Technology Strategy in a Software Products Company

Marc H. Meyer*
Luis Lopez

April 1995 WP # 123-95

Sloan WP # 3816

*Northeastern University

Accepted for publication in the Journal of Product Innovation Management

© 1995 Massachusetts Institute of Technology

Sloan School of Management
Massachusetts Institute of Technology
38 Memorial Drive, E56-390
Cambridge, MA 02139
ABSTRACT

The primary contribution of the article is to provide a method to help understand the relationship between nurturing core technologies and the evolution of a product family. This method embraces product family mapping and allocation of R&D investments to elements within the evolving product family. The product platform concept, common for physical assembled products, is found to be meaningful for software products and a important aspect of managing development.

The methods are applied to a software company. We show the consequences of different types of investment in core technologies on renewal of its software product platforms, and ultimately, on its commercial performance. The firm's choice of technology strategy was found to be associated with its commercial success. When the company focused on one core technology at the expense of another, its products faired poorly. When it pursued a more complete and integrative technology strategy, the opposite effect was observed. The study also shows the need for continuous renewal of underlying product platforms to realize sustained success.
Marc H. Meyer is Associate Professor of Management at Northeastern University's College of Business Administration, the Walsh Research Professor (1993-1995), and Director of the Center for Technology Management (at Northeastern University). His primary research interest is managing new product development. His work has been published in journals that include *Management Science, Research Policy, Sloan Management Review,* and *MIS Quarterly.* He holds degrees Harvard (AB) and MIT (MSc, and PhD). Dr. Meyer works with industry to study the planning and management of product development for assembled products. His current research focuses on extending product family and platform concepts to nonphysical products such as software and information products.

Luis E. Lopez is on the faculty of the Central American Institute of Business Administration (INCAE) in Costa Rica and is pursuing doctoral studies at MIT's Sloan School of Management. His primary research interests are in the management of new product development in service businesses, and in the impact of technology on services.

Please direct comments to:

Marc H. Meyer
Walsh Research Professor
Northeastern University
360 Huntington Ave
Boston, MA 02115
617 373 5948

Luis Lopez
Sloan School of Management
Massachusetts Institute of Technology
50 Memorial Drive
Cambridge, MA 92129
617 253 3721
Introduction

This article explores the relationship between investment in technological capabilities of the software product company and the evolution of the firm's product over time. Ketteringham and White [6] aggregated individual technological capabilities into "core technologies" with the intent better understanding a firm's strategy with respect to new products. Meyer and Roberts [10] used the core technology construct as a unit of analysis for tracking and assessing focus in the evolution of technologies embodied in emerging product families for technology-based startups. Core competencies [12] and capabilities [9] are now common terms in the management vocabulary. Teece et. al. described such capabilities as "a set of differentiated skills, complementary assets, and routines that provide the basis for a firm's competitive capacities and sustainable advantage in a particular business [17, p. 28]" Whichever term one wishes to employ -- core technologies, competencies, or capabilities -- a firm's ability to recognize and nurture the technologies required in the evolving designs of its product families is an important aspect of managing technological innovation within firms.

The "product family" is the second grounding concept for this article. For many if not most product making firms, individual products may be closely related by virtue of underlying architectures and market applications. This was clearly shown by Meyer and Roberts [10] and Roberts [14] for small, growing firms, and by Sanderson and Uzumeri [15], Wheelwright and Clark [19], and Meyer and Utterback [9] for larger firms. The technological basis of a product family is the product "platform" from which derivative or follow-on products may be efficiently created. Product platforms are renewed to create successive versions or generations of the product family by integrating the results of internal R&D or external technologies into improved designs. Companies such as Intel and Hewlett Packard are strong believers in continuous platform renewal, even while follow-on products are being generated from existing product platforms.

In this article we will focus on the relationship between investments in a firm's core capabilities and the evolution of a product family that utilizes these capabilities. If a firm requires one or several core technologies/capabilities in its product families, investment in these underlying technologies must be continuous to insure technically successful platform renewal and commercially success follow-on products. Discontinuous investment, ie. shifts in focus among embodied core technologies, may lead to undesirable outcomes. To study this issue, we will:
a) present a method to map the evolution of a software firm's product family,

b) identify the embodied technological core capabilities in the product family, and track investments made in these areas over time,

c) understand the relationship between (a) and (b), showing management's decisions with respect to core technology and observing the consequences of those decisions.

The Company

We gathered data within a firm that provides Unix (an operating system) for Intel-based desktop computers. The company has developed an efficient version of Unix for smaller, PC environments. Second, the firm develops and integrates a real-time scheduling module into the standard Unix kernel. This allows customers to build PC-based systems for data acquisition and analysis in scientific, manufacturing, telecommunications, and medical measurement applications. Furthermore, many of these systems are special purpose devices containing "embedded" Unix operating systems and applications software that drive process control applications.

Research into real-time Unix was initiated by one of the company founders under contract to a major university. Upon startup, the company licensed the original source code for Unix from AT&T (and later, from Interactive Systems Corporation and Novell), and made its enhancements to the operating system kernel. It then began selling a general purpose software development environment for minicomputers, and then, PC's. Later, its product set differentiated into two basic elements: a relatively expensive, fully featured software development environment for process control applications, and an inexpensive run-time enhanced Unix kernel for embedded systems. The company is small. Its sales will remain under $10 million in 1994.

We selected this company in part to broaden our understanding of what a "platform" means, and the type of leverage that may be achieved from it, for different classes of products. Over the past several years, we had studied assembled products and nonassembled products. This firm provided the opportunity to study platform development for pure software.
Defining and Charting the Firm's Product Family

The first step of the research was to map the company's Unix product family. We stated that a product family is defined as a set of products that share common core technology and address a related set of market applications. This can be validated for a group of products by applying the technology and market newness criteria proposed by Meyer and Roberts [10]. The commonality of technologies and markets leads to efficiency and effectiveness in manufacturing, distribution, and service, where the firm tailors each general resource or capability to the needs of specific products and niches. The technological foundation of the product family is the product platform. A platform is the physical implementation of a technical design that serves as the base architecture for a series of derivative products.

Working with three company founders, we proceeded to map the product family of the company under study, identifying its platform versions and follow-on products with an approach adapted from work on assembled products. A number of authors have stressed the importance of managing the evolution and renewal of product architecture for sustained competitive success [5, 18]. The architecture of a product refers to its overall design concept. Different architectures result in variation with respect to product functionality, cost, quality and performance. Architectures are both a basis for product innovation and a constraint on the variety of product versions that can be offered. For example, Intel and Motorola have competed for years in supplying computer chips for personal computers. IBM aligned with the Intel architecture (chips designated 8086, 80286, 80386, 80486, etc.), while Apple aligned with the Motorola architecture (chips designated 65000, 65010, 65020, 65030, 65040, etc.). Both chips relegated the earlier Zilog architecture (the Z80 chip used with early Tandy /Radio Shack PCs) to an insignificant presence in today's chip market.

The design concepts comprising a product's architecture are physically implemented as product and process "platforms". By platforms, we mean the basic product or process design from which different product versions can be derived. For example, platforms have formed the basis for creating families of automobile products. Many readers will be familiar with Chrysler's K-car platform from which the company derived many successful versions of their basic automobile offering. A well-designed platform enables a company to rapidly and efficiently create new versions of its basic product in response to or anticipation of changing market needs.
A platform design is comprised of subsystems or modules and the interfaces between these modules. For an effective product design, the interfaces between the parts of the product must be "seamless" and standardized [16]. In a camera, for example, one finds the shutter, the lens, the various operator controls and focus mechanisms, the flash, the battery power source, and the housing as major systems. The fittings and electronics by which these subsystems are integrated are the interfaces within the camera design. The roll of film used in cameras also has subsystems: the spool around which the film is wrapped, the cover over the film, and the end caps. The width of the spool, the length of the film leader, and the spacing of the edge perforations on the film are all standard interfaces designed to allow the roll to be used in a particular class of cameras. The film itself, a nonassembled product, has numerous laminations, such as a polyester or mylar film base, and various chemistries to provide black & white or color, and various speeds and image quality. Experience suggests that the critical importance of specifying clear internal and external interfaces in new product design, standardized to the point where the interfaces may have "part numbers" just as the components have part numbers.

In a software product, subsystems and interfaces between subsystems can also be identified. Subsystems in software products tend to be called software modules. In the company whose data are presented here, the major subsystems within its products can be grouped as: a) the kernel modules (including subsystems for scheduling, I/O, file management, task management, user management, and network management, b) systems administration utilities, and c) user utilities (the various "shells" and shell utilities and editors that are provided with standard Unix).

Software products also show the difference between "internal" interfaces and "external" interfaces within platform designs. In the case of the Unix software product, the internal interfaces are those protocols, rules, and mechanisms by which data are exchanged between the Unix kernel, systems administration, and utilities modules. The Unix shell command interpreter and the graphical user interfaces provided with the operating system are examples of external interfaces. Going back to our camera example, the instrumentation and manual or automatic controls for setting the shutter speed, the aperture, and the flash are external interfaces.

The notion of subsystems and interfaces leads to a typology of changing in the evolution of a product family (see Figure 1):

INSERT FIGURE 1 ABOUT HERE
The initial platform of a product family consists of the subsystems and subsystem interfaces of the basic product design. These subsystems are incrementally reengineered or refined to generate specific product offerings.

A platform extension occurs when: (a) particular subsystems within the existing platform design are substantially changed and enhanced, and/or (b) new subsystems are added to the design without disturbing the primary subsystems and interfaces in the existing design. For example, a computer manufacturer may replace a particular processor subsystem with a faster processor offered by a chip supplier.

A platform renewal occurs when the product design is redesigned to incorporate major new subsystems and new subsystem interfaces, i.e. to create a new platform.

Figure 2 shows the application of this typology to the PC Unix product family. The product family has had three product platforms (labeled Platform 1, Platform 2, and Platform 3) and fourteen derivative products. These three platforms had clearly different architectures. At the same time, all fourteen products were enveloped within a common technological and market space. Figure 3 contains the newness mapping of the firm's successive products. None of the successive products were deemed to contain new, unrelated core technology nor unrelated market applications.

INSERT FIGURE 2 ABOUT HERE

INSERT FIGURE 3 ABOUT HERE

The company's first product platform, Platform 1, was based on AT&T Unix Version 6, and targeted the large computer manufacturer's minicomputer hardware environment. The design of the platform (shown in Figure 4) was comprised of the following basic subsystems:

- the base Unix kernel provided by AT&T (which included subsystems for multi-tasking scheduling, I/O management, file management, user management, and a limited networking subsystem).
o the company's own real-time scheduling module which was added to the kernel (to allow the user application to preempt normal multi-tasking for a particular purpose such as data acquisition),

o the company's own "device drivers" programmed specifically to accommodate PDP input and output peripheral devices,

o the user programming and systems administration "utilities" provided by AT&T with Unix.

**INSERT FIGURE 4 ABOUT HERE**

The second product platform, Platform 2, was based on Unix Version 7 and System V. (Note that AT&T changed its product conventions at this point in time from just "versions" to "systems" with versions.) The work occurred in the early 1980's. The company worked on the new platform concurrently while derivative products were being developed on Platform 1. The thrust of the initial platform effort of Platform 2 was to "port" Unix, i.e. to make it work, on Intel-based PC's. Compared to Platform 1, Platform 2 had new subsystems and new interfaces between subsystems. First, AT&T added features or modules to the basic kernel to permit more efficient multi-tasking. Second, the company had to write new device drivers suitable for PCs. Third, AT&T expanded the number and functionality of the user utilities provided with Unix. It is important to note, however, that the company carried forward with little change its real-time scheduling module and integrated it into the new Unix kernel. The company achieved its port on the IBM PC several years ahead of competitors.

In 1987, the company adjusted its strategy away from general purpose computing towards real-time data acquisition applications using PCs. While all the subsystems and interfaces remained the same in number and type (the company simply swapped in new versions of externally supplied Unix kernels and user utilities, and did not add new device drivers), it reengineered and substantially improved its real-time scheduling subsystem. Response time dropped by an order of magnitude, from 10 milliseconds in Platform 2.1 to less than 1 millisecond by the end of Platform 2.2. Because a major subsystem in the architecture was completely reengineered, the effort produced a platform extension within our typology.
In 1990, the company shifted further into the real-time market by developing a new product architecture designed specifically to enable embedded Unix systems. Embedded applications are specialized systems in which the real-time Unix kernel and the user's applications programs are "burned into" non erasable memory chips. In addition to what the company called ROM-able, real-time Unix, a more comprehensive development environment was created. The company integrated a specialized process control user interface development tool (licensed from another computer products company) and implemented new windowing interfaces. This development allowed the customer to develop and fully test a real-time control application (typically on a 486 or Pentium computer), and then seamlessly download the application onto a run-time environment. Using specialized hardware and software tools, the code development on general computer purpose computers would be etched into the memory chips selected by the customer. These development tools and embedded run-time kernels entailed the development of several new proprietary subsystems and new interfaces between them and existing device drivers and user utilities.

Platform 3 had two platform extensions. The first was the incorporation of a new and substantially improved versions of Unix licensed from Novell (particularly in the areas of networking and graphical user interfaces), leading to platform version 3.2. Second, the firm continued to improve its own real-time capabilities, leading to platform version 3.3 in which a response time of .2ms was achieved.

We found that discussing changes at the level of major subsystems and interfaces in the underlying product platform proved a very useful technique for understanding technological evolution of the product family. We were then able to add to the product family map key market events and strategy changes as shown in Figure 2.

Technology Strategies Pursued over Time

We then identified the core technologies within this product family. The founders agreed that the firm had two significant technical strengths: the ability to make Unix work on PC environments (which they referred to as "porting Unix"), and adding real-time capability to Unix.
The company provided man-years spent on the development of its products. While the firm had no formal project accounting system for R&D, management was able to allocate person year with confidence to specific product developments. Being a small firm, the person-years in R&D represented a limited number of individuals who were dedicated to specific R&D projects from beginning to end. In recent years, these amounts had been tracked for tax accounting purposes (to depreciate R&D investments). The firm's R&D projects tended to be either "ports" of Unix to new machines (or of new versions of Unix to existing hardware), or real-time enhancements, and each new product was comprised a limited number of specific R&D projects. This allowed us to apportion emphases for each new product and allocate product person-years to each of the two areas of core capability. Allocation of resources to areas of core capability is shown in Table 1.

It should be noted that a significant amount of time was also spent by engineers supporting and providing technical service to existing customers. These data were purposefully not included in the figures above.

The allocation of engineering resource by area of core capability is plotted on a percentage basis by year in Figure 5. Also shown in that Figure are sales derived from the Unix product family on an annual basis. These data are presented on a scale for each respective dimension so as to disguise the actual dollar amounts involved.

INSERT FIGURE 5 ABOUT HERE

The firm has pursued two technology strategies over time. The first is a bimodal approach characterized by dramatic shifts between one area of capability and other(s). In this case, before company startup, the lead technologist and founder of the company did pioneering R&D in the area of real-time technology under a research contract with a major university. Once the company was launched, the firm's engineering focus shifted decidedly over to porting Unix onto the successive computer hardware environments.

Starting with Platform Version 2.2 in 1987, management realized that a more balanced technology strategy was required by its customers. Not only did its products have to run on new hardware environments and incorporate new versions of the Unix operating system, but the real-time capabilities of the software had to be brought to a higher performance level to be useful for the broader market of real-time applications. While the emphasis between porting and real-time
enhancements continued to shift after 1987, the degree of that shift was far less and neither area of technology was ever left to lapse.

The lagged-relationship in Figure 5 between the bimodal and balanced technology approaches and company sales is clear. The initial focus on the real-time area produced a level of sales growth of two to three years after the initiative started. The subsequent focus on porting the software to different computer architectures produced an equivalent level of sales two to three years after the initiative started. The initial six year period of weak performance by the firm was sufficient, in combination with a continuous market growth rate during that time, to convince us that the subsequent improvement in performance was indeed attributable in no small part to the new technology strategy pursued by management. A combined focus provided the real-time performance and timely hardware porting as demanded by the market. Sales began to accelerate two to three years after the strategy change. Note further from Table 1 that total amount of R&D spending over the years in the firm, while lower in the initial five year startup period, has had great variation over the years. The experiences of this firm confirm our hypothesis that successive product platforms in a product family must present both balance and strength in embodied core technology to achieve desired amounts of technical and commercial leverage.

It is therefore reasonable to posit that a balanced technology strategy leads to more effective product platforms and actual products where market applications require excellence in more than one core technology. Might sales have shown a marked increased after a full ten years of product announcements and distribution simply because the firm was becoming better known in the industry? We think not. Discussions with management indicated that the acceleration of sales was due primarily to the fact that the firm was developing and selling vastly better products by virtue of their improved functionality and the range of hardware upon which they could function. At the same time, the relationship observed between technology strategy and sales must be tempered by the further observation that the company studied had continuously sought to become more "market oriented" and to create a more effective selling capability. The technology strategy of the company co-evolves with many factors that affects the timing of product introductions to the market and sales gained from them. As observed by Roberts [14], in successful technology-based enterprises increased orientation towards the market produces strategic adjustments in both technology strategy and specific product developments. These orientations and strategies in turn produce higher sales.
Competitive intensity in the real-time market has been rather stable, with three or four suppliers of Unix-based or Unix-derived real-time software environments. The release of directly competitive products can force a firm to package the technology that it has presently available into products for shipment, and delay the incorporation of an optimal technological capability for a later product release. While always a market leader in terms of real-time capability, the firm under study came to be a market follower in the porting arena (which is a composite of new versions of Unix running on new processors). For example, Platform 3 was rushed to market within a year (Unix 5.3.2 on Intel 80486-based hardware), forcing a surge of investment in the porting arena during 1990. In this company's particular industry, product releases are often timed to exhibitions at major annual trade shows.

Market growth is another factor. According to industry experts, the rate of market growth for Unix on PC's has consistently ranged between 20% and 35% in the time period shown in the Figures. The rate of growth market growth for real-time applications has also ranged between 20% and 35% over the past five years. Thus, one might observe that the market provided a certain consistency over time with respect to the potential for achieving sales growth with strong products.

Distribution is another critical influence on sales. The company has maintained two distribution channels throughout its history: distributors or value-added resellers for more general purpose Unix applications, and its own direct sales force for embedded real-time markets. Management believes that its sales capability has never truly been great, nor poor, but on par with its major competitors.

**Influences on Product Evolution**

Why did the firm described above choose when to renew its product line and why were particular technological foci chosen at these times? Does its experience have broader application to the more general issue of platform renewal?

The two most direct factors shaping management's decisions regarding technological focus and the timing of projects were firm size and management's perception of the market. A small firm may find itself believing that resources are too few to invest concurrently in several or more core technologies. Yet, this may be a trap (as it was for the Unix software company), since the product
excellence required to propel growth may rely on strength and balance among embodied core technologies. The need for balance might not be clearly perceived. For example, in this case, management envisioned Unix "taking over" the office automation desktop computing marketplace during the latter part of the 1980's, a market that would not require real-time data acquisition. When this market failed to materialize, the firm refocused back towards technical markets needing real-time systems. Technology does not evolve in isolation from a firm's target markets.

The availability, adoption, and integration of external technology has also had major bearing on the evolution of the company's product family. The firm has always licensed Unix from external suppliers, and then added its own value to the software. Looking again at the product family map, the reader can observe the stream of external technology flowing into the firm, expressed in the form of successive versions of Unix. Management spends a substantial amount of time developing and nurturing relationships with various Unix licensers (such as AT&T or Novell), always seeking to learn about their respective platform renewal efforts. Adoption of external technology must be judicious. For example, in Platform 2, the company skipped implementing Unix System 3, and waited instead for the next release, Systems V. Management felt that the functionality of Unix System 3 did not merit the effort needed to port it, and was assured by its contacts within AT&T that more powerful software would soon be delivered and deserve to be the basis of the firm's next product platform.

Integration of external technologies is platform driven. Al Lehnerd (formerly an executive at Black and Decker, SunBeam and SteelCase) speaks of a "composite design" process where, for physical, assembled products, each new platform version may be visualized on paper by its key subsystems and the interfaces between them [7]. We believe that the composite design approach applies well to software products. For example, Figure 4 showed the "composite design" of the Unix software company's recent product platform. Armed with this design, the development team can then look globally to find best in class suppliers for each major subsystem and take the steps necessary and appropriate to fully understand the techniques and processes used by these suppliers to excel.

The composite design process can also reveal where a firm might best focus its own value added and the partners that it might work with to create the complete solution. Further, before any platform engineering work is started, the team can first test the "composite design" with customers, having them do the type of tradeoff assessments between product features and then
between product features and price that is standard practice in many consumer products companies. The feedback from this market research can make development more focused and effective. In our experience, however, the new product designs of software firms are driven far more frequently by the personal vision of technical entrepreneurs than by any systematic market research anchored in composite design proxies.

The idea of a composite design applies directly to this company's customers. Many of the company's present customers make embedded systems, comprised of general purpose computing hardware, custom electronics, the real-time Unix operating system, run-time versions of various data management and user interface software development tools, and the customer's own applications software. These systems are then sold to end-user organizations, such as factories, hospitals, telephone operations, and airlines.

The nature of these customers' respective businesses has produced a particular constraint for the company that has had direct bearing on its own technology strategy. Its customers often had far longer time lines concerning the need for platform renewal than the company itself. For example, one of these customers was a medical equipment manufacturer. Over the last several decades, the manufacturer has spent three to five years developing its own "platforms" for products such as patient monitors. The real-time operating system software would be just one key component among many others in the overall design of these medical products. Once the equipment platform was completed, the manufacturer would then typically generate numerous derivative products for five to ten years before introducing the next generation of medical equipment. The costs of transitioning to a new platform versions, obsoleting existing ones, were simply too high to undertake renewal with greater frequency.

Therefore, while the Unix software company had been able to deliver substantively new technology every two to three years, many of it customers managed their own platform renewals at time cycles approximately twice that duration. As overall integrators of special purposes systems, these customers have had to synchronize the technology advances of their numerous suppliers with their own respective product development plans. Suppliers of component technologies, such as this Unix company, have had to maintain support for their older technologies while finding new applications for their "next" platforms.
Conclusion

This article makes a contribution at several levels. First, we found that the platform concept may be productively employed to study and improve management for software products, and in particular, for streams of closely related products derived from a common technical platform. The need for continuous renewal of underlying product platforms also became clearly apparent as we studied this firm. One of the more fundamental aspects of such renewal is comprised of the engineers hired and assigned to advance core technologies.

Our work clearly argues for a technology strategy that is inclusive of all necessary technologies required in current and future product architectures. We studied a firm that had shifted the relative emphasis placed on its core technologies in order to gain access to new markets. The firm's choice of technology strategy was found to be associated with commercial success. When the company focused on one core technology at the expense of another, its products fared poorly. When it pursued a more complete and integrative technology strategy, the opposite effect was observed. Thus, the focus and level of R&D investments will have strong bearing on the nature and quality of a firm's evolving product platforms and the products derived from them.

Lastly, the work presents methods to understand the evolution of technology strategy and its relationship to both the architecture of products and firm performance. We created a product family map for a software company's products, differentiating between platform improvements and derivative product developments. Using this product map, we then examined R&D budgets and allocated this spending to basic areas of core capability embodied in the firm's products. Coming to this research, we had been both intrigued by the whole notion of core competencies and capabilities, and frustrated by the near absense of any work actually measure changes in these phenomenon and their business impact. This article has presented a method to examine changes in technical capabilities as they affect and shape a firm's product developments.

We have observed that the gathering and visual presentation of data associated with product family maps and core capability investments can help a company's management more clearly see the consequences of past decisions and learn from them. We also believe that these techniques may be a catalyst for considering a new type of product generation and resource allocation process that will stand in striking contrast to the single product, single period, portfolio-minded methods commonly used today.
References


The levels of change between represented by any given product relative to all products made by the firm in the past are:

Technological change:

1. Minor improvements to core technology(ies) that exist in earlier products
2. Major enhancements to existing core technology(ies)
3. The addition of a new core technology, which is combined or otherwise integrated with existing core technologies in earlier products to create the new product
4. New, unrelated core technology

Market Applications Newness: For each product, observe relative to all prior products, any change in a) customer group, b) functional use of the product by the same or new customer group, or c) distribution channel.

1. No change in (a), (b) or (c)
2. Change in one of (a), (b) or (c)
3. Change in any two of (a), (b) or (c)
4. Concurrent change in (a), (b) or (c)

We also asked colleagues in our computer science departments to identify core technologies in real time Unix products. They concurred that for a firm which enhances Unix licensed from other companies, the porting of the operating system and real-time enhancements to the kernel would be two essential core technologies.

In small firms, human resource allocations can be similar for startup periods versus the second stage of growth for two reasons: startup funding to pay for early engineering, and sweat equity, where lead engineers receive stock compensation in lieu of salary.

This market research technique is called conjoint analysis.
<table>
<thead>
<tr>
<th>Platform</th>
<th>Platform Version</th>
<th>Product</th>
<th>Start R&amp;D</th>
<th>End R&amp;D</th>
<th>Real-Time Level (milliseconds)</th>
<th>Man Year Allocation to Real Time</th>
<th>UNIX Version</th>
<th>Computer Hardware</th>
<th>Man Year Allocation to Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Investigation</td>
<td>No product</td>
<td>1/76</td>
<td>1/80</td>
<td>From None to 10</td>
<td>2</td>
<td>ATT Version 6</td>
<td>DEC PDP</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Platform 1</td>
<td>Initial</td>
<td>Product 1</td>
<td>1/79</td>
<td>1/80</td>
<td>10</td>
<td>0</td>
<td>ATT Version 6</td>
<td>PDP-11/23</td>
<td>2</td>
</tr>
<tr>
<td>Platform 1</td>
<td>Initial</td>
<td>Product 2</td>
<td>1/80</td>
<td>1/81</td>
<td>10</td>
<td>0</td>
<td>ATT Version 6</td>
<td>Other PDP Mini's</td>
<td>2</td>
</tr>
<tr>
<td>Platform 1</td>
<td>Initial</td>
<td>Product 3</td>
<td>1/83</td>
<td>1/84</td>
<td>10</td>
<td>0</td>
<td>ATT Version 6</td>
<td>DEC Professional</td>
<td>2</td>
</tr>
<tr>
<td>Platform 2</td>
<td>Initial</td>
<td>Product 4</td>
<td>1/82</td>
<td>1/84</td>
<td>10</td>
<td>0</td>
<td>ATT Version 7</td>
<td>Intel 8086 PC's</td>
<td>8</td>
</tr>
<tr>
<td>Platform 2</td>
<td>Initial</td>
<td>Product 5</td>
<td>1/85</td>
<td>1/86</td>
<td>10</td>
<td>0</td>
<td>ATT Unix 5.1</td>
<td>Intel 80286</td>
<td>2</td>
</tr>
<tr>
<td>Platform 2</td>
<td>Initial</td>
<td>Product 6</td>
<td>6/86</td>
<td>6/87</td>
<td>10</td>
<td>0</td>
<td>ATT Unix 5.2</td>
<td>Intel 80286</td>
<td>2</td>
</tr>
<tr>
<td>Platform 2</td>
<td>Second</td>
<td>Product 7</td>
<td>1/87</td>
<td>1/88</td>
<td>5</td>
<td>2</td>
<td>ISC's Unix 5.3</td>
<td>Intel 80386</td>
<td>1</td>
</tr>
<tr>
<td>Platform 2</td>
<td>Second</td>
<td>Product 8</td>
<td>1/88</td>
<td>1/90</td>
<td>1</td>
<td>2</td>
<td>ISC's Unix 5.3.1</td>
<td>Intel 80386</td>
<td>1</td>
</tr>
<tr>
<td>Platform 3</td>
<td>Initial</td>
<td>Product 9</td>
<td>1/90</td>
<td>1/91</td>
<td>1</td>
<td>0</td>
<td>ISC's Unix 5.3.2</td>
<td>Intel 80486</td>
<td>2</td>
</tr>
<tr>
<td>Platform 3</td>
<td>Initial</td>
<td>Product 10</td>
<td>1/90</td>
<td>1/91</td>
<td>1</td>
<td>0</td>
<td>ISC's Unix 5.3.2</td>
<td>Intel 80486</td>
<td>.5</td>
</tr>
<tr>
<td>Platform 3</td>
<td>Initial</td>
<td>Product 11</td>
<td>1/90</td>
<td>1/91</td>
<td>1</td>
<td>1.5</td>
<td>ISC's Unix 5.3.2</td>
<td>Intel 80486</td>
<td>0</td>
</tr>
<tr>
<td>Platform 3</td>
<td>Second</td>
<td>Product 12</td>
<td>1/91</td>
<td>9/93</td>
<td>1</td>
<td>0</td>
<td>Novell Unix 5.4.2</td>
<td>Intel 80486</td>
<td>2</td>
</tr>
<tr>
<td>Platform 3</td>
<td>Second</td>
<td>Product 13</td>
<td>7/92</td>
<td>9/93</td>
<td>1</td>
<td>1.5</td>
<td>Novell Unix 5.4.2</td>
<td>Intel 80486</td>
<td>1</td>
</tr>
<tr>
<td>Platform 3</td>
<td>Third</td>
<td>Product 14</td>
<td>1/93</td>
<td>4/94</td>
<td>.2</td>
<td>2</td>
<td>Novel Unix 5.4.2</td>
<td>Intel 80486/586</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 1
Typology of Platform Change in Product Family Evolution

Platform Architecture: Common Subsystems and Interfaces for Multiple Products

Platform Extensions: A new generation where number and types of subsystems and interfaces remain constant, but where subsystems and interfaces are enhanced.

Platform Renewal: A New Architecture, where subsystems and interfaces from prior generations may be carried forward and combined with new subsystems and interfaces in the new design.
Figure 2

The Evolution of a Software Product Family
Initial R&D: Founder's Unix/real-time software expertise gained in Physics department.

Company Founded: Making Unix work in 64k address space of the 11-23 was a great challenge. First licensed Unix real-time product.

Platform 1 (ATT Unix Version 6 with Real-time Scheduler)

Product 1: for scientific and other technical applications, PDP 11-23 version, 10ms response time.
R&D: 1/76 to 1/80, 2 MY

Product 2: Port to other PDP 11 hardware
R&D: 1/80 to 1/81, 2 MY

Product 3: DEC Professional port
R&D: 1/83 to 1/84, 2 MY's

ATT releases Unix 3.0 (bust) and 5.1 and 5.2

Platform 2.1: ATT Unix (Version 7 and then System V with real-time for Intel hardware)

Initial R&D: Surmounting poor memory management in 8086 was a major feat.

ATT Unix V 7
1982-1984

Product 4: 8086 hardware port with real-time, 10 ms response time.
R&D: 1/82 to 1/84, 8 MY's

ATT Unix 5.1
Product 5: 80286 hardware port, improved general kernel performance
R&D: 1/85 to 1/86, 2 MY's

ATT Unix 5.2
Product 2: Kernel enhancements for binary compatibility with ATT's 5.2 Unix 286 port.
R&D: 6/86 to 1/87, 2 MY's

ATT releases Version 5.3: Interactive Systems becomes major OEM for Intel Port

Platform 2.2: Interactive System's Intel Unix with real-time enhancements

IS Unix 5.3
Product 7: Unix 5.3 on 80386 hardware, improved real-time performance, 5ms
R&D: 1/87 to 1/88, 3 MY's

IS Unix 5.3.1
Product 8: Fully Preemptive Kernel, 1 ms.
R&D: 1/88 to 1/90, 3 M's
Pursued high software content, Real Time Unix embedded applications in industrial, telecommunications, and medical markets. Developed full-scale development and run-time products.

ATT releases Version 5.3.2 and 5.4

Platform 3: Embedded Systems Run-time and Development Environments, System V Unix

Initial R&D: 1/90 to 1/91, 4 MY's, $360k

<table>
<thead>
<tr>
<th>IS Unix 5.3.2</th>
<th>Product 9: Interactive Systems Unix 5.3.2, on 486 hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS Unix toolkit</td>
<td>Product 10: Development Tools for Testing and GUI</td>
</tr>
<tr>
<td>ATT Unix 5.3.2</td>
<td>Product 11: AT&amp;T Unix 5.3.2 Kernel for run-time, still 1 ms response</td>
</tr>
<tr>
<td>Run-time license</td>
<td>Novell acquires ATT's Unix Labs</td>
</tr>
<tr>
<td></td>
<td>Creates Unixware for Intel, Unix 5.4.2</td>
</tr>
</tbody>
</table>

Platform 3.2: Novell Unix V.4

Novell's Unixware Unix 5.4.2

Product 12: Novell's Unixware on 486 hardware, development environment
R&D: 1/92 to 9/93, 2 MY's
Product 13: Product with real time Kernel
R&D: 7/92 to 9/93, 2.5 MY's

Platform 3.3: New Run-Time Module

Product 14: Embedded Runtime, .2ms
R&D: 1/93 to 4/94, 3 MY's
Figure 3
Technology and Market Newness
Assessed for a Software Product Family

Technology Newness

New, Unrelated
Core Technology

New, Related
Core Technology

Major Enhancement
Existing Core Technology

Minor Improvement
Existing Core Technology

Market Applications Newness
Customer Groups, Functional Uses, Distribution Channels

<table>
<thead>
<tr>
<th>Product</th>
<th>Technology</th>
<th>Market Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 1</td>
<td>Basis Point: Real Time Unix, PDP11-23</td>
<td>Scientific markets, direct sales</td>
</tr>
<tr>
<td>Product 2</td>
<td>Minor improvement to core technology</td>
<td>New Group: Engineering Companies</td>
</tr>
<tr>
<td>Product 3</td>
<td>Major enhancement, DEC Pro port</td>
<td>New Channel, government market</td>
</tr>
<tr>
<td>Product 4</td>
<td>Major enhancement, Intel Port</td>
<td>New market (Office Automation), New Channel</td>
</tr>
<tr>
<td>Product 5</td>
<td>Minor improvement</td>
<td>Same</td>
</tr>
<tr>
<td>Product 6</td>
<td>Minor improvement</td>
<td>Same</td>
</tr>
<tr>
<td>Product 7</td>
<td>Major enhancement, new hardware port and real time advances</td>
<td>New Use, Manufacturing embedded systems</td>
</tr>
<tr>
<td>Product 8</td>
<td>Major enhancement, to real time</td>
<td>Same</td>
</tr>
<tr>
<td>Product 9</td>
<td>Minor improvement, new Unix port</td>
<td>New channel formed, and new markets (medical)</td>
</tr>
<tr>
<td>Product 10</td>
<td>New, related core technologies, new GUI tools for process control integrated with Unix product</td>
<td>Same</td>
</tr>
<tr>
<td>Product 11</td>
<td>Minor improvement (embedded run-time)</td>
<td>Same</td>
</tr>
<tr>
<td>Product 12</td>
<td>Major enhancement (new hardware port)</td>
<td>New market (telecommunications)</td>
</tr>
<tr>
<td>Product 13</td>
<td>Minor improvement (embedded version)</td>
<td>Same</td>
</tr>
<tr>
<td>Product 14</td>
<td>Major enhancement, real-time</td>
<td>New market (telecommunications)</td>
</tr>
</tbody>
</table>
Figure 4
System Architecture: Subsystems and Interfaces

Application Level

User Interfaces

Custom Applications

Standard Applications

Software Development Tools

Real Time System Calls

Unix System Calls

Operating System Level

Standard Unix Kernel

Real Time Process Scheduler

Real Time File System Extensions

Embedded Systems Delivery Kernel

Custom Device Drivers

Standard Device Drivers

Hardware Level

Computers, Devices & Peripherals
Figure 5: Sales Growth and Resource Allocation

- Percentage of Total R&D Spent on Each Core Capability
- Sales of Product Family Normalized on Scale of 1 to 10

Legend:
- % Real Time Capability
- % Porting Capability
- Sales