Managing Embedded Software Development in China

by

Wei Hu

M.S. Electronic Engineering (1996)
Southeast University, China

B.S. Electronic Engineering (1993)
Northwestern Polytechnical University, China

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

[February 2004]
January 2004

Signature of Author_____

Wei Hu
System Design and Management Program
January 2004

Certified by_____

Michael A. Cusumano
Thesis Supervisor
Sloan Management Review Professor of Management

Accepted by_____

Thomas J. Allen
Co-Director, LFM/SDM
Howard W. Johnson Professor of Management

Accepted by_____

David Simchi-Levi
Co-Director, LFM/SDM
Professor of Engineering Systems

Chapter 1 Embedded Software Overview

1.1 Introduction to Embedded Software

As microprocessors have become smaller and cheaper, they are embedded in more and more non-computing products, such as washing machines, elevators, MP3 players and printers. It has been estimated that these products consumed 99% of the worldwide production of microprocessors. [1]

In general, "Embedded system" means a computer system sitting inside a product other than a computer to make the product more flexible and controllable. For example, a modern washing machine has a control software system to execute different "washing programs" for different types of clothes.

Embedded systems usually have strict requirements on response time, and the response must be generated within a finite and specified period, though depending on the situation, the time could be within a few milliseconds or a few seconds. Because of the special requirement on response time, embedded systems are sometimes called real-time systems.

Embedded systems can be divided into two categories: hard and soft, according to the degree of required "timeliness"[2]. A hard embedded system is stringent on that the response must occur within a specified timeline. Typical examples are flight-control systems and missile control systems. A soft embedded system is less strict: response time is important but the system still can function properly given occasionally missed deadline. Examples are mobile phones, printers, and medical devices. This paper is only concerned with development of the soft-embedded systems, and hence the term "embedded systems" in the paper means "soft embedded systems".

1.2 Characteristics of Embedded Systems

Though embedded systems are computer systems, they are different from personal computers (PCs) we are familiar with in both hardware and software aspects.

1.2.1 Hardware characteristics

Unlike a PC, which stores program and data on external storage devices, such as hard disks, and loads them into memory when system boots up, an embedded system usually doesn't have external storage devices and must store its program and data in memory and keep them in memory even without power.

Embedded systems often have standard serial ports, network interface and hardware to interact with sensors and activators, but lack the rich peripherals a personal computer has.

---

[1] Alessandro Pasetti, Software Frameworks and Embedded Control Systems, Springer
[2] J.E. Cooling, Real-time Software Systems, Thomson Computer Press

- A monitor. Some systems may have a liquid crystal display (LCD) that can display several lines of characters, and some only have a few diodes to indicate the system's states.
- A keyboard. Most embedded systems only have a few buttons for input.
- Disk drives. As both program and data are stored in memory, and embedded systems don't need such devices.

Embedded systems usually don't require high computing power, and tend to use less advanced CPU models, and also have less memory.

The following table shows some CPUs commonly used in embedded systems, and their memory and speed information.

Processor	Bus Width	Memory	Max Speed (MIPS)
Zilog Z8 family	8	64K (maximal)	1
Intel 8051 family	8	64K program + 64K data	1
Zilog Z80 family	8	64K; 1M	2
Intel 80188	8	1M	2
Intel 80386 family	16	64M	5
Motorola 68000 family	32	4G	10
Motorola PowerPC family	32	64M	75
Intel StrongArm	32		2.1
Motorola DragonBall	32		5.4

Table 1.1

1.2.2 Software Characteristics

Embedded systems deal with real-world inputs through hardware peripherals. The need to respond to and control external hardware often results in strict timing requirements, and most embedded systems must then be built to ensure certain minimal response time or to guarantee a certain minimal throughput. Meeting such requirements can have a significant impact on the architecture of the software and programming paradigm, and timing aspects becomes one of the most important design drivers of embedded software.

Because embedded software is usually used to collect inputs and control devices, it usually lacks the complex user interface conventional software systems have, and has a much smaller proportion of UI related code, whereas in the convention software application with graphical user interface, the proportion of UI related code can reach 90%[3].

Embedded systems are designed to operate without direct human supervision. Compared with an information processing system or desktop system, it requires a higher degree of autonomy and reliability, guaranteed response time and lower maintenance cost.

[3] Alessandro Pasetti, Software Frameworks and Embedded Control Systems, Springer

Embedded software systems usually have two different system architectures depending on whether they are built on a real-time operation system (RTOS). Figure 1.1 illustrates the architecture diagram built on a RTOS.

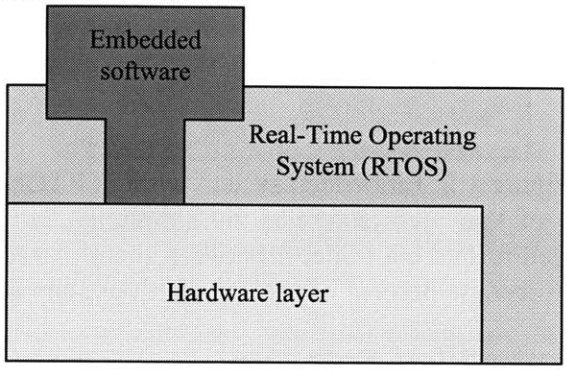

Figure 1.1 Embedded system with RTOS

A RTOS provides infrastructure services to ease the work of developing embedded software. Typical services include scheduling service, tools to protect shared data and facilitate inter-task communications, such as semaphores and message queues, and timing service. To save memory and improve performance, a RTOS typically include just the services that are necessary for embedded systems, and most RTOSs allow users to configure them extensively to streamline features further.

In spite of similar names, most real-time operating systems are quite different from desktop machine operating systems such as Windows or UNIX. The biggest difference lies in the relationship between OSs and applications. On a desktop computer, the OS takes control of the machine as soon as it is turned on and then lets users to start other applications. In contrast, in an embedded system, an application is linked into RTOS, and at boot-up time, the application gets control first and it then starts the RTOS. RTOSs also lack the memory protection mechanism Windows and UNIX have. Consequently, an application crash can easily bring down the whole system. The tight coupling reflects the fact that most embedded software is designed for a specific task, and it does not make much sense to keep the OS alive if the application crashes.

Popular RTOSs include VxWorks, Windows CE, Palm OS, Real-time Linux and etc.

Because RTOSs are general commercial products, to lower cost and potential overhead, many small projects choose not to use a RTOS and build their own modules to handle boot-up, interruption processing, tasking scheduling and simple mechanism to protect shared data. Figure 1.2 illustrates the architecture diagram without a RTOS.

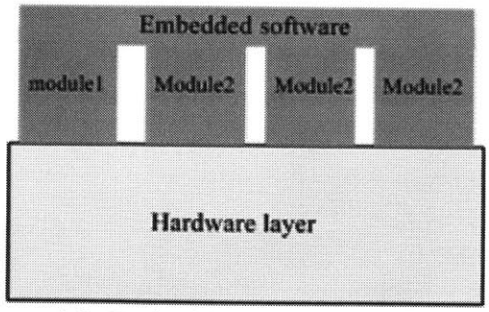

Figure 1.2: Embedded system without RTOS

The biggest advantage of this architecture is the simplicity, but it is not flexible and scalable. Developers need to make a tradeoff between them. A recent survey shows that in Asia-pacific region, forty-one percent of projects don't use any RTOS. [4]

1.3 Developing Embedded Software

1.3.1 Design methodologies

The most important objective in the development of any embedded system is to make a design that meets requirements. In this aspect, embedded systems are no different from other software application. A typical cycle of an embedded software application consists five stages.[5]

- Requirement specification
- Architectural design
- Detailed design
- Implementation
- Testing

Because of hardware constraints and performance requirements, in the early stage, assembly and C language have been the dominant implementation language in developing embedded software, and they are still popular even today. For example, in 1996, a control system of a large satellite was written in assembler. Structured bottom-up approach is the major programming paradigm. Embedded software was often considered a supplement to hardware, and many software projects were not well planned and managed, and grew as hardware evolved and requirement increased. For example, the engine control software of a line of middle-sized cars consisted about 300,000 lines of C code in 1997, and was poorly structured.[6] Also due to the limited hardware capacity, the research on methodologies for embedded software lags far behind the practice, as the software systems becomes more complex, many embedded systems suffer from poor reusability and maintainability.

Due to the fast improvement of CPU and memory design and manufacturing technologies, the hardware constraints have loosened up significantly. Consequently, many efforts have been spent to bring good practices in developing conventional software products, such as

[4] Gartner Report: Asia/Pacific: Embedded Systems Design, Software Decisions, May 2003
[5] A. Burns and A. Wellings, Real-time Systems and Programming Languages, 3rd edition, Addison-Wesley
[6] Alessandro Pasetti, Software Frameworks and Embedded Control Systems, Springer

Modeling languages, Object-Oriented design and programming, Component-based programming, to developing embedded software systems.

Gomaa suggested that there are four important objectives for a real-time design method:[7]

- To structure a system in concurrent tasks
- To support the development of reusable components through information hiding
- To define the behavior aspects using finite-state machines
- To analyze the performance of a design to determine its real-time properties

An embedded system usually consists of computers and several coexisting external elements with which the computer programs must interact simultaneously, and concurrency is often inherent in embedded software systems. SDL-RT addresses these issues well and is popular in the industry. Though UML is designed for embedded software, many efforts have been spent to extend it for embedded software.

1.3.1.1 SDL-RT[8]

Specification and Description Language (SDL) is an object oriented, formal language defined by International Telecommunications Union (ITU, formerly CCITT). The language is design to specify complex telecommunication systems, which are usually event-driven, real-time, and interactive, and involve concurrent activities communicating with discrete signals.

Though SDL has been developed in the first place for telecommunication applications, experience showed that some of its basic principles could be applied in a wide variety of real time and embedded systems.

SDL-RT inherits the merits of standard SDL and improves it on several aspects:

- It replaced SDL obsolete date types with ANSI C data types
- It added "semaphore" symbols to the model
- It removed some message types that are rarely used in embedded systems.
- It adopted UML for modeling object oriented content in v2.0.

This section gives brief description of SDL-RT, and demonstrates how it can be used to model special characteristics, such as time constraint and concurrence, of an embedded system. Though it focuses on SDL-RT, most of the content applies to SDL as well.

In SDL-RT, an element in the system is called an agent, and there are two kinds of agents: blocks and processes. A system itself can be viewed as the outermost block. A block can be further decomposed into smaller and simpler blocks, and a lowest level block can be composed of one or several processes, which fulfill the block's functionality.

A process is modeled as a finite state machine, and it has a set of states and conditions that cause states to transit. A process communicates with other processes through

[7] Software design methods for the design of large scale real-time systems, Journal of Systems and Software
[8] SDL-RT standard, http://www.sdl-rt.org

message exchanges, and its state transitions are usually triggered by messages it receives. Processes can be concurrent, and it is also allowed to have several instances of the same process running independently.

Message sequence chart (MSC) is the major tool to show process state transition. Figure 1.3 shows server process enters idle state after startup, and process caller in its start transition sends a conReq to server and goes to state idle. Process server returns a conConf message and enters connected state. The caller in idle state transits to connected state after receiving conConf message.

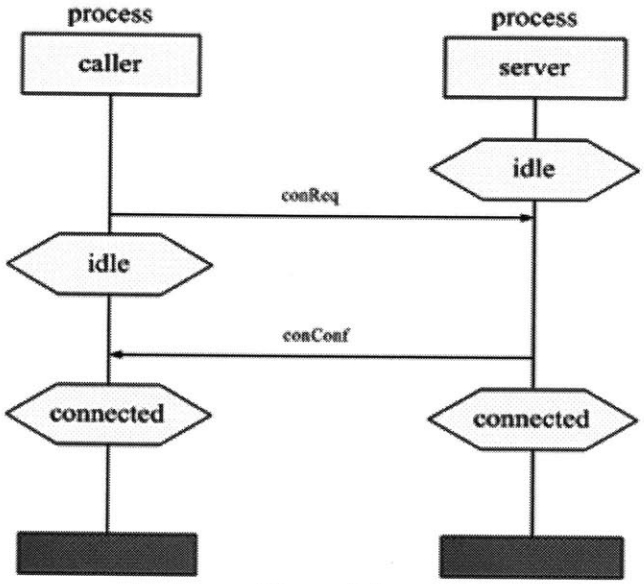

Figure 1.3

With timers defined in SDL-RT, the time constraint can be expressed conveniently in the model. Figure 1.4 shows an example with some time constraints.

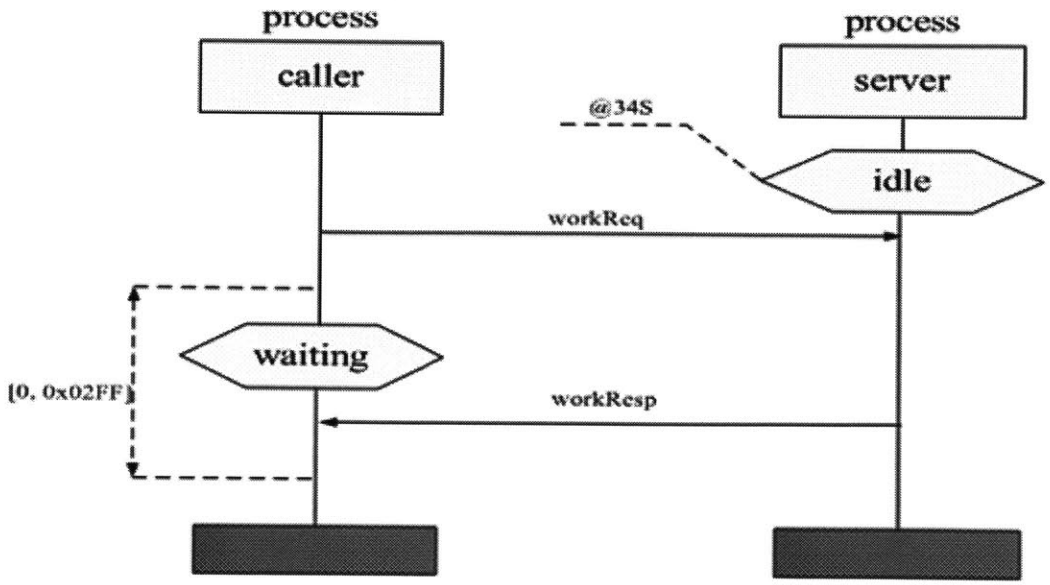

Figure 1.4

This MSC diagram shows that server process reaches idle state at absolute time 34 sec, and caller process requests server process to compute some work in less than 0x02FF time units.

As a process is a finite state machine, MSC describes its internal state transition effectively and efficiently. However, a system usually consists of multiple processes, and these processes can run in parallel or alternatively. It is sometimes needed to describe such relationships between processes. SDL-RT provides high-level MSC (HMSC) diagram to model the relationships.

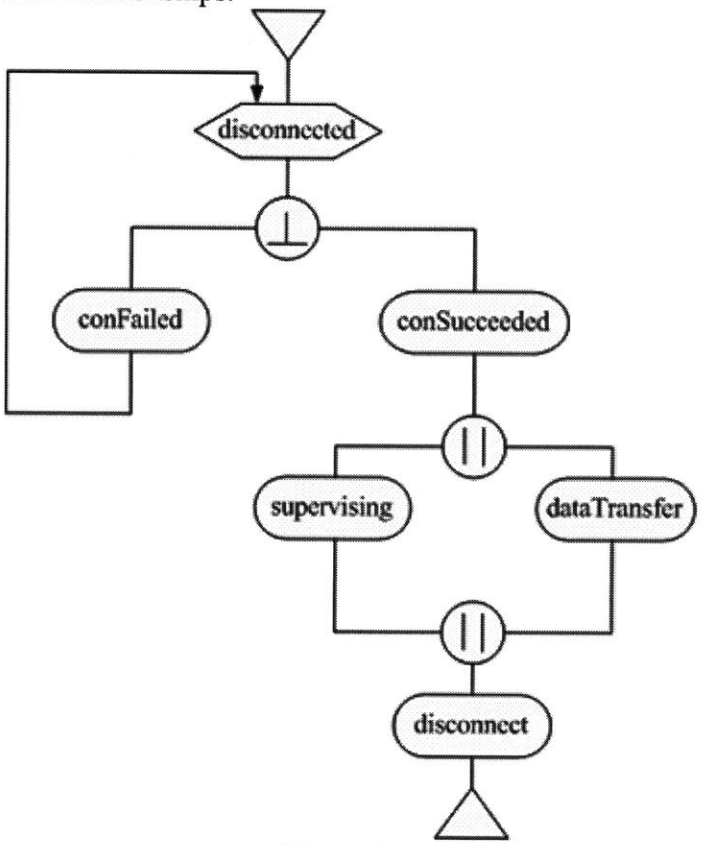

Figure 1.5

In Figure 1.5, the system starts in disconnected state, and either process conFailed or conSucceeded is executed depending on the return result of connection attempt. In the scenario of conSucceeded, the process supervising and dataTransfer run concurrently, and both end in disconnect process.

SDL-RT uses UML to describe classes, relationships among classes, package and deployment.

SDL-RT is a formal language, and in theory, the model can be translated to executable codes directly. However, such tools are not available yet.

1.3.1.2 Unified Modeling Language

Over the last ten years, object-oriented (OO) analysis and design has become the mainstream methodology in software development, and the Unified Modeling Language (UML) was adopted as the standard OO notation. Many of key UML features are suitable for modeling real-time embedded systems. For example, the object model and packaging can be used to capture system architecture and organize elements into groups. To better support modeling real-time applications, the proposed real-time UML standard includes features to model key concepts in embedded software.[9](All the figures in this section are from the same reference.)

In the "CommonBase" package, it introduces a Resource package and a Time and a Concurrency package that derive from the Resource package, as shown in Figure 1.6.

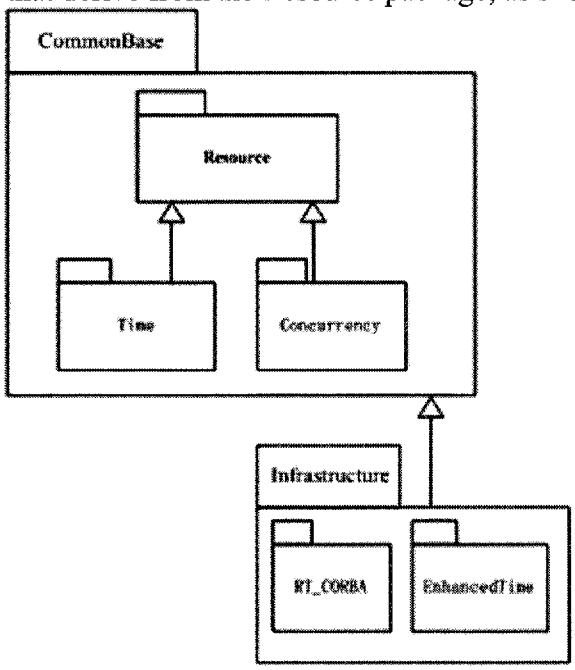

Figure 1.6 The Structure of real-time profile

The base package is extendable to support specific technologies, such as Real-Time CORBA.

Resource models the physical aspect of software, and a resource is viewed as a server that provides one or more services to its clients. The physical limitations of a resource are represented through its quality of service attributes (QoS). QoS attributes can be used to model many different properties. In most cases, the values are quantitative, for example, size, service time and capacity, but they can be qualitative as well, and scheduling policy and failure model are two such examples.

[9] Bran Selic, The Emerging Real-Time Standard, Proceedings Sixth International Workshop on Objected-Oriented Real-Time Dependable Systems

In the proposed real-time UML, a situation involving resources and their clients is referred to an analysis context. The context can be static and dynamic. The relationship between resource and QoS attributes is referred to general resource model, which is the basis for all other aspects of the real-time UML profile. Figure 1.7 is the relationship diagram.

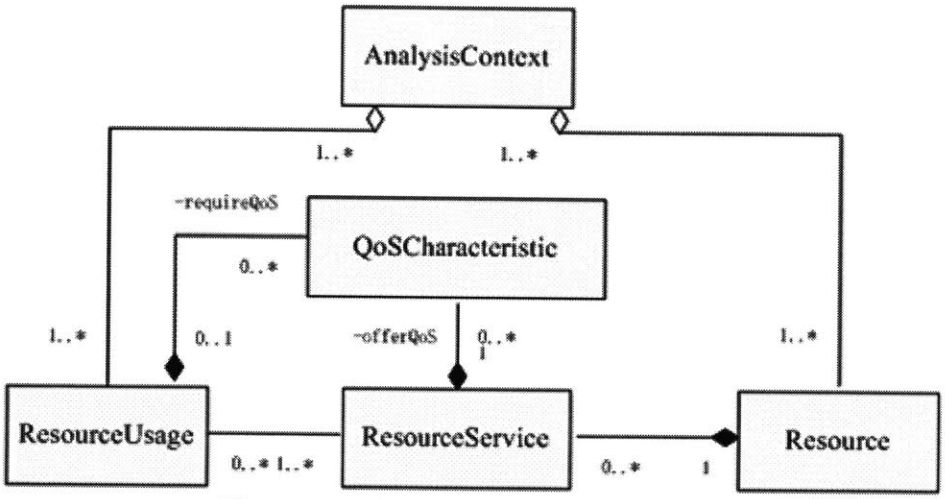

Figure 1.7 The General Resource model

Figure 1.8 is an example of a simple static analysis context. It shows two active and potentially concurrent clients accessing a shared monitor resource. The time constraint is modeled as QoS attributes.

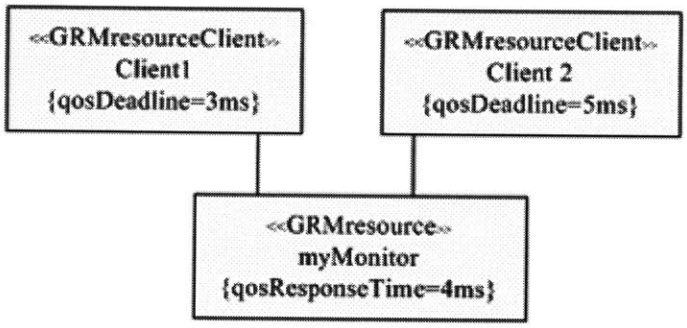

Figure 1.8 Example of a service representation

The Time package that derives from Resource is used to model the time constraint of embedded applications. Physical time is modeled as a set of physical time instants, and the count of the number of expired cycles is used to measure the progress of physical time.

The real-time URL proposal distinguishes two types of timing mechanisms: timers and clocks. Clocks periodically generate special kinds of time events called clock ticks and timers only generate a single timeout event when their duration expires, unless they are periodic timers, which act like clocks. The relationship among physical time, clocks and timers is demonstrated in figure 1.9.

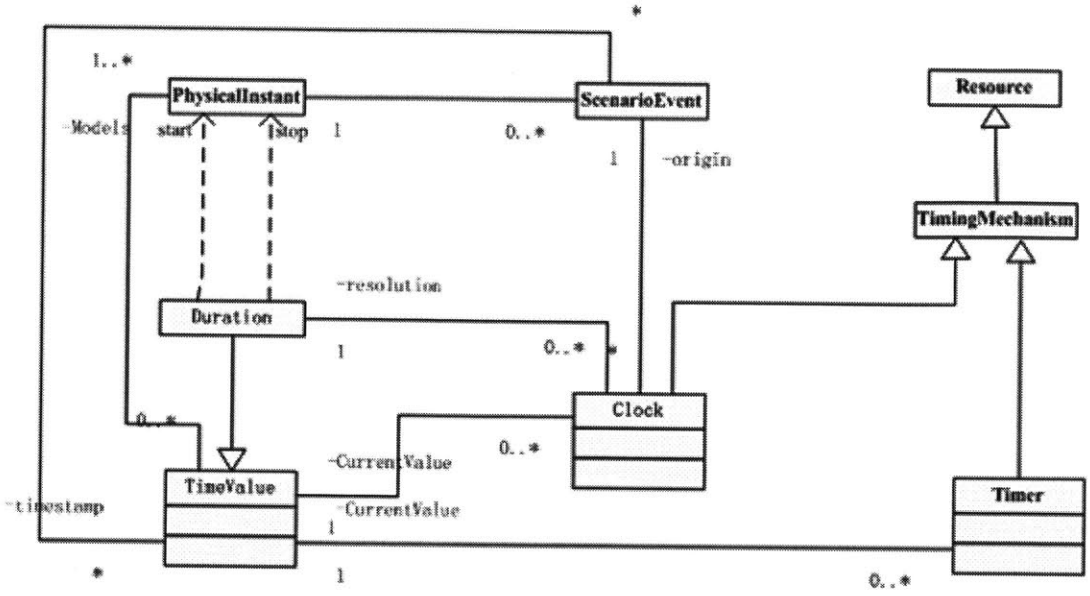

Figure 1.9

What real-time UML provides is a common language to express a design – not a design method. Several methodologies using UML have been proposed, ROPES (Rapid Object-oriented Process for Embedded Systems, Douglass, 1999) is one of them. It defines an iterative development cycle of analysis, design, implementation and testing. Use-case model is used to capture user requirements, and the identified use-cases are ordered according to priority, risk and commonality. However, some researchers argue that use-case model may not be appropriate for embedded systems, as requirements in most cases are non-functional requirements, such as response time and robustness[10]. Though UML will be used more and more widely in embedded software development,[11] the adoption of these methods remains to be seen.

1.3.2 Programming Languages

Initially, most embedded systems were developed with the assembly language of the embedded computer. This is because that assembly programs could achieve efficient implementation through direct access to the hardware, it is also because high-level programming languages were not well supported on microprocessors used in embedded systems.

As computers became more powerful, programming languages more mature, and compiler technologies progressed, the advantages of developing embedded software in a high-level language outweighed the disadvantages. Though some languages specifically for embedded systems have be designed, general purpose programming languages, such as C/C++, ADA, are more widely used in developing embedded software applications.

[10] J. Arlow, I. Neustadt, UML and Unified Process, Addison-Wesley, 2002
[11] Gartner Report, Asia/Pacific: Embedded Systems Design, Software Decisions, May 2003

The languages can be roughly grouped into two groups, sequential and concurrent languages.

Typically sequential languages include FORTRAN, C and C++. These languages are weak in the facilities for real-time control and reliability, such as thread and thread synchronization tools, and they often rely on operating system support.

In contrast, concurrent languages like ADA and JAVA have build-in support for these features. Though JAVA initially was not suitable for embedded programming, many efforts have been spent recently to develop a real-time version of JAVA.

1.4 Challenges in developing embedded software

Embedded software is rarely pure software, and it has close relationship with the underlying hardware. Whether the requirements can be achieved not only depends on the software but also the hardware capacity. Further, some minor hardware changes could simplify software implementation greatly. Therefore, the software programmers have to understand how the hardware works before making software design, and make reasonable suggestions to hardware designers. In most situations, the software is developed in parallel with hardware development, and software needs to be robust and yet flexible to cater for potential hardware changes.

Embedded software needs to be highly reliable, but it is difficult to test and debug a real-time program. Some intractable errors are usually the result of subtle interactions between processes or an unexpected combination of conditions. It is common that a system works perfectly under development environment, while works abnormally during the real testing. Such errors tend to be very difficult to reproduce in debug mode or using a simulator, and programmers usually have to check all related codes to guess the possible reasons. In extreme cases, programmers have to rewrite the whole module to resolve the problem.

It is also challenging to achieve reusability in embedded software development. To get the optimum performance, the software needs to fully take advantage of the hardware capacity and characteristics, making it tightly coupled with the specific hardware. Though it is recommended to separate the generic code from hardware related code, it is difficult to do in practice due to small memory space and the lack of good framework.

Chapter 2 China's Embedded Software Industry

2.1 China's Software Industry at Large

China's software industry has experienced fast growth in the past decade, and the annual growth rate has been above 25% over eight consecutive years. In 2002, the industry had 4,700 companies and hired 590,000 employees, and the total sales revenue reached RMB 110 billion (US$ 13.5 billion)[12]. Figure 2.1 illustrates China's software sales from 1992 to 2002.

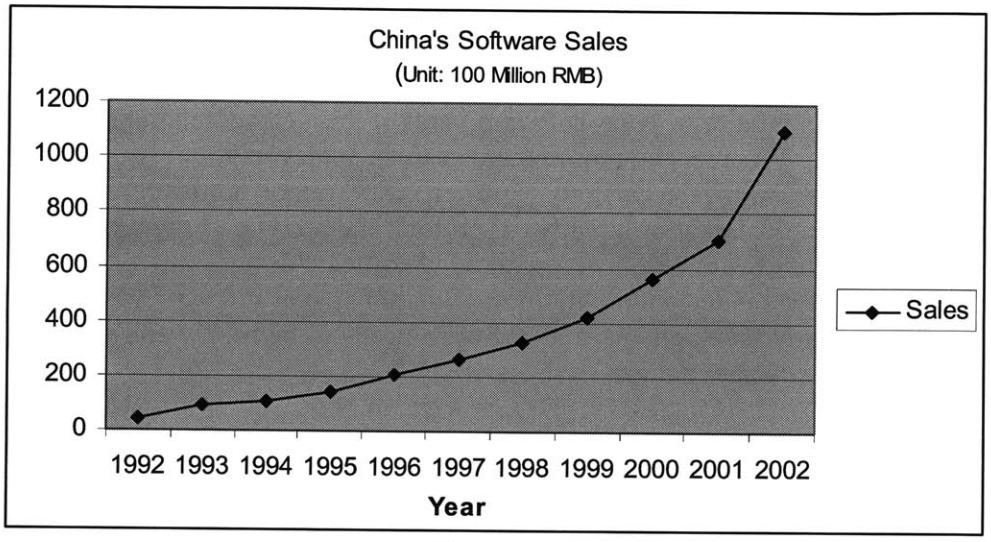

Figure 2.1

Despite the fast growth, China's software industry remains small. In 2002, it took only 2% share in global software market, whereas U.S. and west Europe took 40% and 31% respectively.

China's software industry is largely driven by its domestic needs. In 2002, the total export was US$1.5 billion, only 11% of the total sales.

2.1.1 Companies and Products

Among the 4,700 companies, small companies are the majority. 67% of companies have employees less than 50 and few companies have more than 1,000 employees. Only 214 companies have annual sales more than RMB 100 million, and 368 companies have sales more than RMB 50 million[13]. Figure 2.2 shows the company distribution according to sales and employee numbers.

[12] Ministry of Information Industry of China 2002 Annual Report
[13] MII 2003 survey

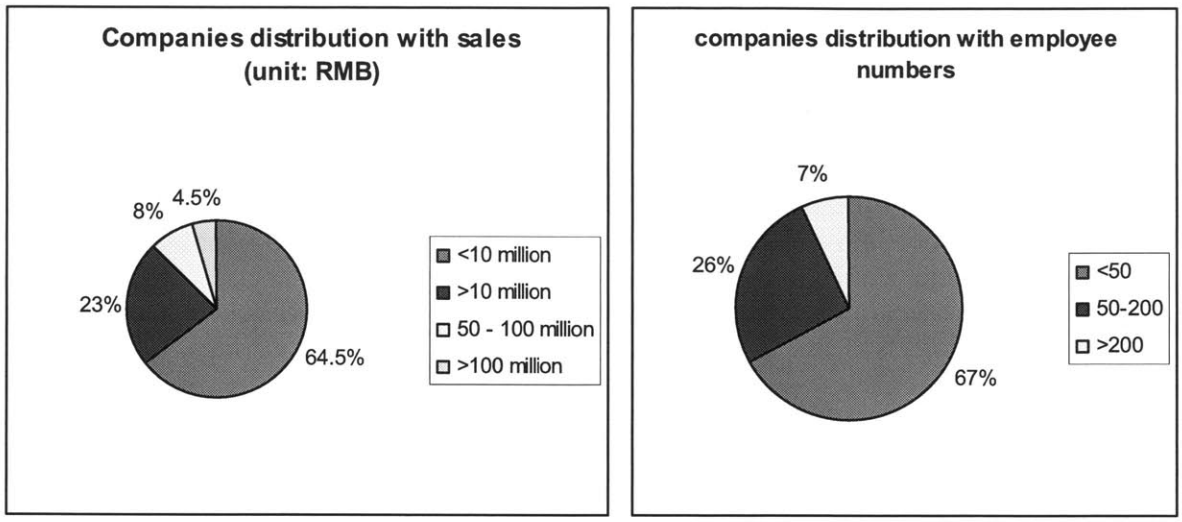

Figure 2.2

The overall software market can be divided into three segments: application software, system/tools software and middle-ware[14]. In 2002, their proportions are 64.5%, 28.9% and 6.6%, respectively.

2.1.1.1 Mid-ware products

Mid-ware products include group-ware, Web application servers and CORBA-related products. This market is dominated by foreign players. Lotus Notes is the market leader in group-ware, and IBM WebSphere and BEA WebLogic are mainstream web application servers. Some domestic companies also development some mid-ware products, but so far these products have failed to make significant impact.

2.1.1.2 System/tools products

System/tools software includes desktop and server operating software, database systems, network management software, and general development tools.

Foreign companies and products dominate this market, taking 95.3% market share in 2002. Microsoft Windows can be found on almost any PC in China. Oracle, Sybase and IBM take the database market. Borland's C++ Builder, JBuilder and Delphi, Microsoft's Visual Studio, Marcomedia's DreamWeaver are the most popular development tools in China.

Though Chinese companies currently don't have much presence at this segment, the advent of OpenSource movement may change the situation in the next several years. There are big Linux communities in China, and several indigenous companies have developed operating systems based on Linux. Meanwhile, the heavy reliance on foreign proprietary operating systems has caused the government some serious concerns with national information security. As a result, the government has started to take a pro-

[14] Embedded software market will be discussed in the next section.

indigenous-product policy in government purchasing to finance these products. Beijing municipal and Guangdong provincial government have taken the lead to adopt some Linux variants, such as Red Flag Linux.

2.1.1.3 Application software

The application software market covers a wide variety of products and services. It can roughly be divided into two sub-segments: commercial packages and IT solution projects. The former includes ERP, office applications, corporate financial, MIS, desktop publishing and security software; the latter includes numerous government and company information projects. Table 2.1 illustrates some major application markets and some key players.

Market	**Major players**
ERP	SAP, Oracle UFSoft, KingDee, Genersoft
Financial	UFSoft, New Grand KingDee, Genersoft
Office	Microsoft, KingSoft CS&S, Evermore
Publishing	Adobe Founder
Network Security	KingSoft, Rising Jiangmin, Neusoft Symantec

Table 2.1

To develop successful application products, companies need to understand China's unique cultural, economic and political systems, and regulations. Therefore, indigenous players have significant advantages, even with less advanced technologies. For example, in the ERP market, domestic products took nearly 80% market share in 2002. Table 2.2 shows the top players. Only two foreign vendors, SAP and Oracle, had significant market share (PeopleSoft had 2.0% share).

Rank	Company	Sales (US dollar):million	Market share
1	UFSoft	40.12	21.60%
2	SAP	25.08	13.50%
3	KingDee	23.22	12.50%
4	Oracle	11.89	6.40%
5	DCMS	10.40	5.60%
6	Genersoft	7.99	4.30%
7	HJSoft	7.62	4.10%
8	NeuSoft	6.69	3.60%
	Others	52.75	28.40%
	Total	185.75	

Table 2.2

Data source: CCID (China Center of Information Industry Development)

2.2 Embedded Software in China

China's booming economy created a large market for consumer electronics, digital devices and telecom equipment. China is the largest mobile phone market in the world, the number of mobile phone users reached 167 million in April 2001[15]. As the government poured huge amount of money to update its old communication infrastructure in the past ten years, China is becoming world's largest telecom equipment market. Also in the past a few years, the prevalence PCs has brought strong demands for digital devices, such printers and digital cameras.

As these products require embedded software, a big market for embedded software is forming. In 2001, China's embedded software sales exceeded $580 million, about 14.5% of total annual software sales. A survey showed that 30% of software companies were involved in embedded software development, and another 20% were considering entering the industry[16].

2.2.1 Product segments

Similar with computer software market, embedded software can be roughly divided into two segments, operating system/tools, and product specific software.

In real-time operating system (RTOS) market, VxWorks, Windows CE and embedded Linux are the most popular platforms[17], but indigenous vendors also have strong presence, and have developed and are marketing a dozen of RTOS products. Table 2.3 lists some of these products.

RTOS name	Vendor name
HOPEN	Beijing Software Engineering Center
Embedded Linux	CS&S
Red Flag Linux	Red Flag Software (Beijing)
Delta OS	Core Tek Systems (Beijing)
CC-Linux	Beijing CCOSS
XTinux	Net Tiger
Athenew	Shanghai GoTop
Start Embedded OS	Fuzhou Start
EJE-OS	Xi'an EJE
GoldLinc 927	Shenzhen GoldLinc
ZyCo	Beijing Ketai Century

Table 2.3

Data source: Embedded software development in China, Dehua Ju, 2002

[15] Embedded software Development in China, Dehua Ju, 2002
[16] Embedded software development in China, Dehua Ju, 2002
[17] Gartner 2002 Asia/Pacific survey

As open-source movement is taking roots in China. Linux is also taking off in embedded software market. According to a report from the China Center of Information Industry Development (CCID), in next three years Linux will be the preferred OS for embedded devices with 54.8 percent of the China market. According to the same CCID report, 57.2 percent of developers for embedded devices now intend to focus primarily on Linux.

In product specific software segment, telecom equipment vendors are the biggest producer of embedded software, which has become an integral part of the equipment. In fact, the top three Chinese software companies are all telecomm vendors and five out of the top ten are in telecomm industry (Beijing Eriksson and Datang are also telecomm vendors).

Table 2.4 shows the top 10 Chinese software companies.

Name	Rank	Software and service revenue (RMB, thousand)	Software revenue (RMB, thousand)
Hua Wei	1	6,040,390	6,040,390
Zhong Xing	2	3,786,160	3,546,160
Putian Dongfang	3	2,432,240	2,432,240
Shenzhou Shuma	4	2,321,230	525,400
Beijing Eriksson	5	2,088,680	2,088,680
Founder	6	1,556,410	1,556,410
Microsoft China	7	1,464,360	1,464,360
NeuSoft	8	1,421,520	653,620
CS&S	9	1,329,490	512,100
Datang	10	1,283,320	102,580

Table 2.4

Data source: MII report.

2.2.2 China vs. India in Embedded Software

In contrast to China, whose IT industry is largely driven by domestic needs, India's software business is built for export. India is the largest country for software outsourcing, and in 2002, India exported almost $10 billion software and IT services.

One reason behind the difference is that India has a much smaller domestic market. Table 2.5 shows the comparison between India with China in some IT respects.

	India	China
Per Capita Income	$400	$800
Number of Internet Connections	8 million	22 million
PC sales (2000)	$1.5 billion	$10 billion
Cell phone (per 1000)	2	34
Export	$45 billion	$180 billion

Table 2.5[18]

[18] China Report, Y. Gao, etc MT Software Business class presentation, 2003

As embedded software is usually sold as one part of other products, such as cell phones and other consumer electronics, the small domestic demands determine that India's embedded software is also export oriented. Two forms of exports exist in India: direct product export and indirect export as value-added internal transactions.

In 2000, India estimated it could export $100 million embedded software product to Japan, mostly in high-tech toys, and plant and machinery.[19] After the great success in software outsourcing, India is well-regarded as a place that produces high-quality IT products with low-costs. As the economy in US and Europe is still at the downturn, a large number of western companies consider moving basic product research and development to India a long term cost cutting strategy. For example, Cisco has a big development center with 600 employees. Intel revealed its plan to design the next version of its Xeon processor for mainframes and write some embedded software for its Centrino mobile computer chip in India.[20] As western companies shift their product R&D to India, India's embedded software grew rapidly in the past several years.

2.2.3 Leading Telecomm Vendors in China

The huge and fast growing telecomm market not only made a lot of big indigenous telecomm companies, but also attracted many foreign manufacturers to set up their facilities in China. Huawei is the largest and probably the best-known Chinese telecomm vendor, and Motorola is the largest foreign investor in China's telecomm industry.

2.2.3.1 Huawei Technologies

Huawei is a private company founded in 1988. The company came to prominence with China's rapidly expanding telecomm market in the late 1990s. Its avenue reached RMB 22 billion (US$2.68 billion) in 2002 and the projected revenue for 2003 is RMB 30 billion (US$3.66 billion). Huawei is expending to international market as well, and its products are being sold in over 40 countries. The international revenue was US$552 million in 2002, and reached US$350 million in the first half of 2003.[21]

The company's growth is driven by three areas: DSL, Software and services, and data communications. In 2002, its software and service revenue was RMB 6 billion, about 27.3% of its total revenue, and the company forecast a 32% increase in 2003.

Huawei has major domestic software development centers in Beijing, Shanghai and Shenzhen, and established its first overseas center in India in 1999. The India and Beijing center have achieved CMM4 certification.

Huawei has developed almost everything it needs to write software for its routers and switches. The company has its own real-time operating systems, communication protocol libraries and testing tools.

---

[19] http://www.expressindia.com/fe/daily/20001024/fco24003.html
[20] http://nation.ittefaq.com/artman/exec/view.cgi/8/3067
[21] Gartner report: Huawei: China's leading equipment vendor returns to growth, July 2003

2.2.3.2 Motorola China Software Group

Motorola entered China market in 1987, and since then it has set up a mobile phone factory, a semiconductor factory and 16 R&D centers. Motorola China is one of the biggest foreign companies in China, and it plans to make China its global production and R&D base and increase its annual output to US$10 billion in China by 2006.

Its software development center was founded in 1993 in Beijing. As the first practitioner of CMM, Motorola brought CMM to China. The software center has over 530 staff and has three branches in Beijing, Nanjing and Chengdu, all are at CMM 5.

Motorola's China software center focuses on developing application and system software for wireless and mobile telecom equipment. Embedded system is the core technology, as they are "the workhorse powering interactive TV, information home appliances, GPS, smart control systems, cellular phones and satellite communications."[22]

Motorola has been a champion for Linux and has ambitious Linux plans. In April 2000, the Beijing center released the first Chinese language version of an embedded Linux system (based on TurboLinux) running on Motorola's PowerPC 8240 chips.[23]

2.3 Development Process Management

Chinese companies have realized the importance of process management. One example is that the CMM is being accepted by more and more companies. Since Advanced Systems Development Company (ASDC) became the first company passed CMM certification (CMM2), the number of vendors with CMM or ISO certificates has been increasing. Among them, Neusoft, Legend, Genersoft, UFSoft and Huawei achieved CMM4[24], and Neusoft achieved CMM 5.[25]

Beside CMM, other good software engineering practices, such as Spiral model, and variants of agile programming model, are also being widely used in Chinese companies, especially in small and medium companies.

In the next three chapters, the practice in developing and managing embedded software in Motorola Beijing Design Center (BJDC), Huawei Shenzhen, and Glocom Shanghai will be presented and analyzed in details. The following table gives basic information about the three companies.

---

[22] http://www.turbolinux.com/news/pr/motorola.html
[23] http://www.csdn.net/news/newstopic/6/6461.shtml
[24] Gartner report, COM-14-9027, 9-Jan-2002
[25] http://www.pconline.com.cn/pcedu/softnews/yejie/10307/192087.html

	BJDC	Huawei	Glocom
Product	Software system embedded in mobile phones	Software system for switches and routers	Software for satellite communication terminals
Process	CMM 3	CMM4 equivalent	Home-made process
Development model	V-Model	V-Model	Iterative model
Modeling Language	SDL	UML for use cases	SDL and state chart
RTOS	Embedded Linux	Home-made RTOS	RTXC
Language	Java/C/Assemble	C/Assemble	C/Assemble

Table 2.5

The practices in the three companies are quite representative, and will give an overall picture about how embedded software development is managed in China.

Chapter 3 Embedded Software Development

- at Motorola Beijing Design Center

Motorola Beijing Design Center (BJDC), a CMM3 organization, was merged from the former Motorola Mobile Research Center and Motorola Pager Research Center in late 1990s. It hires about 300 engineers to conduct a wide variety of researches on software, electrical, mechanical and industrial design development. Its R&D focuses in 2003 include touch-screen smart phones and platform development, refreshes of GSM cellular phones, and localization of CDMA cellular phones[26].

3.1 Product

BJDC develops embedded software systems that make mobile phones function. It has developed the systems for Motorola's V60, V66, V70, and T720 series mobile phones, and it is developing the software system for A760 series.

3.2 Development process

BJDC is a CMM3 organization. Figure 3.1 illustrates the key requirements of CMM 3 and the relationships between level 3 and lower levels. To achieve a standard consistent process and product quality required by CMM 3, BDC relies on standardize requirement analysis process, historical data, a set of process management tools, and strict build and testing procedures.

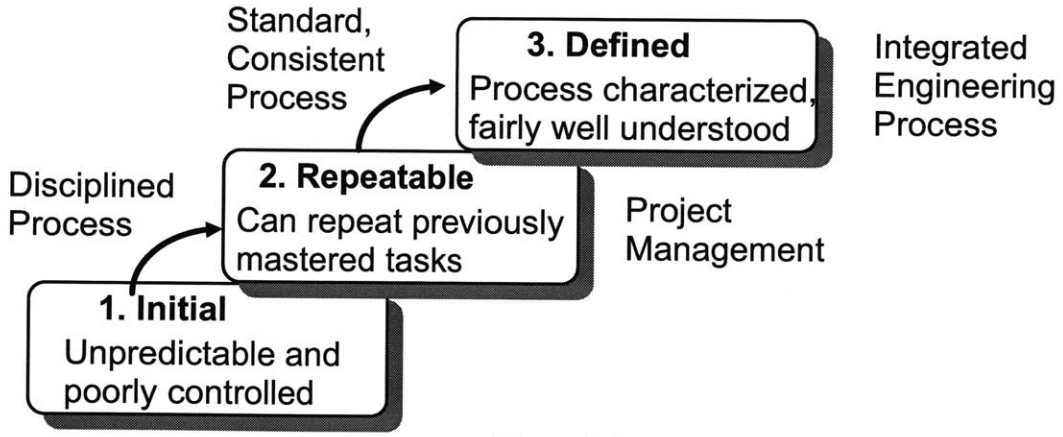

Figure 3.1

The development process follows a typical V-model. It starts at system requirement definition, followed by subsystem design and component implementation, then followed by testing and verification procedures. Figure 3.2 illustrates the process and major tasks at each stage.

[26] From Motorola China's website

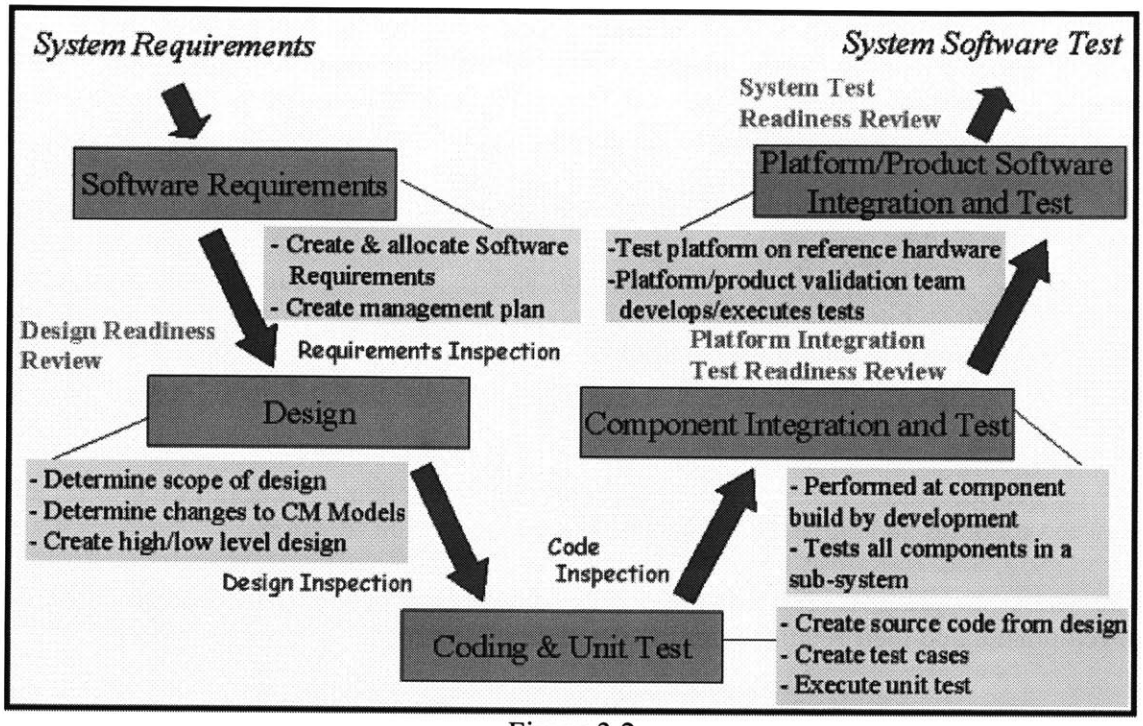

Figure 3.2

3.2.1 Requirement analysis

The marketing department is responsible for initial user requirements. After receiving the requirement, the hardware team writes hardware specification, and the software team writes a set of documents, including:

Software Requirement Specification (SRS)
Software Architecture Specification (SAS)
Software Interface Specification (SIS)

The requirements from marketing department are described in users' language, and they are usually ambiguous and sometimes conflicting. Some examples are:

- The device can function normally in suburban area where the signal is weak.
- Users want a light-weight device with large display and long talking time.
- Users want a fashionable device with a price tag around RMB 800.
- The device supports multiple Chinese character inputs, and has a large volume address book.

To provide a roadmap for product development, the initial user requirements must be translated into accurate and measurable engineering requirements.

The user requirements can be mapped to categories: hardware and software requirements. The hardware spec usually specifies the requirements on physical components, such as

the weight, dimension, and battery working time. Tradeoffs are made when user requirements are in conflict, such as battery size (weight) and talking time, price and display size. SRS, as the overall software spec, defines the general software requirements, such as user friendly interface, and data storage types. SRS also includes the real-time requirements, such as date writing/reading time.

The software system is further decomposed into modules (subsystems), for example, data-storage module, and protocol handling module. The system functions are allocated to each module. The hardware configuration and real-time constraints are considered when software architecture is designed and module functions are defined. For example, to meet the low power consumption requirement, some modules are required to have a low-power mode. And to meet the real-time response requirement, some modules are required to use multithread architecture.

SIS further describes the relationships between software modules. It defines the interface between modules and correlation sequences to realize system functions.

The three documents together provide a complete roadmap for software system development, and their relationships are illustrated in figure 3.3.

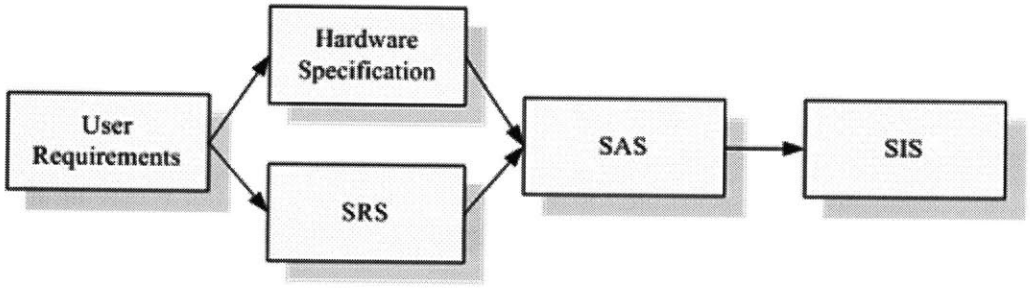

Figure 3.3

These documents will also be used later to verify the functionalities of the product.

3.2.2 Module Design

After SAS and SIS are finalized, the software team proceeds to module design. Top-down approach is used to decompose the modules to several components (classes), and to allocate module functions to them. At BJDC, a module design consists of a high-level and a low-level design. In a high-level design, SDL (Specification & Description Language) is used to describe the component functions and correlations. The output of architecture design is a set of detailed component composition diagrams and the correlation sequence diagram. Figure 3.4 shows the three components and how they work together.

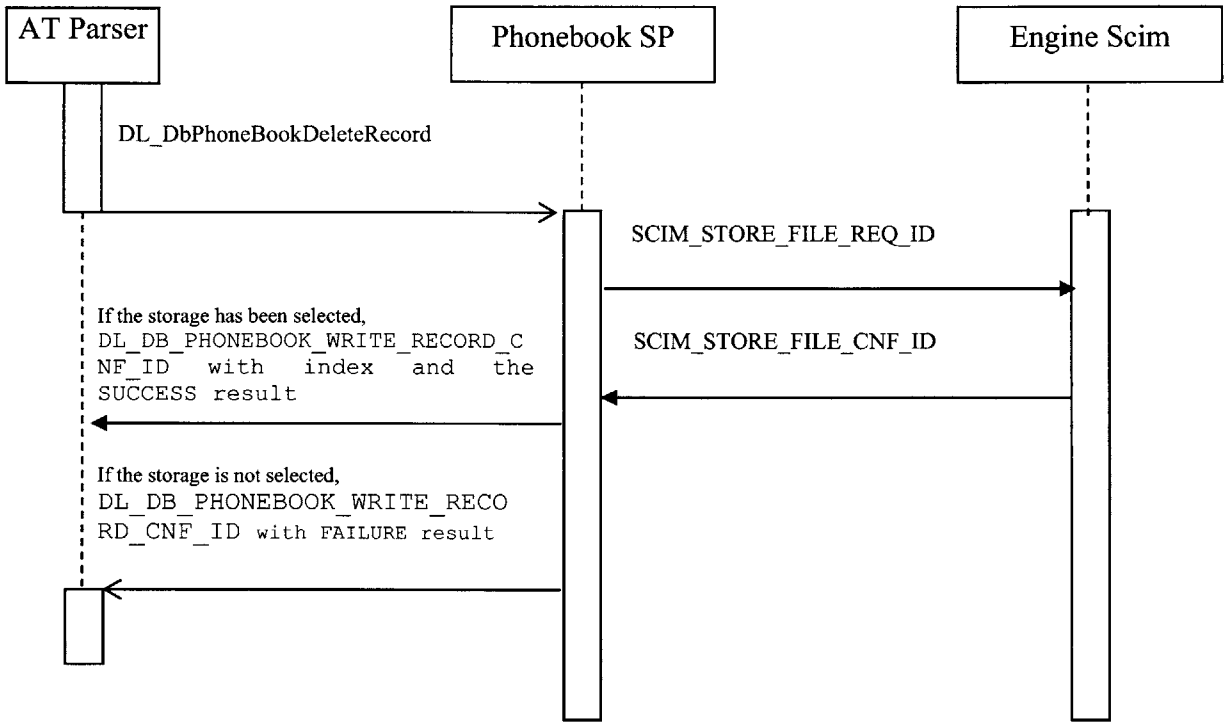

Figure 3.4

The low-level design is more concerned with the implementation logic. Usually several alternatives are proposed and evaluated. For the selected implementation, detailed pseudo code is developed to describe the logic, and the pseudo code should be detailed enough to allow direct mapping to real code. The following is the pseudo code-snap to set up communication.

```
IF is_cell_available == TRUE AND (cell is not low priority OR radio is in low power
scan OR ...) THEN
        Setup camp in this cell
ELSE
        Setup to try another cell
        IF cell_is_available == TRUE THEN
                ...
        ELSE IF ( something != 0 ) THEN
                ...
        ENDIF
ENDIF
```

3.2.3 Process management

After the detailed feature set is defined and pseudo codes are developed at the end of design process, the team proceeds to feature implementation. With the help of historical data, the feature set is further broken into individual features that can be implemented

with about 80 man hours. The related features are typically assigned to a feature team of three to eight developers depending on the complexity and urgency of the features.

The development period varies from product to product. For example, the A760 system requires about three months implementing all the features[27]. But, in general, BJDC's schedule estimation based on the historical data is quite close to the actual schedule.

As features are developed by team in parallel, proper tools and processes are required to manage the source code and to synchronize the efforts of all developers.

BJDC has an array of tools to manage the artifacts of the development process. It uses Rational's ClearCase to store and control versions for all of deliveries, including source code, documents, tools, and compiled code, and uses Clear DDTs to track Change Requests.

It also has strict source code checking rules. Before a build starts, all the developers are required to check in their changes, and the developer must test the changes locally to make sure the changes don't break the build process.

3.2.4 Testing

BJDC has strict testing procedures. In parallel with the feature development, the quality assurance (QA) team develops test cases based on the design documents. During the product development process, testing at three levels are conducted at different stage: Unit testing, component integration testing and product/platform software integration testing.

Unit testing starts after the main functions of a feature are implemented. The focus of unit testing is to verify that the designed functions are properly implemented with acceptable quality.

Component integration testing is performed after the subsystems are implemented. It focuses on the interactions between subsystems. The integration testing validates that the system can fulfill the designed functions.

The last testing, hardware/software integration testing, tests the interaction between the software and the hardware system.

BJDC has standardized bug logging and tracking procedures. The QA engineers categorize bugs into several different types according to the severity, and log the bugs with the procedures to reproduce them in a bug database. Then a QA engineer files a bug-fix-request to the development team, and the request is routed to the right developer. After fixing the problem, the developer marks the bug as "fixed by developer". The QA engineer verifies the fix, and closes the bug if the bug is fixed.

[27] The development period is changed for confidential reason.

3.3 Development Team Structure

BJDC usually uses a devoted project oriented team structure. Team size varies from project to project, but a team usually consists of a project manager, three to four team leads and about 20 developers. During critical periods, a task force may be set up to address some technical difficulties, and members of a task force usually are selected from experienced developers across teams. In such cases, the team structure has a favor of a matrix structure.

An independent SQA (Software Quality Assurance) team is responsible for testing all products BJDC develops. QA engineers don't report to any project managers.

Figure 3.5 illustrates the team structure.

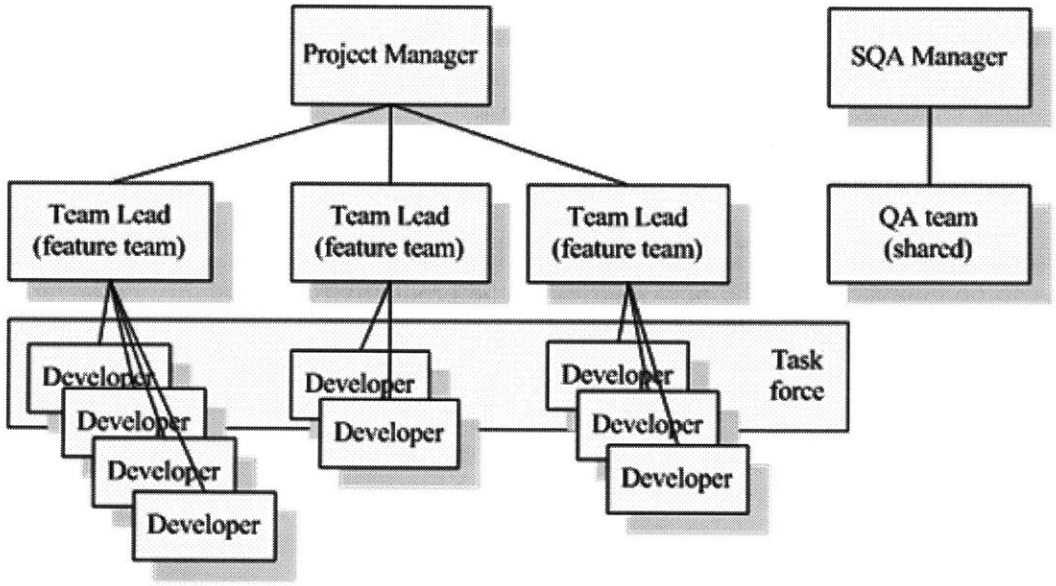

Figure 3.5

3.4 Efficiency and Lessons learned

BJDC has used CMM3 process management in several previous projects, and all the projects were finished within schedule and with expected quality. However, in this project, the process management ran into some problems, and the company did realize that some aspects need to be improved to better deal with technology risks and changing requirements.

CMM's major vehicle to make project schedule is to productivity based on historic data. To estimate the schedule of a project, the project is first broke down into modules and then ultimately small sub-modules whose workload can be estimated using historic productivity data. For example, the granularity in this project is 80 man-hours. This approach works well with products that have high similarities with previous products, but tend to be less accurate when the product is being developed in a new environment and with new tools. This is the first product BJDC has ever developed on embedded Linux platform, and the development tool (QT) is new to many developers. Because of the

changes in platform and development tool, estimations based on the historical data can be misleading, and in fact, many features that appear to be similar with those in previous products turned out to be harder and required longer time to develop. Even though some buffer time is added in the project schedule, the project was behind schedule, and both the productivity and quality were lower than expected.

BJDC uses a V-model in its development process. As a variant of waterfall model, V-model works well with projects that have clear and stable requirements before development starts. It is still largely the case in this project, but the number of requirement changes has increased a lot. (The company has a request tracking system to track the changes in requirements.) The impacts of changing requirements lie in two aspects: interruptions to the normal work process and generated rework, which tend to lower productivity and delay schedules.

Another problem is that the expected heavy workload also squeezes the time that allocated for other CMM activities, such as code review and document updates, which may cause some potential problems in later stages.

The company believes that these difficulties are caused by the lack of experience and data with the new technologies, and expects that developers could catch up once developers get familiar with the new platform and tools. But more importantly, even the project is going to be delayed, with the new metrics of the productivity and quality collected, the delay could be predicted early and accurately.

Chapter 4 Embedded Software Development

- at Glocom Shanghai

Glocom, a US company founded in 1988, develops mobile satellite terminals for Inmarsat market[28]. Its main products are terminals for Inmarsat's M, B, M4, F services.

The company is headquartered at Maryland USA, currently hires about 120 employees global wide, over 60 percent of which are hardware and software developers. Its Shanghai subsidiary, Glocom Shanghai Limited, was found in 2001, and has about 20 employees. Over the past three years, the subsidiary had developed several successful products, and become the company's product development center.

4.1 Product introduction

Glocom Shanghai recently developed the company's next generation product: Inmarsat F77 ship station, which supports voice, high speed data and Inmarsat Packet Data Service. The product consists of two parts: above deck equipment (ADE) and below deck equipment (BDE). ADE is RF (Radio Frequency) component, and BDE processes data packages according to Inmarsat communication protocols and provides human-machine interface for input and output.

BDE has a big embedded software system. It uses a Motorola PowerPC-based platform which supports various analog interfaces, such as ISDN driver, RS-232 driver, smart card interface, and so on. The software system is built on RTXC, or Real-Time eXecutive in C, a multitask real time operating system, and developed with C language.

4. 2 Development Process

The development process can be roughly divided into five phases: product requirement analysis, product architecture design, hardware and software design and implementation, and integration testing and field testing. In F77, the process (exclusive requirement analysis) took about six months.

4.2.1 Product Requirement Analysis

Inmarsat defines detailed protocols for any device working with its satellite system, and any product has to conform to the standard. Given the standard external interface, products from different vendors are highly exchangeable. To differentiate their products, during requirement analysis, engineers and marketing personnel sit together to identify attributes highly valued by some customer segments, and then create a product specification.

F77, as a ship station, needs to work in a highly humid environment with a wide range of temperatures, and hence requires high stability and robustness, which are the two most

[28] Inmarsat is the world's major global mobile satellite communications operator and provides a wide range of communications services to maritime, land-mobile, aeronautical and other users.

important factors in the specification. In addition, the specification also defines the types of service provided, number of telephone handsets supported, data interface used, price, weight, power consumption and product dimensions.

4.2.2 System Architecture Design

Based on the requirements, a pilot team consisting of experienced engineers is put up to explore the technical feasibility. Engineers usually build several hardware prototypes, compare pros and cons of each prototype, and identify a configuration that is most likely to meet the performance requirements.

To some extent, the process of system architecture design is the process of developing prototypes. There is no clear definition on when the architecture design phase should end, but it usually ends in two or three weeks when the senior engineers feel comfortable with one prototype.

After selecting the prototype, the team then draws detailed schematic diagrams for the hardware configuration. Based on the configuration, the management makes a rough estimation of the required resources and schedule. This phase focuses mainly on hardware configurations, and software system is only briefly addressed in resource preparation.

In F77, it took a team of three engineers three weeks to find out the most promising configuration, and initial estimation was eight months.

4.2.3 Software Design

After the basic configuration is decided, software and hardware design are conducted in parallel. The software system mainly consists of two parts: user interface (UI) and communication protocol module.

As the product is targeted for professional users, UI is relatively simple and straight-forward. The front panel has some buttons to take inputs and a LCD screen to display the working status of the equipment. The protocol module implements communication protocols, and are quite complex. In F77, two types of protocols are required: Inmarsat protocols, which are based on the system definition manual (SDM)[29], and internal protocols, which are defined by the company to transmit data between tasks.

Just as system architecture design is a process to find the most promising hardware configuration, the purpose of software design is find out the software architecture that best suits product requirements.

At this stage, the hardware team provides the software engineers detailed information about the preliminary hardware configuration, including CPU pin allocation, port allocation, analog driver chip data sheet and its connection to CPU. With the information,

[29] For more information about SDM, refer to "http://www.inmarsat.org/glossary.cfm?letter=S"

software engineers can experiment their ideas. For example, like many telecomm equipment, F77 has strict requirements on response time in some situations, so software developers experiment different designs and select the fastest one. Engineers also develop

There is no clear deadline for software design phase, usually it ends when software developers feel they have addressed the key difficulties. At the end of this stage, the developers also are asked to refine their schedule.

In the case of F77, this phase took about two weeks. Combined with progress in hardware design, the development schedule was also shortened from eight months to six.

4.2.4 Software Implementation

As software developers have decided their approaches and identified difficulties in the design phase, implementation is to complete the designs and make them into production code.

As the initial hardware configuration usually changes during the development period, to minimize the impact of future hardware changes, at the beginning of implementation, software developers first define some interfaces to separate software from hardware.

In F77, two sets interfaces are defined at low and high level. The low level interface is the interface between the hardware peripherals and the software. Based on the initial hardware configuration, the hardware developers develop some software drivers that provide basic functions to access the peripherals while hiding the device specific details. The high level interface wraps the drivers further and provides some convenient high level functions, such as timeout, and data buffer management.

After the interfaces are defined, software developers started to implement the user interface and protocol modules.

Because UI is simply, only one engineer is assigned to the work. The protocol module is the main work of the software implementation. In F77, four engineers work on this module and it is estimated that 90 percent of coding efforts are spent on it.

Even though developers try to separate software from hardware changes with interfaces, in the practice, it often turned out that the interfaces are not properly defined at the first place. Moreover, to achieve high performance, the communication module sometimes uses some hardware specific features. It is also common that software developers ask for some hardware changes to ease their difficulties in software implementation. The coupled relationship between software and hardware has significant impact on schedule and process management, and will be further discussed in next section.

In F77 project, it took the team about one and a half month to implement the software.

4.2.5 Quality Assurance and Testing

Because both hardware and software are in constant changes, it is critical to keep track of the changes and synchronize the codes. The team uses Microsoft SourceSafe to manage the software versions and labels code before any major changes. The developers are also asked to check in their changes in the evening and check out latest codes in the evening. During the coding process, the team make build once or twice in the first one and two month into implementation, but regular weekly build later on and daily builds before release.

The company doesn't have dedicated QA engineers, and developers are responsible for testing both their own modules and the product as a whole. Besides the cost savings, the company believes it is technically feasible without dedicated QA engineers for two reasons. First, the product is design to perform one single purpose, which doesn't involve many scenarios, so it is less a problem for engineers to cover only scenarios familiar to them during the unit test. Secondly, after system integration, the product will go through a set of standard test cases defined by Inmarsat type approval, so any problems overlooked in the early tests will be captured here.

The company is very strict on the Inmarsat testing. In F77 project, the team spent about half of product development time to pass the type approval test.

One reason that contributes to such a long testing period is that during development process, to speed up development, engineers tend follow a "good enough" principle, and "ad-hoc" approaches are frequently used to fix the problems. For example, if the response is slower than required, software developers can make it faster by changing some parameters. Without looking into the root causes, developers are actually delaying the work to the integration test.

4.3 Development Process Management

At Glocom, the development process is targeted at delivering the product faster and cheaper. In software development, it adopted early prototyping and multiple iterations. The team refines its schedule based on the prototype and adjusts resources to keep the project on schedule.

4.3.1 Characteristics of Software Development Process

The most striking characteristic is that engineers actually jump into implementation and coding very early – right after the product requirements are defined. Table 4.1 illustrates the major activities and outcomes in each stage.

As shown in the table, the initial hardware configuration is decided in product design phase, and software framework is developed at software design phase.

	Product Design	Software Design	Software Implementation
Activities	Experimenting different hardware configurations	Experimenting software designs	defining interfaces between hardware and software
	Studying feasibility	Comparing performance against goals	developing device drivers and software wrappers
			Implementing software design
Output	Selected hardware prototype	developed software framework	Complete software system
	initial estimations of schedule and required resources	Refined schedule and required resources	

Table 4.1

However, as the software implementation starts with the preliminary hardware configuration, which changes a lot during the process, the software has to change when hardware changes. To mitigate the negative impact, engineers define interfaces between hardware and software. As mentioned before, the interfaces are defined in the early stage of the both software and hardware development, and they may not capture all the needs of software components and often fail to wrap all hardware details. Consequently, the interfaces themselves also have to be expended and updated frequently. It is also common that software engineers bypass the interface and access the hardware directly to have better performance.

On the other hand, when designing the hardware architecture, the hardware engineers usually do not give enough considerations to the software components, and sometimes, the hardware configuration introduces unnecessary difficulties to software developers. In the process, software engineers often have to ask hardware engineers to make some changes.

Because of dynamic nature and mutual dependence between hardware and software, the product development requires several iterations and close cross team cooperation. Figure 4.1 illustrates the iterative nature of the process.

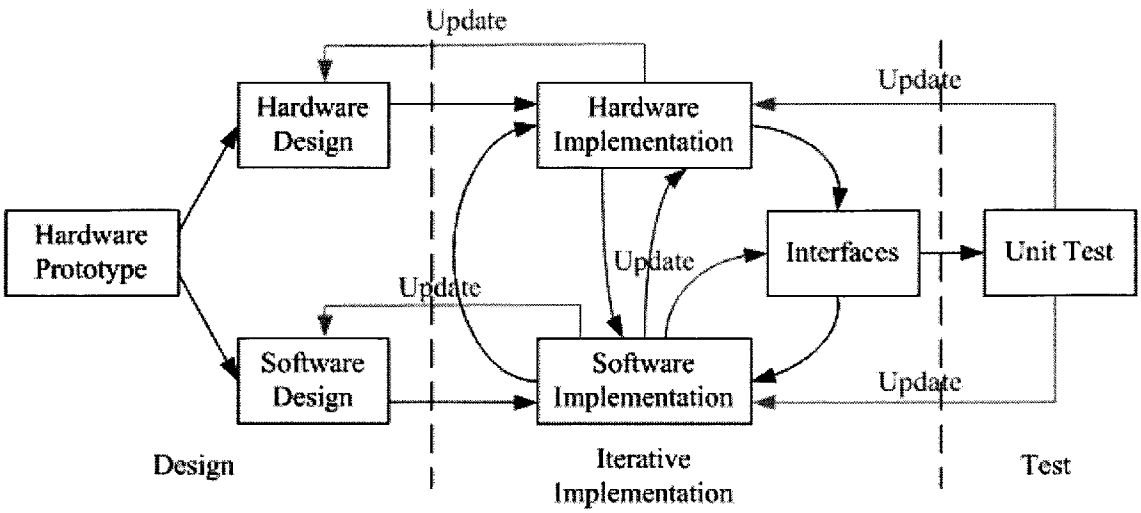

Figure 4.1

The mutual dependence between software and hardware also requires close interactions between software and hardware developers. In F77, the whole team, six hardware engineers and five software developers, work in one big room so that they can have formal and informal discussions, and the project manager also arrange regularly progress briefing to facilitate communications.

Another characteristic of the process is the lack of detailed documentation. The only detailed document is the system/module specification, and other documents, such as interface definition, system general design document, and testing document, are brief and not updated timely. To compensate the lack of detailed documents, the project manager communicates with developers frequently to understand the current designs and difficulties the team is facing.

4.3.2 Schedule Management

At Glocom, a project starts with a rough schedule, and but it gets clearer after software design, which is about five weeks into the development. In F77, the initial estimation was eight months, after system design, the schedule was refined to six months. Once the schedule is decided, the project manager will try to meet the schedule. The project manager tracks the progress through weekly meetings, regular builds and informal discussion with engineers. When a project faces potential schedule slip, the manager will bring more engineers to keep the schedule. In F77, the number of hardware engineers increased from three to five, and number of software engineers from four to six.

4.4 Efficiency and Lessons learned

Glocom believes its process management is very effective. F77 took six months from architecture design to complete, which is quite close to the scheduled development time, though the number of developers involved increased from seven to eleven. The company has not benchmarked its productivity with other companies in the same industry, but it believes its productivity is very competitive.

In the period of six months, it is estimated that 30% of time is spent on the prototypes, 20% on implementation, and 50% on integration testing.

The company admits that its process management is not without problems. First, the success heavily relies on smart and dedicated engineers. As the project manager of F77 commented that incapable developers often become the bottleneck of the development process. However, because the knowledge is not captured in documents, it is difficult to replace incapable engineers responsible for key components without hurting the schedule. Secondly, the lack of careful upfront design and reliance on incremental improvements make the development vulnerable to 90% syndrome. During the final testing of the F77, it took engineers tremendous efforts to keep the schedule on track.

Chapter 5 Managing the Development Process

- Software Quality Assurance at Huawei

5.1 Background

When Huawei was founded in 1995, its main business was low-end telephone switches. To avoid direct competition with foreign vendors, it targeted mainly at small or remote cities or countryside in China.

Though Huawei had no technology advantages compared with the leading foreign vendors, such as Nortel, Siemens and Lucent, it competed with them effectively on deep knowledge about Chinese market, flexible marketing strategies, low price, and especially excellent customer services.

As the telecommunication market expended rapidly in late 1990s, the whole company was geared up to release more products in shorter time. Without a formal development process in place, product breakdown happened frequently, and it was not uncommon for Huawei to fly service engineers to a remote county to stay weeks to fix problems.

Taking ride on the economic boom and in 1990s, Huawei grew into a company with 22,000 employees and a wide range of product line from telephone switches to network routers. The old way to fix problems is not only time-consuming and costly, but it didn't scale.

The company realized that to do it right at the first time was the best growth strategy, and to establish a suitable product development process was the key to fulfill the strategy. The process management model Huawei found out is Integrated Product Development or IPD.

5.2 Brief on IPD

IPD was developed by IBM in late 1993. The idea origins from Product and Cycle-time Excellence (PACE) model, an approach to manage product development process developed by PRTM (Pittiglio Rabin Todd & McGrath).

PACE emphasizes the importance of breaking department boundaries, and creating an empowered cross-functional team involving management, marketing, technology, product development and manufacturing at each stage of product development.

PACE model considers "the only sustainable source of product advantage is a superior product development process."[30] To achieve the goal, PACE model divides a product development process into seven elements, and introduces a collection of concepts, processes, techniques and frameworks to manage these elements efficiently.

[30] Product Development: Success Through Product and Cycle-time Excellence, M. McGrath, etc Butterworth-Heinemann 1992

The following is a brief discussion of the seven elements in PACE model.

- Decision making. New product decision making is implemented through a Phase Review Process that requires decisions at specifically defined points. A project must achieve clearly defined objectives within expected time frame in order to move to next phase. Product Approval Committee (PAC), a team consists of senior management members, reviews a project on a phase-by-phase basis, and it empowers project development team with authority, responsibility and resources to implement the next phase if the project goes through a phase.
- Project team organization. A core team is a dedicated small cross-functional project team that is empowered by PAC to manage all of the tasks associated with the development of the product.
- The structure of development activity. A structured process consists of several layers. Within the framework provided by Phase Review Process there are typically 15 – 20 major steps, and each of these steps is further broken into 10 – 30 tasks. The tasks define standard cycle times for each step so that steps can be used as building blocks for scheduling, estimating resource requirements, planning and management.
- Product strategy. The strategy is the framework used by PAC to make decisions and set priorities in the Phase Review Process, and it is the guideline for the Core Team in defining products. It includes defining the opportunities for expanding current product lines and innovating new product lines.
- Technology management. PACE focuses on the interrelationship of product development and technology. It clarifies the distinction between product development and technology development and defines the interface between them.
- Design techniques and automated development tools. PACE outlines a number of design techniques and tools, such as Quality Function Deployment (QFD), Design for Assembly (DFA), and Design for Manufacturing (DFM), and provides guidelines on applying the right technique or tool at the right time and within the context of an overall product development process.
- Cross-project management. PACE focuses on resource scheduling, business-system interface, product development process engineering and the interfaces into functional organizations.

The key benefits of PACE include:

- It significantly shortens development cycle times, and consequently reduces development costs, minimizes the possibilities of product overrun and overbudget.
- It reduces development risks and waste with a formalized phase by phase approval process.
- It leads to higher quality products with features customers really need.

IPD is IBM's implementation of PACE. It embodies all the key ideas of PACE and adds some tools and frameworks IBM developed in its practice. IBM used IPD to reform its

product development process in early 1990s, and successful turned the company around. In 1994, Huawei brought in IBM consultants to establish IPD process at Huawei.

5.3 IPD at Huawei

Huawei is a product oriented company. Depending on the market size, product complexity and development locations, the size of development teams varies from 10 to around 1,000 developers.

In the past, the development center developed products, passed on to the marketing department to sell them, and left service and support department to fix problems. As there was no product development process, the development center enjoyed the freedom to develop any products it deemed profitable.

Under IPD, a product development process is divided into product definition, product design, paralleled hardware and software development, integration and testing. Figure 5.1 illustrates the stages.

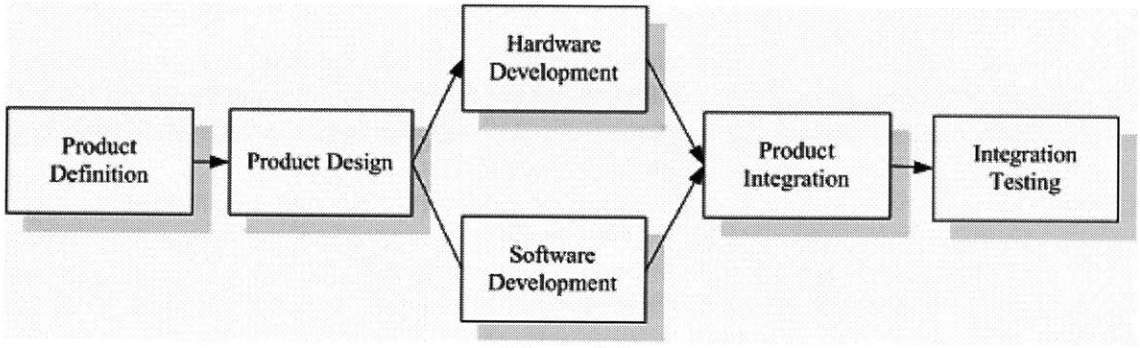

Figure 5.1

The biggest changes come from the process implementation. For example, during the product definition stage, a cross-function team of engineering, marketing and customer service is formed to define product features. Unlike before, the team must be led by a senior marketing person rather than a senior engineer. In the definition phase of the first product after IPD, service and support engineers made more 100 requirements on product maintenanability[31], which greatly surprised many developers.

Huawei's products typically have hardware and software components. The company is a highly vertically integrated manufacturer, and it has large manufacturing facilities and software development centers. In software, the company developed its real-time operation system (RTOS), low level communication protocol libraries, and high level utility libraries. All software systems for its products are developed in house, and it doesn't works with any contractors.

[31]华为 土狼向狮子的演进, IT 经理世界(CEO & CIO in Information Times) 2002/10

In product development process, the company developed its own process standard for hardware, and it adopted CMM for software.

As CMM provides only a framework, Huawei developed its company standard V-model, and updates the standard regularly.

Figure 5.2 illustrates the stages defined in Huawei's V-model. The estimated time spent on system definition and design, implementation, and testing is about 40% - 50%, 20%, 30% - 40% of the whole development period, respectively.

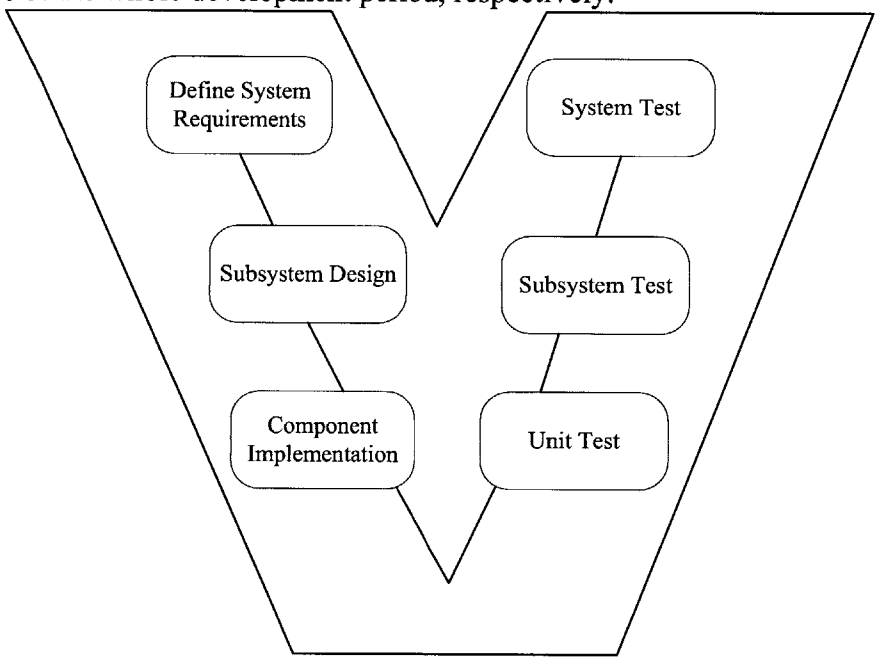

Figure 5.2

5.4 Software Quality Assurance at Huawei Shenzhen

Huawei Shenzhen is Huawei's headquarter. Though it doesn't have any CMM ranking, it doesn't lack CMM expertise and practice experience, as Huawei has a CMM5 branch, Huawei India branch, and two CMM4 branches, Huawei Beijing and Nanjing branch. At Shenzhen, the software development process follows CMM4 standard.

To promote and enforce the implementation of CMM, Huawei Shenzhen created Process Quality Department, which has a dedicated Software Quality Assurance Department (SQA) and a Hardware Quality Assurance (HQA) Department.

SQA's responsibilities include providing CMM process training, assisting project managers in practicing CMM key practice areas (KPA), guiding and enforcing developers to follow CMM process, urging project managers and developers to write/update documents, and collecting process data. SQA has the authority to suspend a project if it doesn't follow SQA guidelines.

Since 2001, SQA has been providing training programs on CMM process. All new software developers are requested to take it and others are advised to do it as well. To present, most developers have taken the training.

SQA also collects process data to establish productivity and schedule bottom lines, and defect rate standard. Productivity is calculated from the lines of code in all finished products and development period. SQA has established productivity index for five product categories: telephone switches, wireless device, wireless equipment, network data equipment, administration software. During the definition of a new product, a project manager uses the productivity index to estimate project schedule, and decide module gratuity. With the help of these data, SQA can specify time needed for each development phases and allowed variations.

SQA has about 20 engineers, and each one is responsible for three to four projects. These engineers work in development teams, but don't report to the project managers. SQA engineers get involved in whole software development process, especially the definition phase and implementation phase.

In product definition phase, a SQA engineer advises project managers on what the key areas they must go through, such as risk analysis, and proper tools used to perform the tasks. During requirement analysis, Huawei uses industry standard tools, such as context analysis, and use cases analysis. Special hardware constraints and real-time requirements are identified and captured in documents. SQA engineers also helps project managers to decompose a project to manageable feature sets and make project schedule based on historical data.

During implementation phase, SQA engineers help organize activities CMM standard requires. One important activity is code review. In embedded software development, there are many tricky but quite common problems, such as protection of shared variables, proper usage of intra-process communication mechanisms. Inexperience developers tend to fall into pitfalls, and their program works well in simple testing but often fails in a comprehensive testing. Code review is not only an effective way to eliminate these problems early, but an efficient way for knowledge transfer.

During the whole development process, SQA engineers urge project managers and developers to document their design and implementation plans and update these documents regularly.

5.5 Impact

When Huawei started to introduce IPD, it encountered strong resistance from project managers. As observed in many organizations, the initial change from CMM1 to CMM2 tends to increase, rather than reduce, the development period.[32] The first product using IPD model was forced by senior management, and it turned out to be very successful in

[32] The Capability Maturity Model for Software, M. Paulk, etc, http://www.sei.cmu.edu

terms of quality and schedule. Huawei used the project as an example to educate project managers and more and more project managers started to use the new process model. Since 2001, all products at Huawei have adopted IPD process, and when a product starts, project manager always ask SQA to send engineers to help them with process management.

Four years after implementing CMM4 based process management, Huawei's software quality improved significantly. The average defect rate is ten times lower than before, and software quality in algorithm intensive modules improved even further. After adopting CMM4 process management, the average software development time in general increased, but the improved quality dramatically reduced the testing time and improved the predictability of release date. It is estimated that most of all products are released on schedule, and delay time for late products is effectively controlled. [33]

5.6 Success factors

Establishing a product development process usually involves cultural changes, as the company moves toward an approach that uses quantitive measures to evaluate productivities and qualities, and requires a large amount of artifacts and rituals, such as documentations and phase reviews.

Huawei's success in IPD and CMM in software development largely attributed to three factors, the commitment of senior management, buy-in from mid-management and developer, and persistent guidance and enforcement.

At Huawei, the initial impetus came from the top management. The CEO once said "Huawei must establish IPD, and all employees must understand, and practice IPD, whoever disagrees shall leave the company".[34]

Top management's enthusiasm is instrumental to set up the process, but to actually practice it needs buy-in from the mid-management. In software development, Huawei achieved it using approaches. On one hand, it created a model project that demonstrated the benefits of CMM and the true values it could bring about; on the other hand, it created a dedicated SQA to help project managers mitigate the frustrations and confusions that are bound to happen in any standard implementations.

Developers' participation is also critical to the success of a process management. SQA organizes regularly activities that help developers improve code quality, such as code reviews, and it also creates templates and automated process to help developers write and update documents.

Even after the process is in place, and engineers and managers are willing to follow the process, proper enforcement and guidance are still needed to make sure that the standards

[33] Company interview

[34] 华为 土狼向狮子的演进, IT 经理世界(CEO & CIO in Information Times) 2002/10

are correctly understood and properly followed. Not all process activities are welcomed by developers. A CMM process requires developers to write detailed design and implementation documents and update them regularly. As programmers rarely have interests in writing documents, and sometimes SQA engineers have to really push hard to get them to write documents.

5.7 How much process is enough?

When time-market pressure is low, it is relatively easy to follow a standard process. However, communication equipment is a highly competitive market, and delay in key product release could bring disastrous consequences. It is a big challenge for Huawei to find a balance between a formal process and schedule. For example, in 2001, after IPD was introduced, Huawei's development on a high-end router lagged behind its competitors, the CEO demanded that all resources should be directed to the project, and project managers were authorized to take whatever measures they could do to speed up the development, and actually they did break the IPD process completely. [35]

As Huawei gained more experience in process management, it gradually established a procedure for such urgent situations. The procedure takes into consideration the relationship between quality, schedule and resource, and allows project managers to sacrifice some elements from others. For example, in a recent project, because the client pushed hard on early release, the project manager negotiated with SQA, and increased the acceptable defect rate.

[35] 华为 土狼向狮子的演进, IT 经理世界(CEO & CIO in Information Times) 2002/10

Chapter 6 China's Capability in Embedded Software

As discussed in Chapter 1, embedded software applications used to be different from conventional computer applications in several dimensions.

- Embedded software is used to control the device, and in general it is tied up with hardware devices.
- Embedded software is burned into chips and difficult to validate software logic.
- The development processes and methods are heavily influenced by slow CPU, limited memory, and real-time response constraints.

From the practices of the three Chinese companies, it appears that these differences are diminishing. Due to the fast improvements in hardware and software development tools, embedded software is more and more developed in the similar environment as computer software.

6.1 Similar Development Environment

CPU speed and memory speed and volume have improved significantly over the past decade. It is not uncommon for a device to have computing power of Intel 486. The improvement in CPUs and memory chips almost completely eliminate the need to use assembly code to squeeze the code size and increase the execution speed, and also allow developers to use high-level languages, such as C/C++, and Java. The shift from assembly to high level languages allows developers to focus on application logic, and use modern OO methodologies.

Development tools and environment for embedded software are improved too. Logic validation and debugging used to be the most difficult part of embedded software applications, but advanced simulators and testing equipment have made the jobs much easier. Programmers can debug their source code step-by-step on a simulator on a PC, and simulate potential deadlocks and race conditions. Even after the code is burned into chips, programmers still can trace the code execution with some advanced testing equipment.

Consequently, many developers think that writing code for an embedded device is not much different from writing code for PC software. First, to a great degree, the underlying hardware becomes transparent to developers. This is more obvious in large companies. For example, at Motorola, after the system architecture is defined, developers spend 80% of their time programming in a PC environment and testing with simulator running on their PCs. At Huawei, the software development and hardware development are managed as largely two independent processes after product definition and before system integration. Secondly, developers do not have many constraints when they write code with high level languages. Some companies do have general guidelines on allocating memory with C++ or Java, but the guidelines do not have big influence on program structures, and are easy to follow.

6.2 Different Process Management

Despite the apparent similarities at programming level, from a higher perspective, embedded software still has some significant differences from computer software, and these differences have strong influence on development process management.

- The fundamental purpose of embedded software is to control hardware and to provide an interface between users and hardware. The functions and scope of the embedded software for a device are usually well-defined and quite stable in the product development phase. The software development is quite a straightforward implementation process, and it doesn't need many software innovations, therefore, the workload and schedule are more predictable than computer software.
- Embedded software ultimately will be burned into chips and to upgrade the software usually requires to stop the system and to change the hardware manually. In contrast, computer software can be dynamically updated by running "patches". Further, the devices are used in various environments and required to run correctly for long periods. These impose much higher requirements on the quality of embedded software.
- Embedded software is usually a part of a tangible product, which requires time and financial resources to manufacture. Any slip in release schedule not only increases the development cost of the software but also delays the release of the whole product, which can be disastrous in a competitive market.
- Unlike computer software, which are getting more complex as computers become more powerful and users ask for more friendly user interface more functionalities, embedded software is bounded with the hardware device, and for many devices, and the software may not necessarily be getting complex.

Because of these characters, innovative software features are less important in embedded software, while Quality and Schedule are the two most important factors.

6.3 Chinese Companies' practice in Process Management

In China, large and small companies take two different approaches to take on the two issues. Large companies, represented by Motorola China and Huawei, tend to install a high CMM level, strict process management, and small firms usually use home-made, iterative process.

In companies using CMM methodologies:

- The software development process is divided into several clearly defined phases. Each phase is required to meet certain standards and generate some artifacts, such as detailed diagrams and prototypes.
- Waterfall model is widely used for its simplicity and accountability. Each team is assigned to be responsible for a phase.
- Process data are collected regularly to reflect the productivity and to guide the process.
- To prevent the process management from becoming nominal, companies usually have independent process enforcement teams; Huawei's SQA is one example.

The benefits of such an approach include accurate estimation of workload, predictable and realistic schedule, and high quality.

From the practice of Huawei and Motorola China, CMM3 and 4 are very efficient way to manage quality and development schedule. At Huawei, after adopting strict CMM4 method, 90% of its products are released on schedule and defect rate is dropped significantly. At Motorola BJDC, even though it encountered some problems after switching to a new platform, following CMM process allows it to adjust its initial schedule and predict the possible delay early and accurately.

Though CMM is efficient, it remains a heavy weight process. It requires significant investment in training and process enforcement, and developers spend a large portion of time in documenting everything rather than programming. Small companies often find they could not afford such an approach, and instead they look for light-weight process management methods. Glocom's approach is quite representative in China, and it has the following characteristics.

- Developers jump into implementation after a brief system design phase. No detailed architecture design, but external requirements are defined.
- During implementation, software developers and hardware developers work closely. Multiple iterations are required to achieve the desired performance.
- Software developers usually have some electronic engineering background, and they can make reasonable suggestions to hardware designer to ease the software.
- Developers only do simple unit testing in developing process, instead, they have a long, thorough testing before product release.

The biggest benefits are flexible and cheap. With the early workable product, problems are identified earlier and product risks are reduced as well.

Though this approach does suffer from some problems, such as 90% syndrome and long testing period, and developers have to work really hard before product release. With a capable and dedicated development team, companies still can deliver quality products fast and cheaply.

The wide adoption of CMM and creation of home-made process management symbolize that Chinese companies are getting mature in managing embedded software. As China has become an important player in global economy, China's capability in embedded software will have impacts on both Chinese and foreign companies, which will be discussed further in the next chapter.

Chapter 7 Opportunities and Challenges

Two decades after China started economic reform, boosted by strong domestic needs and export, China has become the world's factory floor for consumer electronic products and also one of the biggest markets for these products. As Chinese companies become mature in developing embedded software, China has great potentials to be global production base for sophisticated products that require both hardware and software components.

At present, most Chinese companies only sell their products to domestic market. After China's entry to WTO, it will be natural for Chinese companies to exploit international markets.

As China merges further into global economy, in the next several years, the competition centered on embedded software will have a big impact on the competitive dynamics between Chinese and foreign vendors.

7.1 Opportunities and Challenges to Chinese companies

7.1.1 Rise of Chinese Vendors

At present, China is the largest producer of commodity consumer electronic products, such as TV sets, VCD/DVD players, microwave ovens and refrigerators. The profit margins in this segment are very low. For example, it is estimated that on average a Chinese company need to sell 15 TV sets to make the same profit as Sony makes in one[36]. It is in Chinese companies' interests to move to more sophisticated products, which usually have higher profit margins. For example, some toy manufacturers are developing electronic toys that can recognize voices and have some meaning conversation with kids.

Because of China's large and cheap labor pool, Chinese companies usually enjoy much lower manufacturing costs than their western counter parts. In parallel to their move to high-end markets, some companies are leveraging the price advantage to expend overseas.

Huawei, the leading Chinese telecomm equipment vendor, is expending quickly to overseas market. It received $552 million from international contracts in 2002, and $350 million in the first six months of 2003[37].

Galanz, China's largest microwave oven manufacturer, started to export its products in 1998. In just four years, the company captured 40% of global microwave oven market (45% in European countries)[38] in 2002.

7.1.2 Obstacle to Overseas Market

However, in general, Chinese companies lack research capacities. Due to the lack of core technology, most Chinese companies have to buy key components from foreign vendors. In fact, many companies southern provinces are more like assemble plants. The lack of

[36] http://www.i-power.com.cn/ipower/knowledge/tv/071801.htm
[37] Gartner Report: Huawei: China's Leading Vendor Returns to Growth, July 2003
[38] http://www.galanz.com.cn/about/index.asp

self-owned advanced technology hampers many companies' efforts to expend overseas, and it was highlighted in a dispute over DVD loyalty fee in 2002.

China is the largest manufacturer of MPEG2 based DVD players. In 2001, it produced 26 million units, and exported 10.7 million units, which was about 25% of overseas consumption. However, the key technologies are owned by three foreign consortiums, 3C, 6C and 1C[39], and all Chinese manufacturers buy key parts, such as decoder and laser head from the companies of these consortiums. After seeing that the cheap and quality Chinese products were eroding their home market shares, they fought back with patents.

In early 2002, on requests from Phillips, who claimed that Chinese manufacturers did not pay it the due patent fee, British and Germany customs detained about 10,000 units of DVD players shipped from China. The dispute ended after several months tense negotiations, and Chinese DVD manufacturers agreed to pay $9 for each exported DVD player to the three consortiums ($5 to 3C)[40]. Considering that the price of Chinese products is below $100, the patent fee would increase product costs significantly.

The similar pattern is going to repeat in digital camera market. Chinese companies have commoditized low-end cameras (less than 2M pixels). In China, the price for 2M pixel digital cameras can be as low as $50. However, Chinese manufacturers met great difficulties to enter the high-end and high profit market (3M pixels or above). Similar with DVD technology, the key technologies of digital cameras are owned by some Japanese companies. To protect their profits, they are unwilling to sell the key parts or license the technologies to Chinese manufacturers. Further, encouraged by the DVD case, they are planning to sue Chinese companies who acquired the parts from other channels.

7.1.3 Possible solutions

To further increase the competence of their products, Chinese companies need to narrow the technology gap, and develop proper strategies to enter global markets.

To overcome the technology barrier, Chinese companies need to invest more on technology research. In 2001, the number of patents that Chinese companies had is only 1/40 of that American companies, and 1/10 of that Korean companies had, and the majority of the Chinese patents were about product designs[41]. In some high tech industries, such as next generation wireless network, where industrial standards have not been formed, Chinese companies should consider participating more actively in the standard setup processes.

The competitive strength of Chinese products is built on low price. However, once Chinese company go overseas, shipping and distribution channel costs, customer service costs, and prolonged supply chain will significantly increase the product costs. If a company wants to sell the products under its own brand, it has to spend resources and

[39] 3C consists of Phillips, Sony and Pioneer. 6C consists of Toshiba, Mitsubishi, Panasonic, Hitachi, JVC and Time-Warner. 1C is backed by Thompson.
[40] http://www.china.com.cn/chinese/zhuanti/wtobg2003/351820.htm
[41] http://news.sina.com.cn/c/2003-09-17/12061758745.shtml

time to promote its brands. To lower the costs of entry into global market, Chinese companies may consider establishing alliances with some foreign companies. Recently the joint venture between Huawei and 3Com represents the strategy. With the new joint venture, Huawei would gain access to Europe and US market through 3Com's vast sales channels, an important step toward a truly global enterprise.[42]

7.2 Opportunities and Challenges to Foreign Companies

For foreign companies, China's market is too attractive to ignore. But making money in China is becoming more and more challenging due to fierce competition from indigenous companies. A brief review on the history of foreign companies in China helps understand the current situations and their possible future strategies.

7.2.1 from Dumping Ground to Battle Field

In the early days, foreign companies simply treated China as a dump ground for their out-dated products and assemble lines. After isolated from outside world for three decades, China was lagging behind western countries in almost all aspects, and even very old products in western countries could be sold as new products for premium prices. For example, in consumer electronic appliances, Japanese companies were the first to benefit from selling TV sets, audio and video products in China. As the technologies in these early products are highly mature, Chinese companies, attracted by the high profits, started to manufacture these products with bought production lines. Compared with foreign companies, Chinese companies enjoyed cheaper labor and distribution costs, and usually had better understanding of the market needs. Consequently, the indigenous products became very competitive in both price and design, and prices dropped rapidly. But in general, foreign products dominated China's market during this period.

To foster technology transfer and indigenous manufacturers, from early 1990s, Chinese government, on one hand, exacted high taxes on imported products, and on the other hand, established regulation requirements on technology transfer and local production in many industries. To lower production costs and meet government regulations, many foreign companies started to set up joint ventures and their own subsidiaries to manufacture their products in China. In telecomm industry, the successful examples include Beijing International Switching System Corp. (BISC), a joint venture between Siemens and several Beijing based companies, Shanghai Bell, a joint venture between Shanghai government and Alcaltel, and Motorola China. These jointed ventures or subsidiaries of foreign companies also gave Chinese companies access to the advanced technologies and modern management knowledge. In the early development of China's high-tech industry, the products more or less had some similarities with some foreign products. For example, Huawei's first generation telecomm switch systems resembled Fujitsu's in many aspects.

At present, there are thousands foreign companies operating in China. In electronic and telecomm industry, nearly all big international companies have facilities in China[43]. And Chinese companies are getting mature in production development and manufacturing.

[42] http://www.chinabyte.com/homepage/21900512406693478 4/20030926/1732191.shtml
[43] http://tel.21cn.com/news/2002-12-18/875955.html

Consequently, the competition in China is very fierce, and it was no longer a sure thing that foreign companies can make profits in China, especially, in the commodity market. Whirlpool, one of the largest American home electronic appliances manufacturers, was estimated to lose about $400 million in China from 1995 to 2002, and was forced to leave China. Even successful companies are feeling the intensified competition. For example, in 2003, Motorola lost to a Chinese company in half-year mobile phone sales for the first time since it entered China in 1990s. In fact, the market share of indigenous brands has risen to 55.3%.[44]

7.2.2 Strategies to Leverage China

To compete more effectively in the increasing competitive environment, it appears that foreign companies are taking two strategies. One is to further localize their research and production to lower production costs, and the other is to leverage their technology advantages to produce high end products, which have higher technology entry barriers.

Since price has been the most effective weapon for Chinese companies, to narrow the gap between foreign and indigenous products helps increase their competence. Foreign companies have significantly increased their investments in China since 2000. The total investments are over $10 billion in the past three years ($4 billion in 2001 and 2002). Nokia has invested $1.2 billion (RMB 10 billion) in the first phase of its industrial park in China, and is planning to invest another $1.2 billion in the second phase. Motorola planned to invest $10 billion in China before 2006 to increase its R&D and production capacity in China. As China companies lack the key technologies, their products concentrate on low end products. Foreign companies have more advantages in high-end markets. For example, Motorola and Nokia recently released a wide variety of color-LCD phones and smart phones in China. Even though Motorola lost the first place in overall mobile phone sales, its high end mobile phones still have big advantages over products from Chinese companies.

Besides making profits directly in China, due to China's cheap and big talent pool, it may make a lot of economic sense for foreign companies to incorporate China into their global product development process. Denso, a big Japanese auto supplier, established Denso Create Shanghai in 2002, to develop embedded software for Denso's control systems and information system products for automobiles. "We came to Shanghai to take advantage of the many excellent software engineers based in the area"[45], and Denso is further planning to make the Shanghai subsidiary a part of its global R&D chain. This strategy applies to many small and medium companies as well. Golden River is a case of point. The company has moved its global product development base Shanghai.

7.3 Outlook of Embedded Software in China

In the past decade, embedded software has been widely used in many products, and it has penetrated into many industries which used to be pure mechanical. For example, most modern car engines have an embedded software control system to control the movements of mechanical parts and adjust their working states. It has been a trend to use software to

[44] http://www.cnii.com.cn/20030218/ca144564.htm
[45] Embedded Software Development in China, Dehua Ju, 2002

improve product performance and increase product functionalities. As Moore's law has predicted, the number of transistors per integrated circuit would double every eighteen months, the computing power will be more powerful and cheaper, therefore, in the foreseeable future, embedded software will play a more critical role in a wide variety of industries.

Because of the increasing importance of software and China's future impact, to some degree, how to manage the embedded software development in China will be critical to foreign companies regardless those focus in China market or those want to incorporate China into their global development chain.

As discussed in the previous section, embedded software emphasizes more on quality and schedule, and less on innovations. These characteristics may imply that the development could be carried out in a "software factory" fashion, in which software products are "manufactured" in a similar way as hardware products.

Japanese companies are among the first to exploit the "software factory" concepts to develop software applications for ATM, banking transactions and telecomm. In a software factory, college graduates are responsible for product design, while coders, who take only several months training in programming, implement the design. As embedded software often tends to have well defined user requirements, and doesn't require innovations, it is feasible to use such a "factory" fashion to develop embedded software in China.

To achieve high productivity, a software factory requires "workers" to be proficient in one a two programming languages and some special industrial tools. In China, programmers, who have education in computer science at colleges, tend to be generalists, and don't fit well with the software factory concept. Seeing the future demand for this type of programming "workers", many foreign IT and educational companies started IT training programs in China. In Aug, 2003, Motorola and Beijing University launched a special institute on using Metrowerks and Motorola's PowerPC chips. The institute offers one or two month training programs to help engineers learn quickly the special development environment and tools. Some India IT training companies also started their business in China, and offer various programming certificate programs.

China also has to face competitions from India, the largest outsource software development center. Because of the relative small size, embedded software was not highly valued in India. For example, an editorial in a national media commented on software outsourcing: "For India it is application outsourcing, ITO, BPO, product development, network management and contact centers. For China it is embedded software, hardware services, localization and application development."[46] However, embedded software is quickly gaining attentions in India. In NASSCOM-McKinsey Study 2002, a strategic analysis of India's IT industry, embedded software market is listed as one important growth opportunities.

[46] http://www.hinduonnet.com/thehindu/2002/10/10/stories/2002101002651800.htm

One significant advantage India enjoys is its close relationship with western companies. A large number of US and European IT companies already have big software or product development centers in India, and have successfully run them for several years. It would be natural for these companies to add their embedded software capacities to their available facilities in India.

In conclusion, China's vast market and talent pool have made it an attractive place for global embedded software development. But to make it a global factory for embedded software, China still has a long way to go.