wesley, Reading, MA, 1986. Aimed at large companies. Contains some good advice and techniques for looking at products that a start-up can also use for product positioning.

+ **Davidow, William:** *High Technology Marketing: An Insider's View*, Free Press, New York, 1986. A fine book of stories. I recommend spending about two hours to outline the material and get the author's advice. The 16 rules (i.e., questions, just like the Bell-Mason Diagnostic) presented in Chapter 11 ("Do You Have Marketing?") are worth understanding and following.

•+ **Davis, Robert T., F. Gordon Smith:** *Marketing in Emerging Companies*, Addison-Wesley, Reading, MA, 1984. Contains good insights and much good advice about marketing and selling.

Deal, Terrence E., Allan A. Kennedy: *Corporate Cultures: The Rites and Rituals of Corporate Life*, Addison-Wesley, Reading, MA, 1982.

DeMarco, Tom: *Controlling Software Projects: Management, Measurement and Estimation*, Yourdon Press, a division of Prentice Hall, Englewood Cliffs, NJ, 1982.

DePree, Max: *Leadership Is an Art*, Doubleday, New York, 1989. An excellent book describing the culture of Herman Miller, Inc.

Drucker, Peter F.: *Innovation and Entrepreneurship*, Harper & Row, New York, 1985. A work that should be read rapidly and outlined if time permits.

•+ **Fairley, Richard:** *Software Engineering Concepts*, McGraw-Hill, New York, 1985. Presents concepts that should be understood if the start-up is engaged in software engineering.

Gershman, Michael: *Getting It Right the Second Time*, Addison-Wesley, Reading, MA, 1990. Covers marketing dos and don'ts.

Gilder, George: *Microcosm: The Quantum Revolution in Economics and Technology*, Simon & Schuster, New York, 1989.

‡**Gladstone, David** 1988: *Venture Capital Handbook*, Prentice-Hall, New York, 1988. An essential book on raising capital, with a good discussion of the business plan.

Goldberg, Adele, ed.: *A History of Personal Workstations*, Proceedings of the History of Personal Workstations Conference (Jan. 1986). Addison-Wesley, Reading, MA, 1988.

‡**Grove, Andrew S:** *High Output Management*, Random House, New York, 1983. An excellent book on how to manage and how to increase management productivity. It would certainly be great if everyone read and in some way practiced this kind of management.

+ **Grove, Andy:** *One-on-One with Andy Grove*, Putnam, New York, 1987. Presents questions and answers about management.

•+ **Humphrey, Watts S:** *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989. Essential for software-engineering management.

• **JIAN:** *BizPlanBuilder*, JIAN Co., Los Altos, CA, 1988. A template that can be used on a PC or Macintosh to write a business plan by revising and responding to material contained in the template, including spreadsheets.

• **Juliussen, Karen, Egil Juliussen:** *The Computer Industry Almanac, 1990*, Simon & Schuster, New York, 1990. Contains very useful information about companies, organizations, markets, people, and products.

•+ **Kawasaki, Guy:** *The Macintosh Way: The Art of Guerrilla Management*, Scott, Foresman, Glenview, IL, 1989. Presents many critical rules for marketing products.

Kotler, Philip: *Principles of Marketing*, 3d ed., Prentice Hall, Englewood Cliffs, NJ, 1986. A traditional marketing textbook. Shows why MBAs can be replaced by a series of computer programs.

Kurzweil, Ray: *The Age of Intelligent Machines*, The MIT Press, Cambridge, MA, 1990.

•+ **Levitt, Theodore:** *The Marketing Imagination*, Free Press, New York, 1986. *The* book on marketing. Reprints the classic "Marketing Myopia" from *Harvard Business Review* (Jul-Aug 1960).

‡+ **McKenna, Regis:** *The Regis Touch*, Addison-Wesley, Reading, MA, 1985. A great guide to all aspects of high-tech marketing.

+ **McKenna, Regis:** *Who's Afraid of Big Blue?* Addison-Wesley, Reading, MA, 1989. I recommend spending an hour to read and outline its two pages of advice.

• **Nesheim, John L.:** *Startup: Founding a High Tech Company and Securing Multi-Round Financing*, Electronic Trend Publications, Saratoga, CA, 1988. Contains many details about what to do, along with numerous, clearly marked rules. This book is expensive, however, and most start-ups are unlikely to spend the several hundred dollars it costs.

Osborne, Adam, John Dvorak: *Hypergrowth: The Rise and Fall of Osborne Computer Corporation*, Idthekkethan Publishing Co., Berkeley, CA, 1984. Recommended reading to see what can go wrong in stage IVb, as a product takes off. Clearly illustrates the flaw of introducing a new product that can't yet be shipped while the company is still selling a product whose revenue is vital.

Peters, Tom J., Robert H. Waterman: *In Search of Excellence*, Harper & Row, New York, 1982. Presents a good discussion of corporate cultures in large organizations based on a survey of successful companies. Some ideas may be useful to a start-up.

Rifkin, Glenn, George Harrar: *The Ultimate Entrepreneur*, Contemporary Books, Chicago, 1988. The story of Digital Equipment Corporation, a great role model for CEOs and for establishing corporate culture. Ken Olsen founded DEC in 1957 and led it to become the world's second-largest computer company, staying in charge longer than any other CEO.

Roberts, Edward B.: *Generating Technological Innovation*, Oxford University Press, Oxford, England, 1987. Contains many case studies. Any large-company bureaucrat managing research and development should understand its contents.

Rogers, Everett M., Judith K. Larsen: *Silicon Valley Fever: Growth of High Technology Culture*, Basic Books, New York, 1984. Helps in understanding the culture of employees, customers, and investors-if the start-up is doing business in Silicon Valley.

• **Schlit, W. Keith:** *The Entrepreneur's Guide to Preparing a Winning Business Plan and Raising Venture Capital*, Prentice-Hall, Englewood Cliffs, NJ, 1990. Worth owning. Contains lots of useful plan formats, definitions, and sources of capital.

Shim, Jal K., Joel G. Siegel, Abraham I. Simon: *The Vest-Pocket MBA*, Prentice Hall, Englewood Cliffs, NJ, 1986. Presents useful guidelines for understanding the subtleties of financial statements and financial decision making. Assumes that the reader is familiar with basic accounting principles.

‡**Silver, A. David:** *Venture Capital: The Complete Guide for Investors*, Ronald Press, John Wiley & Sons, New York, 1985. Explains how customers—i.e., investors—think when doing financing. *The* book about the venture capital community. Describes how the financing of funds and of companies works.

Smith, Douglas K., Robert C. Alexander: *Fumbling the Future*, Morrow, New York, 1988. Story of Xerox's inventions in distributed computing and its attempts to enter the information-processing business. Useful for understanding the management of research and the technology-transfer process.

• **Walker, John:** *The Autodesk File*, New Riders Publishing, Thousand Oaks, CA, 1987. A great book on starting a software company. I recommend reading it and using its memos directly in managing a company.

‡**White, Richard M.:** *The Entrepreneur's Manual*, Chilton Book Co., Radnor, PA, 1977. Presents a wonderful set of rules for understanding and managing all aspects of a business (e.g., salespeople and how to close sales). Chronicles in an almost encyclopedic fashion many aspects of a start-up. Also contains a good discussion of building a plan.