Symbian OS ABSTRACT

Symbian OS is designed for the mobile phone environment. It addresses constraints of mobile phones by providing a framework to handle low memory situations, a power management model, and a rich software layer implementing industry standards for communications, telephony and data rendering. Even with these abundant features, Symbian OS puts no constraints on the integration of other peripheral hardware. This flexibility allows handset manufacturers to pursue innovative and original designs.

Symbian OS is proven on several platforms. It started life as the operating system for the Psion series of consumer PDA products (including Series 5mx, Revo and netBook), and various adaptations by Diamond, Oregon Scientific and Ericsson. The first dedicated mobile phone incorporating Symbian OS was the Ericsson R380 Smart phone, which incorporated a flip-open keypad to reveal a touch screen display and several connected applications. Most recently available is the Nokia 9210 Communicator, a mobile phone that has a QWERTY keyboard and color display, and is fully open to third-party applications written in Java or C++.

The five key points - small mobile devices, mass-market, intermittent wireless connectivity, diversity of products and an open platform for independent software developers - are the premises on which Symbian OS was designed and developed. This makes it distinct from any desktop, workstation or server operating system. This also makes Symbian OS different from embedded operating systems, or any of it! Competitors, which weren't designed with all these key points in mind.

Symbian is committed to open standards. Symbian OS has a POSEX-complains interface and a Sun-approved JVM, and the company is actively working with emerging standards, such as J2ME, Bluetooth, MMS, SyncML, IPv6 and WCDMA As well as its own developer support organization, books, papers and courses Symbian delivers a global network of third-party competency and training centers - the Symbian Competence Centers and Symbian Training Centers. These are specifically directed at enabling other organizations and developers to take part in this new economy. Symbian has announced and implemented a strategy that will set Symbian OS running on many advanced open mobile phones.

# CONTENTS

1 INTRODUCTION .............................................................................5

2 NEED FOR SYMBIAN OS ..............................................................8

## 3 SYMBIAN HISTORY .....................................................................14

4 SYMBIAN OS ...................................................................................16

5 COMMERCIAL BENEFITS ..........................................................18

6 FUNDAMENTAL REQUIREMENTS ..........................................20

## 7 ARCHITECTURE ...........................................................................21

8. APPLICATION ................................................................................23

9 WRITING APPLICATIONS FOR SYMBIAN OS ......................27

10 CONCLUSION .................................................................................31

11 REFERENCES .................................................................................32

INTRODUCTION

Small devices come in many shapes and sizes, each addressing distinct target markets that have different requirements. The market segment we are interested in is that of the mobile phone. The primary requirement of this market segment is that all products are great phones. This segment spans voice-centric phones with information capability to information-centric devices with voice capability. These advanced mobile phones integrate fully featured personal digital assistant (PDA) capabilities with those of a traditional mobile phone in a single unit. There are several critical factors for the need of operating systems in this market. It is important to look at the mobile phone market in isolation. It has specific needs that make it unlike markets for PCs or fixed domestic appliances. Scaling down a PC operating system, or bolting communication capabilities onto a small and basic operating system, results in too many fundamental compromises. Symbian believes that the mobile phone market has five key characteristics that make it unique, and result in the need for a specifically designed operating system:

1. Mobile phones are both small and mobile.

2) Mobile phones are ubiquitous - they target a mass-market of consumer, enterprise and professional users.

3) Mobile phones are occasionally connected - they can be used when connected to the wireless phone network, locally to other devices, or on their own.

4) Manufacturers need to differentiate their products in order to innovate and compete in a fast-evolving market.

5) The platform has to be open to enable independent technology and software vendors to develop third-party applications, technologies and services.

The way to grow the mobile phone market is to create good products - and the only way to create good products is to address each of these characteristics and ensure that technology doesn't limit functionality. Meeting the impressive growth forecast by analysts in a reasonable time frame is only possible with the right operating system.

Symbian and its licensees aim to create a mass market for advanced open mobile phones. To deliver products that satisfy mobile phone users, an operating system must be engineered to take into account key functional demands of advanced communications on 2.5G and 3G networks.

To fit into the limited amount of memory a mobile phone may have, the operating system must be compact. However, it must still provide a rich set of functionality. What is needed to power a mobile phone is not a mini-operating system but a different operating system - one that is tailored. Symbian is dedicated to mobile phones and Symbian OS has been designed to meet the sophisticated requirements of the mobile phone market that mini-operating systems can't. They simply run out of steam

The five key points - small mobile devices, mass-market, intermittent wireless connectivity, diversity of products and an open platform for independent software developers - are the premises on which Symbian OS was designed and developed. This makes it distinct from any desktop, workstation or server operating system. This also makes Symbian OS different from embedded operating systems, or any of its competitors, which weren't designed with all these key points in mind.

Symbian is committed to open standards. Symbian OS has a POSIX-compliant interface and a Sun-approved JVM, and the company is actively working with emerging standards, such as J2ME, Bluetooth, MMS, SyncML, IPv6 and WCDMA. As well as its own developer support organization, books, papers and courses, Symbian delivers a global network of third-party competency and training centers -the Symbian Competence Centers and Symbian Training Centers. These are specifically directed at enabling other organizations and developers to take part in this new economy. Symbian has announced and implemented a strategy that will see Symbian OS running on many advanced open mobile phones. Products launched, such as the Sony Ericsson P800 smart phone, the Nokia 9200 Communicator series and the NTT DoCoMo Fujitsu 2102V [2], show the diversity of mobile phones that can be created with Symbian OS. Other Symbian OS licensees include BenQ Motorola, Panasonic, Samsung, Sendo and Siemens. Over the next year, we can look forward to an even wider range of mobile phones.

NEED FOR SYMBIAN OS

This describes the key characteristics required of an operating system designed for mobile phones and explains why Symbian OS is the best-in-class mobile operating system.

1 Small and mobile, but always available

Mobile phones are both small and, by definition, mobile. This creates high user expectations. For instance, if you have your agenda on a phone that you also use to make calls and exchange data, you expect to be able to carry it with you at all times and to be instantly available whenever you want to use it. Fulfilling these expectations makes considerable demands on power management. The device needs to be responsive in all situations and cannot afford to go through a long boot sequence when it is turned on. In fact, the device should never be powered down completely since it needs to activate timed alarms or handle incoming calls. At the same time, a mobile phone must provide many hours of operation on a single charge or set of batteries. Meeting these contradictory requirements can only be done if the whole operating system is designed for efficiency.

2 Addressing the mass-market

Reliability is a major issue for mass-market phones. Data loss in a personal mobile phone causes a loss of trust between the user and the phone. A mobile phone therefore must be at least as resilient as paper diaries and agendas. Recalling phones to install service packs is a commercial and practical last resort - a mobile phone should never lock up or come with a major software defect. In fact, to use a PC term, it should never ever need a "reboot"! This is a far cry from desktop computers where bugs, crashes and reboots are expected. It may come as a surprise to many computer users that a robust and reliable operating system is perfectly achievable. Even though nobody can guarantee bug-free software, a good operating system can make it much easier to write robust and reliable applications. Reliability requires good software engineering (including object-orientation) and a good error-handling framework. Engineering best practice greatly helps reduce the number and severity of bugs while the error-handling framework enables graceful recovery from run-time errors, such as running out of memory, low battery power or dropping a communication link.

Reducing the possibility of user code making the whole system unstable goes a long way towards achieving robustness. Ideally, the kernel, with its privileged code, should be small. System servers running without special privilege should handle much of the functionality conventionally handled by device drivers.

An effective memory management system is needed to prevent memory leaks. System resources should be released as soon as they are no longer needed and an effective, easy-to-use error-handling framework should manage out-of-memory errors properly. For systems that are never completely shut down and cannot be rebooted, keeping an accurate track of resources makes the difference between peak performance at all times and slow degradation to partial, or total, lack of usability. Applications and system modules that allocate blocks of memory should cater for the possibility that none might be available. Defensive programming has to be applied from the operating system through to the application level.

However, reliability alone is not enough to make good products. Sound consumer design is also necessary, where:

1) Product applications take advantage of the mobile phone's unique characteristics as well as its environment

2) Products should be designed to meet current usability and future developments in wireless technology

1. Consistency of style is paramount - if a feature is too complex to use, then it cannot justify either the time it took to develop or the space it takes in the device.

An operating system targeted at mobile phones must support these design principals by offering a high level of integration with communication and personal information management (PIM) functionality. Symbian OS combines high functionality middleware with superior wireless communications through an integrated mailbox and the integration of Java and PIM functionality (agenda and contacts).

### 3. Handling occasional connectivity

Accessing remote data, sending email or synchronizing calendars requires some type of connection. Mobility constraints generally make a wireless connection preferable - whether wide area (using wireless telephony) or personal area (such as infrared or Bluetooth). Wireless connectivity is patchy, caused by different protocols around the world, fade-outs while moving and incomplete coverage a€" especially in remote areas, in some buildings or while airborne. It is unwise to rely on a permanent mobile connection - it is very frustrating for the user if such a connection is assumed. Wide area wireless networks are - and always will be - much slower than wired networks. An operating system must take this into account by delivering rich applications that are designed to manipulate the user's data while it is on the phone even when no connection is established. In short, the mobile phone must function as an advanced client, not a thin client, and the operating system must support this. There must be a smooth transition between being a window on the network and a self-sufficient device.

Connectivity requires an operating system with genuine multi­tasking, communications-capable real-time performance and a rich suite of communications protocols. In addition to the real-time requirements to maintain connections, the operating system must provide mechanisms to handle dropped connections gracefully and inform the user appropriately. To provide a smooth transition to the user and to be able to support forthcoming standards (such as third-generation W-CDMA and its evolution), network stacks must be abstracted in such a way that the application-level interface remains consistent no matter what type of protocol stack is used. The operating system has to provide a rich set of APIs to ensure that applications can benefit fully from current connectivity possibilities and be easily adapted to take advantage of new protocols as they are implemented.

4. Product diversity

There is an apparent contradiction between software developers who want to develop for just one popular platform and manufacturers who each want to have a range of distinctive and innovative products. The circle can be squared by separating the user interface from the core operating system. Advanced mobile phones or "Smart phones" will come in all sorts of shapes - from traditional designs resembling today's mobile phones with main input via the phone keypad, to a tablet form factor operated with a stylus, to phones with larger screens and small keyboards.

The different input mechanisms and form factors strongly influence the intended primary use of devices. With a very small screen and just a keypad, the main use tends to be voice calls. With pen input, browsing is quite convenient, but data entry is not. A keyboard is obviously the most practical mechanism to enter a large amount of data. These distinctions imply that user interfaces are ultimately both device and market dependent. Product differentiation is not just a matter of operating system design. The operating system vendor must allow its licensees freedom to innovate and develop new product lines. Whether or not a vendor allows this is a key feature of its commercial model.

To support distinct phone families and yet maximize code reuse, Symbian focuses on the common code: Symbian OS, which includes a multi-tasking multithreaded core, a user interface framework, data services enablers, application engines and integrated PIM functionality and wireless communications. Licensees are active participants in software development, creating a large development organization to extend Symbian OS. This results in thousands of developers among licensees and partners having access to source code and ensuring that Symbian OS remains an "open standard" - open and advanced. This strategy ensures that Symbian OS phone manufacturers can create highly differentiated products while sharing a technology platform and keeping the learning curve to a minimum.

#### 5 Open platforms

An operating system for the mass-market must be open for third-party development - by independent software vendors, enterprise IT departments, network operators and Symbian OS licensees. In turn, this implies a manageable learning curve, standard languages such as C++ and Java, along with SDKs, tools, documentation, books, technical support and training. Symbian OS has a rich set of APIs for independent software developers, partners and licensees to write their applications.

Even though mobile phones are small and mobile, they can offer facilities as rich as those on desktop computers, in addition to basic functions such as voice and data communication. The operating system has to support both conventional and mobile computing paradigms, and developers need knowledge of both.

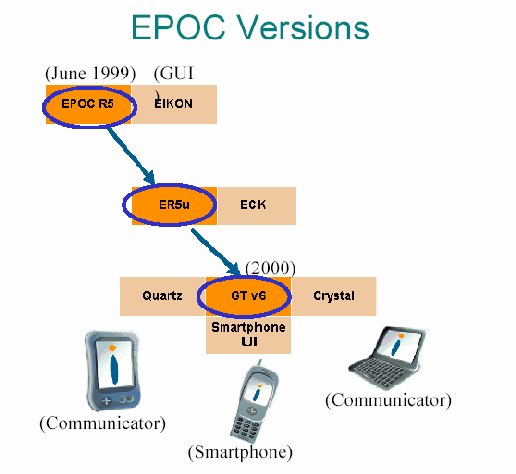
To reduce the time-to-market, developers should become proficient in as short a time as possible. Supporting standards that they may already know or can easily learn from a multitude of sources is necessary. Standards also make the platform more open and hence attract more developers.

Traditional standards such as Unicode for internationalization, a POSIX API, and Java are a must, but for an operating system to take its place in the connected world, open standards such as TCP/IP, POP3, IMAP4, SMTP, SMS, MMS, Bluetooth, OBEX, WAP, i-mode, Java and SyncML should also be supported.

Symbian has trusted leading partners in the mobile phone market and actively participates in standards organizations (such as the Open Mobile Alliance and the Java Community Process). Through these, Symbian has advance knowledge of future technologies and can test Symbian OS with many different phone systems. This ensures the stability and the future place of Symbian OS. Furthermore, a user interface framework, data service enablers and application engines provide a solid base for application developers to target.

##### SYMBIAN HISTORY

Symbian OS started life as EPOC - the operating system used for many years in Psion handheld devices. When Symbian was formed in 1998, Psion contributed EPOC into the group. EPOC was renamed Symbian OS and has been progressively updated, incorporating both voice and data telephony technologies of ever greater sophistication with every product release.



THE COMPANY:

Headquartered in London, Symbian Ltd. is owned by Ericsson, Nokia, Panasonic, Psion, Siemens and Sony-Ericsson.

CUSTOMERS:

Symbian's customers include all of its shareholders, but any company is free to license the product - Symbian OS is open to all on equal terms. So far, in addition to the shareholders, Sony, Sanyo, Kenwood and Fujitsu have all taken licenses.

BASIC PRINCIPLES:

The cornerstone of Symbian's modus operandi is to use open - agreed -standards wherever possible. Symbian is focused squarely on one part of the value chain - providing the base operating system for mobile internet devices. This enables manufacturers, networks and application developers to work together on a common platform.

SYMBIAN OS

By setting the standard for wireless value computing and telephony, Symbian brings together the wireless value chain. Symbian OS drives standards for the interoperation of data-enabled mobile phones with mobile networks, content applications and services:

1 A platform for wireless services:

Symbian delivers an advanced, open, standard operating system to its licensees. Symbian OS is flexible and scalable enough to be used in the variety of mobile phones needed to meet a wide range of user requirements. Symbian OS supports complex requirements of network protocols worldwide and enables a broad, international community.

2 Providing wireless services:

Open standards ensure global network interoperability, allowing mobile phones users to communicate with anyone, anyway, at anytime. The compelling advanced data services that operators can provide on Symbian OS phones will help minimize churn and maximize revenue.

3 Developing wireless services:

Software developers are able, for the first time, to build applications and services for a global mass market of advanced, open, programmable, mobile phones. A set of standard application programming interfaces (APIs) across all Symbian OS phones and the advanced computing and communication capabilities of Symbian OS, enable development of advanced services.

Symbian OS is a powerful aligning force for the wireless value chain. Mobile phone manufacturers, network operators and software developers are assured that they are working with an industry standard, open operating system that allows customization and is focused on the mass market, driving the wireless community.

COMMERCIAL BENEFITS

The widespread establishment of Symbian OS will bring significant commercial benefits, both direct and indirect.

Operators:

1) Operators will benefit from having a wide pool of interoperable devices, built on open standards. They will be able to select from a wide range of terminal and infrastructure manufacturers with a rich set of interoperable solutions.

1. In terms of value that operators can add, applications and content can all be made more cost effectively supplied - given the common OS shared across phones.

Developers:

* Developers will benefit from being able to target a greater number of consumers across one platform. Their porting and development costs will dramatically decline as the common OS means that applications will need to be developed once.
* Applications can be written by virtually anybody. This software could be stand-alone, used only by the user of the device. However. Just as easily, the software could be a networking application, enabling users to communicate with other users, or to access a resource somewhere in the internet.
* Equally, whilst the cost are reduced, potential returns are increased as a wider pool of users is accessible - a win-win situation for all concerned.

Indirect benefits for the whole industry:

* The above benefits assume that the number of users stays constant in establishing Symbian OS, Nokia and the other industry players believe that there will be a Metcalfe effect - whereby the value of a network is the square of the number of users. As users proliferate, they will attract more, attracting even more users and consequently, more application developers, and content. This will benefit the whole industry.

Symbian OS is the key to creation of this virtuous circle.

SYMBIAN OS; FUNDAMENTAL REQUIREMENTS

There are some fundamental requirements, which are very much essential for an OS for mobile phones.

1) It must work on stand-alone portable devices.

2) It must work on different sorts of devices.

3) It must be future proof.

4) It must be open to all to license on fair and equal terms.

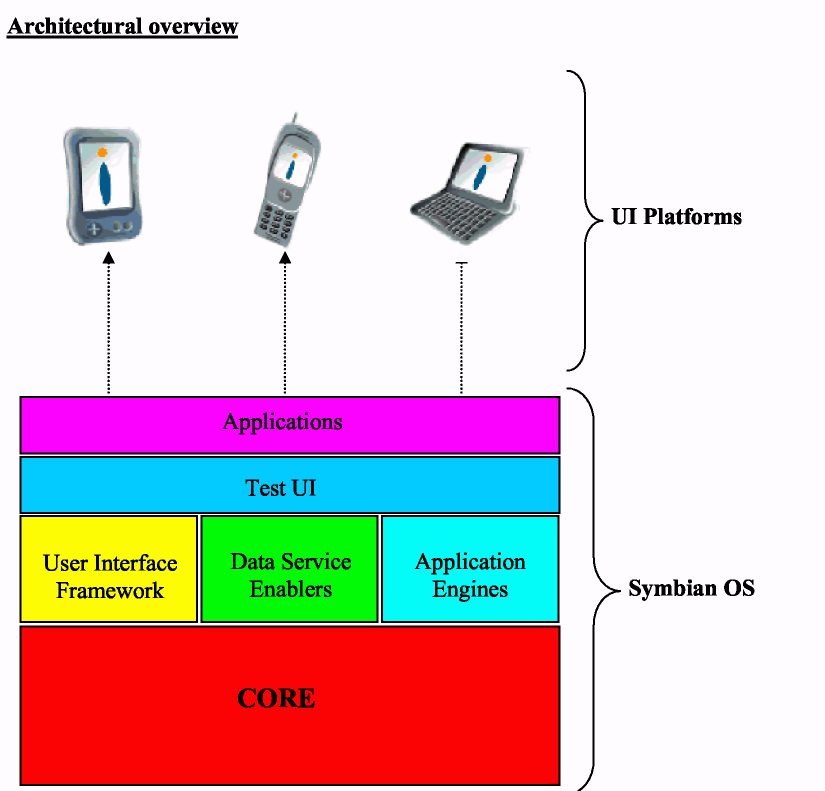
5) It must be open to all to develop applications - again with a level playing field for all.

6) It must be based on open standards.

Perhaps the most important requirement is to work on a stand-alone device. Symbian OS is fundamentally designed for mobile phones - with highly advanced features - but they must still function primarily as mobile phones. This means that expectations are already set - for a user to consider buying Symbian OS based phones they must outperform the user's current model in some areas and be at least equal in all others. The performance benchmark for Symbian OS is not the PC or portable computing devices but the phones that around one billion people already have in their pockets.

SYMBIAN OS; ARCHITECTURE

Symbian OS architecture is designed to meet a number of requirements. It must be hardware independent so it can be used on a variety of phone types, it must be extendable so it can cope with future developments, and it must be open to all to develop for.



1) Core - Symbian OS core is common to all devices, i.e. kernel, file server, memory management and device drivers. Above this core, components can be added or removed depending on the product requirements.

2) System Layer - The system layer provides communication and computing services such as TCP/IP, IMAP4, SMS and database management.

3) Application Engines - Above the System Layer sits the Application Engines, enabling software developers (be they either employed by the phone manufacturer or independent) to create user interface to data.

4) User Interface Software - USI can be made or licensed by manufacturers.

5) Applications - Applications are slotted in above the user interface.

APPLICATIONS OF SYMBIAN OS

There are many features that make Symbian OS ideal for mobile devices. Some of these are briefly explained below.

Client-Server Architecture:

The power of the client-server framework is widely acknowledged in the software community. In Symbian OS, clients are programs that have user interfaces, and servers are programs that can only be accessed via a well-defined interface from other programs. The role of a client is to serve the user, while servers ensure timely response to all the clients while controlling the access to the resources of the actual system. Additionally, in practice, one server will often have many extra servers relying on the original server.

Event Management:

Event management has long been considered core strength of Symbian OS -reflecting the fact that Symbian OS was designed from the start to have event-based time-sharing in a single thread. Rather than more conventional methods of having multi threaded applications, Symbian OS enables the developer to think in terms of interactions and behaviors as the main artifacts. Enabling this shift from procedural to interactive designs have been one of the main challenges of modern software engineering, and this is one reason why Symbian OS has earned its reputation for advanced design.

Object Oriented Design:

Because Symbian OS has an object oriented design, it is easy to configure for different sorts of hardware, and being component based, it allows manufacturers to add or remove components. This id crucial in enabling manufacturers to make devices that best suit their customers needs. This flexibility extends even to the user interface -again allowing a variety of different device designs to work from the same operating system. For Symbian itself, the design allows new technology to be slotted into an already stable platform. This will provide a stable base as the telecommunications industry moves from 2G to 2.5G to 3G to 4G, with the further introduction of new technologies such as SyncML, Bluetooth, Multimedia Messaging amongst many. The picture will grow ever more complicated, especially when technologies are used in combination, but Symbian OS is ready! For application developers, this separation of components allows them to program far richer applications - getting into the middle of the operating system.

Power Management:

Symbian OS users are used to the performance of mobile phones - and so demand similar performance in terms of weight and operating times when they adopt new devices. Power management is built into the kernel of Symbian OS and is designed to make efficient use of the processors and peripherals and so minimize power usage. When peripherals are not being used they are switched off by the system. This lowers battery consumption, prolonging usage and allows for smaller batteries.

This meets the requirement to work on stand-alone portable devices, enabling manufacturers to make phones that capture the optimum combination of size and weight for their target market.

Robust and Dependable:

Symbian OS users will have experienced the performance levels achieved in this area by mobile phones. Devices should not lose user data, crash or require rebooting.

Symbian achieves this in two ways:

1) Each process runs in a protected address space, thus it is not possible for one application to overwrite another's address space.

2) The kernel also runs in a protected address space, so that a bug in one application cannot overwrite the kernel's stack or heap.

The client-server architecture of Symbian OS allows applications to exchange data without compromising overall system integrity. This meets the requirement to work on stand alone portable devices, even though Symbian devices offer greatly enhanced functionality over standard mobile phones.

Memory Management:

For stand-alone portable devices, memory management is important. The need to minimize weight, device size and cost means the amount of memory available on a Symbian OS device is often quite limited. Symbian OS always assumes that the memory available is limited, and minimizes consumption at every turn. Consequently, less memory is actually required by the system. Also having less memory helps to keep down power consumption.

Full Multitasking:

Symbian OS runs each application as a separate process, allowing multiple applications to run concurrently. For instance, if a user is checking the calendar, and receives a call, the system must allow the user to switch between applications instantaneously. Equally, should the phone call result in an appointment, the user must be able to check the calendar - and still maintain the phone call. As phones become more data enabled, this ability will become ever more important.

An Open Operating System:

Symbian OS is an open OS. The different aspect of this statement is explained below.

1) **Open to anyone to license:**

All manufacturers are treated equally - licensing Symbian OS is open to all on fair and equal terms.

2) Open to anyone to develop applications:

The even-handed approach adopted towards manufacturers extends towards developers. API's are made available as a matter of course. Support for 3rd party developers is a key tenet of Symbian OS so full of SDKs and support are available for all products. Anyone can build an application for Symbian OS and again there is fair and equal access for all.

3) Based on open standards:

Symbian focuses on one clear part of the value chain - providing a platform for all to build upon. Consequently Symbian avoids proprietary standards. It is an active participant in many standards forums - often drawing on the expertise of its shareholders and licensees. The components of Symbian OS are based on agreed open standards.

4) Owned by the industry:

Symbian has steadily increased the number of shareholders since it was inaugurated. With the addition of Siemens as the latest shareholder, Symbian shareholders now make over 70% of the phones sold globally. This breadth of ownership ensures that Symbian acts in the interests of the whole industry, driving open standards and promoting interoperability.

###### WRITING APPLICATIONS FOR SYMBIAN OS

Symbian OS is written in C++, so it is natural fit to develop applications also in C++. This provides the developer with the most flexibility and scope. However, this flexibility brings with it complexity, and in some cases it may be more appropriate to develop an application in Java, which is also well-supported Symbian OS devices.

Symbian's use of C++ is efficient and thoroughly object-oriented. The design of the OS focuses on getting the most out of the limited hardware resources of mobile devices and this affects the way that code is written throughout the system including at the application level. This requires developers to get used to a few programming idioms that aren't common in other systems. However, these idioms help in making efficient use of the hardware resources, especially the very limited amount of memory. They also help simplify some of the more difficult tasks in application development.

Some of the idioms are:

1 The cleanup stack - A straightforward method for claiming back memory if a memory allocation fails partway through a function.

2 The rule that a C++ constructor cannot leave (i.e. cause an exception). This results in a two-phase construction system for objects (i.e. make a empty new object first, then allocate the memory in a second step), which makes the cleanup stack system keep working even for complicated class constructions.

1. Various naming conventions. E.g. C, T and R type classes, L (leaving) and non-L functions. The conventions quickly tell the developer useful information about the class or method without having to look up the definition.

Multitasking:

One of the major design decisions taken in developing Symbian OS was to optimize the system for efficient event handling from the ground up. Native Symbian OS programs are written from the viewpoint of the events that occur rather than the traditional programming model of a main control program that regularly polls for events and then performs the appropriate actions. This traditional model often requires multiple threads to be used to perform these actions and this results in the complicated problem of synchronizing access to resources.

Symbian OS multitasking system eliminates this problem by having only a single thread that responds to events as they happen. An Active Scheduler implements non-preemptive multitasking within the context of this single thread. The Active Scheduler catches the events as they occur and then runs the appropriate Active Object for that event. The Active Object does the processing for that event and then returns control to the Active Scheduler. If several events occur in quick succession, they are stored and each Active Object is run in turn. There is a priority system to determine which Active Object should be run first, but if there is an Active Object is already running it will run to completion before the next one can be run, even if the next one is of a higher priority. Thus we have multitasking that is non-preemptive. Since a Active Object function can't be preempted there is no need to use mutexers, semaphores, critical sections or any kind of synchronization to protect against the activities of other Active Objects in the thread. However, to keep the system responsive, the processing of each event must be quick so that control is returned in order for the next event to be processed. Traditional multithreading is also implemented in Symbian OS. Multiple application and servers can be run simultaneously. Threads implement preemptive multitasking, so one thread can preempt another if it has to handle an event - for instance, the window server can handle a key - press event while an application is running, by preempting the running application thread. The ability of one thread to preempt another depends on thread priority. The most critical threads in the system are given the highest priorities - with the kernel, including device drivers, the highest priority of all.

Application Architecture:

Symbian OS has an application architecture that helps developers manage the complexity of graphical user interface based applications. A Symbian OS application is made up of several parts. An Application Engine that contains all the non-UI parts of an application, an Application UI that handles the application events coming from the user and calls the Engine, and then there is the Application View (or several views) which contains the actual windows and controls (e.g. buttons, text an graphics) that show on the screen of a Symbian OS device. The Application Architecture has a built in Active Scheduler so that developers don't need to understand the ins and outs of the Active Object system when writing normal applications.

The tools that come with Symbian OS SDK can be used to generate an application with this basic structure. This provides the developer with a good guide for how to continue the development of the application.

Java:

All Symbian OS devices have Java available on them. The higher end devices tend to have Personal Java and the more popular devices have MIDP Java.

Development environment (SDKs):

The main part of an SDK are a device emulator that runs on a PC, a cross compiler for compiling software for the device and assorted tools that are required for application development. There is also a large amount of documentation and plenty of example applications in the SDK that help a developer get started with using the system.

CONCLUSION

Symbian OS is a robust multi-tasking operating system, designed specifically for real-world wireless environments and the constraints of mobile phones (including limited amount of memory). Symbian OS is natively IP-based, with fully integrated communications and messaging. It supports all the leading industry standards that will be essential for this generation of data-enabled mobile phones. Symbian OS enables a large community of developers. The open platform allows the installation of third party software to further enhance the platform.

REFERENCES

1) Professional Symbian Programming by Martin Tasker et al.

2) Nokia White paper.

3) [www.seminarsonly.com](http://www.seminarsonly.com)

4) [www.google.com](http://www.google.com)

5) chatgpt