A Platform Based Approach for Embedded Systems Software Development

by

Deepak Seth

M.S., Computer Systems Engineering
Northeastern University, 1999

B.S., Electrical Engineering
Northeastern University, 1995

Submitted to the System Design and Management program
in partial fulfillment of the requirements for the degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

June 2006

Signature of Author:_____
Deepak Seth
System Design & Management Program
May 2006

Certified by:_____
Michael A. Cusumano
Sloan Management Review Distinguished Professor of Management
Thesis Supervisor

Accepted by:_____
Patrick Hale
Director, System Design & Management Program

THIS PAGE IS INTENTIONALLY LEFT BLANK

A Platform Based Approach for Embedded Systems Software Development

by

Deepak Seth

Submitted to the System Design and Management program
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering and Management

Abstract

A platform based approach for product development allows companies to eliminate redundancies, efficiently utilize its resources and provide products for a wider market. The basic idea is to develop and share key components and to introduce new technologies in as many products as possible. The automobile industry has for long used the concept of product platforms and has successfully achieved savings in development costs and seen a growth in sales and market share.

By creating a common software platform, this concept can be applied to software development for embedded systems where software modules and applications can be shared across products within a product family. This provides better code reuse and increases standardizations across products.

This thesis will examine how the concept of platforms can be applied to software development from the viewpoint of the telecommunications industry. By using the power of a common software platform, telecommunication equipment makers can accelerate product delivery and introduce new technologies to a wider range of customers. With the right strategy, they can also make their products into platforms that serve as a foundation on which other companies can develop products and offer their services.

Thesis Supervisor: Michael Cusumano
Title: Sloan Management Review Distinguished Professor of Management

THIS PAGE IS INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

I would like to first thank my thesis advisor, Professor Michael Cusumano, for his guidance and insight through this research effort. I owe a debt of gratitude to my directors at Nortel Networks, Louis Farina and Lance McNally, for their encouragement and support of my education at MIT. Finally, I wish to thank my family for their patience and continued support during my thesis writing.

The SDM faculty and staff have been very helpful throughout the program and my fellow classmates have made my experience at MIT a rewarding and enjoyable one.

THIS PAGE IS INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1: INTRODUCTION

For many companies, the path to success, growth, and customer loyalty has been built on excellence in product development. Only a handful of companies have been able to sustain that excellence through periods of changing technology and market conditions. Some companies like Microsoft, Intel, Toyota and Cisco advance their products while others fail to do so. Some companies offer exciting new products and others do not. The long term success of an enterprise depends on a stream of new products, which is a key to corporate prosperity.

The basic idea of a platform based approach is to create new products that share key components but provide enough differentiation such that each product will attract different customers. The automobile industry has for long used the concept of product platforms and has successfully achieved savings in development costs and seen a growth in sales and market share. Ideas from the automobile industry can be applied to embedded products in the telecommunications industry, an industry still recovering from the collapse of 2001. Telecom equipment suppliers are faced with challenges to reduced development costs and yet still need to achieve growth and market share in an industry undergoing consolidation among carriers and equipment suppliers.

Development using a common platform

The long term success of an enterprise depends on a stream of new products. Some are for replacing existing products and some for new markets. When products are developed

one at a time, there are redundancies in the product development effort and it is not the most efficient use of resources. In some cases technologies are developed, just to serve a single product. It is not possible to dominate large markets by developing and mass producing one product at a time. If the technology designed for one product is used in multiple products, it can lower design and development costs by reducing design cycle time. The best companies today view projects as part of a portfolio and make the most of their project investments by introducing new technologies in as many products as frequently as possible[1].

By using a common platform, and concentrating on the development and sharing of key components, a set of derivative products can be created. Thus a company can develop a foundation for a range of products and a product family can be created. A company can capture a wider market share by launching a number of products instead of one segment at a time. Also, with a platform based approach to product development, a company can make its product a foundation on which other companies build their products or offer their services.

Embedded system products

Advances in design and fabrication techniques of semiconductor devices have resulted in an increased availability of microcontrollers with decreasing price and increasing performance. This has enabled the proliferation of embedded systems which provide dedicated and integrated services to end users. Devices include PDAs, cell phones, MP3 players and digital cameras. These products contain embedded software that provides end

users with a range of features and functionality in these products. Software is increasingly the portion of the system that enables these unique behavioral characteristics [2]. Competitive developers of end-user system products find themselves increasingly developing software, even though the system itself combines both hardware and software. Since hardware-manufacturing cycles are more expensive and time-consuming, software based implementation has become more popular. The increased computational power of processors and correspondingly decreased size and cost let designers move increasingly more functionality to software.

In recent years, the functions demanded for embedded systems have grown so numerous and complex that development time is increasingly difficult to predict and control[3]. To cope with the growing requests for new embedded system products and to fulfill consumer appetite for greater functionality and services, makers of embedded products need to reduce the development time for the embedded system software. By using a platform based approach for software development, embedded system makers can enhance productivity, improve predictability, decrease the time to market, and increase quality of their products.

In embedded products, a common software architecture across the same product family can provide many benefits. By sharing the same operating system, system services and software components across different product lines, product developers that create applications from one product can be shared across all products within the same family. This also simplifies porting, maintenance and quality assurance of these applications. The

heterogeneity of hardware architectures, the diversity of operating systems and stiff global competition are making it less feasible for development organizations to develop applications from scratch. As a result, companies need to increase reuse of components within a product family. In a Common Software Platform, common software components as well as a standardized set of off-the-shelf components can be leveraged across products to capitalize on synergies and accelerate product delivery.

Software Reuse

Software reuse is the process of creating software systems from existing software rather than building software systems from scratch[4]. The primary advantage of reusing software is that it reduces the time and effort to develop software systems. Software reuse in practice has been limited, however. While this concept is so powerful, the reuse of application software has not lived up to its promises, and at times has resulted in losses and led to accidents[5]. In practice, software reuse has been limited to the cutting and pasting of code from one project to another. A common software platform approach, where software modules are shared across products, can provide effective code reuse. Also this results in software commonality and standardization across product lines.

The Telecommunications Industry

In certain industries, the functionality provided by software had become a commodity. For example, in the networking industry, the protocols that provide end-to-end connectivity and govern the communications capability between two nodes (IP, BGP, OSPF, etc) have become a minimum requirement for any internet class router. Likewise,

every Personal Computer has an IP stack. Here, these protocols are developed to a standard defined by the Internet Engineering Task Force (IETF). Since there is limited room for innovation within these protocols, new networking products are being brought to market in a radically different way. The new business realities for network equipment manufacturers - downsized engineering, outsourced development, and the need for value added differentiation - demand changes to traditional business models to remain competitive. By leveraging third party software and using a platform based approach, Original Equipment Manufacturers (OEMs) can bring products to markets at lower price points.

Telecom equipment manufacturers need to look at their products in a different way. While they don't directly sell routers and switches to end users, demand for network bandwidth and telecom equipment doesn't come directly from carriers or ISPs, it comes from society. As the demand increases for high-capacity transmission, especially with the rising volume of Internet data, telecommunications companies have been expanding and upgrading their networks to increase the amount of available bandwidth[6]. Applications such as Voice over IP, IPTV, Video on Demand, all create demand for telecom products. Telecom equipment makers need to create a communication platform that can not only drive innovation in this industry but also serves as a foundation where ISPs and telecom carriers can provide their direct customers with enhanced communication capabilities.

The success of a company in a marketplace depends on how it manages technology to create and deliver products and services to its customers. Utilizing the platform software

paradigm requires specific system architectures. It also requires an organization structure in place to take advantage of this approach. Additionally, the organization needs to have business process in place to utilize this concept. This thesis will examine how the concept of platforms can be applied to software development from the viewpoint of the telecommunications industry. It will also explore how product development groups can adopt this concept.

Plan of this thesis

This document is organized into six chapters. Chapter 2 provides an overview of product platforms and product families. It also provides examples of platforms in the automotive and aerospace industry where common components and technologies have been shared. This chapter also discusses industry platforms where companies such as Intel and Microsoft have made their product foundations on which other companies build their products and offer their services. Chapter 3 provides some background into the telecommunication industry and equipment manufacturers for this industry. Chapter 4 discusses embedded systems and the concept of common software platforms. It describes how to create a common software platform for a product family comprising of embedded system products. It also describes strategies for multiple projects and mechanism in which technology can be transferred within projects. Chapter 5 discusses a platform strategy for the telecom equipment manufacturers. The Four Levers framework on platform leadership developed by Gawer and Cusumano is used for formulating a strategy. In addition, organizational structures, business processes and software

development environments for creating common platforms are presented. Chapter 6 provides conclusions of this study and final thoughts.

CHAPTER 2: PRODUCT PLATFORMS AND PRODUCT FAMILIES

2.1 Definitions of Product Platforms

McGrath[7] defines product platforms as a collection of the common elements, especially the underlying core technology, implemented across a range of products. A product platform is primarily a definition for planning, development, and strategic decision making. Product platforms are designed to serve specific group whereas the product architecture is not necessarily developed with this group in mind.

A product platform is a set of subsystems and interfaces that form a common structure from which a stream of derivative products can be efficiently developed and produced.

Oliver deWeck[8] defines a platform as set of common or shared elements and its interface definition. Elements can be all kinds of architectural elements, such as parts components, systems, processes and organization.

Gawer and Cusumano[9] define a platform as an evolving system made of independent pieces that can be innovated upon. It can refer to a foundation product that has the most value when it works as the core of a system of components made by one or more firms.

Meyer and Utterback[10] define platform as set of subsystems and interfaces that form a common structure from which a stream of derivative products can be efficiently developed and produced.

2.2 Product Families

A Product Family is a group of related products that share common features, components, and satisfy a variety of markets. Product families do not have to emerge one product at a time. In fact, they are planned so that a number of derivative products can be efficiently created from the foundation of common core technology.

According to Meyer and Utterback[10], products that share a common platform but have specific features and functionality required by different sets of customers comprise a product family. They discuss in detail about product family evolution, platform renewal, and new product creation.

2.3 Motivation for product platforms

Since many companies design new products one at a time, the focus on individual customers and products often results in a failure to embrace commonality, compatibility, standardization, or modularization among different products or product lines. As a result, a "mushrooming" or diversification of products and components takes place, with proliferating variety and costs[11].

The long term success of companies depends on new product creation, some to replace older ones and others for entering new markets. Companies typically design new products, one product at a time. Traditional methods for such a type of product development fail to deliver in the long run, because the single product has to compete for

resources against other projects in the corporation's portfolio. Every product team must justify its own existence throughout the process of development and commercialization. Approval gates swing open and shot as single development projects move forward. Budget, break-even and cycle-time measures are all typically calculated on the basis of single products. The end result of a single project focus is a failure to embrace commonality, compatibility, standardization or modularization among different products[11].

Much current management thought addresses developing single products as rapidly as possible. This approach leads to a redundancy of both technical and marketing efforts as well as a lack of long-term consistency and focus[10]. If companies build an entire family of products that utilized the common underlying technology for a particular market, it would make product development a lot more efficient. Rather than having separate development teams working on single projects, wouldn't it be better to have them join forces in building a common platform or a design from which a host of derivative products could be effectively and efficiently created.

Companies can remain competitive by utilizing product platforms and product families. This can lead to increase in product variety, shorten product lead time, and maintain economies of scale to reduce costs.

2.5 Multi project Management

In *Thinking Beyond Lean*, Cusumano and Nobeoka[1] talk about multi-project management and the benefits this kind of thinking can bring to projects and to companies. Managers can view new product development in two ways. They can either take the viewpoint as if each project and products exists in isolation or they can view each project as part of a broader portfolio of projects – existing in the past present and the future. By following multi-project thinking, managers can maximize the chances that the organization produces a stream of new products that cover a range of market segments and makes the best possible use of R&D investments.

Lean refers to a general way of thinking and specific practices that emphasize less of everything – fewer people, less time, lower costs. However, for managers, multi-project thinking fits reality much better than focusing on single projects, which lean practices in product development emphasize. Most companies have more that one product, and many companies have more than one new product under development at the same time. Companies may not deliberately link projects by sharing components. Nonetheless, any firm requires some sort of multi-project management if its projects compete for key engineers or financial resources, or target similar customer in the marketplace. Research by Cusumano and Nobeoka suggests that, over the long term, aiming for new designs or hit products in isolated projects in not enough. The best companies today view projects as part of a portfolio and make the most of their project investments by introducing new technologies in as many products as frequently as possible[1].

2.6 Platforms in the Automotive Industry

We can learn many lessons from the automotive industry. Automobile manufactures have for long faced challenges in creating many product lines with many project to co-ordinate, and with complex products with many components. They have numerous products line with lots of ongoing projects simultaneously. Research conducted by Cusomano and Nobeoka indicates that the best way to work for the good of the firm and create a portfolio of products at low cost is to shift the company mindset into a *multi project mode*[12], where the firm develops some totally new products but focus an equal amount of attention on developing common core components and quickly sharing these across multiple projects. Companies such as Toyota have followed a deliberate approach to achieve this and have created families of well-integrated products that share design concepts as well as key components and basic technologies.

2.6.1 Platforms at Toyota

Toyota[13] has served as a benchmark in many industries for manufacturing and product development. Using a project-centered management system, Toyota has set new standards in the auto industry.

During 1992-93 Toyota adopted a strategy and structure specifically for multi-project management of product development. Toyota created three vehicle development centers that group similar projects together, based on common platforms. A fourth center provides common components to the different development centers.

Center 1 was responsible for rear-wheel-drive platforms and vehicles.

Center 2 was responsible for front-wheel-drive platforms and vehicles.

Center 3 was responsible for utility vehicle/van platforms and vehicles.

Center 4 was responsible to develop components for all systems for all vehicle projects.

Toyota had considered alternative groupings such as by product segment (luxury versus economy versus sporty cards, or small versus medium versus large cars). Toyota management chose platform similarity because this would lead to the highest level of technology sharing among projects with a center. Toyota managers concluded that because new platform development requires extensive resources, using common platform designs for multiple product lines would provide savings in engineering investments and reduce production costs most efficiently.

This structure differed from Toyota's former project centered organization (used prior to 1992) and it provided the firm with significant improvements in its ability to coordinate across projects as well as to integrate different engineering function projects and related sets of projects.

In the mid-1990s, Toyota offered 5 distinct platforms for low cost vehicles (at Center 2):

1. Celica/Carina ED/Daren
2. Camry/Vista
3. Corona/Carina
4. Corolla/Sprinter
5. Tercel/Corsa/Starlet

2.6.2 VW Platform approach

Volkswagen saves about $1.7 billion annually on development costs using a platform based architecture[14]. The Volkswagen, Audi, Seat and Skoda are based off the same platform sharing the floor group, drive system, running gear and unseen part of cockpit with numerous other elements.

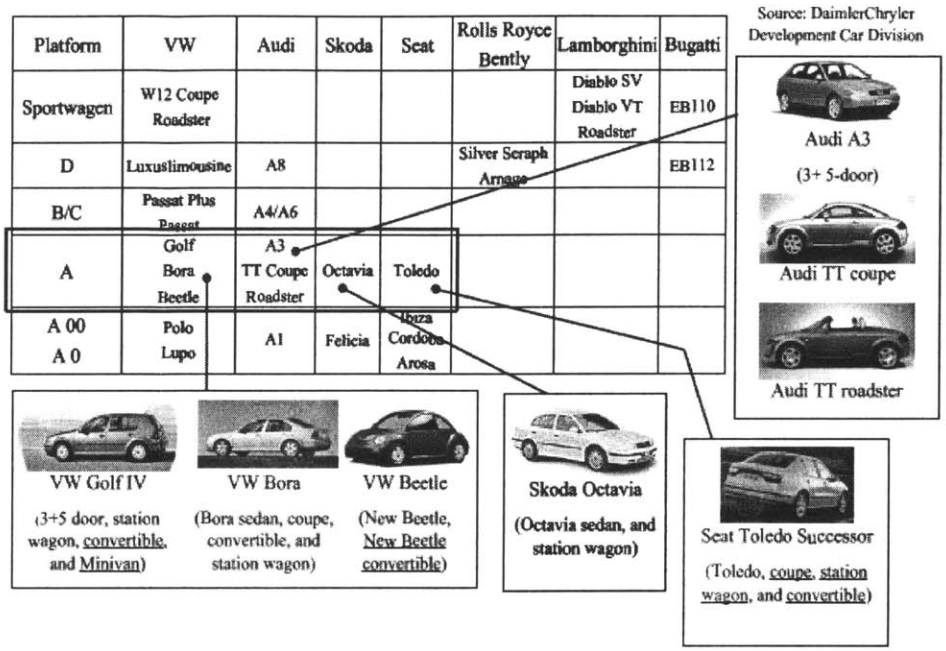

Figure 2.1: Volkswagen A Platform and Product Family

In the late 1990's, Volkswagen was recognized as the leading product platform implementer in the automotive industry. Volkswagen defines a product platform as a unit that has no impact on the vehicle's outer skin and that is a chassis including the inner wheelhouses[14]. Volkswagen's product platform consists of common components, such as front axles, rear axles, front end, rear end, wheels, steering system, brake system, center floor, fuel tanks, exhaust systems, and seat frames. In 1998, Volkswagen owned four out

of top ten vehicle platforms (by production volume). Figure 2.1 shows product variants derived from the Volkswagen A Platform[15].

At one time, Volkswagen shared over 65% of its components within its product families, resulting in huge cost savings from economies of scale. However, its common component platform strategy also had some drawbacks. Volkswagen suffered from cannibalization by Skoda, its platform sharing partner. There were issues concerning unforeseen performance drawbacks on the Audi TT. Finally, brand blending from sharing too many components became an issue.

2.7 Platforms in the Aerospace Industry

2.7.1 The Boeing 737

The Boeing 737 is divided into 3 platforms[8]:

- Initial-model (100 and 200)
- Classic (300, 400, and 500)
- Next generation (600, 700, 800, and 900 model

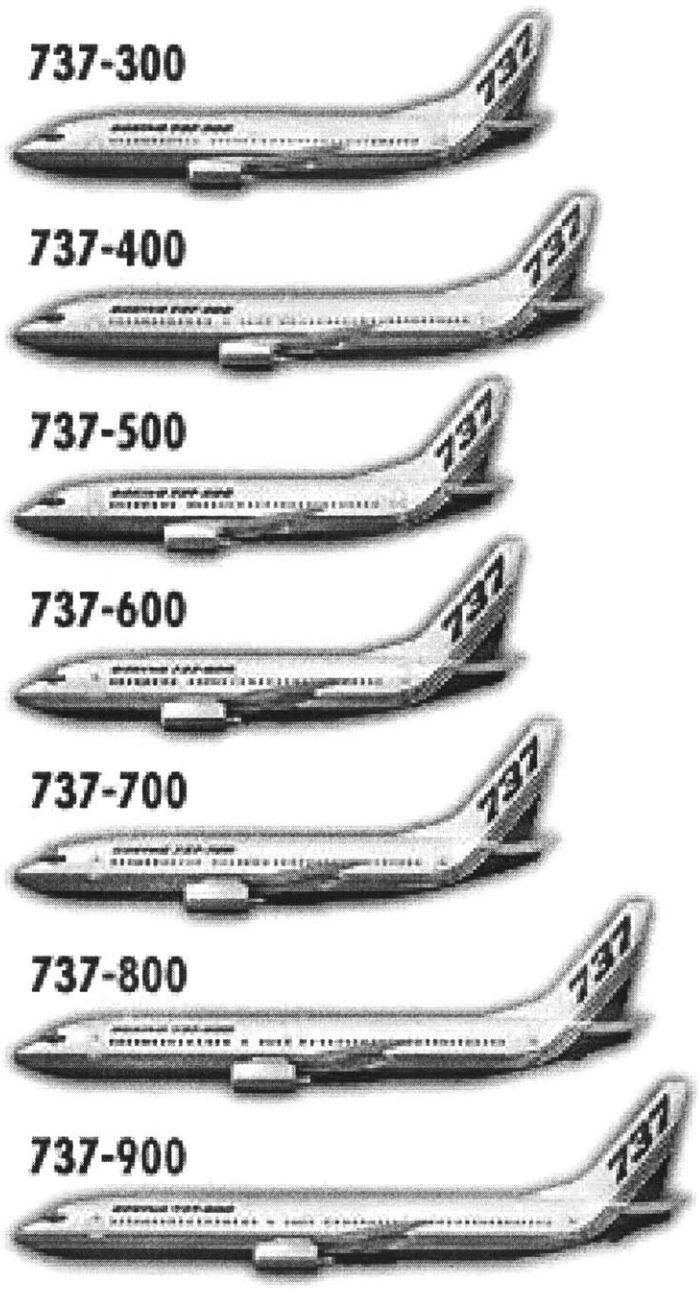

Figure 2.2: The Boeing 737 product family

All three aircraft share common height and width, but their fuselage lengths are different as shown in Figure 2.3.

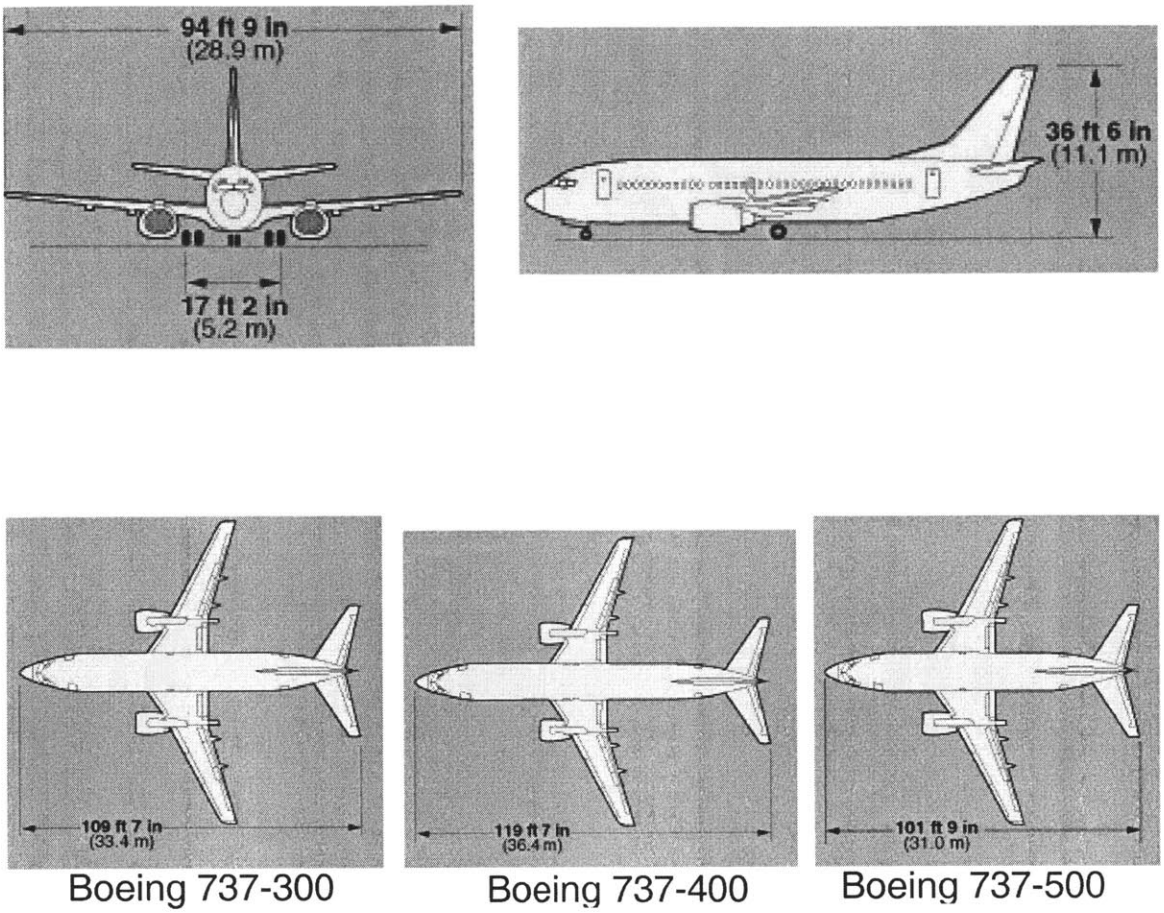

Figure 2.3: Boeing 737 product dimensions

Likewise, Airbus also uses common wings, nose and fuselage modules to create aircraft of different lengths and capacities

2.7.2 The Blended Wing Body

The Blended Wing Body (BWB), shown in Figure 2.4, is a new concept in aircraft design[16]. The BWB is a hybrid shape that mainly resembles a flying wing, but also incorporates some features of a conventional airliner. The futuristic airframe is a unique

merger of efficient high-lift wings and a wide airfoil-shaped body, causing the entire aircraft to generate lift and minimize drag, thereby increasing fuel economy. Passenger and cargo areas are located within the center body portion of the aircraft. The BWB has significant benefits over families of tube and wing transports with its ability to cover the large airplane market with one cross section The BWB type aircraft can be scaled from 200 passengers to 450 passengers with identical wings, identical cockpit as well as identical and similar bays as shown in Figure 2.5. Each bay in the BWB is an identical "cross-section" and thus lends itself to high part/weight commonality amongst the family members. By sharing a common wing, cockpit and centerbody element, the BWB can be used as commercial aircraft, a tanker, a bomber or a global reach freighter.

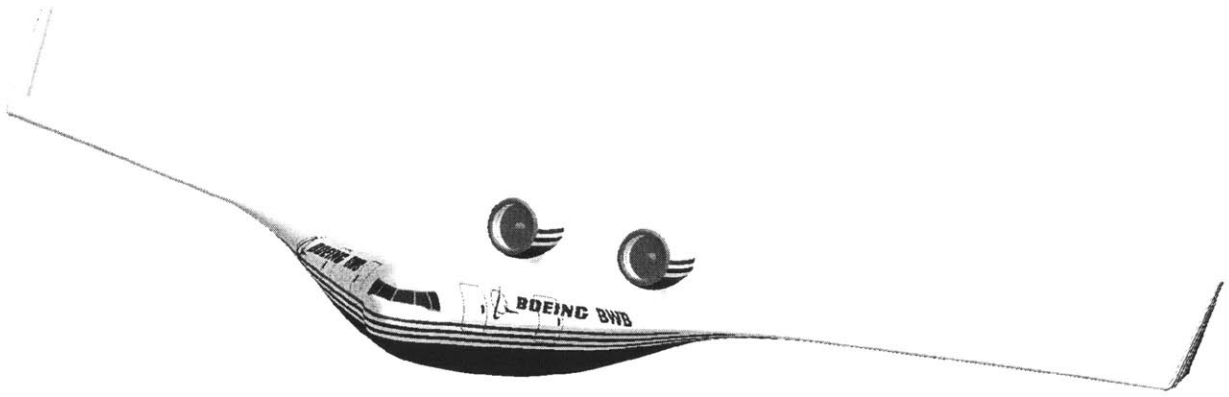

Figure 2.4: The Blended Wing Body Aircraft

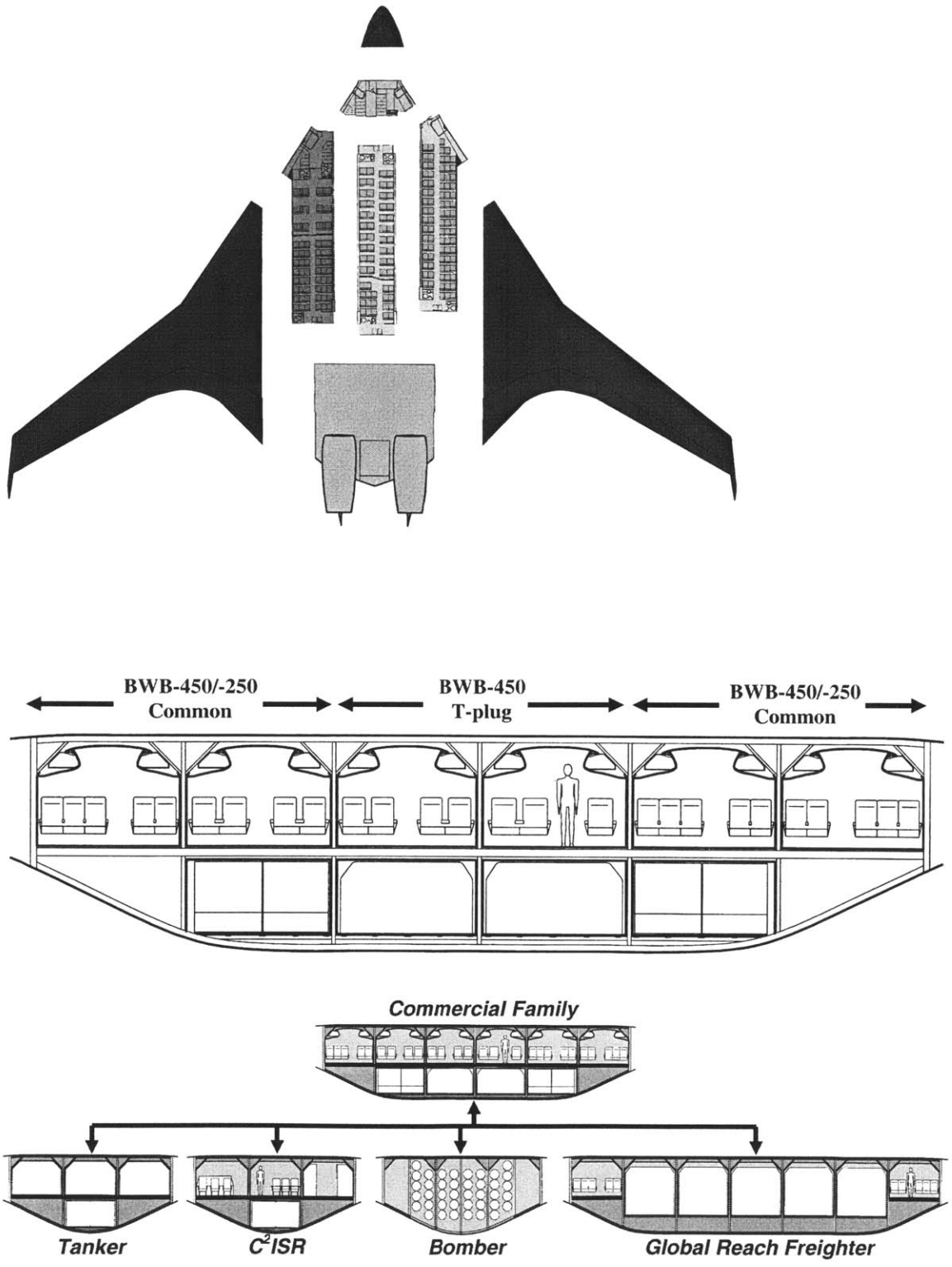

Figure 2.5: Different configurations of a BWB Aircraft

2.8 Other examples

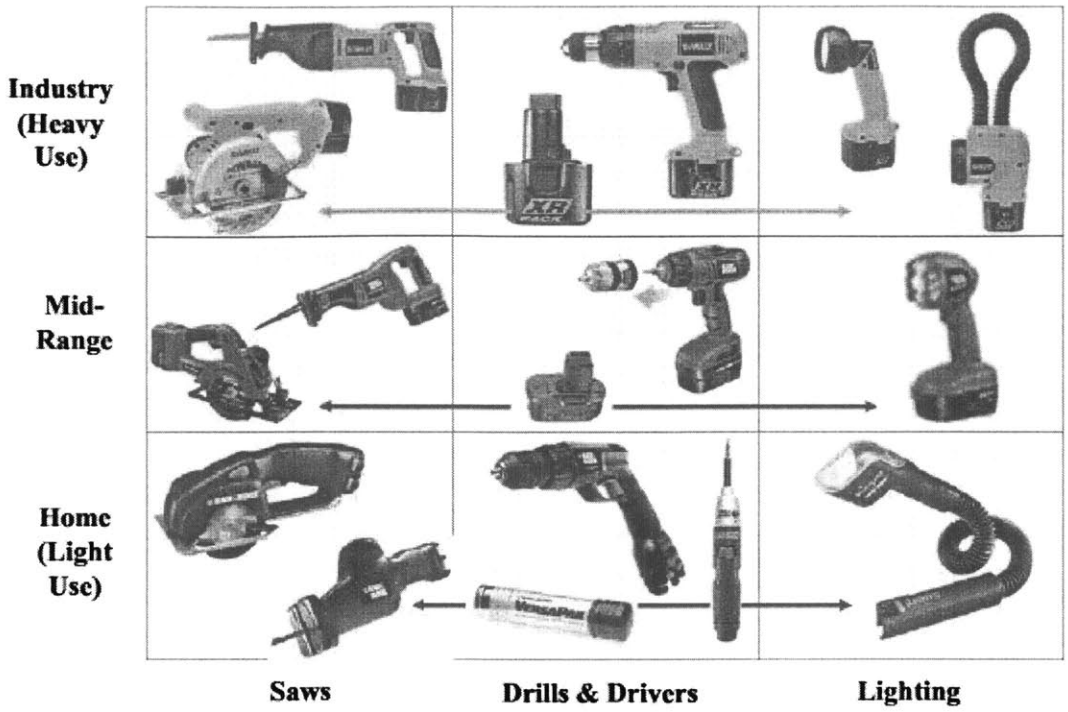

Figure 2.6: Black and Decker Power Tool Family

Black & Decker, a power tool company, decided to create a common product platform, which consisted of an electric motor module and standard interfaces to the motor module. Product differentiation was achieved by varying the motor lengths for different market segments (requiring different power outputs) and business application ends (handle and power tool). By standardizing their core components and processes, they enjoyed immense success, capturing major market share through rapid product differentiation. Figure 2.6 shows various product families offered by Black & Decker, with corresponding market segments[15].

In the personal portable stereo market, Sony introduced almost 250 models in the 1980s. Sony dominated his market (estimated worth $1 billion) with a market share of 40% primarily due to a product family approach[17].

HP strategy for ink jet printers consisted of developing derivatives from existing product platforms, enhancing those platforms to address new market niches or reduce costs and creating entirely new platforms – all at the same time. The Intel 486 processor was based off the 386 line. It was twice the performance and was fully compatible with the software developed for the 386[17].

2.9 The Wintel Platform

Intel and Microsoft have made their products into platforms which have become a foundation on which other companies build their products and offer their services. Intel and Microsoft are considered platforms leaders in the PC Industry. Both companies have taken steps to not only innovate in their core products, microprocessors for Intel and operating systems for Microsoft, but significantly influenced the PC industry and yet retained their platform leadership positions while doing so.

According to the Gawer and Cusumano study on Platform Leadership[9], in an industry platform a company develops and sells a core product that is

(1) part of system that is itself evolving, and

(2) not valuable itself without complementary products or services.

Intel considers the personal computer running Intel microprocessor and the Windows operating system to be the target platform for its microprocessors and chip sets. Microsoft considers Windows to be a platform technology for applications producers and makers of others kinds of complements. The complementary products have played a role in the success of platforms at Intel and Microsoft.

2.9.1 Platforms at Intel

Intel has continued to design microprocessors in accordance to Moore's Law, where the performance of the microprocessor doubles every 12-18 months due to improvements in design and manufacturing processes resulting in improved scalability, power and efficiency. In Intel's case, end users don't directly buy microprocessors, instead they buy PCs. Although the microprocessor is an essential part of the PC, the user experience with computers is affected by products such as software applications, peripherals and other devices that Intel does not make. For Intel to capture value it creates with the higher performance microprocessors, it needs demand for the extra processing power that the newer microprocessors provide. If there is no demand for this extra computing power, then PC manufacturers, software developers and peripheral manufactures have no incentive to make PCs with the latest Intel technology.

Intel benefits, if firms that develop complementary products would innovate in a way that took advantage of the improved performance and scaling available in the next version of the microprocessor. So this way, as Intel evolves its microprocessor, the complimentors evolves their products taking advantage of the latest Intel technology and as a result

consumers have new applications that ultimately require extra processing power that Intel continues to provide. This approach continues demand for Intel's latest microprocessors every 12-18 months.

The PCI Bus

In 1991, Intel took the lead in the development of the PCI bus and driving industry adoption of this technology. Prior to the PCI bus, the PC architectures was dominated by IBM PC-AT introduced by IBM in 1984. This bus architecture was affecting the overall performance of the PC. The speed of the ISA bus was very slow, and as a result end users could not experience the benefit of the performance of the Intel microprocessors because the bus interface to the microprocessor was slow. To provide end users with a better computing experience and increase the performance of the PC system, Intel proposed replacing the PC-AT architecture with the PCI bus. Thus using a higher performing internal bus, customers could now experience a performance improvement when they bought PCs with the latest microprocessors from Intel.

The USP and AGP

Intel's PCI bus initiative was an effort of transform the internal architecture of PC. The PCI architecture became an industry standard and was a major development in the evolution of the PC. With a higher performance bus, end users could now take advantage of the latest Intel technology. After the PCI bus initiative, Intel made other improvements in the evolution of the PC. To continue demands for higher performance microprocessors, Intel continued to innovate to remove performance bottlenecks in the PC architecture.

Intel invented the Universal Serial Bus (USB), which increased the bandwidth between the PC and peripheral devices such as printers, scanners, joysticks and digital cameras. This way Intel could stimulate innovation from third party companies and create business opportunities for them in devices that could connect to the USB interface on the PC. As a result, many companies became complimentors of the platform by adopting the USB interface. This indirectly created demand for newer Intel microprocessors. By having third parties develop products for the USB, Intel increased the valued of the PC platform and it became the platform of choice for many applications.

Similarly, Intel took a lead role in improving the graphics capabilities of PC. The Accelerated Graphics Port (AGP) increased the data transfer rate between the microprocessor and the graphics cards.

2.9.2 Platforms at Microsoft

Microsoft's core product is the Windows operating system. A computer operating system by itself has little or no value. This product derives most of its value for its capacity to function as a platform for other products. As a platform, the demand for PC operating systems is influenced by network effects. Easy file sharing, widespread familiarity with the same GUI and compatibility of software applications across computers are benefits from the use of a common platform.

The availability of a large number of complementary Windows-compatible software encourages potential customers to buy Windows-based computers. Windows is a

platform which runs a variety of application software. In the case of Microsoft, Windows by itself had little or not value without applications. Because it has the resources, Microsoft developed it's own complements – applications such as Word, Excel, Outlook, Exchange, Internet Explorer, and as a result ensured that a new generation of its platform would be successful. Although, most of the Windows operating system consisted of proprietary technology, the company made available interface specifications (APIs) to complimentors such as PC hardware and peripheral manufacturers, software applications companies, consumer electronics and telecommunications companies so they could develop software for the Windows operating system.

Over time, Microsoft evolved its operating system which became a foundation for applications developed by Microsoft as well as for applications developed by third parties for the Windows operating system. It also provided backwards compatibility so its operating systems could run applications written for pervious versions.

The .NET Platform

Microsoft evolved its Windows software platform where computing and communication would converge, by allowing capabilities such as Internet hosting and browsing. This Microsoft platform included everything a business needs to develop and deploy a Web service-connected IT architecture: servers to host Web services, development tools to create them, applications to use them. According to Microsoft[18], .NET is the Microsoft Web services strategy to connect information, people, systems, and devices through software. Integrated across the Microsoft platform, .NET technology provides the ability

to quickly build, deploy, manage, and use connected, security-enhanced solutions with Web services. .NET-connected solutions enable businesses to integrate their systems more rapidly and in a more agile manner, by using the Windows operating system.

Support for Software Developers

In addition to providing a software platform and applications for end users, Microsoft also made available key enabling technologies such as Object Linking and Embedding (OLE) to software developers. Microsoft provided the software industry with a platform for applications and useful development support tools. Additionally, thousands of companies have effectively used Microsoft technologies for years. Microsoft simulated a huge complements business as well and fed this market directly.

2.10 Benefits of a Platform Strategy

There are many examples in industry that show the benefit of platform strategy[17]. Fuji introduced the Quick Snap single use camera in the US market in 1987. In the coming years the market in this field was expected to grow by over 50% per year. A year later when Kodak introduced its first model Fuji had already developed a second model. Yet by 1994 Kodak captured 70% of US market back from Fuji[19]. Between 1989 and 1990, Kodak redesigned its base model and introduced three more models, all sharing common components and common production process steps. Because Kodak shared components among four products it was able to develop products faster and cheaper. These models appealed to different customers in the market and resulted in increased market share.

Platform strategies are important for reducing development costs, increasing market share and the ability to survive longer in a competitive market.

CHAPTER 3: NETWORK EQUIPMENT PRODUCTS

This chapter provides some background information into telecommunications networks and a brief overview of router products. Although communication capabilities are provided by other networking devices such as switches, security and VPN devices, application gateways and telephony equipment, the focus in this chapter is on routers and routing technologies; however the technology trends in the industry are applicable to all network equipment devices.

A **router** is a computer networking device that forwards data packet across a network toward their destinations. A router acts as a junction between two networks to transfer data packets among them. Figure 3.1 shows a block diagram of a router.

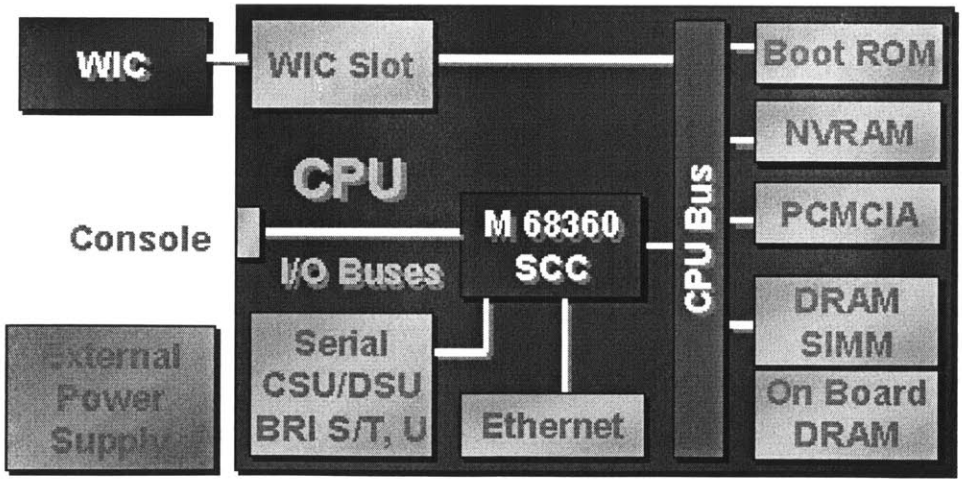

Figure 3.1: Hardware Architecture of a Cisco 1600 Router

In the above diagram, the Motorola 68360 Serial Communications Controller is used as a CPU. The router architecture consists of memory, WAN ports, serial ports and Ethernet

interfaces. Each Router consists of a CPU; DRAM for storage of the routing and forwarding tables and other processes; a compact flash disk (PCMCIA) for primary storage of software images, configuration files, and microcode; a hard disk for secondary storage; a PC card slot for storage of software upgrades; and interfaces for out-of-band management access.

3.1 The Communications Model

A communication network as shown in Figure 3.2 can be viewed as a 7-layer OSI model, developed by the International Standards Organization. Routers operate at Layer 2 and Layer 3 of this model. Using this model we can view the communication task in terms of a column of layers, each of which contains protocols. By using a layering approach, the communication functions are partitioned into a vertical set of layers. Each layer performs a related subset of functions required for communication with another device. It relies on the next lower layer to perform more primitive functions and to conceal the details of those functions. Additionally, it provides services to the next higher layer.

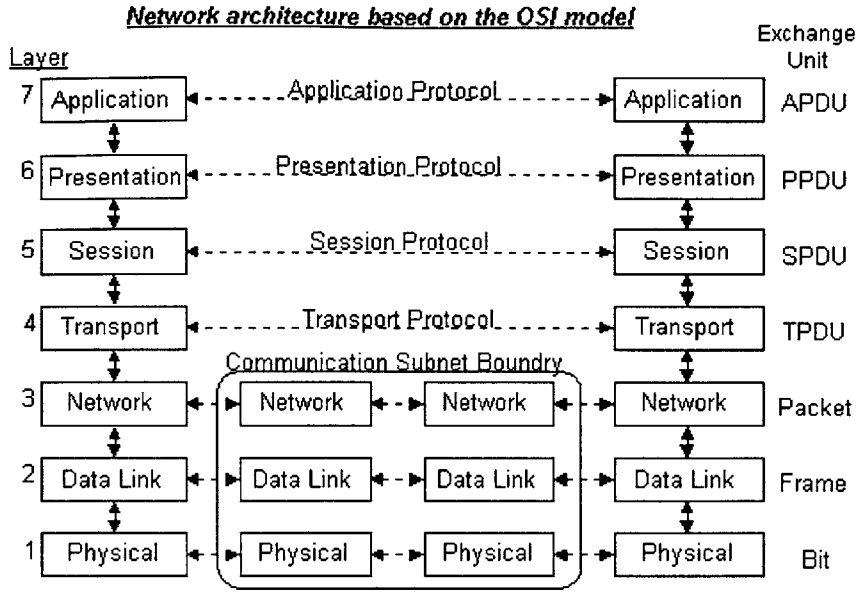

Figure 3.2: The 7-Layer OSI Model

3.2 Software Architecture for routers

A router is a device that acts a "traffic cop" on the Internet and determines the next network point to which a packet should be forwarded towards its destination. The device is connected to at least two networks and decides which way to send each information packet based on its current understanding of the state of the networks it is connected to. A router is located at the gateway where it directs the flow and determines the route of packets as they travel from one network to another network.

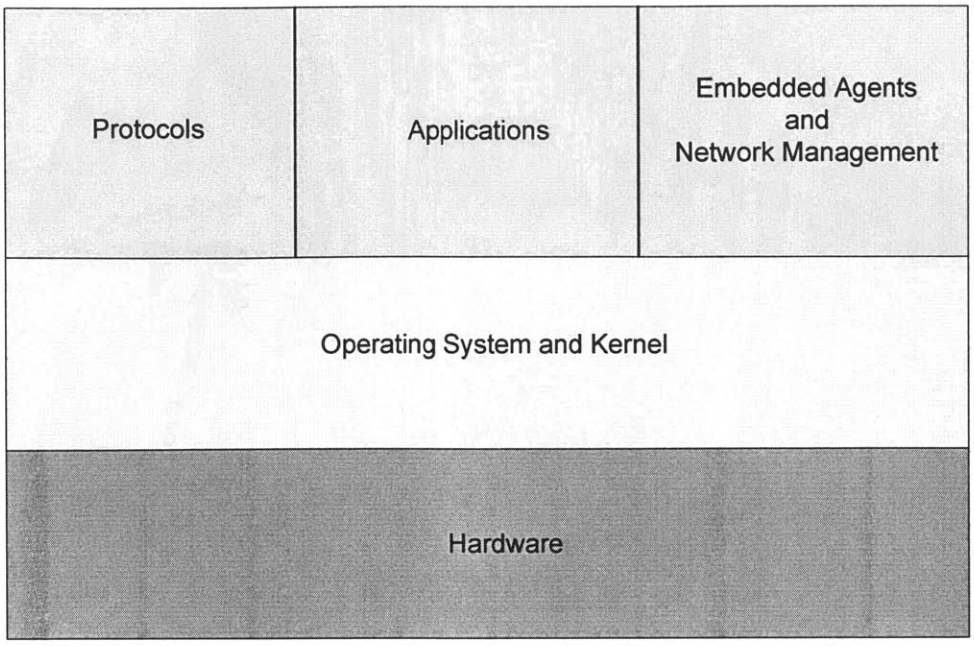

Figure 3.3: Software Architecture of a Router

Networks of computerized switching equipment, called packet switched networks, route the packets. Packets may take separate paths to their destination and may share the paths with packets from other users. At the destination, the packets are reassembled, and the transmission is complete. Because packet switching considers alternate routes, and allows multiple transmissions to share the same route, it results in a more efficient use of telecommunications capacity as packets are routed along less congested routes.

3.2.1 Legacy Router Architectures

The software architecture of a router plays a significant role in performance, reliability and scalability of the device. Legacy routers architectures depended on a CPU for software based real-time packed forwarding. The CPU was responsible for best route

calculation, building the routing table, network management and user interface. Routers performance could be increased by using multiprocessor architectures, but the software running on a CPU was ultimately responsible for disassembling packets, examining the packet header and contents, and forwarding the packet based on a best available route at that time based on the current state of the network.

3.2.2 Modern Router Architectures

In the mid- to late 1990s, the wide-scale commercial acceptance and deployment of the Internet dramatically changed the router requirements. Consequently, the legacy architectures were unable to meet the bandwidth and scalability need of their customers. Modern router architectures separate the control functionality from the packet forwarding functionality for the rest of the system. The separation of the two functions provides additional performance improvement in terms of packet throughput.

The Control Plane

The function of the Control Plane software is to construct and maintain the routing and forwarding tables that the Data Plane hardware uses to forward the actual data packets along the appropriate paths. According to Nexthop Technologies[20], the control place is defined as the intelligence behind the network as shown in Figure 3.4. Whenever changes occur in network topology or rechability, the control plane software uses IETF-prescribed protocols (BGP, OSPF, RIP) to communicate this information with its neighboring peer routers, and updates the local forwarding table appropriately. It maintains the routing tables and manages the routing protocols used on the router.

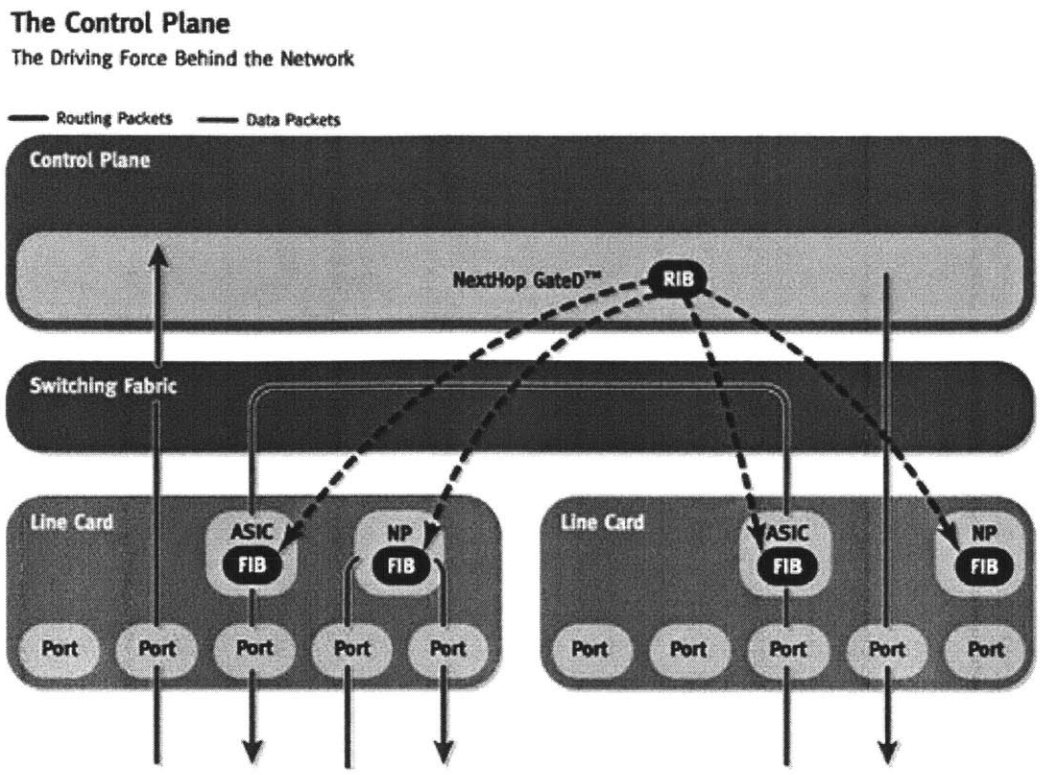

Figure 3.4: The Control Plane Architecture

(Source: Nexthop Technologies)

The Packet Forwarding Plane

The Forwarding Place is a multi-component system that uses application-specific integrated circuits (ASICs) to perform packet switching, route lookups, and packet forwarding. The majority of the traffic entering the router is processed by the Forwarding place. Control packets, or packets that do not have a destination network in the packet are sent to the Control Plane for protocol processing. If there is a change in network information, a new route table is downloaded to the ASICS in the forwarding plane.

The Routing Engine

The Routing Engine is responsible for executing the routing process for the entire system. Routing Protocols, interface management, chassis management, SNMP management, system security and the user interface processes all interact with the operating system as subsystems.

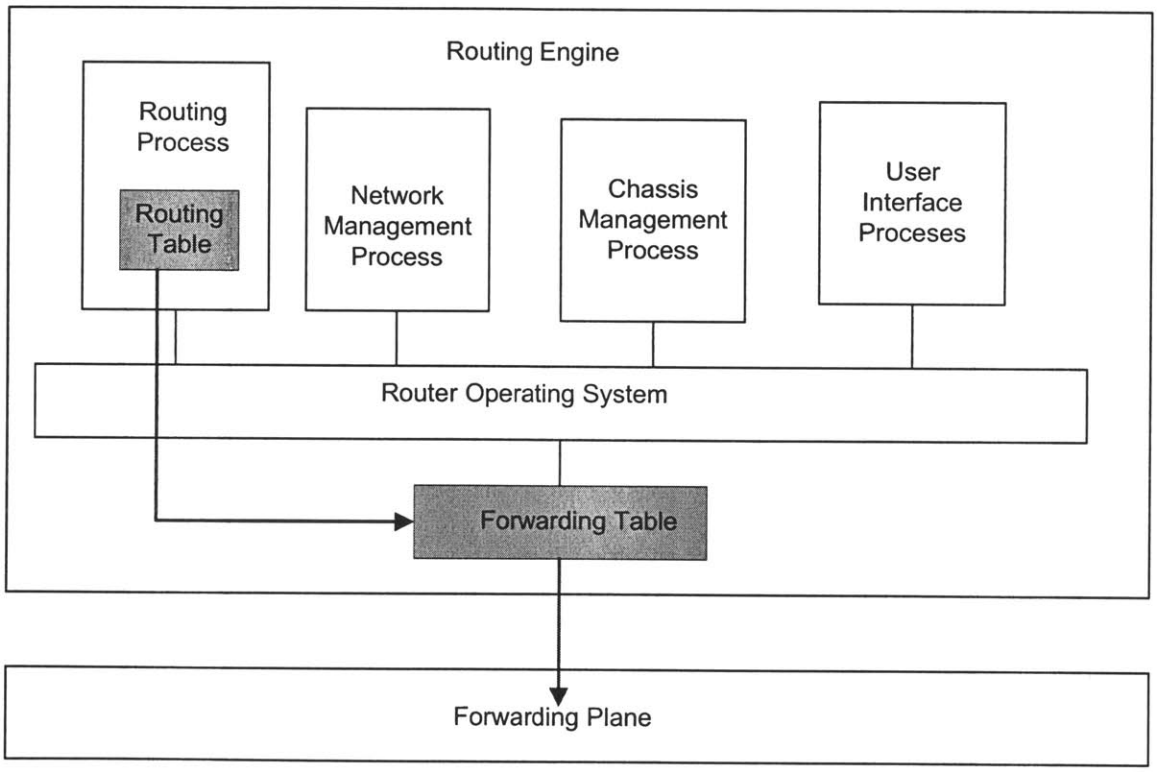

Figure 3.5: Architecture of a Routing Engine

The routing table contains the routes learned from neighbors and through static configuration. The forwarding table is derived from the routing table. The Packet Forwarding Engine uses the contents of the forwarding table, to make its ultimate forwarding decision, based on the destination address of in incoming packet.

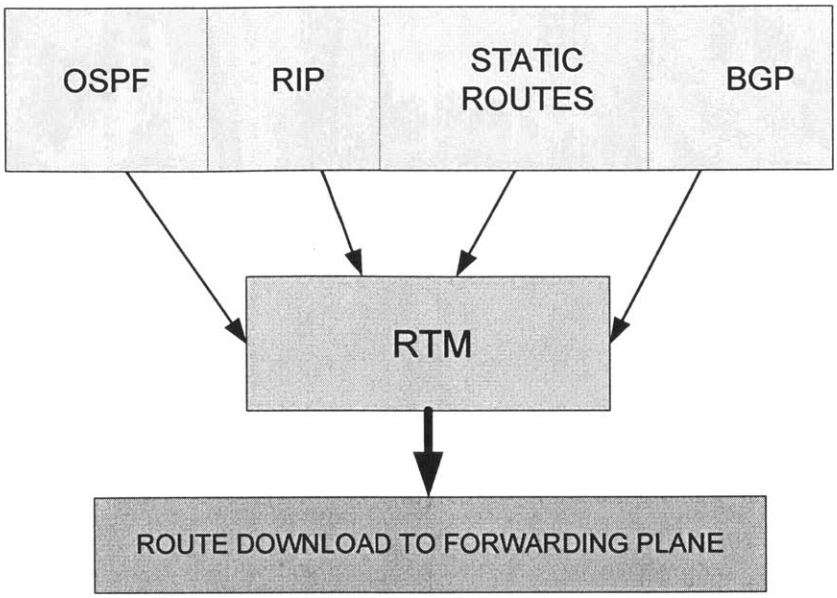

Figure 3.6: Best Route Selection

3.3 The Converged Network

Networks have dramatically evolved over the last decade. The Internet provides transport for data and it has become a backbone where information can be exchanged. With the growth of the Internet, increasing of volumes data flow across networks. For example, the AT&T network alone is carrying 5.2 petabytes of traffic on an average business day to nearly every continent and country, with up to 99.999% availability [21]. Today's information economy is heavily dependent on a telecommunications infrastructure. The telecommunication industry has transformed the way people work, learn and play. This industry is now at the forefront of the information economy. In the past, separate networks were needed for voice and data transmission, but now 'triple play' voice, video and data services can be achieved on a single unified network at ever increasing speeds.

VoIP (Voice over IP) is a technology that uses the Internet or a data network to transmit phone calls. VoIP converts the voice signals from a telephone into digital signals that travel over a packet switched data network compared to the circuit switched network used by a conventional phone. With this technology, the phone conversation gets broken into packets which are transmitted over a high-speed Internet connection. Cable companies are using the technology to offer phone services without building a conventional phone network. All of the major sectors of the telecommunications industry are migrating to VoIP[6].

3.4 The U.S. telecommunications market

Emerging technology is disrupting industry dynamics. Newer, cheaper and more flexible technology is providing alternative means of voice communication compared to the traditional circuit-switched telephone network. Increasing levels of digital content, combined with the adoption of the Internet Protocol (IP) for managing communications traffic has broken down boundaries between voice and data networks. No longer are separate networks needed for voice and data. Both can be carried over the same lines that deliver broadband internet access and related content services. Services will become more software driven, convergent and network independent, relying on the ability to switch networks based on the most efficient option at the time[22].

Changes in technology and regulation now allow cable television providers to compete directly with telephone companies. An important change has been the rapid increase in

two-way communications capacity. Conventional cable television service provided only one way transmission services from the cable operator to the home. Due to capacity limitations and signal interference, two way communication was not possible. However, as cable operators use new technologies to reduce signal interference and improve data compression, and increase capacity by installing fiber optic cables, operators can now offer two-way services such as high speed internet access, video-on-demand, IPTV and telephony using VoIP[6].

3.5 The Telecom Equipment Manufacturers

The telecom industry is going through a transition. Equipment maker are facing threats from Chinese firms like Hwawei and Harbor Networks. These suppliers are beginning to enter the truf dominated by Cisco, Lucent, Nortel and Alcatel with cheaper products.

Telecom equipment manufacturers make technology that is largely invisible to consumers but underlies many of the services they use such as telecommunication networks transmitting voice, video and Internet traffic[23]. The telecommunications industry in the United States is going through a consolidation, which is still struggling to recover from the collapse that began in 2001. Deals worth $200 billion have been announced in the past two years. Most recently AT&T announced plans to buy Bell South. Other transactions have included Verizon's acquisition of MCI, and the earlier purchase of AT&T by the former SBC Communications, which took the AT&T name.

Equipment makers are following the wave of mergers among telecom carriers that has marked the telecommunications industry in recent years. In April 2006, Lucent Technologies and Alcatel announced plans for a $33 billion merger. Consolidation pressure has been growing in the equipment business amid a wave of telecom deals and the emergence of more efficient technologies. That has left limited the amount of money that big phone companies have to spend on equipment. The result has been too many equipment makers chasing a limited amount of business[24]. Also, a flood of new technologies and new competitors have driven down the prices of traditional phone service while introducing a host of new services that are dramatically changing the way consumers and business communicate. While phone service over the Internet is taking off, the cable industry is entering the traditional residential landline business of telephone companies like Verizon and AT&T. In turn, phone companies are upgrading their networks and starting to offer television service.

These changes are creating opportunities for some equipment makers, but the also pose challenges because improved efficiency enable carriers to replace huge switches and other pieces of equipment with computers and software.

CHAPTER 4: THE SOFTWARE PLATFORM

Initially, "platform" simply referred to the computing hardware that executed software. It was later generalized to include software-based platforms such as operating systems. Currently, the term implies any set of hardware or software mechanisms that enable execution of software applications. This means providing external mechanisms or services used by one more software applications on a given system. A software platform provides a collection of capabilities to the system. This includes the traditional services provided by operating systems such as task scheduling, inter-process communication, file systems, memory management, input/output and multitasking. Additional capabilities include a configuration management system, user interface (CLI or GUI), system logs, statistics collection, redundancy, etc. These services are made accessible to system applications by Application Program Interfaces (APIs)[11].

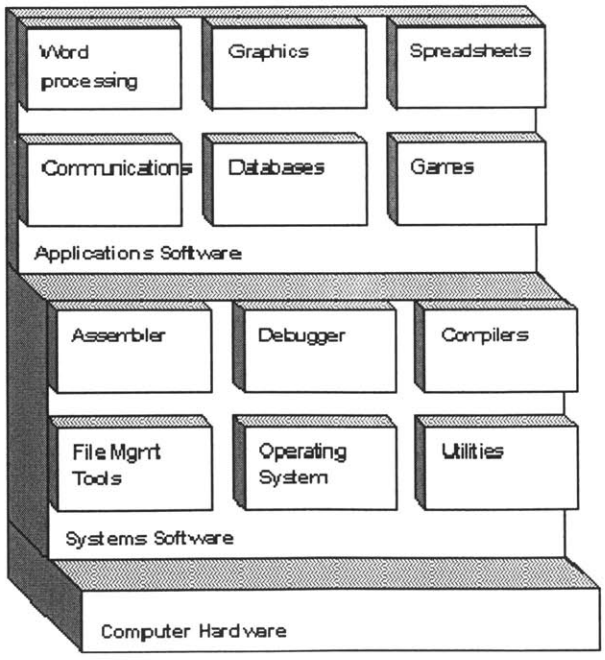

Figure 4.1: A Software Platform

Here the choice of operating system, architecture of the system components, and robustness of these features can have a direct outcome on the characteristics of the overall system. The performance, reliability and scalability are directly influenced by the architecture of the software platform. Therefore it is crucial that system architects design a platform that can operate in the desired environments.

4.1 Operating Systems and Platforms

Every computer requires a central processing unit (CPU) and an operating system (OS). The CPU consists of one more microprocessors which perform program execution as well as arithmetic and logic functions. The OS manages the CPU and other hardware such as keyboard, monitor, storage media and communication devices. Hardware management functions are often combined with an interface between the user and the computer. This interface allows the user to access and manipulate files, run programs and operate the hardware, either directly or through instructions generated by application software. In turn, the hardware management and user interface functions are often combined with a software platform into a single operating system product[25].

A software platform contains Applications Programming Interfaces (APIs) that specify how a software developer can use system functions or access useful modules of code built into the platform. By using these APIs and underlying code modules for developing new application software, the software developer can reuse exiting functions without having to rewrite new code. Essentially, the application software calls on the processing functions built into the platform product, which reduces the need for application

developers to write code that performs routine functions. Microsoft Windows, for example, contains thousand of APIs that can be accessed by software applications and that are relied upon by software developers. In this way, a software platform supports the development and operation of software applications.

4.2 Embedded systems software

An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, designed to perform a dedicated function. Embedded systems can be stand alone devices such as mobile telephones, PDAs, medical devices or digital cameras. In some cases, embedded systems are part of a larger system or product, as in the case of an antilock braking system in a car, an ATM machine or avionics for aircraft and missiles.

The software for the embedded system drives the microcontroller or processor and associated hardware components to provide the required system functionality.

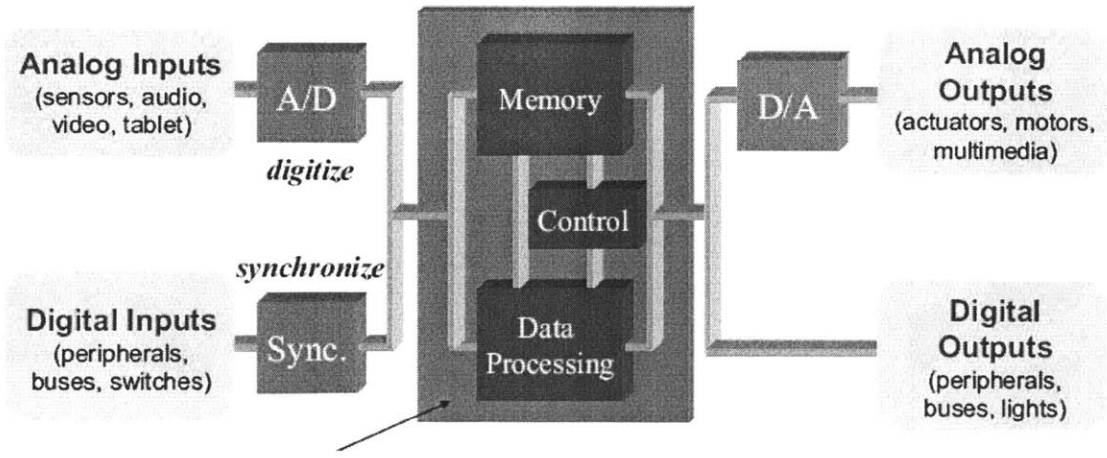

Figure 4.2: Architecture of an Embedded Digital System

4.3 The State of Embedded SW Development

Due to advances in information and communication technology, the market for embedded products is expected to continue to grow. Also the functionality provided by software in embedded systems will rise. The growing complexity of the systems will create challenges for software developers. To addresses these challenge, organizations need a software development environment that allows delivery of innovative products with quality in a timely manner.

A wide variety of companies in different industries have reported a dramatic shift in the relative engineering efforts devoted to hardware and software. Where 10 to 15 years ago the ratio was 70% hardware to 30% software, it is now typically reversed, and the software fraction is continuing to grow. Embedded systems products for the consumer market segment typically have short product life cycle time due to the changing buyer behavior, appetite for newer technologies and the introduction of new products from competitors. Therefore, embedded system providers have to keep on developing new products based on new hardware components as well as software to address user requirements for functionality, user interfaces, and customized products. In order to continuously fulfill customer demands for new products the embedded systems industry needs a robust product architecture and a software development environment where quality products can be delivered to customers.

4.4 Product Development Strategies

In designing networking systems, engineers can choose from variety of components. The choice primarily depends on the market segment, functionality and cost of the device. For example, the WAN ports can be Frame Relay, ATM, DSL, or T1/E1. The applications are networking protocols which include IP, BGP, TCP, IPSec, SSL, SNMP, Firewalls, etc. The operating systems can be Linux, QNX, Windows, VxWorks or even a proprietary operating system.

Figure 4.3 shows three different products in the same product family of routers. Although these are products provide similar functionality, they don't have a lot of common components even though they run the same set of applications. They may have resulted from having separate development groups for each product or became part of an organization's portfolio though acquisitions.

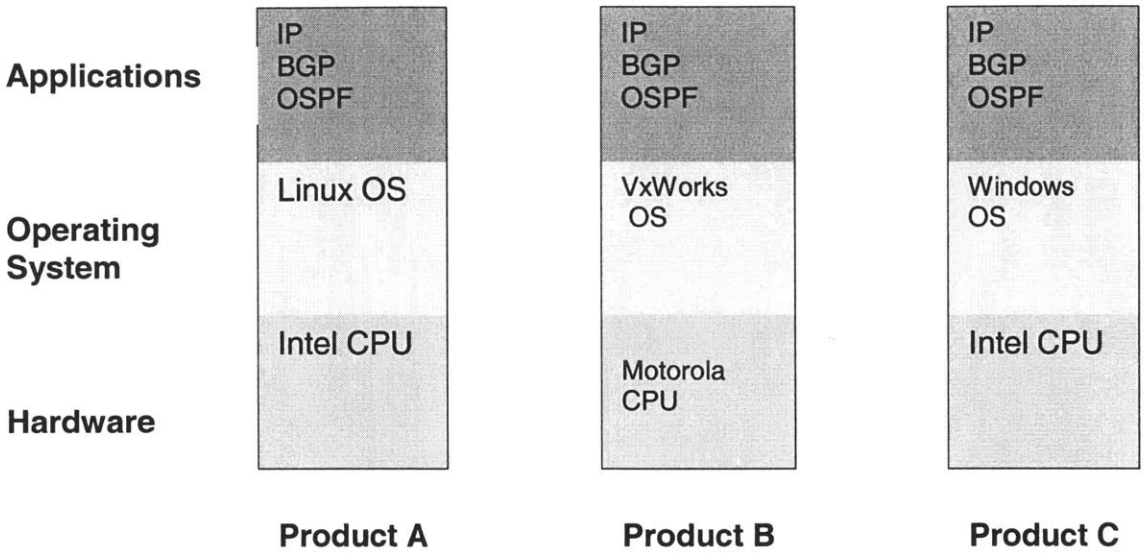

Figure 4.3: Products within a family

In the arrangement shown above, although the applications provide a similar functionality to the end user, the components used to develop these products are quite different.

Despite the lack of commonality in Operating Systems and hardware it does offer some advantages. A particular type of operating system might be well suited where high reliability is required or where certain applications have specific performance requirements. Cost also might be a factor. Product A could be high end product for use by large Enterprises and ISPs, where as product B could be a low end product used by consumers of the Small Office/Home office market.. This approach, allows the Original

Equipment Manufacturers (OEMs) to offer a very tailored product for particular market or an application.

Although the above approach has its advantages, it may not be the most efficient in terms of development efforts.

1) First in terms of application development, the application software for Product A might have to be written separately from Product B, because the APIs for each operating system are different. If product C gets developed later in time, after A and B have been developed, their application software cannot be ported to product C, because they are running on a different Operating Systems. Here there is little reuse of software.

2) Separate efforts are required for product validation. Even though the applications for Products A, B and C might perform the same function, each product has to get validated separately.

3) During the maintenance phase, defects have to be fixed separately for each product. Solutions for defects on one product do not automatically address the same defects on another product.

4) Separate development processes are required for each product, thereby increasing the cost overhead.

For organizations looking for R&D efficiency, the above approach is not optimal. To increase efficiency, reduce time to market, and increase quality of the entire product line a common software platform can provide tremendous benefits.

4.5 A Common Software Platform

As embedded system continue to proliferate, the software running on these products will have customers in every vertical market and will be required to serve multiple application purposes. Companies that develop these complex products will be challenged with rising complexity, shorter market windows, tighter quality control and evolving standards. The key benefit of a common software platform results from the ability to develop a component once and share the implementation of components in several products. This results in overall lower development and maintenance costs.

To implement a common software platform approach, a common Operating System is needed. To allow the existence of separate hardware platforms within a product family, a Hardware Abstraction Layer (HAL) exists between the hardware and the operating System. The HAL functions as an interface between a system's hardware and software, providing a consistent hardware platform on which to run applications. When a HAL is employed, applications do not access hardware directly but access the abstraction layer provided by the HAL. The HAL is similar to a software API, which allows applications to be device-independent because they abstract information from such systems as caches, I/O buses and interrupts, and use this data to give the software a way to interact with the specific requirements of the hardware on which it is running.

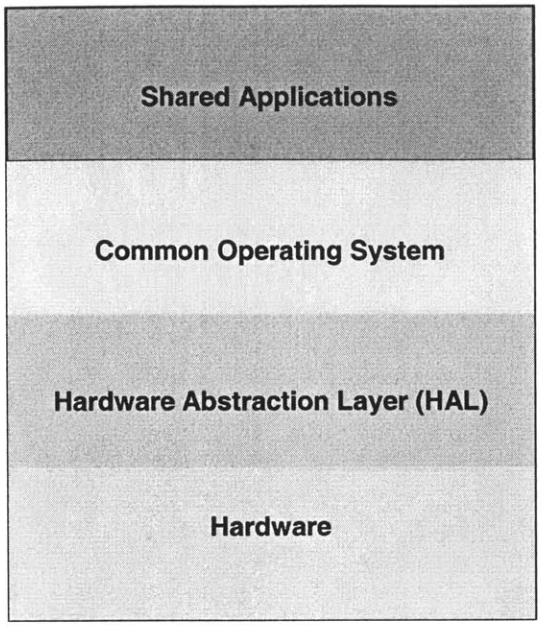

Figure 4.4: A common Software Platform architecture

The HAL layer takes into account different CPUs, different device drivers and different hardware architectures. From this common software platform, a number of derivative products can be created, all sharing the same applications.

This biggest advantage with this approach is that it permits software applications to be shared across different products, even though the products use different hardware components. With this there is not only reuse of software, but also reuse of product validation and business process that support the software development activities. This increases productivity of the team. Results from Bellcore suggest a reuse of 1 million lines of code and gains in productivity[26].

Since we have reduced the number of operating systems in this product family, the support costs for maintaining it are lowered. With this approach, a product development organization can focus on its core competency, of designing and developing applications instead of writing and maintaining operating systems. For most applications, a number of commercially available Real Time operating systems are available, and it makes little sense to write operating systems from scratch.

This approach allows different product groups to "own" a particular application technology instead of having multiple groups work on the same application for different platforms. For example, one product group could specialize in the development of firewall software, another group could specialize in encryption technology. In this approach, applications created for one product are readily available for use on the other products without any re-writing of software. This encourages innovation within an application domain.

If the APIs are similar, product development organizations can further accelerate product delivery by using Commercial off-the-shelf (COTS) software. By using common components, development activity can be standardized across the enterprise.

4.6 Examples of Common Software Platforms

4.6.1 Examples in Nokia Mobile phones

Nokia develops and maintains several advanced software platforms that enable different user interfaces and displays, product concepts, and feature configurations[27]. The *Series 30* is the lowest-cost platform, designed for entry-level mobile phones that feature voice and basic messaging functionalities. The *Series 40* is a versatile, efficient and highly cost-effective feature phone platform, particularly suited for the mobile phone product range. The *S60 platform* is a smartphone software platform and is licensed to OEMs. The *Series 80* is a high-end software platform optimized for enterprise Communicators and smartphones, enabling a two-hand operated QWERTY keyboard and a wide screen

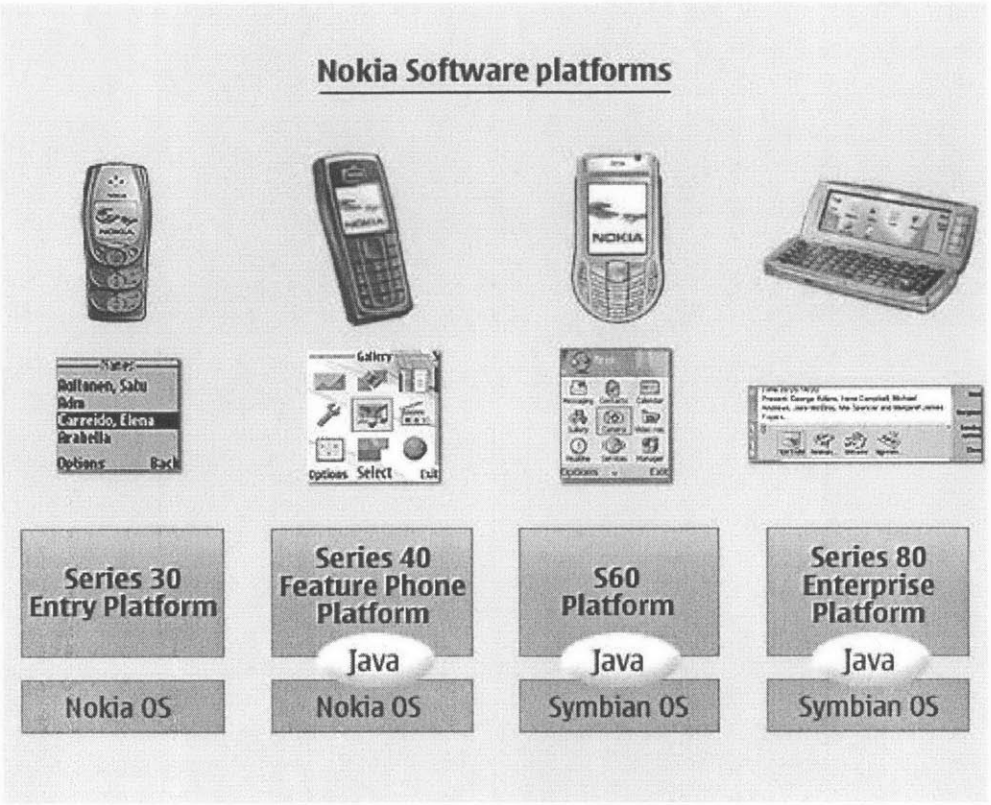

Figure 4.5: Nokia Software Platforms

This Nokia software platform provides a programming framework for software developers. It hides the device complexity from applications and ensures ease of programming, so software written for one telephone can be used by all telephones that share the same common software platform. This leads to a growing number of applications and services for mobile telephone users.

4.6.2 Cisco IOS

Cisco IOS (Internetwork Operating System) is the software used on the vast majority of Cisco Systems routers and some network switches. IOS is a package of routing, switching, internetworking and telecommunications functions tightly integrated with a multitasking operating system. IOS evolved from being a software code base for router products to a software platform eventually linking different technologies such as ATM, Frame Relay and MPLS switches, WAN aggregation, telephony and broadband solutions, DSL and wireless equipment. Cisco acquired many of these technologies through its numerous acquisitions and incorporated them into IOS so it could offer products with these newer technologies to its customers.

IOS is a single code base and a common software platform that provides software for all of Cisco's different hardware equipment, which includes enterprise, network core, aggregation/edge and branch office routers. This includes routers from the Cisco 800 to the Cisco 12000 as shown in Figure 4.6[28]. Cisco's website describes the IOS software as[29], "The world's leading network infrastructure software, delivering a seamless integration of technology innovation, business-critical services, and hardware platform support." Because the deployment of IOS was so prevalent, vendors of networking equipment had to license IOS from Cisco and test their software to ensure it interoperated with Cisco's products. Cisco worked with industry groups and standards bodies to set new networking standards and would often be the first company to include these newly defined protocols to IOS. In some cases Cisco would develop proprietary extensions to a protocol based on customer requests and eventually push for them to become standards.

Cisco's IOS and its Command Line Interface (CLI) became a de facto industry standard. As a result, Cisco's customers were reluctant to remove Cisco gear from their networks and in some cases demanded other equipment manufacturers to use a Cisco like CLI for configuring and monitoring their network equipment products.

Cisco was a leader in evolving the networking industry, by making the router a platform for software-based communication standards that interoperated with other routers and communications equipment. It broadened the set of technologies that fit into the category of a router platform, such as VPNs and security, ATM, broadband/cable, DSL, MPLS, QOS, ethernet switching, voice, remote access and storage technologies. Cisco used IOS as the "glue" to make these disparate technologies work together[30].

Although IOS appears as a powerful common software platform for Cisco equipment, its origins go back to Cisco's early days where it began as an abstract concept to describe the various network services and routing protocols that Cisco's source code handled[31]. Later it included technologies from acquisitions. For many years IOS was more of a marketing tool than an actual integrated software system. IOS was also modified to run on products that came from acquisitions. For example, the company added IOS code to StrataCom switches so that is could work with the existing Cisco product line. It wasn't until 1997 that that IOS became an actual integrated system and a common software platform for Cisco's product lines, not just routers.

IOS has somewhat of a modular and layered architecture, but was not a top-down-designed operating system like Windows NT/2000 and UNIX[32]. IOS contains proprietary software code and is not an open system in the sense that Linux is an open system where the source code is available for others to modify. However, IOS is built according to open standards whose specifications define the interfaces for connecting with other communications equipment. Integrating technologies into IOS was not easy also. The company had to go through great pains to develop quality software for the StrataCom products and fully incorporate it into IOS with troubleshooting, management and policy-routing features[33].

Unlike Windows, IOS was not a product by itself. It only ran on Cisco developed or supported hardware. IOS also added APIs for Cisco business units to share their internally developed applications and reuse code. Cisco IOS as become extremely large, and now in it's 12th revision, has become complex with so many streams that make it challenging and costly to support released software running on millions of deployed systems.

During the early releases of IOS, Cisco delivered a single software release which provides new features and functionality as well as incremental software fixes for all Cisco products. In recent years, with the Internet boom, Cisco added thousands of new features and applications to wide array of hardware products. Cisco IOS Software diversified from one to multiple release families, each of which supported different feature sets for different customer needs, which Cisco calls "Trains." The E Train for

enterprise customers, the S Train for service provider customers, and the B Train for supporting broadband features.

Having to make available separate software trains shows the challenges in having one common software stream for all Cisco products. In Cisco's case some diversification was necessary to fulfill requirements for certain customers (Carrier and Service Provider) and yet keeping a stable code base for their Enterprise customers. IOS did not evolve overnight. It took a lot of effort to get integrate disparate technologies and to have IOS run on systems of acquired companies. Despite its limitations, it has provided Cisco with a common software platform for all its products.

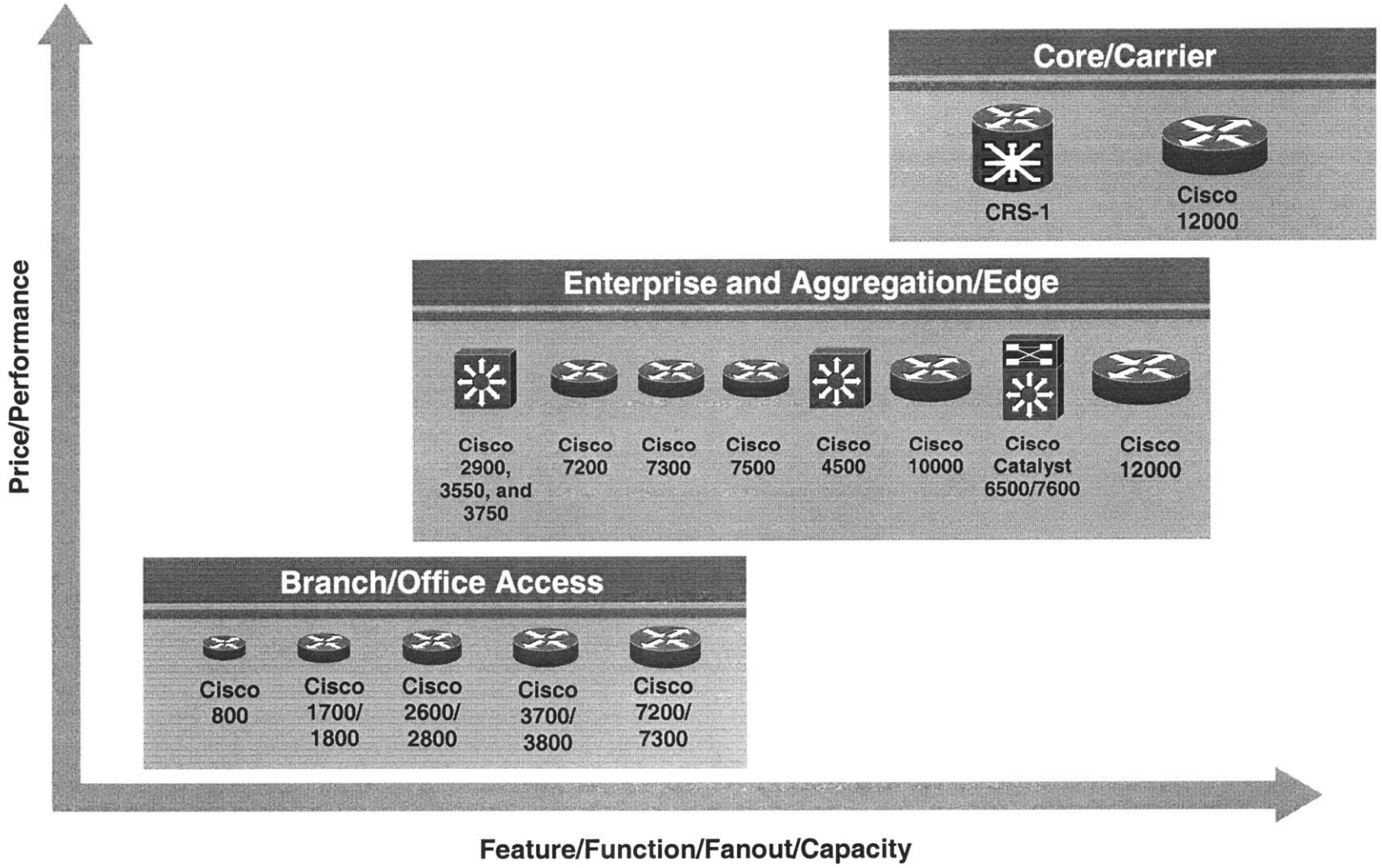

Figure 4.6: Cisco Platforms Using Common Software

4.7 Architecting a common software platform.

A common software platform can be synthesized through a parallel process of the overall system architecture and the business case for the product[34]. The customer needs provide input to the business case which generates product requirements and goals for the architecture. The systems architecture serves as a solution to the business case.

In addition to the customer needs, the competitive environment (other products, barriers to entry, market timing), corporate strategy and distribution channels of how the product will be marketed and sold to customers should be considered in creating the business case. Regulations and standards are external drivers which primarily impact the architecture. Also the system under design will have some connection to legacy products. The extent to which compatibility with legacy systems is required and migratory paths for existing customers to the new system are strong drivers of architecture.

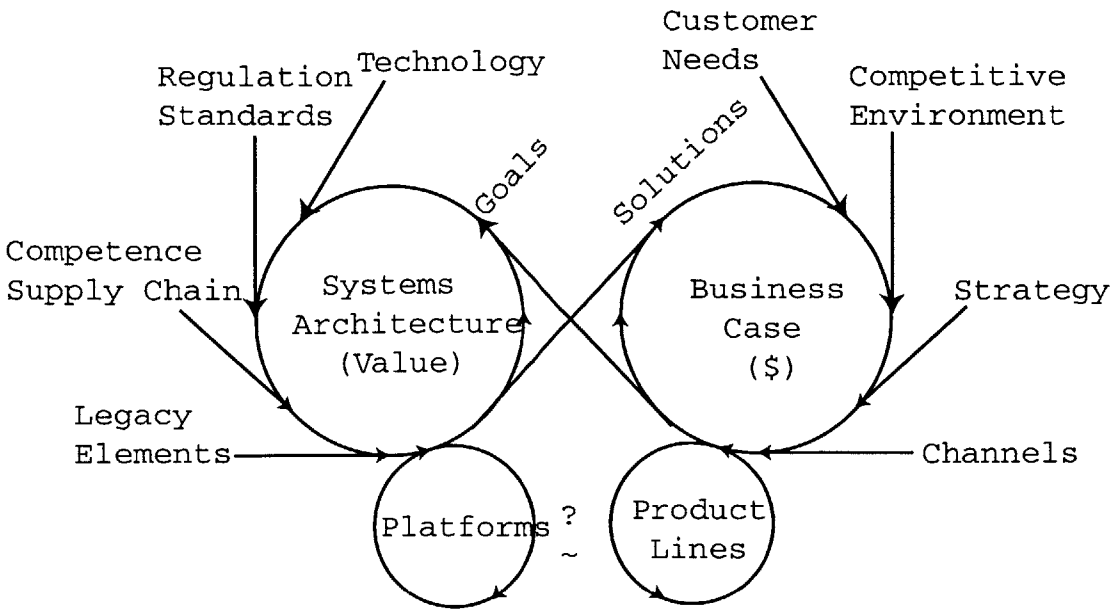

Figure 4.7: Parallel process of architecture and business case

The architecture should be designed to make use of platform elements that are common to all the products within the same product family. Commonality and distinctiveness are two important aspects for a successful product family. *Commonality* leads to a platform shared between variants and *differentiating characteristics* distinguish one model from other. Further, core needs will lead to a common platform and distinctive needs will lead to variety or differentiating modules[17].

The architecture of the system and the functionality of the system allocated to software need to be well understood. This is crucial because software can have a profound influence on the overall system architecture. Well-architected software can rapidly evolve.

Therefore software architectures must be explicitly designed to accommodate future changes in the system. A good software architecture will influence the quality and reliability of the system as well as the system's ability to evolve over time. The software components that will be shared should be identified with the view point of the product family utilizing the common software platform, with APIs available to access sub-systems that are unique to a particular product.

4.8 Strategies for multiple projects

Cusumano and Nobeoka[35] present a framework for project strategies which focuses on how companies can link projects in order to share platform technologies. Their topology categorizes new projects into four types, depending on the extent of new technology or changes in a platform, how projects use the platform technology, and how fast or slow a company transfers a platform from one project to the other.

The four categories they describe are new design, concurrent technology transfer, sequential technology transfer and design modification. Projects that develop platforms primary from scratch are categorized as *new design*. This strategy is most appropriate for incorporating the latest technology or totally new designs into a product without placing many restrictions on the development team. In *concurrent technology transfer*, a new project begins to borrow a platform from a base or preceding project before the base project has completed its design work. Generally, this transfer occurs within two years after the introduction of the original or base platform. In *sequential technology transfer*, a project inherits a platform from a base project that has finished its design work. Here the

second project reuses a platform design that already exists. The reused platform is already relatively old compared to designs transferred while a base model is being developed, as in concurrent technology transfer. Concurrent technology transfer and sequential technology transfer require projects do share a platform with other projects in the firm. In *design modification*, a project replaces an existing product but without creating a new platform from another product line. The modification project inherits or reuses the platform from the predecessor model in the same product line, perhaps with some minor changes.

Concurrent Technology Transfer and Common Software Platforms

According to research by Cusumano and Nobeoka, projects and companies that rely more heavily on concurrent technology perform better in terms of market share and sales growth compared to companies that rely more on the other strategies presented above. In other words, when a new platform is developed and its technology quickly transferred to another project, the company grows faster than its competitors. For embedded systems development using a common software platform, a company can develop a mid-range product, and could shortly thereafter introduce a low-end product with the same software capabilities for a different market segment. Here, the only additional engineering effort would be the development of new hardware and software modifications to the Hardware Abstraction Layer. All the software and applications developed for the mid-range product can be reused on the low-end product. For example, a telecom equipment manufacturer could first develop a mid range router for the enterprise core market and later could reuse the same applications and protocol software (initially developed for the mid range router)

in the branch office router. Concurrent technology transfer allows developers of embedded system products to efficiently add products to their product portfolio and more quickly enter new markets.

CHAPTER 5: PLATFORM STRATEGY FOR THE TELECOMMUNICATIONS INDUSTRY

In Chapter 2, I provided examples of platforms and described how Intel and Microsoft evolved their products to become platform leaders. In this chapter, I will apply the Four Levers framework of Platform Leadership developed by Gawer and Cusumao to devise a product strategy for telecom equipment makers. I will also outline some business processes and organizational structures for developing common software platforms within a product development group.

5.1 The Four Levers Framework

The Four Levers of Platform Leadership are defined below:

1) Scope of the Firm: Is it preferable to create product complements internally or let the market produce them.
2) Product Technology (architecture, interfaces, intellectual property) What degree of modularity is appropriate? Should product interfaces be open or closed? What information should leaders disclose to outside firms?
3) Relationships with external complementors: How can the company balance the competition and collaboration with outside firms.
4) Internal organization structures: What processes and systems will allow the company to manage internal and external conflicts of interest most effectively?

Lever 1: Scope of the Firm

Depending on the product and technology the firm plans to make, equipment makers need to focus on products that can give them a competitive edge. Most of the core routing technologies have become commodities, and since they are standards based protocols, there is limited room for product innovation. Companies such as NextHop Technologies provide protocol software (IP, BGP, OSPF, RIP) that is compliant with standards set by the IETF. It makes no sense for telecom equipment firms to implement these protocols in software from scratch. Instead these firms should use code developed by these third parties and focus their resources on scalability, security and reliability of the systems. OEMs can leverage third party code and modify it based on their specific hardware architectures. Networking source code from third parties now enables OEMs and telecom equipment manufacturers to quickly bring routers to market at significantly lower price points than existing alternatives.

Telecom equipment makers should take a very close look at complimentary technologies. Essentially, it is the complementary technologies, such as VoIP, IPTV, Video on demand, multimedia services and network convergence that ultimately direct traffic to a router or a switch. It is unlikely that router manufacturers can make these technologies or develop the end user applications, but they can work closely with service providers and complimentors to bring these technologies to the end consumers, by working on defining new standards for interoperability and helping create solutions for them.

Routers serve as platforms for building IP-based networks and also serve as aggregation points for WAN access. Additionally, telecom carriers and ISPs use routers for providing traffic engineering capabilities and revenue generating services such as VPNs, bandwidth guarantees, Quality of Services, and long haul transport. Unlike ISPs and telecom carriers, which buy telecom equipment directly from equipment suppliers, enterprise customers look for complete solutions and often go through resellers. The solutions involve core routers, branch routers and switches along with other communication equipment. Also enterprise customers require interoperability between Internet routers and other types of networking and communications devices.

If the equipment manufacturers are missing key technologies and unable to deliver products in time with current R&D resources, they should consider acquisitions or joint ventures, like Cisco to fill gaps in its product portfolio and integrate those technologies within the router platform. In addition to having a vision of what product or platform to develop in the future, they need to focus on developing a platform where they will have an advantage over competitors and where the core technologies will be difficult to imitate.

Lever 2: Product Technology

A platform architecture – the high level design of the system and the interface designs that determines how components or subsystems work together – can have a profound and lasting impact on the structure of an industry and the nature of innovation[36]. For telecom equipment firms it is important to chose modular architectures as opposed to monolithic

architectures. Modularity facilitates the development of components and modular designs can reduce the costs of innovations for outside firms.

Modular Software Design

In networking equipment, a modular design starts with a clear separation of the data plane and the control plane. Not only does this provide performance benefits, but many improvements made in the area of protocol software can easily be propagated to the rest of the system. Also a modular architecture makes it easier to leverage third party code. For example, if compression or encryption technology is needed, it can easily be included in the system. A modular architecture makes it possible to break down the tasks necessary to build components of the platform and a family of compliments. It facilitates the dispatching of tasks to distinct groups. It is an ideal approach for system design in order to take advantage of third party innovation[36].

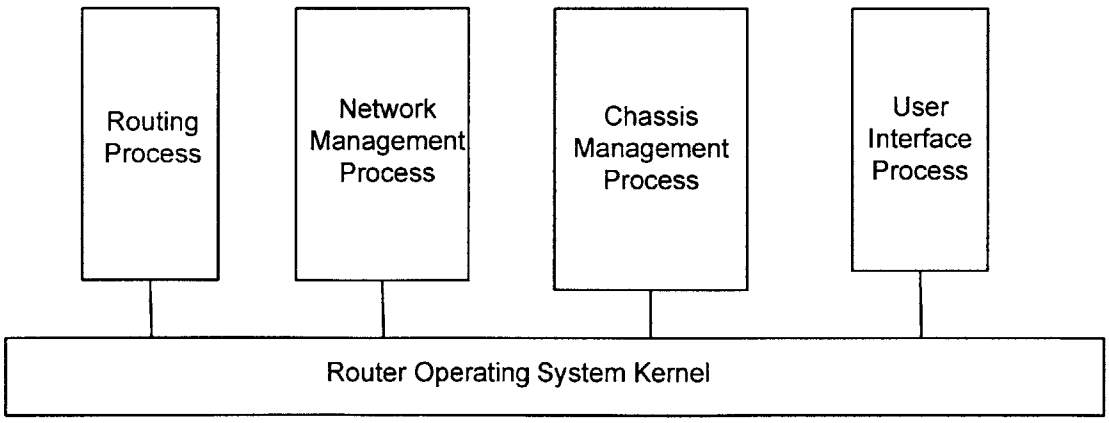

Figure 5.1: A Modular Operating System

A modular operating system allows each individual process to be independently restarted or changed without affecting the operation of the router and protects the entire operating system from crashing due to the failure of a single module. Modularity also allows users to upgrade a specific software process without rebooting the entire system[37].

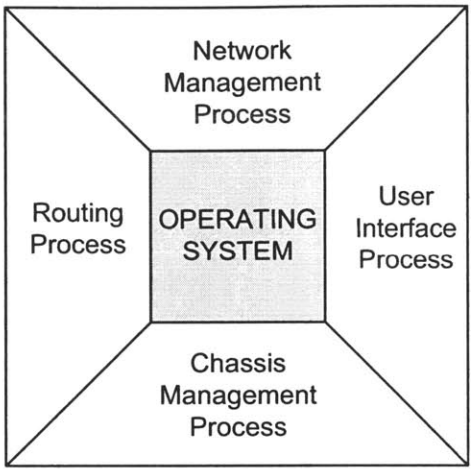

Figure 5.2: Monolithic Operating System

In choosing operating systems for routers, it makes little sense to write an Operating System from scratch, unless the application require certain system services that existing operating systems cannot provide. Over the last decade, there have been numerous improvements in the availability of operating systems. Additionally, if commercially available operating systems such as Linux or VxWorks are used, it simplifies porting of third party applications. Also, this makes it easier for complimentors to develop applications for a standardized operating system as opposed to a proprietary operating system. Many developers are migrating to Linux and open-systems software in embedded systems design. Open standards allow developers to leverage freely available software

and spend their valuable resources on product differentiators. This also permits having open interfaces where third parties can develop complements.

Industry standards

Customers in the telecommunication industry expect products to adhere to standards and interoperate with equipment from other suppliers. Firms in a platform leadership position can propose standards, but that the actual definition and approval is governed by a committee. Since this is a standards based industry, proprietary technologies that give the firm a competitive advantage in term of performance such as ASICs or hardware architecture should be safeguarded. Since implementation and source code of standardized protocols (BGP, OSPF, IP, RIP, IS-IS, etc) are available from third parties, telecom equipment manufacturers should focus on key differentiating attributes such as scalability, performance, security, and robustness of their systems.

Lever 3: Shape relationships with external complementors

Internet routers are products not directly sold to consumers. End users experience the Internet based on their speed of the connection and services it provides. It is important for router developers to collaborate with ISPs and enterprises to ensue that they have the right set of features so ISPs can provides the new services that their customer demand. Also since these devices need to interoperate with other communication equipment, router manufactures need to take a leadership rose in evolving the industry and driving industry innovation where more data traffic is directed to the router and it continues to serve as a platform where IP services an can delivered. This may require proposing new

standards. Intel did this with the microprocessor and led the industry is defining the architecture of the PCI bus and later with the USB interface.

Also, corporations are concerned about network security and protecting their networks from attacks. Equipment suppliers already provide solutions such as Deep Packet Inspection, Firewalls and Intrusion Detection, but these solutions provide security only at the periphery of the corporate network and do not go all the way to a desktop in case a malicious packet does enter the network. For solutions to such threats, telecom equipment makers should partner with companies with expertise in desktop information security to extend security provisioning from the core of the network to the desktop.

For selling to telecom carriers and ISPs, equipment manufacturers usually have long term contracts and relationships with customers such as AT&T, Bell South, Verizon, BT etc. The equipment manufacturer usually sells their product directly to the carrier or ISP. However, enterprise customers usually buy communications equipment through a reseller or a distributor and do not go directly to the equipment manufacturer. Resellers usually carry equipment from a number of manufacturers. In order to sell a particular vendor's equipment, they need an incentive since most communications equipment is interoperable and standards based. Therefore telecom equipment makers need to have relationships with resellers and should collaborate with them in developing router features that enable the reseller to provide new and innovative communications solutions to end customers.

Lever 4: Internal organization structures

In the telecommunication industry, where customer needs are changing, new technology is emerging and standards are evolving, telecom equipment makers need an organization structure and business processes in place to manage internal R&D efforts. Additionally, they need to create a platform and manage relationships with external complimentors. They also need to manage their internal development where they can leverage technology developed by complementors and third parties, and yet continue to innovate in core technologies that provide them with a competitive edge. These firms also need to continuously replace existing products with newer technologies and occasionally expand product lines with innovative, high-quality products that minimize development and product costs.

For software development efforts they need to create a common software platform that provides internal R&D efficiency. Additionally, this platform should enable other firms to innovate upon by either developing routing software components or developing complementary technologies which interoperate with other telecommunication equipment.

5.2 The Software Development Process

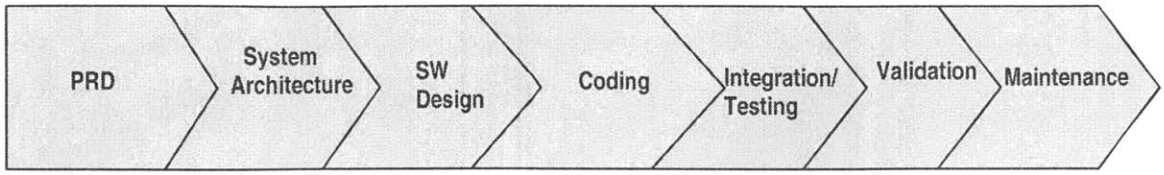

Figure 5.3: Software Development Life Cycle

Figure 5.1 shows the life cycle model for developing software. The software development process starts off with a product requirements document. This is typically written by a Product Manager responsible for the product and is based on customer needs and market requirements. System architects generate the overall architecture for the product and design of the software. The implementation phase includes coding, integration and unit testing. After the development effort is complete, the software goes through a validation phase and defects found in this phase are corrected. After the software is released to customers, it enters a maintenance phase where customer reported defects are corrected or the development team continues to add new features.

5.3 Software Development Organizations

Software development teams are typically organized either in a functional arrangement or a project arrangement as shown in Figure 5.4 and Figure 5.5[38] respectively. For large software development organizations, these teams are part of a line of business within a corporation.

The team structure shown in Figure 5.4 and Figure 5.4 is typical for creating products described in section 4.4. The products created have their own applications with limited common components. Also they have customized internal development processes and can use separate verification and validation teams for each product. For developing a common software platform, having the right organization structure is important so the platform can evolve over time as new features are added and as the company's strategy changes due market requirements.

As shown in Figure 5.6, applications developed by individual product groups operating in silos get unified into a common development platform though which a range of products get generated. Previously, duplicate application development teams created the same applications for separate products.

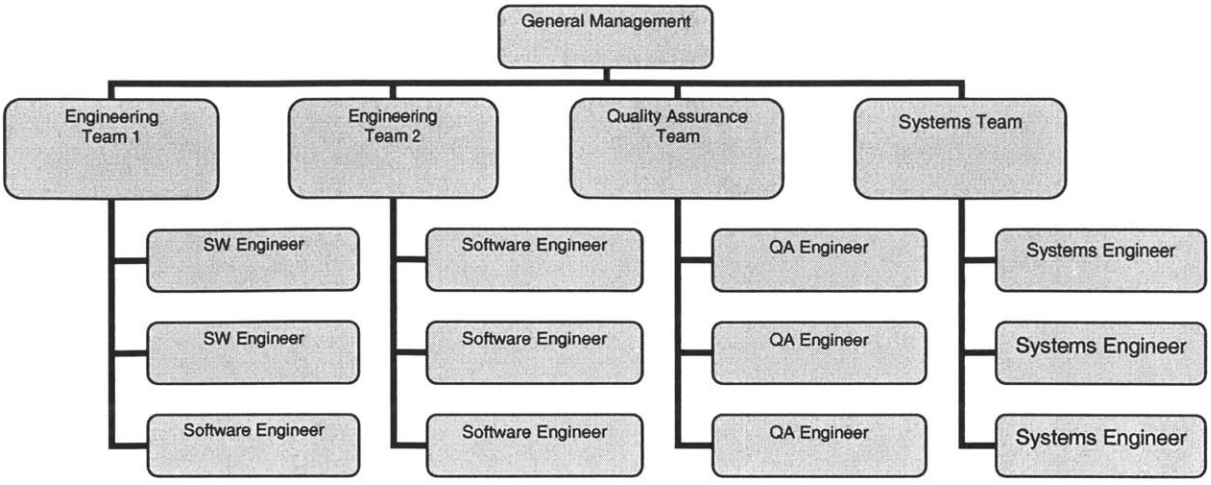

Figure 5.4: A Functional Organization

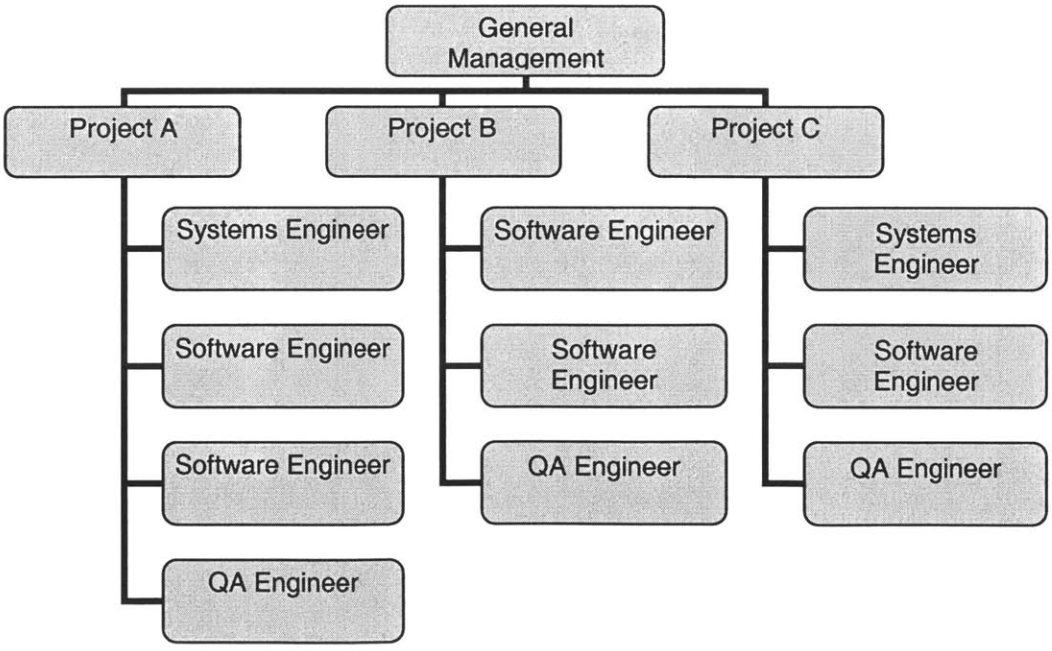

Figure 5.5: A Project Organization

An Integrated origination structure, as shown in Figure 5.6, allows for innovation to occur within the teams. Instead of the teams rushing to meet product deadlines and duplicating effort in the areas of application development and product verification/validation, specialized teams can focus on creating new application or enhancing exiting applications. Likewise the QA teams can invest in automated test tools and perform system tests with the same resources for both Products and Products B. Similarly, software developers can develop applications X and Y for products A and product B.

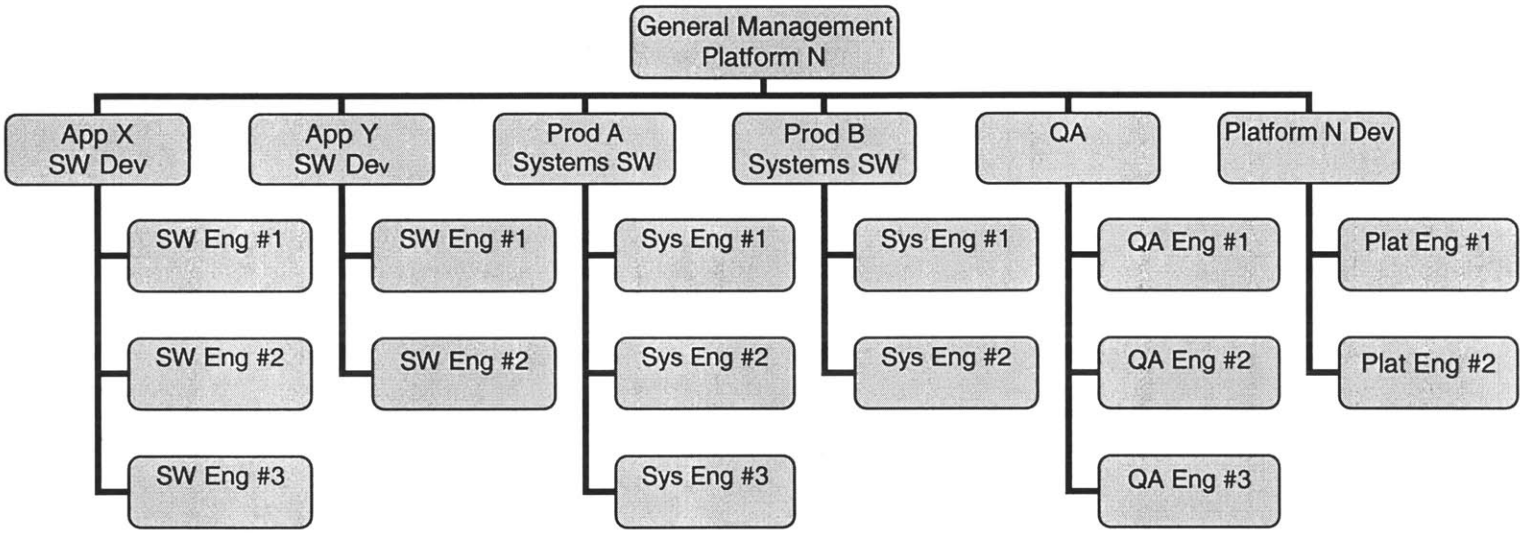

Figure 5.6: An Integrated Organization

5.4 Business Processes

To achieve different products in the same product family using a common software platform, development processes within an organization are necessary to create software for each product that gets derived from the common platform. As shown in Figure 5.7, without a common platform approach, as described in Section 4.4, each product within the same product family would need its own software development environment. Additionally, each product has its own development processes as well its product verification teams. In a common software platform approach shown in Figure 5.8, duplicate business process can be eliminated and verification/validation teams are unified where new products are derived from the same common software platform.

For each platform, a common source control system is needed where application developers can submit their software changes. By having a common source code control system along with an integrated build environment capable of generating multiple targets, executables for each product are created from the common code base. A Release Management system provides periodic software updates for products that use the common platform so it evolves with time.

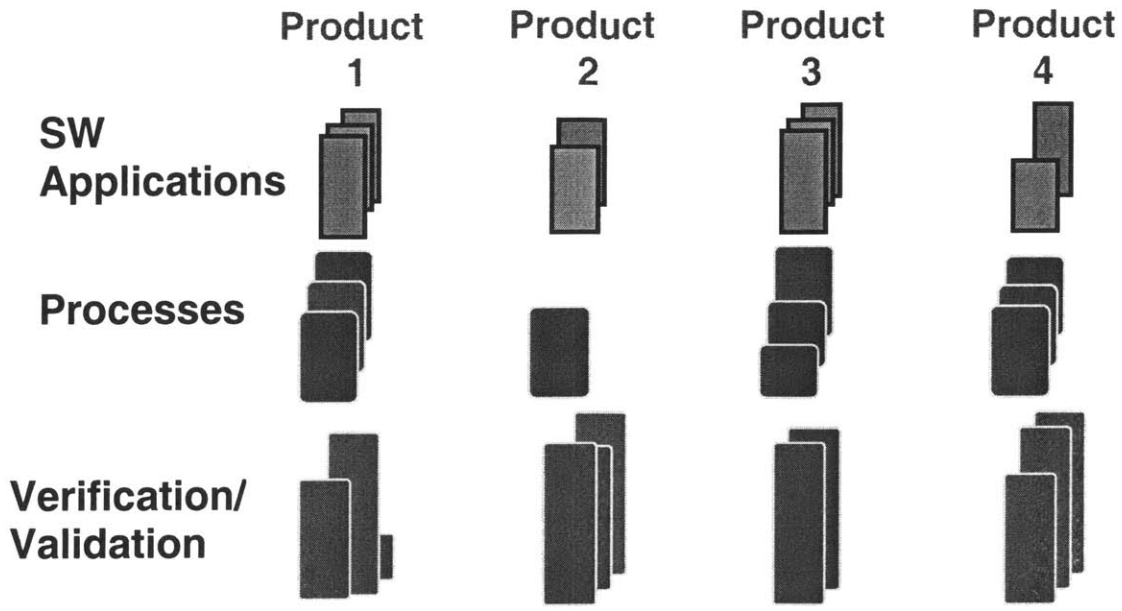

Figure 5.7: Product development without a common software platform

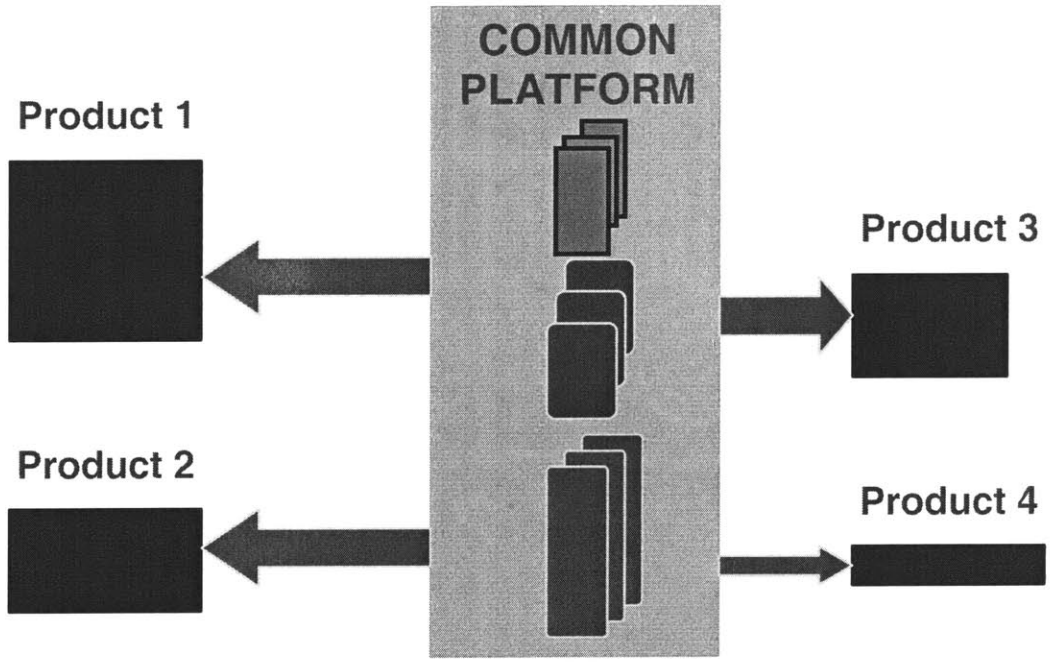

Figure 5.8: Products derived from a common software platform

5.4.1 Common Source Code Control System

If a software developer adds new code or modifies the source code for a particular application, that change can now get propagated to all the products by using a common source code control system. So a change in one source file is effective for all products. Compare this approach to the one where a developer would have to physically modify separate files for each product using the same application. Since there are fewer files to modify, the chances of making an error get decreased. Additionally, only one code inspection is needed instead of multiple inspections for multiple code changes.

5.4.2 Integrated Build Environment

An integrated build environment provides tools for developers so targets for multiple products can be created. Since the products using the common code base have unique hardware architectures, different compliers, linkers and utilities must be part of the build environment. The application source code is common across the products, but device drivers, interrupt handlers and other hardware specific modules can be unique to the specific product. For the integrated build environment to work, this platform specific code must also be part of the build environment. Specialized scripts and Makefiles can provide a mechanism for building and distributing executable images for products using the common platform.

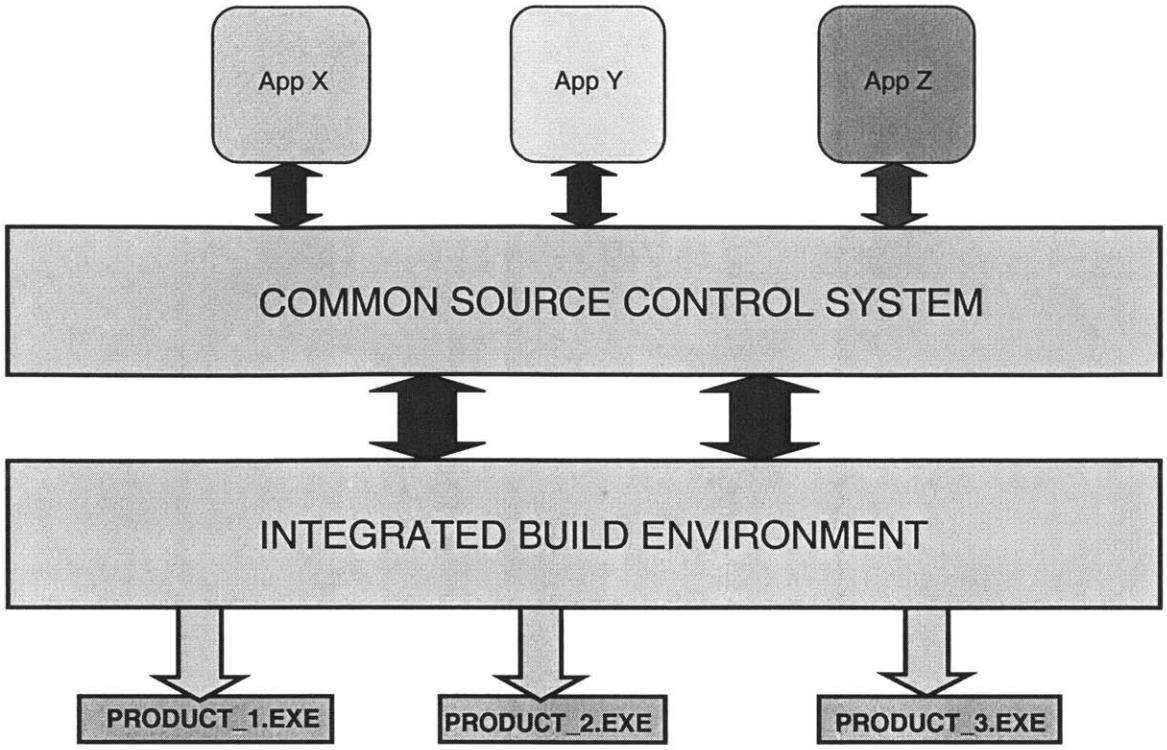

Figure 5.9: A Software Development Environment

5.4.3 Release Management

A release management system is required where new versions of software are available from all products within a family. When a release is created, it includes executable software for all products within the same platform. Effective release management ensures that customers have a process of receiving new software as well as upgrading their software to install patches to correct problems. Figure 5.10 shows a software release roadmap.

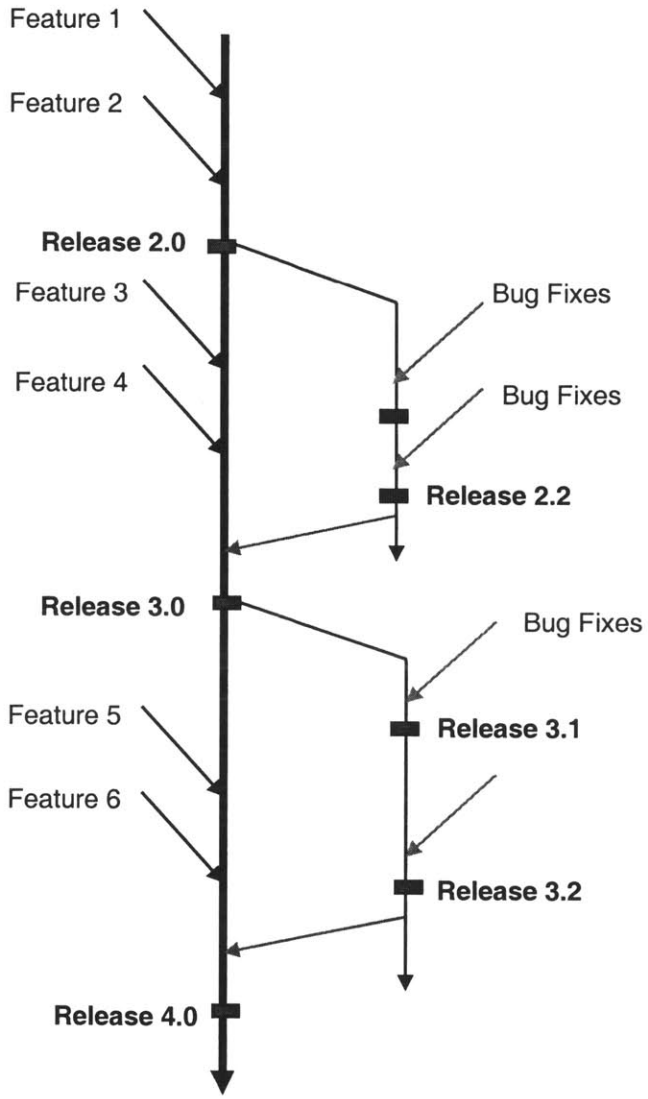

Figure 5.10: Roadmap of Software Releases

5.5 Software Platform Evolution

As more and more functionality in embedded devices moves to software, a software platform needs the ability to support new technologies and standards as they emerge. This capability depends on the software architecture of the system. The platform also needs to support new market requirements and technology trends, otherwise customers and third parties will loose incentives in building complements or offering complementary services. Evolution of the platform is necessary, otherwise a platform leader can fall prey to the competition.

For example, due to limitations in Cisco's IOS, the platform was unsuitable for high speed routers with carrier grade and high availability requirements, and essential feature for ISPs or routers operating in the core of an enterprise network. Due to its monolithic architecture, IOS was unable to meet the performance, scalability and security requirements for carrier customers. IOS runs as a single image and all processes share the same memory space with no memory protection between processes. It lost market share in high end routers to Juniper Networks (the M40, and M160 product lines), whose products were superior to Cisco's in performance and scalability. Cisco's response was to develop a new version of Cisco IOS called IOS-XR used on the Cisco 12000. It also had to rewrite a large part of the IOS code to provide, modularity, memory protection, pre-emptive scheduling, and the ability to independently restart failed processes, which are necessary features to support carrier grade requirements. IOS-XR also uses a 3rd party real-time operating system (QNX). Despite this undertaking, newer modular operating

systems are still technologically superior to a modified IOS. This makes it challenging to integrate technologies from third parties or from acquisitions.

In evolving software platforms, companies should consider providing a migratory path from older platforms to the newer ones if necessary. Thus customers with investments in older platforms can seamlessly move their applications and installed based to use the new platforms. For example, Microsoft provided backwards compatibility for applications that used newer versions of the Windows operating system.

CHAPTER 6: CONCLUSIONS AND FINAL THOUGHTS

A platform based approach provides a foundation for creating new products that share key components, but provides enough differentiation to allow a firm to capture a wider market for its products. By using a common platform and focusing on the development and sharing of components, a set of derivative products can be created. This allows a company to view its products as part of a portfolio and to make project investments by introducing new technologies in as many products as possible. Product platforms can eliminate redundancies in product development and permit organizations to make the best possible use of R&D investments.

Using this concept in embedded systems software development, where software modules are shared across products, embedded developers can achieve effective reuse of software. By using a common software platform, companies can create software applications for one product that can be shared across all products within the same product family. This allows for a component to be developed once and its implementation shared in several products, which results in lower software development and maintenance costs. By having a multi-project mindset and focusing its thinking and resources from single products to a family of products, a company can achieve engineering efficiencies and can quickly utilize its newly developed technology and software applications across multiple products. Using concurrent technology transfer, a company can introduce a new product for a different market segment shortly after developing a base platform. This way, a company

can reuse technology from a base platform to a new product as well as achieve sales growth and increase in market share.

While the concept of common software platform is powerful, the value realized may be limited if an effective organizational structure is not in place. Additionally, commitment from the firm's senior managers in platform development is vital for projects utilizing a common software platform to achieve success. Managers need to understand the distinction between product development and platform development. The goal of the platform development team is not to directly create a new product, but to create pieces or elements that enable the development of common components and subsequent products. In the case of embedded products, it is software applications and modules that are common to all products within a family. The product teams are ultimately responsible for integrating the unique software applications and technologies developed by the platform teams and creating new products using their work. Based on my experience, companies that have the same team responsible for both product and platform development fail to realize the benefits of a common software platform. Typically, the team focuses it efforts on short term product specific customer requirements. Therefore, the development of common software, which would benefits other products in the portfolio, becomes a lower priority. Often this results in project failure and negatively impacts product teams that are expecting common software components for their specific product.

Companies planning to utilize the concept of a common software platform also need to decide on funding model for the development of common components. The common

software and application groups must have adequate resources to innovate and develop features that provide the firm a competitive edge. Investments in a new platform should not be justified or planned based on a single product, but rather on the entire product family that will be based on that platform. Therefore it is essential for success of a common software platform to have the appropriate investments in technology and resources.

For continued success in new product development, a firm must renew its platform architectures. To evolve the platform, it is important to make technological improvements in the platform by keeping it current with the latest standards, advances in technology and market needs. Managing a software platform's transition from one technological environment to the next is one of the greatest challenges for software developers. When done well, this migration is largely transparent to users, and the product family continues to grow as application developers exploit the more powerful capabilities incorporated into the new platform[39]. Successfully managed software platforms can evolve as new ideas and technological advances are integrated.

For telecommunication equipment manufacturers, a platform based approach for developing software provides a pathway for growth and R&D efficiencies. Communications products are far too complex to develop one product at a time. It is far more efficient to create a platform and launch derivate products. By developing a technology or software application once, it can easily be made available to all products within the product family. Furthermore, by using concurrent technology transfer, new

products for different market segments can be introduced a lot quicker. An industry still struggling after the telecom crash of 2001, telecom equipment makers need to create a communication platform that can not only provides new features and functionality based on customer demand or technological change, but also serves as a foundation where ISPs and telecom carriers can provide their direct customers with enhanced communication capabilities. It is new services and communication capabilities which increase network traffic and create demand for routers and other communications products.

A common software platform encourages a firm to take along term view of its product strategy. The key to success in the high technology industry is innovation and new product development. With the right strategy, a firm can evolve its platform as a foundation for other companies to develop complements and offer services. By doing so, a company ensures continued demand for its products.

ABBREVIATIONS

AGP	Advanced Graphics Port
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
BWB	Blended Wing Body
CLI	Command Line Interface
COTS	Commercial off-the-shelf
CPU	Central Processing Unit
DRAM	Dynamic Random Access Memory
DSL	Digital Subscriber Line
GUI	Graphical User Interface
IETF	Internet Engineering Task Force
IOS	Internetwork Operating System
IP	Internet Protocol
IPSec	Internet Protocol Security
IPTV	Internet Protocol Television
ISA	Industry Standard Architecture
ISO	International Standards Organization
ISP	Internet Service Provided
MPLS	Multi Protocol Label Switching
OEM	Original Equipment Manufacturer
OS	Operating System
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Digital Assistant
QA	Quality Assurance
QOS	Quality of Service
R&D	Research and Development
RIP	Routing Information Protocol
SNMP	Simple Network Management Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
USB	Universal Serial Bus
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WAN	Wide Area Network

REFERENCES

[1] Michael A. Cusumano and Kentaro Nobeoka, *Thinking Beyond Lean: How multi-project management in transforming product development at Toyota and other companies,* The Free Press, New York 1998

[2] Mark W. Maier and Eberhardt Rechtin, *The Art of Systems Architecting*, CRC Press, 2002, page 89

[3] Alberto Sangiovanni-Vincentelli and Grant Martin, "Platform-Based Design and Software Design Methodology for Embedded Systems", IEEE Design & Test of Computers, November – December 2001.

[4] Douglas C. Schmidt, "Why Software Reuse has Failed and How to Make It Work for You", University of California, Irvine. http://www.cs.wustl.edu/~schmidt/reuse-lessons.html

[5] Nancy Levesson and Kathryn Ann Weiss, "Making Embedded Systems Software Reuse Practical and Safe", Massachusetts Institute of Technology, 2004

[6] Bureau of Labor Statistics, U.S. Department of Labor, *Career Guide to Industries, 2006-07 Edition*, Telecommunications, on the Internet at http://www.bls.gov/oco/cg/cgs020.htm (visited *April 23, 2006*)

[7] Michael E. McGrath, 1995, *Product Strategy For High-Tech Companies,* New York, NY: Irwin Professional Publishing, 1995.

[8] Oliver de Weck, "Product Family and Platform Strategy," Massachusetts Institute of Technology, 2005

[9] Annabell Gawer and Michael A. Cusumano, *Platform Leadership*, Harvard Business School Press 2002

[10] Meyer, M. H. and Utterback, J. M., 1993, "The Product Family and the Dynamics of Core Capability," *Sloan Management Review*, Vol. 34, p. 29-47.

[11] Meyer, M.H. and Lehnerd, A. P., *The power of product platforms: Building Value and Cost Leadership,* Free Press, NY 1997

[12] Michael A. Cusumano and Kentaro Nobeoka, *Thinking Beyond Lean: How multi-project management in transforming product development at Toyota and other companies,* The Free Press, New York 1998, page 8

[13] Michael A. Cusumano and Kentaro Nobeoka, *Thinking Beyond Lean: How multi-project management in transforming product development at Toyota and other companies,* The Free Press, New York 1998, pp 19-49

[14] R. Bremmer, "Cutting edge platforms," *Financial Times Automotive World,* June 1999, p.30-41

[15] Eun Suk Suh, "Flexible Product Platforms", Ph. D. Thesis, Massachusetts Institute of Technology, 2005

[16] Blended Wing Body: National Aeronautics and Space Administration, http://oea.larc.nasa.gov/PAIS/BWB.html

[17] Ravindra M. Kurtadikar and Robert B. Stone, "Investigation of Customer Needs Frequency vs. Weight in Product Platform Planning," Proceedings of IMECHE 2003, ASME International Mechanical Engineering Congress and R & D Expo November 2003.

[18] Microsoft Corporation Website, http://www.microsoft.com/net/default.mspx

[19] D. Robertson and K. Ulrich, K., "Planning for Product Platforms," *Sloan Management Review* 1998

[20] Nexthop Technologies Inc. website, http://www.nexthop.com/technology/routing.html

[21] AT&T corporate website, http://att.sbc.com/gen/investor-relations?pid=5711

[22] "Meeting your growth goals in challenging times", *Executive Agenda*, A.T. Kearney, Inc. 2005.

[23] "Enemy Lines In Lucent Deal, Two Rivals Face Inroads From China", *The Wall Street Journal*, March 25, 2006.

[24] "Lucent, Alcatel Are Far Along In Merger Talks", *The Wall Street Journal*, March 24, 2006.

[25] S. Davis, J. MacCrisken, K. Murphy, "Economic Perspectives on Software Design: PC Operating Systems and Platforms", National Bureau of Economic Research, Cambridge MA, © 2001

[26] Joseph A. Carfagno, "A common software platform for realizing integrated bellcore software systems," Bell Communication Research, 1991

[27] Nokia website, http://www.forum.nokia.com/main/0,6566,010,00.html

[28] Lily Yu et al, "Cisco IOS Software Based Managed Services Business Overview:, Cisco Systems, Inc. 2004

[29] Cisco IOS Technologies, Cisco Website: http://cisco.com/en/US/products/ps6537/products_ios_sub_category_home.html

[30] Annabell Gawer and Michael A. Cusumano, *Platform Leadership*, Harvard Business School Press 2002, page 266

[31] David Bunnell, Making the Cisco connection: The story behind the real Internet superpower, New York: John Wiley & Sons, 2000, page 116

[32] Annabell Gawer and Michael A. Cusumano, *Platform Leadership*, Harvard Business School Press 2002, page 38

[33] David Bunnell, Making the Cisco connection: The story behind the real Internet superpower, New York: John Wiley & Sons, 2000, page 84

[34] Edward Crawley, Massachusetts Institute of Technology, 2005

[35] Michael A. Cusumano and Kentaro Nobeoka, *Thinking Beyond Lean: How multi-project management in transforming product development at Toyota and other companies,* The Free Press, New York 1998, pp 9-17

[36] Annabell Gawer and Michael A. Cusumano, *Platform Leadership*, Harvard Business School Press 2002, page 252-254

[37] "The Anatomy of Today's Core Routing Operating System", Juniper Networks, Inc., Sunnyvale CA, 2005

]38] Richard H. Thayer, "Software Engineering Project Management", California State University, Sacramento CA.

[39] Meyer, M.H. and Lehnerd, A. P., *The power of product platforms: Building Value and Cost Leadership,* Free Press, NY 1997, page 37 and 192

THIS PAGE IS INTENTIONALLY LEFT BLANK