Developing Software for Symbian OS

A Beginner's Guide to Creating Symbian OS v9 Smartphone Applications in C++

Steve Babin

Reviewed by

Antony Pranata, Bruce Carney, Chris Notton, Douglas Feather, Freddie Gjertsen, Howard Sykes, Jehad Al-Ansari, Jo Stichbury, Laura Sykes, Lucinda Barlow, Mark Jacobs, Matthew O'Donnell, Neil Hepworth, Ricky Junday, Roderick Burns, Steve Rawlings, and Warren Day

Head of Symbian Press

Freddie Gjertsen

Managing Editor

Satu McNabb

John Wiley & Sons, Ltd

Developing Software for Symbian OS

A Beginner's Guide to Creating Symbian OS v9 Smartphone Applications in C++

Developing Software for Symbian OS

A Beginner's Guide to Creating Symbian OS v9 Smartphone Applications in C++

Steve Babin

Reviewed by

Antony Pranata, Bruce Carney, Chris Notton, Douglas Feather, Freddie Gjertsen, Howard Sykes, Jehad Al-Ansari, Jo Stichbury, Laura Sykes, Lucinda Barlow, Mark Jacobs, Matthew O'Donnell, Neil Hepworth, Ricky Junday, Roderick Burns, Steve Rawlings, and Warren Day

Head of Symbian Press

Freddie Gjertsen

Managing Editor

Satu McNabb

John Wiley & Sons, Ltd

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.co.uk
Visit our Home Page on www.wileyeurope.com or www.wiley.com

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 6045 Freemont Blvd, Mississauga, Ontario, L5R 4J3, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Anniversary Logo Design: Richard J. Pacifico

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN: 978-0-470-72570-2

Typeset in 10/12pt Optima by Laserwords Private Limited, Chennai, India
Printed and bound in Great Britain by Bell & Bain, Glasgow
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Contents

Foreword

Jo Stichbury
Symbian Press

Symbian has recently reported combined cumulative licensee sales of over 145 million smartphones worldwide. And with over 120 smartphone models available, Symbian has by far the largest installed base of smartphones, at approximately 72% of the market. Calculations suggest that there are two Symbian smartphones shipping every second.

Symbian smartphone users can buy and install after-market applications, and evidence is that they are doing this in increasing numbers. Symbian Signed has recently reached a milestone, having signed over 20,000 application and content files, ranging from games and multimedia applications to enterprise and messaging utilities.

Symbian OS is a great platform for creating applications such as these, and others. New handsets are shipping with technologies such as Wi-Fi, GPS, DVB-H, HSDPA, IMS, multi-megapixel cameras, multi-GB storage, biometrics, industry-leading security, 3D hardware accelerated graphics, tilt-sensors, DNLA and uPnP (Universal Plug and Play), demand paging, VoIP and much more. . . there's scope to create some of the most imaginative smartphone applications ever.

What about the people creating these applications? Symbian has over 300 Platinum Partner companies and a community of over 55,000 developers. These are not just based in mainland Europe, but are worldwide: in Australia, Canada, Brazil, USA, India, People's Republic of China, Russian Federation, Singapore, Japan, and Thailand, to name just a few of the countries where Accredited Symbian Developers can be found.

Symbian OS offers a range of development environments for application development, from C and C++ to managed runtime environments such as Java, Python, and Ruby. The best access to the smartphone

hardware, and the best performance on Symbian OS, comes from native C++ and Software Development Kits for both major UI platforms – S60 and UIQ – are freely available. But, however available a development environment may be, it isn't necessarily easy to start work on a platform, particularly a mobile operating system that demands robust and efficient code. Newcomers to Symbian OS can be intimidated by the vocabulary and the range of information, libraries, and tools provided.

This book sets out to make it easier to get started. It is aimed primarily at C++ developers who are new to Symbian development. It makes no assumptions of knowledge about Symbian OS, or any other mobile platform, although it does assume a reasonable understanding of C++.

After introducing the Symbian platform and explaining how you can get the tools and kits you need, Steve goes through the fundamental topics needed to write an application to run on either UI platform built on Symbian OS v9 (UIQ 3 and S60 3rd Edition). This new edition of the book includes a chapter on platform security and application signing (Chapter 7), and a set of downloadable sample code for the major code examples in the book, for both UI platforms.

The original book *Developing Software for Symbian OS*, published in 2005 for versions of Symbian OS pre-dating v9, has been a very popular Symbian Press title. It was a ground-breaking book because it was our first aimed squarely at beginners. Readers appreciated Steve's practical approach to the subject, and they report having learnt far more than just the basics – it was a book packed with valuable and accessible information.

Based on the number of times we've seen the original book recommended on developer discussion forums, and our own belief in the quality of the title, we decided to ask Steve to create a new edition for Symbian OS v9. This book is the result, and I believe it is a valuable contribution to the Symbian Press series – our 23rd Symbian Press title. I anticipate it to be a useful addition to the bookshelves of many developers as they learn how to create more applications to help those millions of smartphone users to 'do more with Symbian'.

Foreword

Warren Day
Senior Technical Trainer, Symbian Software Ltd

Why read this book?

It has the best explanations I've read of the fundamental things we must understand about Symbian OS, in that it covers the 'what' and the 'how' of Symbian C++ software development.

And there are several parts to the how.

It is important to understand what facilities are provided (so we don't reinvent the wheel). Just as importantly we should know both how to *think* about these components and how to use them appropriately. A car isn't meant to be driven in just first gear; doing so would quickly empty the fuel tank and put great strain on the vehicle's internal workings.

As it is for a vehicle's transmission system, so it is also true for a software system. Misusing software components can result in everything from short battery life (so phones need frequent recharging) to inappropriate or poor functionality, and phones that are difficult to use. By having a clear insight into the fundamentals (such as memory management, string classes and asynchronous events), your software will be more efficient, because it utilizes the underlying system as intended.

We are living in exciting times. Society is evolving; being always-connected to others and the Internet is standard for many. Now smartphones have the ability to make phone calls, access the Internet, capture video, as well as perform personal organizer, word processor and spreadsheet functionality, all from a small handset. However, smartphones are very different from computers in one fundamental way, people have used phones before! The telephone is easy to use and works correctly about 100% of the time. Smartphones should be equally usable and reliable. This is the case with the other things a smartphone can do too, such as

take photographs, play video and music. People have equivalent experiences of reliability: the devices they are used to just work. Thus we now have raised standards for rich functionality, power efficiency and perhaps most importantly, reliability.

So, this book is a good 'start writing and running programs from scratch' book. It covers similar material to Symbian's OS Essentials training course and shows you how to take your first steps to writing good quality applications for Symbian smartphones as they should be written.

Biography

Steve Babin works at IBM developing enterprise software for smartphones based on Symbian OS and Microsoft Windows Mobile. He has a BSEE from Louisiana State University and over 20 years of software development and leadership experience on a variety of products – including medical devices, Java accelerators, avionics, Internet appliances, and system-on-chip silicon devices – using numerous operating systems. Steve is married to Sharon and has a daughter named Hillary. They live in Austin, TX. He is an Accredited Symbian Developer.

Author Acknowledgments

I would like to thank the people at Symbian Press for their interest in, and enthusiasm for, doing this update. It was a pleasure working with them again. I especially want to thank Jo Stichbury for her very thorough reviews and edits, as well as her excellent suggestions and advice. Special thanks also go to Satu McNabb and all the technical reviewers, including Antony Pranata, Steve Rawlings, Howard Sykes, Laura Sykes, Matthew O'Donnell, Jehad Al-Ansari, Chris Notton, Douglas Feather, Neil Hepworth, Bruce Carney, Ricky Junday, Roderick Burns, and Warren Day. Updating the book for Symbian OS v9 was quite a challenge, but I was glad to have the help of such talented people. Thanks also go to the people at John Wiley for their interest in this project, and their hard work – I can only imagine what it takes to bring a book like this all the way through the process and into the reader's hands. I also want to thank my wife Sharon and my daughter Hillary, for putting up with me on this – again.

Symbian Press Acknowledgments

Symbian Press would like to thank Steve Babin for resolutely revising his much-respected repository of knowledge. Thank you Steve, for being professional, reliable, and a pleasure to work with. We'd also like to thank each of the reviewers, and in particular Antony Pranata and Steve Rawlings, who lent us the weight of their experience on each of the UI platforms at very short notice. Thanks must also go to the Symbian Technical Training team for their feedback and support.

We couldn't have put this book together without the team at John Wiley, and it's thanks to Rosie Kemp that it happened so rapidly. Phil Northam came to the rescue in our darkest hour, and Jo has promised to buy him and Satu lunch, in recognition and with gratitude.

Symbian OS Code Conventions and Notations Used in the Book

For you to get the most out of this book, let's quickly run through the notation we use. The text is straightforward, and where we quote example code, resource files, or project definition files, they will be highlighted as follows:

```
This is example code;
```

C++ code for Symbian OS uses an established naming convention. We encourage you to follow it in order for your own code to be understood most easily by other Symbian OS developers, and because the conventions have been chosen carefully to reflect object cleanup and ownership, and make code more comprehensible. An additional benefit to using the conventions is that your code can then be tested with automatic code analysis tools, which can flag potential bugs or areas to review.

If they are unfamiliar, the best way to get used to the conventions is to look at code examples in this book, and those provided with your chosen SDK.

Capitalization

The first letter of class names is capitalized:

```
class TColor;
```

The words making up variable, class, or function names are adjoining, with the first letter of each word capitalized. Classes and functions have their initial letter capitalized while, in contrast, function parameters, local, global, and member variables have a lowercase first letter.

Apart from the first letter of each word, the rest of each word is given in lower case, including acronyms. For example:

```
void CalculateScore(TInt aCorrectAnswers, TInt aQuestionsAnswered);
class CActiveScheduler;
TInt localVariable;
CShape* iShape;
class CBbc;//Acronyms are not usually written in upper case
```

Prefixes

Member variables are prefixed with a lowercase 'i', which stands for 'instance'.

```
TInt iCount;
CBackground* iBitmap;
```

Parameters are prefixed with a lowercase 'a', which stands for 'argument'. We do not use 'an' for arguments that start with a vowel.

```
void ExampleFunction(TBool aExampleBool, const TDesC& aName);
```

(Note: `TBool aExampleBool` rather than `TBool anExampleBool`).

Local variables have no prefix:

```
TInt localVariable;
CMyClass* ptr = NULL;
```

Class names should be prefixed with the letter appropriate to their Symbian OS type (usually 'C', 'R', 'T', or 'M'):

```
class CActive;
class TParse;
class RFs;
class MCallback;
```

Constants are prefixed with 'K':

```
const TInt KMaxFilenameLength = 256;
#define KMaxFilenameLength 256
```

Enumerations are simple types, and so are prefixed with 'T'. Enumeration members are prefixed with 'E':

```
enum TWeekdays {EMonday, ETuesday, ...};
```

Suffixes

A trailing 'L' on a function name indicates that the function may leave:

```
void AllocL();
```

A trailing 'C' on a function name indicates that the function returns a pointer that has been pushed onto the cleanup stack:

```
CCylon* NewLC();
```

A trailing 'D' on a function name means that it will result in the deletion of the object referred to by the function:

```
TInt ExecuteLD(TInt aResourceId);
```

Underscores

Underscores are avoided in names except in macros (__ASSERT_DEBUG) or resource files (MENU_ITEM).

Code Layout

You'll notice that the curly bracket layout in Symbian OS code, used throughout this book, is to indent the bracket as well as the following statement:

```
void CNotifyChange::StartFilesystemMonitor()
  {// Only allow one request to be submitted at a time
   // Caller must call Cancel() before submitting another
  if (IsActive())
    {
     _LIT(KAOExamplePanic, "CNotifyChange");
    User::Panic(KAOExamplePanic, KErrInUse);
    }
  iFs.NotifyChange(ENotifyAll, iStatus, *iPath);
  SetActive(); // Mark this object active
  }
```

1

Smartphones and Symbian OS

Symbian OS is a full-featured, open, mobile operating system that powers many of today's smartphones. As these smartphones become more powerful and popular, the demand for smartphone software has grown. Symbian smartphones are shipped with a variety of useful pre-loaded and targeted applications, which are selected by each phone's manufacturer. Today, the average Symbian smartphone ships with over 30 pieces of third-party software pre-installed. However, the exciting aspect of Symbian smartphones is that they are 'open', meaning that users can further customize their phone experience by downloading, installing, and uninstalling applications written by third-party developers (or by the users themselves). Users can download applications from a PC to the smartphone through a link such as USB, or Bluetooth technology, or over-the-air via the Internet.

With the largest installed base of smartphones worldwide, Symbian OS offers a great opportunity for software developers to establish themselves in the mobile market by creating novel and exciting software for the growing mass of smartphone users around the world. There is a growing list of Symbian applications available as freeware or as paid downloads on numerous Internet sites (***http://www.handango.com*** and ***http://www.epocware.com*** are good examples). They range from productivity, entertainment, navigation, multimedia, and communications software to programs that can count fast food calories, improve your golf swing, keep diaries, and calculate foreign currency exchange. And business opportunities aside, sometimes it's just plain fun writing your own code to run on your own smartphone.

The purpose of this book is to help and inspire software developers to create good software for Symbian smartphones.

1.1 Notes on this New Edition

Developing Software for Symbian OS was first published in 2005, and in the two years since then smartphones have continued their phenomenal

growth rate. The number of Symbian OS smartphones shipped in 2006 alone was 51 million – a 52% increase from the year before. In the first half of 2007, 47.9 million smartphones were shipped (a 39% increase on the same period in 2006) and the total number of Symbian OS phones in circulation now surpasses 145 million. Smartphones now make up 9% of the total mobile market. Symbian continues to be the most widely shipped smartphone OS. According to Canalys, Symbian's share of the smartphone OS market was 72.4% in Q2 2007.[1] Many new Symbian smartphones have been introduced that run on the latest versions of Symbian OS, that is Symbian OS v9, which was a significant upgrade to previous versions of the operating system.

The main purpose of this edition is to update the original *Developing Software for Symbian OS* for Symbian OS v9. The basic programming concepts of Symbian OS have not changed, so much of the content of the core programming chapters remains. The main areas of change include covering the new Symbian OS v9 software development kits (SDKs) and development environment changes, as well as the significant addition of the platform security architecture to v9, which is used to protect the integrity of the smartphone. Since it affects various aspects of development, platform security is discussed in various places throughout this book as needed, and a new chapter has been added to the book to discuss this subject in depth.

In addition to updating the book for Symbian OS v9, we have made sure the book is updated in general for new developments that have occurred since the original book, and we have fixed a few errata reported against the original.

Before launching into programming for Symbian OS, this chapter introduces the smartphone itself and gives an overview of its features and associated technologies. Understanding the smartphone's range of features helps you as a programmer to exploit these features to their full potential. (For more information about the typical features and design of a smartphone, please consult *How Smartphones Work*, published by Symbian Press in 2006.)

I'll also discuss the company Symbian Ltd, give an introduction to Symbian OS, and discuss how Symbian OS, as well as other operating systems, fit into the marketplace.

1.2 Smartphone Concepts

A mobile phone that fits in your pocket and lets you communicate from and to anywhere in the world is an amazing invention. Like most inventions, mobile phones are built on a chain of prior technological advancements. Without advancements such as integrated circuits,

[1] ***http://www.canalys.com***

microprocessors, semiconductor miniaturization, battery technology and, of course, the invention of telephone and radio, the modern cell phone would not be possible.

Smartphones combine the mobile phone with another stream of technology: the computer, which adds the 'smart' in smartphone. Computers have progressed from centralized mainframes to personal computers with user-downloadable applications and graphical user interfaces. With the introduction of the Internet and email, the PC is a part of everyday life as a productivity, entertainment, and communication device. Laptops were introduced to allow PCs to be portable. Then came the mobile computing device known as the PDA – a true handheld computer.

Since the PDA and the cell phone are both mobile devices, it's only natural that we would want to combine them into one device. After all, you only have so much pocket and/or purse space! This is the basic idea of a smartphone – but a smartphone is more than just a PDA combined with a cell phone. Smartphones also contain features such as a digital camera, video and music players, and GPS, thus combining other portable devices as well.

1.3 Smartphone Features

Like PDAs, smartphones can run applications such as organizers, games, and communications programs (e.g., email, browser). They can, of course, also make telephone calls! The smartphone's goal, however, is not just to limit the number of devices you carry, but also to combine mobile phone and computing technologies in a synergistic way. A simple example is the ability to pull up a person's contact information or even picture, hit a button and automatically dial the person's phone number. Other examples include taking a picture, adding some text, and sending it instantly to a PC or another smartphone user. There are many more examples of this – and certainly many that have not even been thought of yet.

1.3.1 How Smartphones Communicate

Smartphones, like traditional cell phones, use radio to communicate with base towers, which in turn act as gateways into landline-based communication infrastructures. While traditional cell phone systems are based mainly on relaying voice communication between the wireless handset and the wired telephone infrastructure, smartphones provide more features that rely on network data transfer. After all, the basic concept of the smartphone is to combine a mobile phone with a networked PDA. Improving data transfer is the current challenge for next generation mobile communications; unlike voice transfer, which requires a fixed bandwidth, the rule for data transfer is *the faster the better*.

Generations of mobile communication

With faster data speeds come better services. For example, when the bandwidth reaches a certain threshold, applications and services that transfer real-time audio and video become possible. The industry goals in wireless data communications have been categorized into generations – each generation includes a target data bandwidth as well as a set of data services available for it:

First Generation (1G)
Original analog cell phone technology.

Second Generation (2G)
Voice-centric digital systems with increased coverage and capacity. Introduces messaging.

Third Generation Transitional (2.5G)
Stepping stone to 3G. Introduces always-on network connections, bandwidths up to 170 Kbps, allowing better Internet browsing, email, and some audio video. GPRS has been the dominant technology here.

Third Generation (3G)/Fourth Generation (4G)
Supports bandwidths up to 2 Mbps and 200 Mbps, respectively, for high-end services such as video teleconferencing.

The topic of wireless communication protocols is vast and could easily take up another book. But let's briefly cover some of the key communication technologies that apply to smartphones.

GSM

GSM, short for Global System for Mobile Communication, is a digital cell-based communication service that started in Europe, and has quickly spread throughout most of the world. A notable exception is the USA, where CDMA is the dominant standard; however, GSM is gaining popularity there. GSM is the most supported protocol in smartphones.

GSM was designed for circuit-switched voice communication. Circuit-switched means that fixed bandwidth is reserved for each direction of a phone call for the entire duration of the voice call, whether you are talking or not. Although originally designed for voice, GSM now has a variety of higher bandwidth data services (e.g., GPRS and EDGE) available, running on top of the base GSM protocol. This allows for faster data transfer, as we will see shortly.

The following types of GSM exist, each using its own band in the frequency spectrum: GSM 850, GSM 900, GSM 1800, and GSM 1900. The number indicates the frequency band, in MHz, that the protocol uses. Cell phones supporting GSM 900 and GSM 1800 will ensure coverage in Europe and many other areas outside the USA, while GSM 850 and GSM 1900 are used in the USA (mostly GSM 1900).

Fortunately, smartphones support multiple bands to ensure as wide a coverage as possible. It's common to have tri-band phones that support GSM 900, GSM 1800, and GSM 1900 to ensure maximum international coverage – although some still offer separate US models to reduce costs.

A GSM phone uses a Subscriber Identification Module (SIM) to gain access to the GSM network. A SIM contains all the pertinent information regarding a user's account, including the services allowed. It is used to identify the user to the GSM network for billing purposes. The user can switch their SIM from one GSM phone to another, provided that the phone is either not locked to a specific carrier, or locked to the carrier that the SIM is associated with.

CDMA

CDMA, which stands for Code Division Multiple Access, is a cell phone standard that competes with GSM. CDMA currently dominates in the USA and Korea, while GSM dominates virtually everywhere else. CDMA supports a high-speed data mode called CDMA2000 1xRTT, which tends to hover around 50–70 Kbps, bursting up to 144 Kbps.

EV-DO is the high-speed, 3G version of CDMA. EV-DO supports rates up to 2.4 Mbps (with actual speeds averaging closer to 1 Mbps) and is adopted by many services including Verizon and Sprint in the United States.

CSD

CSD, short for Circuit Switched Data, is the most basic mode of transferring data over a circuit-switched connection like GSM. The connection is established by dialing the number of an ISP, in the same manner that a dialup connection is started on a land-based telephone line using a PC modem. With CSD, you do not need any extra data plan like GPRS to send data. You can use up your existing voice minutes.

There are two major disadvantages to using CSD, however. First, it takes a long time to connect since this involves dialing a number and waiting for the server to answer the call. Second, it is slow; data transfer speed is only about 9.6 Kbps.

In GSM-based smartphones, this mode is referred to as 'Dial' or simply as GSM data. Earlier smartphones such as the Nokia 9290 rely entirely on this mode of data communication.

GPRS

GPRS, short for General Packet Radio Service, is a wireless technology that allows the smartphone user to quickly connect to the network and obtain good data rates. Connection time is fast since GPRS does not require any dialing (as CSD does), and the smartphone feels as if it is always connected.

Protocol-wise, GPRS runs on top of GSM. While GSM alone is circuit-switched, GPRS is based on packet-switching technology. This means that the radio bandwidth is used only when data is actually transferred, even though you are constantly connected (circuit switching keeps the full bandwidth reserved throughout a connection).

GPRS, in theory, supports bandwidths up to 170 Kbps. In practice, however, you'll get between 20 and 60 Kbps, depending on network conditions – but this is still significantly faster than the GSM dialup data rate! The best way to think of the speed of GPRS is that it matches approximately with a PC connected to the network via a wired telephone modem. However, GPRS can feel better than dialup since it connects almost instantly to the network without the lengthy delay in dialing a number and establishing a call.

GPRS is categorized as a 2.5G technology due to its speed. Although many networks now support higher-speed protocols, GPRS was an excellent stepping stone and preview for the now-available 3G technologies.

HSCSD

HSCSD is the high-speed version of CSD. HSCSD is another 2.5G standard that supplies a comparable speed to that of GPRS (although on the lower side in many cases), but with a significant difference – the bandwidth is reserved to the smartphone throughout the connection. This is because HSCSD, like CSD and GSM, is a circuit-switched technology. This makes HSCSD better suited for applications that require a constant bit rate, although the practical bandwidth is rather low for good real-time multimedia transfers, and these are the transfers that would benefit the most from constant bit rates.

HSCSD is not widely used due to the higher costs to implement. The Nokia 6600 and the Motorola A920 are examples of smartphones that support HSCSD.

EDGE

EDGE, short for Enhanced Data Rates for GSM Evolution, is a GSM-based protocol that provides theoretical speeds up to 384 Kbps. It is a 2.5G technology that is sometimes referred to as 3G because of its higher speed. It is not yet as widely used as GPRS, but is gaining support. For example, AT&T has deployed EDGE on its GSM networks in the USA, reaching speeds of around 90 Kbps in practice. Most modern GSM smartphones now support EDGE.

UMTS

UMTS, short for Universal Mobile Telecommunication Services, is a high-speed data transfer which supports bandwidths up 2 Mbps. This

protocol is the basis of third generation mobile communications that make many media-rich services a possibility. UMTS is not based on GSM technology – it uses a technology called W-CDMA. However, the UMTS platform is designed to work with GSM systems to ease its deployment. It's exciting that many service providers now have this high-speed service.

HSDPA

HSDPA, short for High-Speed Data Packet Access, is based on UMTS and supports even higher speeds than UMTS (up to 3.6 Mbps). It is known as a 3.5G technology. Many service providers have launched HSDPA support recently.

Wi-Fi

Wi-Fi is a popular communication protocol used to connect devices such as PCs, game consoles, PDAs, and mobile phones to a network wirelessly. Wi-Fi service is found not just at corporations for their employees to access their network, but also for Internet users at coffee shops, hotels, book stores, airports, and even parks.

Wi-Fi is a high-speed protocol when compared with cell tower-based protocols – it supports speeds up to 54 Mbps. Wi-Fi is a much shorter-range protocol when compared with cell-based radio protocols, with its range being about 600 ft.

In the last couple of years, numerous smartphones have been released with Wi-Fi capability. This allows for high-speed data transfer when in range of a Wi-Fi 'hotspot'.

A standard called WiMAX, which is in its beginning stages now, promises to extend the range of Wi-Fi to 2–3 miles for more global area coverage.

1.3.2 Smartphone Messaging

Text messaging, such as email and instant messaging, is widely used on PCs connected to the Internet. It makes sense that similar modes of communication be used in mobile devices. Below are the messaging features supported by smartphones.

SMS

SMS stands for Short Messaging Service. SMS allows mobile phone users to send and receive short text messages up to 160 characters. These messages are sent between phones with only a small delay and can occur even while a voice call is in progress. SMS is well suited for many types of communication exchange, and is less intrusive than making a voice

call. SMS is a part of the GSM communication platform and used by cell phones all over the world. SMS is not yet widely used in the United States, but is slowly growing in popularity. SMS is a standard feature on today's smartphones.

MMS

MMS, short for Multimedia Messaging Service, is an extension of SMS that provides the ability to send media data such as pictures, audio, and even video along with your text message. MMS is a natural complement to smartphones due to its need for, and use of, audio and video capability and, in many cases, an attached video camera. For example, a smartphone user could snap a picture of a landmark, record a quick voice comment on it, and send it instantly to another mobile phone user.

MMS messages can even be sent to people who have only SMS capability by sending a text link to a browser URL containing the MMS message. You can also send and receive MMS messages between a smartphone and an email account used from a PC.

Email

Having the ability to keep up with your email while on the road is a standard feature found in smartphones. With the high-resolution scrollable displays and alphanumeric entry methods, it does not feel much different from email on a PC. Smartphones allow the user to set up multiple POP3 and IMAP email accounts.

Fax

Many smartphones include the ability to send and receive faxes, or can be customized to do so with fax software.

1.3.3 Web Browsing

Internet browsing is a standard feature for smartphones. There are many different browsers available, and they fall into two main types: WAP and HTML.

WAP

WAP, which stands for Wireless Application Protocol, was specifically designed for Internet browsing on resource-constrained devices. It includes lightweight markup languages designed to minimize the processing power and memory needed by the mobile device to render the web page. WAP also ensures that the page is useable on a small screen. Markup languages include WML and xHTML (mobile profile).

In many cases, proxy servers are used, which will automatically translate traditional HTML websites to the WAP markup language before transferring to the mobile device. This is known as *transcoding*.

HTML

Although WAP was considered important for earlier mobile devices, the smartphones today have better memory, processing power, and displays. Because of this, it is feasible to include traditional HTML browsers that load websites directly in their native format, similar to a browser on a PC. Many smartphones have HTML browsers and those usually include WAP capability as well – sometimes combined in one browser.

1.3.4 Local Device Communication Features

Smartphones have a variety of communication features in addition to basic access to the cellular network. These features allow a smartphone to link directly with other devices, including PCs, PDAs, wireless headsets, and other smartphones to undertake a wide variety of data transfer functions. Below are the popular device-to-device communication mediums, along with some of their uses.

USB/Serial cable connection

Smartphones can be connected to a PC via either a USB or serial cable (varies from phone to phone). This high-speed link is normally used for downloading new applications to the smartphone as well as synchronizing user data, such as calendar and contact entries. Many products provide a cradle into which the smartphone can be plugged, for both PC connectivity and for charging the phone's battery.

Infrared (IR)

The smartphone provides the capability to communicate through an infrared port to a PC or other device such as a PDA or phone. You can do all the things that can be done with the USB/serial cable, but without plugging in any wires. IR requires a line-of-sight connection between the devices in the same way that a TV remote control does.

Bluetooth technology

Bluetooth technology is a short-range radio technology that enables devices to find and connect to each other and communicate. While technologies like GSM replace long lengths of wire, Bluetooth technology replaces the rat's nest of short wires connecting various pieces of

equipment. Unlike infrared, Bluetooth technology does not require a line of sight and will even communicate through walls.

With Bluetooth technology, you can connect more conveniently to PCs and PDAs than you can with cable and IR to download applications and synchronize user data. In addition to providing basic PC-to-smartphone linkage, Bluetooth technology makes more device-to-device communication scenarios possible. For instance, you can snap a picture on your smartphone and send it to a nearby printer for printing or connect to a wireless headset for hands-free operation.

Some smartphones allow themselves to be used as a modem, with access to the cellular network. In this case, a device such as a PC connects to the smartphone via Bluetooth technology, cable, or IR to provide the PC with Internet connectivity.

As more devices become available with Bluetooth technology, expect many new possibilities for Bluetooth-enabled smartphones.

1.3.5 Location Based Services

More smartphones are being equipped with features that allow a device to know its physical location. In some cases – for example, the Nokia N95 – Global Positioning Satellite (GPS) hardware is built into the phone. Alternatively, the phone may use the cell service's location information (e.g., the GSM cell id can tell which particular cell tower you are communicating with), which is being increasingly supported for emergency purposes.

Location Based Services (LBS) uses the device's location to provide a personalized experience to the user. The most obvious (and useful!) is the traditional map-based turn-by-turn navigation service. Other interesting applications are:

- Software that shows the hotels, stores, theaters, and other businesses in your immediate vicinity along with any special discounts or other deals offered. Weather and traffic can also be reported for your vicinity.
- Software that allows you to track your children who are carrying LBS-enabled smartphones.
- Software to help with health and fitness by tracking how far and fast you are walking or jogging.
- Games that use your location as input. For example, Blister Entertainment has a fishing game called Swordfish where you find schools of fish based on your phone's real location (launched in North America on Sprint Nextel, Bell Mobility, and Boost Mobile). Once you find them, you can catch them, arcade style. There are a few other LBS games around, and could be many more to come.

LBS is at a very early stage for smartphones at the time of this writing, but its future looks promising. Reference ***http://www.lbszone.com*** for more details on LBS.

1.3.6 Mobile TV

As smartphones get more powerful, have better displays and faster connections, mobile TV video is becoming a reality for these devices. Mobile TV content can consist of live television shows, news and sports, as well as movies on demand and music videos.

Currently, most mobile TV services use the existing cellular network for transferring video data using both 2G and 3G transport protocols. 3G is, of course, the better transport since a good TV picture requires a lot of bandwidth.

However, to realize its full benefits, mobile TV needs a dedicated video communication protocol, and the most promising protocol for this is *DVB-H*. DVB-H is an adaptation of the standard digital TV DVB-T protocol to make it viable for handheld devices. Mobile TV services on a smartphone would use a combination of DVB-H and the 2G/3G network. Although not common yet, some carriers have launched DVB-H mobile TV services (Vodafone in Italy is an example). The Nokia N92 and Nokia N77 are examples of smartphones that support DVB-H.

1.4 The Mobile OS

In the past, portable devices such as cell phones did not require sophisticated operating systems. These earlier devices used simple, and usually proprietary, system software. In many cases they used no operating system at all, and all software remained fixed in the device's Read Only Memory (ROM). Now that mobile devices such as PDAs and smartphones have greater hardware power and implement sophisticated, media-rich (downloadable) applications, it's apparent that a sophisticated operating system is needed.

1.4.1 What Makes a Good Smartphone OS?

Smartphone devices have certain characteristics that are different from traditional desktop computers, and that must be addressed by a smartphone operating system.

Run on resource-limited hardware

Smartphones should be small, have long battery life, and cost as little as possible. To meet these requirements, smartphones, like other mobile devices, have limited memory and processing power compared with

desktop PCs and laptops. The operating system must be frugal in using hardware resources – especially memory. Not only must the OS itself not use much memory, but the architecture should be such that it provides limits and support to help OS applications also limit their use of memory, as well as allowing them to handle low-memory situations gracefully.

Robustness

A user expects a mobile phone to be stable and will not tolerate the device locking up. This is a challenge for any full-featured operating system due to the complexity of the system software itself; however, it is especially challenging for resource-limited devices like smartphones, which also allow third-party applications – that may be of questionable quality – to be downloaded.

Not only must the OS itself be designed to avoid crashing on its own, the OS must also provide support functions and policies for applications to follow, allowing the device to handle application errors and (as alluded to before) out-of-memory situations without locking up the phone.

User interface for limited user hardware

The OS should implement a user interface environment that is efficient and intuitive to use, despite the smaller screen and limited user input capabilities of a smartphone. Also, screen sizes and input capabilities vary between different models of smartphones, so the UI architecture should also be flexible, so that it can be customized for the varying form factors.

Library support for smartphone features

Smartphone operating systems should contain middleware libraries and frameworks with APIs that implement and abstract the functionality of the features of the smartphone. The purpose is to provide functional consistency and to ease software development. Examples of smartphone middleware include libraries and frameworks for email, SMS, MMS, Bluetooth technology, cryptography, multimedia, UI, GSM/GPRS – the more smartphone feature support the better.

Support for application development

Smartphone buyers want to know that there are many good applications available for their device, and that they can expect more and better software for it in the future. In order for this to be a reality, the OS must have good software development tools, support, training, and documentation. The more productive the developers, the more powerful, easy to use, and bug-free applications will appear for the smartphone.

1.5 Symbian OS – A Little History

The creation of Symbian OS can be traced back to a talented team of software developers at a company called Psion, an early pioneer in the

handheld computer market. After successive generations of software for Psion's handheld devices, the team created an object-oriented operating system called EPOC, which was designed specifically for the unique requirements of mobile computing devices.

Psion realized that there was a need for a mobile OS that could be licensed to other manufacturers for use in their mobile products, and that their EPOC operating system was well suited for this. At the time, the mobile phone industry was looking for a general operating system suitable for mobile phones and was interested in using EPOC. In June of 1998, Symbian was formed as a joint venture owned by the major cell phone manufacturers of the day (Nokia, Ericsson, and Motorola) as well as Psion, with the primary goal of licensing the EPOC operating system and improving it.

Fast forward to today, and we find that Symbian's operating system – now known as Symbian OS – is a major player in the smartphone marketplace, residing in the majority of today's smartphone devices. Symbian is jointly owned by Ericsson (15.6%), Nokia (47.9%), Panasonic (10.5%), Samsung (4.5%), Siemens (8.4%), and Sony Ericsson (13.1%), which, together, represent a major portion of the cell phone industry.

1.5.1 Symbian OS Overview

Symbian OS was designed from the ground up for mobile communications devices. While some competing operating systems (such as Microsoft's Windows Mobile for Smartphones OS) evolved from operating systems written for larger, more resource-laden systems, Symbian OS approached it from the other direction. Symbian's earlier versions (when known as EPOC) would run on devices with as little as 2 MB of memory.

Symbian OS is a multitasking operating system with features that include a file system, a graphical user interface framework, multimedia support, a TCP/IP stack, and libraries for all the communication features found on smartphones.

Symbian OS has software development kits available for third-party application development. Furthermore, the hardware layers of the operating system are abstracted, so that phone manufacturers can port the OS to the specific requirements of their phone.

1.5.2 One OS, Various Flavors

It is challenging to create an operating system that provides common core capabilities and a consistent programming environment across all smartphones – yet at the same time allows for manufacturers to differentiate their products. Smartphones come in many different shapes and sizes, with varying screen sizes and user input capabilities; the user interface software needs to vary to fit these differences.

Symbian OS has a flexible architecture that allows for different user interfaces to exist on top of the core operating system functionality. Of course, it is not wise to be too flexible for two reasons: (1) having too many different user interfaces inhibits code reuse among different devices and (2) too much work is required by the original equipment manufacturer (OEM) to create a GUI user interface from scratch for their smartphone.

So, to give the phone makers a starting point, Symbian created a few reference platforms, each packaging the Symbian OS core functionality along with a user interface that matched one of the basic smartphone form factors (screen size and input capability). This was important in the beginning; the idea was for smartphone manufacturers to choose a reference platform that most closely matched their phone's hardware characteristics, and use that as a starting point for their own customized UI layer. This indeed is what happened, and these reference platforms were the origin of the main flavors of Symbian OS you see today – S60, UIQ, and Series 80.

Symbian OS no longer supports the original user interface reference platforms and the smartphone programmer has no contact with these at all. Instead, the developer uses the software development kit (SDK) for the end platform supported by the phone. Also, there is no generic Symbian OS SDK for the developer – all core functionality is also included in the particular platform SDK.

Here are the major platforms for Symbian OS:

Nokia S60

The Nokia S60 (originally known as Series 60) user interface platform was originally designed for lower-end smartphones with small displays (176 × 208) and limited user input, such as a numeric-style keyboard used to enter text. This has changed in that S60 devices are getting more sophisticated, having larger displays, and even full keyboards. Nokia based S60 on the Symbian reference design known as Pearl, although Nokia did make significant modifications to it. S60 is a popular Symbian user interface. S60 is the most shipped platform for Nokia smartphones. At the time of writing, over 100 million S60 smartphones have been shipped.

The Nokia E61, Nokia E90 Communicator, Nokia N76, Nokia N93 and Nokia N95 are examples of phones that run S60 3rd Edition. Earlier S60 phones include the Nokia 6680, Nokia 7610 (which are S60 2nd Edition platforms) and Nokia 3650 (which is a S60 1st Edition platform). At the time of going to press, the latest S60 smartphone to be announced is the Nokia N81.

Nokia also licenses the S60 platform to other manufacturers such as Lenovo, LG, Panasonic, Samsung, and Siemens. At the time of writing,

the most recently announced S60 smartphone from a licensee is the Samsung SGH-i400.

Nokia Series 80

Series 80 is designed for Nokia phones known as communicators. These phones have a half VGA, landscape screen, a foldout keyboard, and hard buttons along the right side of the screen that have dynamic functions as defined by the application. Series 80 is based on a Symbian OS reference design called Crystal. The Nokia 9210/9290 and 9500/9300/9300i communicator devices use the Series 80 user interface.

Nokia is discontinuing Series 80; it will not use it in any new phones going forward. Nokia now includes support for the communicator device form factor in the S60 3rd Edition platform. The Nokia E90 is the first communicator device to use this S60 platform.

UIQ

UIQ originated from a Symbian reference design known as Quartz. UIQ is owned, developed, maintained, and licensed by UIQ Technology AB. UIQ is designed for pen-based (i.e., touchscreen) smartphones with quarter VGA display and no keyboard. A virtual screen keyboard and handwriting recognition is provided for user input.

The Sony Ericsson P1i, W960i, W950i, P990i and M600i, and the Motorola MOTORIZR Z8, are examples of UIQ phones. These phones are based on UIQ version 3, the newest UIQ version at the time of this writing. Earlier UIQ version 2 phones include the Sony Ericsson P800/P900 and Motorola A920, A925, and A1000 smartphones.

Note that originally, S60 was designed for one-handed operation, and UIQ for two-handed operation. This distinction has been blurred, however, since higher-end S60 phones now have full keyboards, and some UIQ phones are now geared toward one-handed operation (e.g., the MOTORIZR Z8).

As mentioned, Symbian OS no longer supports or maintains the original Pearl, Crystal, and Quartz reference platforms; however, they do maintain an internal platform known as *Techview*. This UI is used and maintained internally by Symbian to validate development, and is the basis of Symbian's Training SDKs. Unlike the other UIs, the Training SDK does not support building for any target phone hardware.

1.6 Symbian OS Smartphones

A large variety of Symbian smartphones are on the market today. These devices have various feature sets so that users have a choice of which smartphones to buy based on what features are important to them – and within the price range they are willing to pay.

For example, there are smartphones that target music lovers, emphasizing high-quality audio playback and large storage capability for songs in addition to other smartphone functionality. There are also smartphones that target video enthusiasts, which include high-resolution cameras and video capture. There are enterprise phones that contain large displays and keyboards that are well suited for the business person on the go. Another example is TV-oriented smartphones that support DVB-H capability for high-quality video service. And of course, there many other, more general phones that contain a combination of these and other features.

While Symbian OS powers many high- and medium-end smartphone models, another important – and big – market is the low-end, low-cost phone market. Symbian has been concentrating on getting more into this market in recent years. For example, in 2006 Symbian announced new scalable pricing options for phone manufacturers that lower the cost of licensing Symbian OS for high-volume smartphones.

This section introduces three Symbian OS-based smartphones: the Sony Ericsson P990i, Nokia N95, and Nokia 9300i Communicator. These phones are not necessarily representative of all Symbian OS smartphones, since they are more on the high-end, but they show a good sampling of some of the features we have discussed in this chapter, as well as representing the different UI platforms described in the last section. To see a complete list of Symbian OS smartphones, reference the Symbian phones section of Symbian's website (***http://www.symbian.com/phones***).

1.6.1 Sony Ericsson P990i

The Sony Ericsson P990i (shown in Figure 1.1) is a pen-based smartphone that uses the UIQ user interface. It has a 262K color, 240 × 320 pixel display with touchscreen, and a small keyboard that flips out. The phone has handwriting recognition, along with many pre-packaged organizer and game applications. The device plugs into a cradle that is connected to a PC via USB for downloading applications and syncing user data. IR and Bluetooth technology are also supported. The P990i has an integrated 2 megapixel camera, and will record video also. The phone contains a combination WAP/HTML browser, audio and video playback, email and SMS, as well as MMS. The device has 60 MB of internal memory for storage and supports an external memory card to expand this (a 64 MB memory card comes with the phone, but you can expand up to 4 GB).

The P990i supports UMTS, making it a 3G phone. But note that due to it using 2100 MHz for this, 3G for the P990i will not work in the USA since the UTMS there is typically 850/1900 MHz. The P990i also supports Wi-Fi and GSM 900, 1800, 1900 MHz and GPRS.