

# 1

## Introduction

*Any sufficiently advanced technology is indistinguishable from magic.*

Arthur C. Clarke

This is a book about making software work on the most widely used open mobile device operating system on the planet, in other words, it is about porting to the Symbian platform. It is also a book that describes a new paradigm for the development of native applications<sup>1</sup> on mobile devices, creating and using code that is much more portable than has been possible historically. If that doesn't already sound like something that's worth investing your time and effort in, then hopefully this chapter will convince you.

Before discussing this new era of mobile software development we need to answer a few basic questions:

- What is porting?
- What is portability?
- Why port to mobile platforms?
- Why get interested now?
- Why port to the Symbian platform?

All of this and more will be revealed in the following sections. Along the way, I'll introduce some of the core technologies that will enable the next generation of native mobile applications. Using these technologies is the subject of the bulk of this book. In answering these questions, I'll also look to the future to give a hint at the directions your projects could go in after you've successfully ported them to the Symbian platform.

---

<sup>1</sup> Native applications are delivered in a binary format which the hardware understands directly, rather than being interpreted by some intermediate software layer.

## 1.1 What Is Porting?

Taking the established convention of the Internet age, and consulting Wikipedia as our starting point, we define porting as follows:

*In computer science, **porting** is the process of adapting software so that an executable program can be created for a computing environment that is different from the one for which it was originally designed (e.g. different CPU, operating system, or third party library). The term is also used in a general way to refer to the changing of software/hardware to make them usable in different environments.*

[en.wikipedia.org/wiki/Porting](http://en.wikipedia.org/wiki/Porting)

In other words, ‘porting’ is making software run on different hardware or operating systems, or using different libraries, or any combination of these. There are several examples of porting projects in this book which cover all of these differences. For example, when porting a typical application developed for the Microsoft Windows desktop to the Symbian platform, it will need to run on a different processor (ARM rather than x86 architecture) and take into account differences between the operating systems and user interface libraries, in order to work on a Symbian phone.

## 1.2 What Is Portability?

Closely tied to the process and practice of porting software is the concept of software portability. The Wikipedia page also states:

*Software is portable when the cost of porting it to a new platform is less than the cost of writing it from scratch. The lower the cost of porting software, relative to its implementation cost, the more portable it is said to be.*

How portable a piece of code is will determine how much effort is involved in porting it to another environment. This is not a fixed quantity for a given piece of code but rather a function of the similarity between the original environment and any desired target environment. For example, an application written for an Apple Macintosh may be fairly easy to port to an iPhone but very difficult to port to a Windows Mobile device. However, it is possible for code to be written in such a way that it is portable to a wide range of platforms. Some excellent advice for this practice can be found in Chapter 15. Additionally, the ease of porting to a particular platform can change significantly if the software environment on that platform is modified. This is the case with the Symbian platform, thanks to the addition of several industry standard and other cross-platform programming interfaces in the last couple of

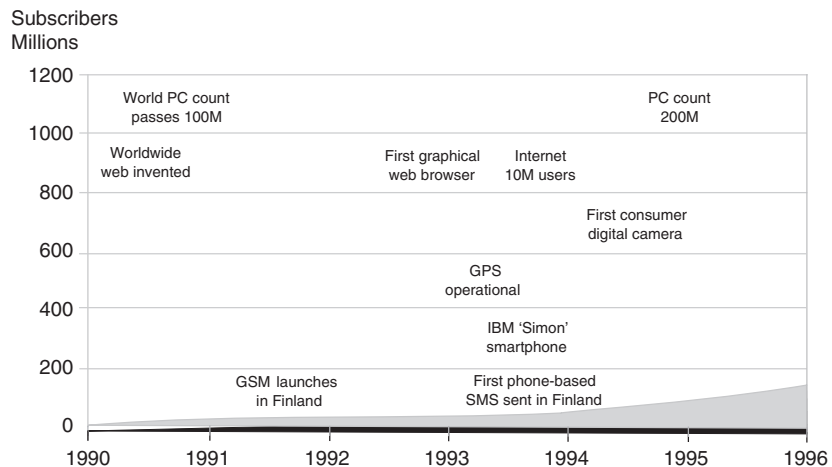
years, as described in Chapters 4 and 6. Such a change to the software environment can make a massive difference to the case for investing in porting your software, whether that's a financial investment or the investment of your time.

## 1.3 Why Port to Mobile Platforms?

Having an understanding of what porting is, we turn our attention to why you'd want to port to mobile platforms at all. To address this issue, I have a few different and complementary answers. Motivation is a very individual thing so there are different reasons for targeting mobile devices depending on your perspective. What follows is an attempt to categorize a few common perspectives.

### 1.3.1 The Marketing Angle

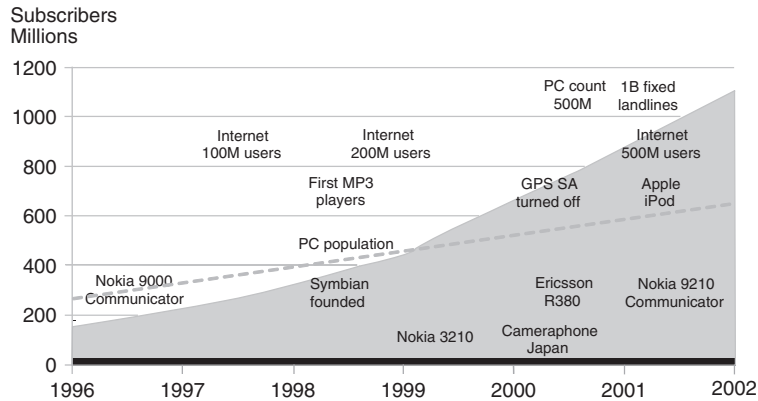
Consider that the global installed base of PCs is just over one billion units<sup>2</sup> and that more mobile phones than that are now sold every single year. In fact, in early 2008 the number of mobile subscriptions exceeded half the global population of 6.7 billion. Although this figure included a fairly large number of people with multiple subscriptions, as you read this it's almost certain that more than half of the people on Earth have a mobile phone (see Figures 1.1–1.3<sup>3</sup>). What's more, that percentage is growing rapidly.



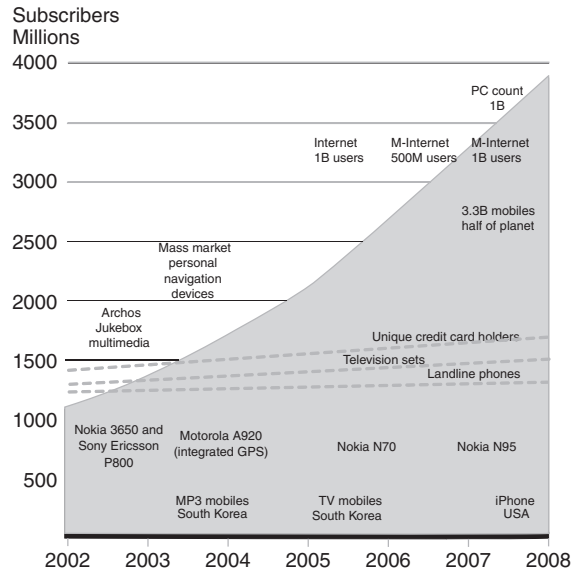
**Figure 1.1** Growth of mobile phone adoption and related technology 1990–96

<sup>2</sup> Source: Gartner, [www.gartner.com/DisplayDocument?ref=g\\_search&id=644708](http://www.gartner.com/DisplayDocument?ref=g_search&id=644708).

<sup>3</sup> Diagrams adapted and reprinted with permission from Tomi Ahonen Consulting.



**Figure 1.2** Growth of mobile phone adoption and related technology 1996–02



**Figure 1.3** Growth of mobile phone adoption and related technology 2002–08

Estimates from Wireless Intelligence<sup>4</sup> (which have historically been far too conservative) suggest that, by the end of 2010, 70% of the global population will be carrying a mobile phone. Although a lot of the recent growth has been in low-end handsets that can't run complex third-party software, reducing hardware costs enable device manufacturers to push more advanced operating systems into mid-range models. Market analysts

<sup>4</sup> [www.wirelessintelligence.com](http://www.wirelessintelligence.com).

Gartner predicted smartphone sales of 190 million units in 2008 alone, with a forecast for over 700 million units in 2012, accounting for 65% of all handset sales.<sup>5</sup> Even though the crisis in the global financial system is likely to reduce the demand from the previously expected level over the next few years, the long-term picture is still intact. A significant fraction of all human beings will soon have a powerful personal computing and communications device with them at all times.

### 1.3.2 The Hacker's Reason

Just to be clear, I'm not talking about people who crack security systems here but those who 'delight in having an intimate understanding of the internal workings of a system, computers and computer networks in particular'.<sup>6</sup> In this sense, the word 'hacker' is generally used as a mark of respect for a particularly skillful and knowledgeable software engineer or programmer. There are a lot of reasons to get excited about mobile technology, which has a much greater level of variety and rate of development than desktop systems. Specific to porting software to mobile platforms, the following points are worthy of consideration:

- Porting can be a good way to start learning a new platform.
- Existing projects can reach thousands or even millions more users.
- There's a great opportunity for new developers to build a reputation.
- New avenues for innovation are opened by combining existing code with mobile-specific features and data.

Also, porting can often have an excellent reward-to-effort ratio, often requiring surprisingly little effort to get some useful results. The recent shift towards open source platforms in the mobile industry, including the recently formed Symbian Foundation, provides an excellent environment for the hacker mentality to thrive.

### 1.3.3 The Geek's Reason

Many (but not all) hackers are also geeks. However, the geek's motivation for porting software to a mobile platform is so that they can use it on that platform, rather than the enjoyment or understanding they gain from the porting task itself. For some people, the idea of having certain favorite applications or services with them at all times is just going to be too tempting to resist. In a world where an increasing number of people are becoming addicted to text messaging, often sending more than

---

<sup>5</sup> [www.cellular-news.com/story/33277.php](http://www.cellular-news.com/story/33277.php).

<sup>6</sup> [tools.ietf.org/html/rfc1392#page-21](http://tools.ietf.org/html/rfc1392#page-21).

100 messages in a day, who knows what this trend may spread to next? Some mobile network operators are already offering unlimited access to mainstream social networking clients such as Facebook and MySpace on pre-paid tariffs. There's even a mobile client for the virtual world, Second Life. Geeks with a taste for slightly less popular applications, social networks or virtual worlds may not want to wait for someone else to give them access from their mobile. With ever-growing levels of Internet and technology addiction in various forms, the opportunity to have 24/7 access may not be viewed by everyone as a good thing. Then again, it might just be the case that having permanent availability of access reduces the compulsion to over-indulge and enables a better balance. Being able to carry their virtual worlds with them might just encourage people to spend more time in the real world and perhaps even combine the best of both.

## 1.4 Why Get Interested Now?

Similar arguments to those in the previous section could have been made at any time in the last few years. My assertion is that the smartphone concept has now reached the point in its technical evolution where it's ready for mass-market adoption by both the device creation industry and the consumer. Ever since we first started recording and processing information in a digital form there has been an ongoing trend towards a globally connected ubiquitous computing environment. Computing technology started out in room-sized isolated units, shared by large numbers of researchers. Since that time it has become increasingly personal, portable and connected. We'll consider the consequences of this long-term shift at the end of this section, but first a little history.

### 1.4.1 A Brief History of Digital Convergence

Every story has to start somewhere; we'll start this one just over 30 years ago. In 1978, there were some important firsts that, with hindsight, could be viewed as the beginning of the current era of global digital convergence. In the telecommunications world, the first TCP/IP networks were being built and tested and at the same time Bell Labs launched a trial of the first commercial cellular network.<sup>7</sup> This was also the year that the first GPS satellite was launched (although it wasn't much use on its own) and the first entirely digitally recorded popular music album<sup>8</sup> was produced. A few years later, in 1981, Sony released the Mavica, the world's first commercial electronic still camera, and IBM released their first PC.

---

<sup>7</sup> Although Finland had the ARP network from 1971, this is not considered a true cellular network as there was no automatic handover between cells.

<sup>8</sup> For the curious, it was Ry Cooder's *Bop Till You Drop*.

It is often said that we tend to overestimate the short-term impact of a new technology and underestimate the long-term implications. This tendency of individuals and society as a whole is sometimes referred to as macro-myopia. I believe this occurs because technological innovation opens up the potential for new markets but also inevitably has teething problems. The market potential encourages several competing solutions to the early problems, resulting in a fragmented market and consumer confusion. It's not until winners emerge, or competitors collaborate to develop a common standard, that consumer adoption increases and mass-market pricing can be achieved. In many cases, further improvements in technology are required before a compelling end-user experience is possible. The converging technologies provide examples of this process.

The original Sony Mavica was not a true digital camera but really just a video camera that could take freeze frames. The product demonstrated the potential for the 'filmless' camera but it then took a further five years for Kodak to develop the first megapixel sensor, capable of producing a standard 5"x7" photo-quality print. From there, Kodak collaborated with others, although it was still a further five years before the first professional digital camera (the Nikon F-3) was released, closely followed by the Photo CD standard. The first digital cameras for the consumer market didn't arrive for another three years, finally hitting the shops in 1994. Fifteen years later, traditional film is hardly used at all.

Similarly, in the digital audio market, the first portable player prototype was developed in 1979. Unfortunately it could only store about three and a half minutes of music and so it was never produced commercially. Consumers had to wait until 1998 for the first mass-produced digital audio players. The inventions of cheaper non-volatile storage devices and digital audio compression technologies were necessary steps to make the original idea a commercial reality. The industry also had to agree on a relatively small set of audio compression techniques and storage formats to make it possible for users to play their content on multiple devices.

Meanwhile in computing, the first IBM PC launched in 1981 was fairly expensive (\$3000) and technologically inferior to the competition that came from Apple over the following years. The Lisa and then the rather more famous Macintosh introduced the first graphical user interfaces. In this case, the industry standardized on the PC because IBM had used off-the-shelf components in their design, plus a third-party operating system, in order to save time and money; this opened the way to much cheaper clones from other manufacturers. However, it took until 1990 for Microsoft to develop a graphical user interface for the PC, version 3.0 of the now ubiquitous Windows operating system, comparable to that on the early Macintosh. The combination of fairly standard hardware and a common operating system with an interface that anyone could learn

to use made the PC an attractive target for a wide range of third-party software.

One such piece of software was the web browser. Although TCP/IP allowed computers all over the world to communicate, there was no convenient standard way for non-technical users to publish and access information on the Internet. Then Tim Berners-Lee, working at CERN, invented the Web and built the first web server and text-based browser in 1990. Three years later, CERN announced that the Web would be free to everyone and in the same year the first graphical browser, Mosaic, was released. With the addition of popular search engines, AltaVista and Yahoo, adoption was rapid. The leader of the team that built Mosaic formed his own company, which became Netscape Communications Corporation. Netscape had early dominance in the market but soon faced strong competition from Microsoft's Internet Explorer. The war between the two browsers resulted in proprietary innovations on both sides, seeking to gain competitive advantage but creating fragmentation for developers and end users. Microsoft eventually ended the war (and browser innovation for several years) by abusing its operating-system monopoly to force Netscape out of the market by 1998.<sup>9</sup> Monopolies can be very bad for innovation!

In the case of GPS, the situation is rather different since the system was built for the US military and controlled by their government. However, the timeline is very similar. The 24th GPS satellite was launched in 1993, bringing the system to what was called 'initial operational capacity'. In the same year, the decision was taken to authorize worldwide civilian use of the system, free of charge. Unfortunately, until 2000, the system included a special feature called 'selective availability' which deliberately introduced errors to the signal for civilian users, providing a position to only 100 m accuracy rather than the sub-20 m accuracy available to the military. Since the feature was removed, there has been a significant increase in the use of the technology, particularly for vehicle navigation systems.

The specification and creation of cellular phone systems across the world was to some extent government controlled, like GPS (exceptions being the United States and Japan). As a result, several countries developed different standards to build their own first generation (analog) systems. An alternative approach was taken in the Nordic region; Denmark, Finland, Norway and Sweden co-operated to specify a common system called NMT. The specifications were free and open, allowing many companies to produce NMT hardware, pushing prices down. The open system and lower cost of hardware encouraged adoption across parts of Europe, the Middle East and Asia. In fact, the first commercial NMT service was introduced in Saudi Arabia in 1981, just ahead of the Swedish network. The success of NMT and the experience

---

<sup>9</sup> [www.usdoj.gov/atr/cases/f3800/msjudgex.htm](http://www.usdoj.gov/atr/cases/f3800/msjudgex.htm).

gained was a significant advantage to Nokia and Ericsson when the second-generation GSM system was developed following a similar open standards approach. The first GSM network was launched in 1991 by Radiolinja in Finland. The standard is now in use by over 80% of the world's mobile subscribers.<sup>10</sup> The standards have since been extended to enable high-speed data communication as well as the original voice and basic messaging features.

The early evolution of mobile phones was closely tied to that of the networks they operated on. The first cellular phones were generally too large for anything but vehicle installation, although some models were designed to be carried in backpacks. The first handheld mobile, the DynaTAC 8000X, was produced by Motorola in 1983. Over the following years technological developments allowed phones to get smaller, lighter and cheaper. Apart from the shift to digital communications, the basic functionality remained the same for a long time. In early 1999, a major shift began with the launch of the Nokia 3210. The device included predictive text input, allowing faster text entry via the numeric keypad; three games, ensuring popularity with younger consumers; changeable covers; and an internal antenna, making it more customizable, attractive and pocketable. The result was a huge commercial success (approximately 160 million units sold) and massively increased the popularity of mobile phones for teenagers and young professionals. Having reached mass-market appeal and pricing, devices started to evolve in two separate directions: some models continued the cost reductions to reach a wider market and other models increased in features and functionality, rapidly becoming more powerful computing and communications devices.

As the various digital technologies discussed above found their way into mass-market consumer electronics devices it became almost inevitable that they would converge. With common requirements for a general-purpose processor, memory, display and user input methods, there was a strong economic incentive to combine them into a single device. Although there have been, and probably will continue to be, niche-market portable convergence products, from the early personal digital assistants (PDA) to small-form-factor, Internet-enabled devices such as the modern netbook; it seems most probable that the only convergence device to be adopted by a significant fraction of the world's population will be an evolution of the mobile phone. There are two important reasons for this: always-on network connectivity is best achieved through the cellular network (even if other bearers are also used when available) and the device that people reliably carry, regardless of culture, is their phone.<sup>11</sup>

---

<sup>10</sup> [www.gsmworld.com/newsroom/market-data/market\\_data\\_summary.htm](http://www.gsmworld.com/newsroom/market-data/market_data_summary.htm).

<sup>11</sup> Extensive research has shown that keys, money and a phone are the only things that most people (who own a mobile phone) almost always carry. See [www.janchipchase.com/blog/archives/2005/11/mobile\\_essentia.html](http://www.janchipchase.com/blog/archives/2005/11/mobile_essentia.html) if you're interested in finding out why.

## 1.4.2 Birth of the Mobile Convergence Device

The potential for a powerful combination of computing and communications devices was recognized very early. The first ever ‘smartphone’, called Simon, was released by IBM in 1993. Nokia’s first smartphone (Nokia 9000), launched in 1996, was an early ‘Communicator’ product that ran GEOS on an x86 CPU. The IBM Simon and the Nokia 9000 were both extremely bulky, expensive and struggled with poor battery life, issues that have plagued many advanced mobile devices since then. It became clear to the leading mobile device manufacturers of the time that what was needed was an advanced operating system that was purpose built for battery-powered, resource-constrained devices which were seldom, if ever, rebooted. A new operating system developed by Psion to power their latest range of personal organizers, called EPOC32, was identified as the most suitable candidate. A joint venture was formed between Ericsson, Motorola, Nokia and Psion to continue the development to meet the needs of the emerging mobile device market. So, in 1998, Symbian was founded and the operating system was renamed Symbian OS.

The first year of the new millennium was also the year the first ever Symbian-based phone went on sale. The Ericsson R380, as it was called, was the size of a standard phone but the keyboard folded back to reveal a touchscreen. It combined a mobile phone with a number of personal organization tools, including an address book, a calendar, email, voice memo and a note pad. However, the R380 was ‘closed’: no additional applications could be installed to the phone. The operating system supported the installation of aftermarket software but it was not permitted on this device. This naturally raises the question of what exactly is a smartphone? The answer has to be that the definition has evolved over time as technology has advanced. Originally the term was applied to any device that combined the features of a phone and a PDA. More recently, the term can only legitimately be applied to phones with an operating system supporting the installation of native applications after a device has been purchased. As such, some might say that the Nokia 9210 Communicator, which launched a year later in 2001 as the first open Symbian phone, was the first genuine smartphone.

## 1.4.3 Inevitable Fragmentation

In the first few years of open smartphone development, the device manufacturers that were collaborating around Symbian OS decided that they could add value and differentiate their offerings from one another at the user interface layer. To this end, several user interfaces and application frameworks were created. At one point Nokia had three of their own: Series 60 (now referred to as S60), Series 80 and Series 90. Sony Ericsson and Motorola used another user interface, called UIQ,

and NTT DoCoMo in Japan created yet another, called MOAP(S). The Symbian platform became badly fragmented for developers at the UI layer. In addition, Microsoft entered the mobile-device operating system market with a version of their Windows CE platform named Windows Mobile<sup>12</sup> and other manufacturers shipped devices based on Windows Mobile or the competing Palm OS from the handheld computer market. Around the same time, Research In Motion (RIM) also started developing their range of pagers and corporate messaging devices into fully featured convergence devices.<sup>13</sup>

There was a multitude of mobile device options for both consumers and developers. To a large extent, developers tended to prefer Palm OS and Windows Mobile (partly due to familiarity from earlier handheld computing platforms) while consumers tended to select Symbian-based devices (probably due to their more phone-like form factors and functionality). Between 2001 and 2005, the device volumes were still relatively low and the technology not yet mature enough to provide really compelling convergence devices. By the middle of 2006, things were starting to change; Nokia had introduced their extremely successful Nseries range<sup>14</sup> of ‘multimedia computers’ and demonstrated the potential for a single model, the N70, to sell several million units. At the same time, technology was evolving to make larger displays a reality at a mass-market price point. A larger display made mobile Internet use a more realistic proposition and that, combined with the high profitability of premium phones, renewed interest from outside the industry. Internet giant Google and luxury device maker Apple started investing heavily to create their own offerings, Android and the iPhone range. Simultaneously, mobile network operators and service providers called for an end to fragmentation so that they could develop applications and services that would work across the whole range of devices.

Many companies saw the free and open nature of Linux as a potential solution. It promised to be an operating system that could provide the basis for a platform that the industry could collaborate to develop. However, initially Linux’s only real qualifications for this role were that it was robust, free and had an enthusiastic developer community. Having been designed and developed for desktop computers, its size, start-up time, power consumption and lack of support for typical mobile device hardware meant that a lot of work was needed to create a serious candidate. To meet this challenge, groups such as the Mobile Linux Initiative and the CE Linux Forum were started. As their efforts started to pay off, other groups began building on their work to create more complete platforms. Unfortunately, rather too many groups were all trying

---

<sup>12</sup> The original Microsoft Pocket PC and Smartphone brands were unified as Windows Mobile.

<sup>13</sup> Although initially open to native applications, RIM’s BlackBerry devices now only allow Java ME applications, like most feature phones.

<sup>14</sup> [www.nseries.com](http://www.nseries.com).

to do roughly the same thing with slightly different motivations; there was the Linux Phone Standards (LiPS) Forum, GNOME Mobile and Embedded, the LiMo Foundation, Ubuntu Mobile, and the Mobile and Internet Linux Project (Moblin). There were also multiple companies and communities trying to leverage these efforts to create similar but not entirely compatible device platforms such as Openmoko and Nokia's Maemo. By the time Google announced the Linux-based Android platform, and yet another group to develop it (the Open Handset Alliance), the situation was starting to look rather farcical.

#### 1.4.4 Reunification

While the various mobile Linux variants were busy creating more fragmentation, Nokia were starting to reduce it in the Symbian world. Initially they reduced their user interface (UI) platforms from three (Series 60, Series 80 and Series 90) to just the most successful one, re-named as S60 and intended to be flexible enough that it could target all the segments previously covered by the others. Then in June 2008, Nokia made an extremely bold move to unify the Symbian platform, announcing their intention to buy all of the shares that they didn't already own in Symbian Ltd and donate the Symbian and S60 software assets to a new non-profit organization called the Symbian Foundation. At the same time, Motorola and Sony Ericsson pledged their UIQ assets and NTT DoCoMo and Fujitsu did the same for MOAP(S). The intention of this was to create a single royalty-free unified UI platform, using the best of all the existing platforms but maintaining backwards compatibility with S60. The acquisition of Symbian was approved at the beginning of December 2008, with the first release of the Symbian platform occurring in the first half of 2009. At launch, the Symbian Foundation made selected components available as open source (under the Eclipse Public Licence) and it is working to establish the most complete mobile software offering available in open source. In the interim, all the contributed software is available to members of the Foundation under a separate royalty-free license.<sup>15</sup> However, this is just while third-party technology is disentangled from the platform and the code base is sanitized; the stated plan is to release the entire platform as open source by the middle of 2010, two years after the initial announcement.

Almost immediately after the Symbian Foundation was announced, two of the Linux standards groups joined forces with LiPS, and folded into the LiMo Foundation. Somewhat ironically for a platform based on an open operating system, the LiMo Foundation platform is not open source, nor is there a public commitment to provide anything other than a binary SDK to non-members at the time of writing.<sup>16</sup> Additionally, many

---

<sup>15</sup> Membership of the Symbian Foundation is open to all at minimal cost (see [www.symbian.org/members](http://www.symbian.org/members) for more details).

<sup>16</sup> See [www.limofoundation.org/images/stories/pdf/limo\\_foundation\\_ipr\\_guide.pdf](http://www.limofoundation.org/images/stories/pdf/limo_foundation_ipr_guide.pdf).

of the devices developed using the LiMo Foundation platform may well not allow the installation of third-party, aftermarket native software; even where this is permitted, there are only source compatibility, not binary compatibility, guarantees so applications may have to be recompiled for each manufacturer's devices. Even so, the LiMo Foundation is going a long way towards consolidating the efforts of the major players in mobile Linux. Also, it has been noted by many industry commentators that the membership of the LiMo Foundation has a significant overlap with that of the Open Handset Alliance. It seems unlikely that manufacturers will ship devices based on both platforms for long, so further consolidation is probable in the future.

Another point of interest is that, just prior to being acquired by Nokia, Trolltech (creators of the Qt cross-platform application framework) joined the LiMo Foundation. Originally most of the Linux platforms were using GTK+ (part of the GNOME project), the most popular open alternative to Qt, for their application frameworks. Now both Maemo and Openmoko have started supporting Qt applications. With Trolltech also having released a Qt port for Windows Mobile in late 2007, it seems that there just might eventually be a single native application framework to target the vast majority of mobile devices. This topic is explored further in Chapter 6 but it's time to return to the question of why now is a good time to get interested in mobile software.

### 1.4.5 A Tipping Point

So, what was the purpose of this journey through history? In fact there were two reasons for providing all of this background information:

- The mobile industry has recently passed a turning point – although mobile device sales continue to grow rapidly the fragmentation problem should now be getting better, not worse.
- Seen in its historical context, the converged mobile device as a product category has been waiting for a number of technological breakthroughs to make it truly compelling as a mass-market product. Those breakthroughs have now been made.

In support of the first point, there is the fact that the Symbian ecosystem is now aggressively moving to support cross-platform standards. In the main, these are standards that are already available for embedded Linux platforms and, to a certain extent, Microsoft's Windows Mobile. Between them, these platforms cover the vast majority of mobile devices for which it's possible to write native aftermarket applications. This makes the case for investment in software development much more compelling.

To defend my second assertion, I could simply point to the statistics in Section 1.3.1. However, somewhat strangely for a book about the

Symbian platform, I'd like to point to the success of Apple's iPhone, particularly in the North American market, as proof of the statement. Unencumbered by the need to support existing features and standards that early converged device adopters have come to expect, Apple has been able to create a device which is optimized for media consumption on the move that also just about passes for a phone and textual communicator. They have done this from scratch in a relatively short timescale and made it attractive enough to sell significant volumes in a market that had previously seen very low uptake of mobile converged devices. In doing so, they've prompted a renewed focus on usability<sup>17</sup> throughout the industry which should help boost all advanced mobile device sales. One final point on the subject of the iPhone is that the Apple App Store has also helped make a convincing case for native mobile applications, making sales revenues of approximately \$1 million per day in its first three months of operation.

From a pure technology perspective, we now have genuine mobile broadband Internet with high-speed packet access (HSPA) and screens of sufficient size and resolution to browse standard websites and watch full-length films. Many advanced mobile devices also incorporate personal navigation via GPS, alternative (and often free) local connectivity via WiFi, and a camera of suitable quality for anyone other than a professional or serious photography enthusiast. The next generation of mobile device hardware will feature multiple processor cores to provide significantly enhanced computing power without compromising battery life when that power isn't required. It will also bring more advanced 2D- and 3D-graphics acceleration, with the potential to further improve the user interface and the gaming experience. If we're not already past it, I believe the wider adoption of these features will take us beyond the tipping point where the inevitability of mobile software and services becoming mass-market on a global scale is assured.

Before we look any further into the future, just for a moment, imagine a person from 30 years ago transported in time to today (or waking from a coma, being rescued from a desert island – whatever you find more believable). The digital technologies we take for granted were only just being invented in their world and someone is showing them the latest high-end mobile device. It's a small wireless phone – amazing. It's also a camera and it can store and play their entire music collection. They can send messages and pictures almost anywhere in the world – that's quite mind blowing. Wait, it can play last night's TV show or live video from the other side of the world. That's not all; it can also find the answer to almost any question, show them where they are on a map and give verbal instructions to get to any other place they want. All of this, they're told, is only just scratching the surface of what this mobile device can do. It really must seem like a little magic box. In just 30 years, technology

---

<sup>17</sup> And, of course, a string of attempted clones from other manufacturers.

has advanced so far as to be almost beyond comprehension. How will technology have changed the world in another 30 years?

### 1.4.6 Mobile Devices of the Future

While the current case for mobile software development is interesting, the future is exciting. Where will mobile technology be in 30 years? It's almost impossible to predict but many clues are available now. Tomi Ahonen, top industry consultant and author, makes the case that mobile is becoming a new and superior mass media channel.<sup>18</sup> The central thesis of the book is that mobile, connected devices can replicate the features of all of the existing mass media channels and also have seven additional unique benefits (paraphrased):

- truly personal mass-media channel
- permanently carried
- always on
- built-in payment system
- available at point of creative impulse, enabling user-generated content
- near-perfect audience data
- context of media consumption.

Many of these unique benefits from a marketing perspective are potential avenues for further enhancement in the future. Devices could become ever more personal if they are worn rather than carried. In early 2008, Nokia showed a Morph concept<sup>19</sup> phone that makes use of nanotechnology in new materials to create a flexible device that can be molded to the desired use but also worn like a bracelet or watch. The advanced materials can even help with the always-on aspect by harvesting solar energy across the entire surface of the device. Even in the much nearer term, nanotechnology promises to help free mobile devices from some of the power-usage constraints imposed by the battery. One of the many new battery technologies that will be available soon uses silicon nanowires to increase the charge density to around 10 times that of existing lithium-ion batteries;<sup>20</sup> it also has the advantage of significantly reduced degradation across charge cycles, giving the battery a much longer life.

With the battery life no longer restricting use of the mobile device, it's likely to be put to many more uses. While it's already possible to pay

---

<sup>18</sup> Ahonen, T. (2008) *Mobile as 7th of the Mass Media*. Futuretext.

<sup>19</sup> See [www.nokia.com/A4852062](http://www.nokia.com/A4852062).

<sup>20</sup> [news-service.stanford.edu/news/2008/january9/nanowire-010908.html](http://news-service.stanford.edu/news/2008/january9/nanowire-010908.html).

for digital content downloads with your phone, that may be extended much further. The keys, money and phone that most people carry with them everywhere could merge into just the phone. Like the early trials of computer networks, which eventually led to a global Internet, there is already a very successful contactless payment system in Hong Kong called Octopus, which is accepted on all public transport systems and in a wide range of shops. It is also commonly used in place of keys for building access. Octopus is based on RFID technology and has already been embedded into mobile phone covers. Another system currently being trialed around the world is PayPass from MasterCard, using very similar technology.<sup>21</sup> It is also being made available in mobile phones. Currently contactless payment systems are only available for low-value transactions but in the future it may be possible to authorize higher value payments via a biometric sensor in a suitably equipped mobile device, allowing the whole process to be seamless.

As mobile devices become increasingly powerful, capable, less constrained by available power and a standard method of payment, they are also likely to become the most common portal for accessing information and services via the Internet. If this seems improbable, consider that it is already the case in India, where wireless technology has leapfrogged fixed lines, but also in Japan and Korea,<sup>22</sup> two of the most advanced digital societies in the world, where standard broadband availability is extremely high. At this point, it should be made clear that mobile devices won't completely replace PCs and mobile will not replace the other media, just that the dominant paradigm will shift. If your application or service isn't available on mobile platforms then it will probably be left behind.

However, the unique benefit that will make developing applications and services for mobile platforms really interesting is 'context'. At present, this is largely limited to some social context (gathered via a social network) and possibly the location of the device. In the future, it is likely to be an area for major expansion with access to a vast array of contextual information from sensors and other systems in the device, elsewhere on the user and also in the environment. This information could be used to make digital services behave in more intelligent and helpful ways. Beyond the obvious and widely discussed medical monitoring and safety applications, there are also enormous opportunities to make our interactions with an increasingly technological world more natural and simple, reducing information overload as the amount of information available continues to increase. It's far too large and complex a topic to discuss here but fortunately an extremely deeply considered and well

---

<sup>21</sup> Visa also has a contactless payment system, called payWaves, and Visa was one of the early members of the Symbian Foundation, so further developments can be expected there too.

<sup>22</sup> According to data from the national regulators in the respective countries.

written account is already available in a book by Adam Greenfield;<sup>23</sup> I highly recommend it. Whatever form ubiquitous computing eventually takes, it seems likely that it will be based around advanced mobile devices for at least the next decade or two. By porting your code to mobile platforms now, you'll be ready to take it in all kinds of new directions that won't be available on the desktop.

## 1.5 Why Port to the Symbian Platform?

If you're convinced by the argument in the previous sections and have decided that porting your software to a mobile device platform is an excellent plan then the next question I need to answer is: Why the Symbian platform? Again, there are several answers:

- market share
- open source and royalty-free
- maturity
- range of hardware
- cross-platform APIs.

We look at each of these factors in the following sub-sections but first, Table 1.1 gives a comparison of the Symbian platform with some of its major competitors at the time of writing.<sup>24</sup>

### 1.5.1 Market Share

Unsurprisingly for such a promising market, there has been fierce competition from the outset. It has been both surprising and impressive that Symbian was consistently able to maintain a market share well above 50% until the third quarter of 2008, with its nearest competitors somewhere in the teens (see Figure 1.4).

Although RIM has a reasonable and growing market share, BlackBerry products don't currently allow native aftermarket applications, only offering a Java-based environment, so we can't really consider them in the context of this book. All Linux-based devices are grouped together, including LiMo and Android devices, although exactly what the breakdown is

---

<sup>23</sup> Greenfield, A. (2006) *Everyware: The dawning age of ubiquitous computing*. New Riders. It's worth noting in support of the case presented here that, shortly after publishing this book, Adam Greenfield became head of design direction for service and user-interface design at Nokia.

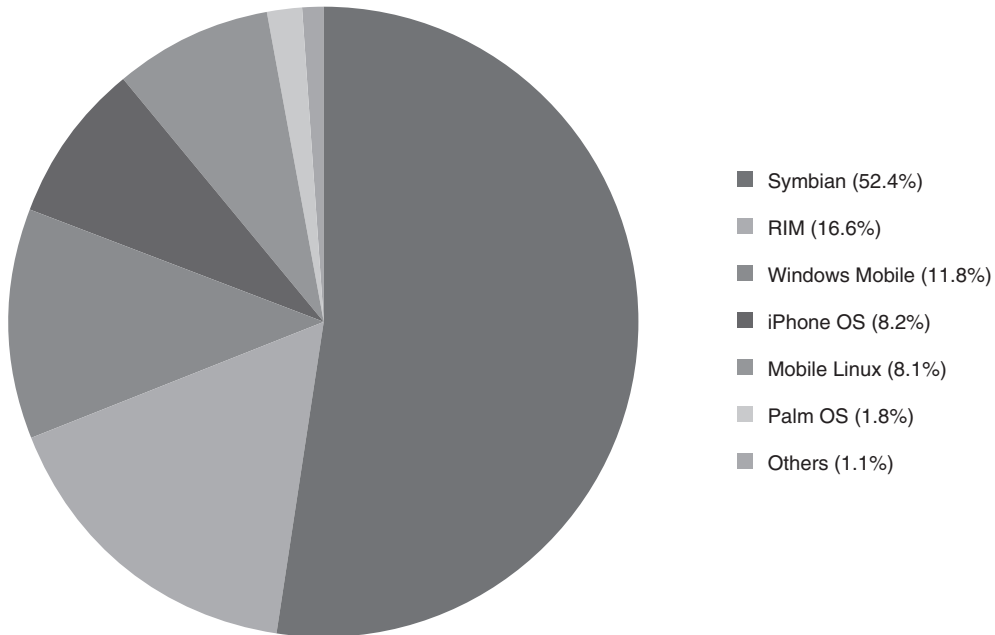
<sup>24</sup> For informed updates on the mobile platform landscape, read the Vision Mobile blog at [www.visionmobile.com/blog](http://www.visionmobile.com/blog).

**Table 1.1** Comparison of Mobile Platforms

	<b>Symbian Foundation</b>	<b>RIM BlackBerry</b>	<b>Apple iPhone</b>	<b>Windows Mobile</b>	<b>LiMo</b>	<b>Android</b>
Market share	High	Medium	Medium	Medium	Low	Low
Source code licensing	Open and free	Closed, not externally licensed	Closed, not externally licensed	Shared, <sup>a</sup> per-unit royalties	Shared, <sup>b</sup> moderate entry cost	Open (gated) and free
Maturity	High	High	Medium	High	Medium	Low
Range of hardware	High	Medium	Very low	High	Medium	Low
Cross-platform APIs	Very wide support	Java only	Limited support	Wide support	Moderate support	Java only

<sup>a</sup>Microsoft has shared source licenses under which it makes some of the platform code available to partners.

<sup>b</sup>LiMo source code is open to consortium members.



**Figure 1.4** Smartphone operating system market share in 2008, clockwise from Symbian (at 52.4%)<sup>25</sup>

<sup>25</sup>Source: Gartner, [www.gartner.com](http://www.gartner.com).

and how many of these can be targeted with native software is not clear at the time of writing. Microsoft, whether fairly or not, has never really been trusted by the mobile industry as there have always been fears that its embedded offerings would suffer from similar reliability issues to many of its desktop operating systems. This wasn't helped by the fact that the early Windows Mobile devices did actually display the infamous 'blue screen of death'<sup>26</sup> to end users. Finally Apple, with its much praised iPhone, has historically been a luxury device maker, focusing on high margins at the higher end of the market. It seems unlikely, although far from impossible, that it will either license the platform to others or make a successful push into the mid-range devices that Symbian has been working hard to reach in the last few years. With the unification of the platform under the Symbian Foundation, it seems extremely probable that you'll be able to target the greater number of devices with a single version of an application on the Symbian platform for many years to come.

### 1.5.2 Open Source and Royalty-Free

The Symbian Foundation plans to make the Symbian platform both open source and royalty-free by June 2010. Having worked on open source systems and on closed systems, both with privileged access to the source and without, I can honestly say that I wouldn't want to work on a time-critical development project without access to the source code for the underlying system libraries if I had any alternative. As mobile platforms become increasingly complex, it seems impossible for documentation and examples to keep up with the development of the system. No matter how good the documentation (and a lot of software documentation really isn't at all), there will always be gaps. When you fall into one of those gaps, then you need to be able to look at the source and work it out for yourself. If you can't do this, you're just left guessing and blindly searching for a solution.

From the platform development side, mobile device software is becoming too large and complex for any single organization to manage and maintain on its own. By opening the platform up to external contributions and shared development, it's possible to take advantage of the best solutions the industry has to offer. Of the platforms in Table 1.1, the Symbian Foundation is the only one with a declared intention to be fully open, although LiMo and Android are also open to contributions from consortium members. With a royalty-free license on the code, two additional options become available. First, smaller OEMs can use the platform to create devices without significant up-front investment or commitment to device volumes, while taking advantage of the full range of third-party software created for the platform. Second, if you've written an application that's dependent on a platform component then you may be able

---

<sup>26</sup> [en.wikipedia.org/wiki/Blue\\_Screen\\_of\\_Death](http://en.wikipedia.org/wiki/Blue_Screen_of_Death).

to port that component to another platform and distribute it with your application, easing future porting to other platforms.<sup>27</sup>

### 1.5.3 Maturity

The Symbian platform has been designed from the ground up for the needs of mobile devices. Unlike some of the alternatives, it hasn't been adapted or converted from a desktop platform. The system has been proven in the market for almost a decade. By contrast, some of the competing operating systems have only just started shipping products in the last year or two. The step from single products to a fully-featured platform capable of shipping tens of differentiated device models every year whilst retaining compatibility and satisfying network operator requirements for security and standards compliance is a painful one. It has to be admitted that in the past some of that pain has reached third-party Symbian developers, but a lot of lessons have been learned. It remains to be seen whether other platforms will make the transition smoothly.

### 1.5.4 Range of Hardware

If you're planning to make your software work across a wide range of mobile devices, then you get the greatest diversity of device hardware platforms on the Symbian platform. Symbian have put a lot of effort into features, such as demand paging<sup>28</sup> and symmetric multi-processing,<sup>29</sup> that enable the platform to scale from mid-range devices right up to the latest high-performance hardware. If your software works on the most resource-constrained, mid-range devices but also scales to take advantage of more advanced hardware when available then you can be confident that performance won't be an issue when porting it to other mobile platforms. This makes the Symbian platform a great choice for your first porting target. However, it should be noted that there are time-to-market benefits involved in targeting a subset of Symbian devices or another platform with a smaller range of hardware for the first release of an application. On the other hand, discovering that your design or architecture doesn't scale after you've got the product in the market could be worse than delaying the launch.

### 1.5.5 Cross-Platform APIs

There are two key factors to consider here: the ease of porting to the Symbian platform and the ease of porting to other platforms from

---

<sup>27</sup> Obviously this will depend on the licensing for your software and any further dependencies of the system component.

<sup>28</sup> [developer.symbian.org/wiki/index.php/Demand\\_Paging\\_on\\_Symbian](http://developer.symbian.org/wiki/index.php/Demand_Paging_on_Symbian).

<sup>29</sup> [developer.symbian.org/wiki/index.php/Roadmap\\_for\\_OS\\_Base\\_Services](http://developer.symbian.org/wiki/index.php/Roadmap_for_OS_Base_Services).

Symbian. As stated in Section 1.2, a significant effort has gone into creating industry-standard and other cross-platform APIs for Symbian devices in recent history. It is now possible to run most standard C and C++ code on Symbian devices, including a lot of code that conforms to POSIX standards – this is enabled by software packages called P.I.P.S. and Open C/C++, which are described in Chapter 4. However, it is still necessary to write the code in a way that respects the limitations and reliability requirements of mobile devices – excellent advice for this can be found in Chapter 3.

Perhaps the most appealing aspect of the Symbian platform from a porting perspective is Nokia's purchase of Trolltech and the subsequent porting of the Qt application framework to S60. This enables the porting of applications already written for Qt to the Symbian platform with relative ease but also porting to Qt on Symbian, which then makes it possible to port to other platforms supported by Qt with a significantly reduced level of effort. Perhaps the greatest potential benefit here comes from Nokia's stated intention to use Qt across most of its device portfolio, including Series 40, the feature phone platform. It is still unknown if or when a port to Series 40 will be available for third-party developers. You'll find more about Qt in Chapter 6, along with details of some other cross-platform APIs that are also available on other embedded platforms, including games consoles, set-top boxes and various mobile Internet devices.

Before you jump into the details of the various APIs, please have a look through Chapter 2, which describes the process for successful porting projects on the Symbian platform. Hopefully it will save you many wasted hours. Good luck and happy porting!

