

Software development for mobile devices: iPhone OS vs. Maemo

Erik Schmidt

Business Information Technology

Authors Erik Schmidt	Group
The title of your thesis Software development for mobile devices: iPhone OS vs. Maemo	Number of pages and appendices 41 + 21
Supervisors Matti Kurki <p>The core of this thesis project is the comparison of the iPhone and the Maemo Software Development Kits (SDK). As a secondary objective, two sample applications for mobile devices are created.</p> <p>The various aspects of the SDKs are described and evaluated, ranging from the hardware and software requirements to the available programming languages, to the availability of support resources and to the operating system and hardware specifications of two exemplary devices (iPhone 3GS and Nokia N900). As an outcome, a list of differences and similarities between the SDKs is presented.</p> <p>The installation process of the iPhone and the Maemo SDK is described and depicted. As a result, a working iPhone SDK, including the Xcode Integrated Development Environment (IDE), is available on a compatible Apple computer. Likewise, the Maemo SDK will be installed and available for production on a Linux-based PC. Two different installation paths are demonstrated for the Maemo SDK, one making use of virtualization software.</p> <p>Putting the SDKs to use and implementing a small sample application is the last part of the thesis project. The applications have similar functionality on both platforms and display the user's location in terms of coordinates. The application development process is presented as a brief summary and the source code for the applications is appended.</p>	
Key words Software development, mobile device, iPhone, Maemo.	

Table of contents

1	Introduction	2
1.1	Objectives	2
1.2	Outline	3
1.3	Scope	3
1.4	Methodology	3
2	Prior research and sources	4
2.1	Prior research	4
2.2	Professional literature and other sources	4
2.3	Source criticism	5
3	Software Development Kit Comparison: iPhone OS vs. Maemo	6
3.1	SDKs - Overview	7
3.2	Hardware requirements	9
3.3	Software requirements	10
3.4	Available programming languages and portability options	11
3.5	Support resources	13
3.6	Devices	15
3.6.1	iPhone 3G S	15
3.6.2	Nokia N900	17
3.7	Operating systems	19
3.8	Summary	23
3.9	Conclusion	23
4	SDK installation	25
4.1	iPhone SDK	25
4.2	Maemo SDK	27
4.2.1	Installation scripts	27
4.2.2	Virtual Image SDK	29
5	Application implementation	32
5.1	iPhone application	32
5.2	N900 application	33

6 Summary and conclusion	33
6.1 Summary	33
6.2 Conclusion.....	35
Bibliography.....	37
Appendices	42
Appendix 1. iPhone SDK installation procedure.....	42
Appendix 2. Maemo SDK installation procedure.....	46
Appendix 3. Source code – iPhone OS	51
Appendix 4. Source code – Maemo OS	56

Glossary

API	Application Programming Interface
Apple	A US computer, telephone and software manufacturer
ARMEL	An ARM-architecture emulator. ARM processors are used in iPhone 3G S and Nokia N900.
Community	A diverse group of people that share the same interest and exchange and contribute actively to a certain topic.
GPS	Global Positioning System, used to determine a device's location.
GTK+ (=GIMP toolkit)	A toolkit to create graphical user interfaces
GNOME	GNU Network Object Model Environment
GUI	Graphical User Interface
Hildon	Application framework used in the Maemo platform, based on GNOME
IDE	Integrated Development Environment
Internet Tablet	Product category of mobile devices optimized for Internet usage
iPhone OS	The operating system of the iPhone 3G S
Linux	Linux operating system, also known as GNU/Linux
Nokia	Finnish telephone and software manufacturer
Mac OS X	Apple's desktop operating system
Maemo, Maemo OS	Software platform for mobile devices, developed by Nokia.
maemo.org	Developer community web site, maintained by Nokia.
Open Source Software	Software, which is available in source code form, often freely available, modifiable and distributable.
Proprietary code	Software that is not open source software. One entity holds the copyright.
rootstrap	Part of the Maemo SDK. The target device's root file system.
toolchain	Part of the Maemo SDK. It contains cross-compilations tools.
Scratchbox	Cross-compilation software, part of the Maemo SDK.
SDK	Software Development Kit
Xcode	Apple's IDE for iPhone and Mac OS X development.

1 Introduction

Nowadays, mobile devices are ubiquitous tools for leisure and work, with capabilities that go far beyond the mere possibility of voice communication. There is a transformation in place that moves the mobile phone away from its former single purpose of voice communication and gives it a multitude of purposes. Mobile broadband data transfers and remote access are a reality as well as gaming, networking, instant messaging, music and video playback and numerous other functions thanks to the ever increasingly powerful handheld devices and services. (Steinbock 2007, 163.) With customer expectations continually rising on what a mobile phone can do, it is worth looking at how developers can keep up with the increased demand and what tools they can use to develop professional applications.

Several platforms representing different philosophies as well as different companies are currently vying for market share and for developers it is important to understand the implications in opting for a certain platform. In this thesis the development environments and tools, also known as “Software Development Kits” (SDK), of two exemplary mobile platforms are evaluated: iPhone OS and Maemo. While iPhone OS is from Apple Inc. and has gained widespread public attention, Maemo, an (mostly) open-source platform developed by Nokia, is only gaining more interest since the release of version 5 in late 2009. Both platforms share certain common aspects like the target device’s form factor, the optimization for touch-sensitive interaction and providing communication features across multiple bandwidths, but there are as well distinctions and differences that shall be found and evaluated.

1.1 Objectives

The primary objective is summarized as follows: List and evaluate the differences (and similarities) of the iPhone OS and Maemo Software Development Environment. The goal is not to make a statement which of the environments is superior to the other but rather to give an objective view of the strengths and weaknesses of both development environments on each platform.

The secondary objective is to create an application on both platforms that makes use of the frameworks’ GPS functionality. To achieve this objective, the SDKs have to be installed, configured and used. The whole development environment will become familiar, which supports and extends the primary objective.

1.2 Outline

The document is made up of two sections. The technical and theoretical aspects of the development environments are researched and evaluated in the first section, then the more practical aspects of applying the previously gained insight and knowledge lead to the creation of applications.

1.3 Scope

The area of software development environments is broad, therefore limitations must be set for the items under evaluation for this document. In order to comply with resource restrictions the scope has been determined as follows:

- Evaluate the hardware requirements of the SDKs for iPhone OS and Maemo.
- Evaluate the software requirements of the SDKs for iPhone OS and Maemo.
- Evaluate the available programming languages for iPhone OS and Maemo as well as the portability options of applications.
- Evaluate the support resources offered for iPhone OS and Maemo.
- Evaluate the hardware and software of the iPhone 3G S and the Nokia N900.
- Implement one application with very similar functionality on both devices.
- Use standard installations and standard settings as far as applicable. References are: iPhone OS 3.1.3 and Maemo 5.

Matters that are out of scope:

- Approaches of open-source software versus proprietary software.
- Modified versions of the iPhone OS or Maemo 5.
- Potential earnings for developers of software for the iPhone OS or Maemo.
- Make projections about the future of the iPhone OS and Maemo platforms.
- Latest developments and releases are not taken into consideration.

1.4 Methodology

For the first part, evaluating the SDKs, theoretical research is applied. The aim is to gather and acquire knowledge from previous research and publications in the related field. This literary study shall serve to gain a well-founded understanding of the subject matter, in detail as well as in a broader scope. In the second part, this knowledge is in turn utilized and applied

for the installation of the SDKs and for the creation of a sample application on both platforms.

2 Prior research and sources

2.1 Prior research

There are many publications surrounding the iPhone platform and especially how to write applications for it. Apple itself provides very extensive documentation of the iPhone OS, its Application Programming Interfaces and frameworks, but as well for its programming language "Objective-C". This is mainly due to the large interest and commercial success of the platform and the close relation it has to the Apple desktop operating system "Mac OS X".

The situation of Maemo presents itself as an almost direct opposite, as very little research has been published. Due to the only very recent rise from a tiny niche operating system usable only on a fraction of Nokia devices to a more mainstream, albeit still niche presence, Maemo has until today not garnered much interest. Therefore most publications and research concerning Maemo are only available online.

Leaving mobile operating system market share statistics aside, the constraints of limited research on Maemo naturally limit the research done involving the two platforms. Thus no prior research on this topic could be found.

2.2 Professional literature and other sources

Professional literature is available in large quantities for the iPhone, but it should be noted that many publications focus either on development for or usage of the platform and not the platform itself. Unfortunately, only very limited amounts of professional literature were found concerning the Maemo platform, as exemplified by only two search results, none of them in English, for the term "maemo" in the thesis publication website www.theseus.fi (see figure 1.). There are printed publications and conference proceedings available, some of them were found suitable for use.

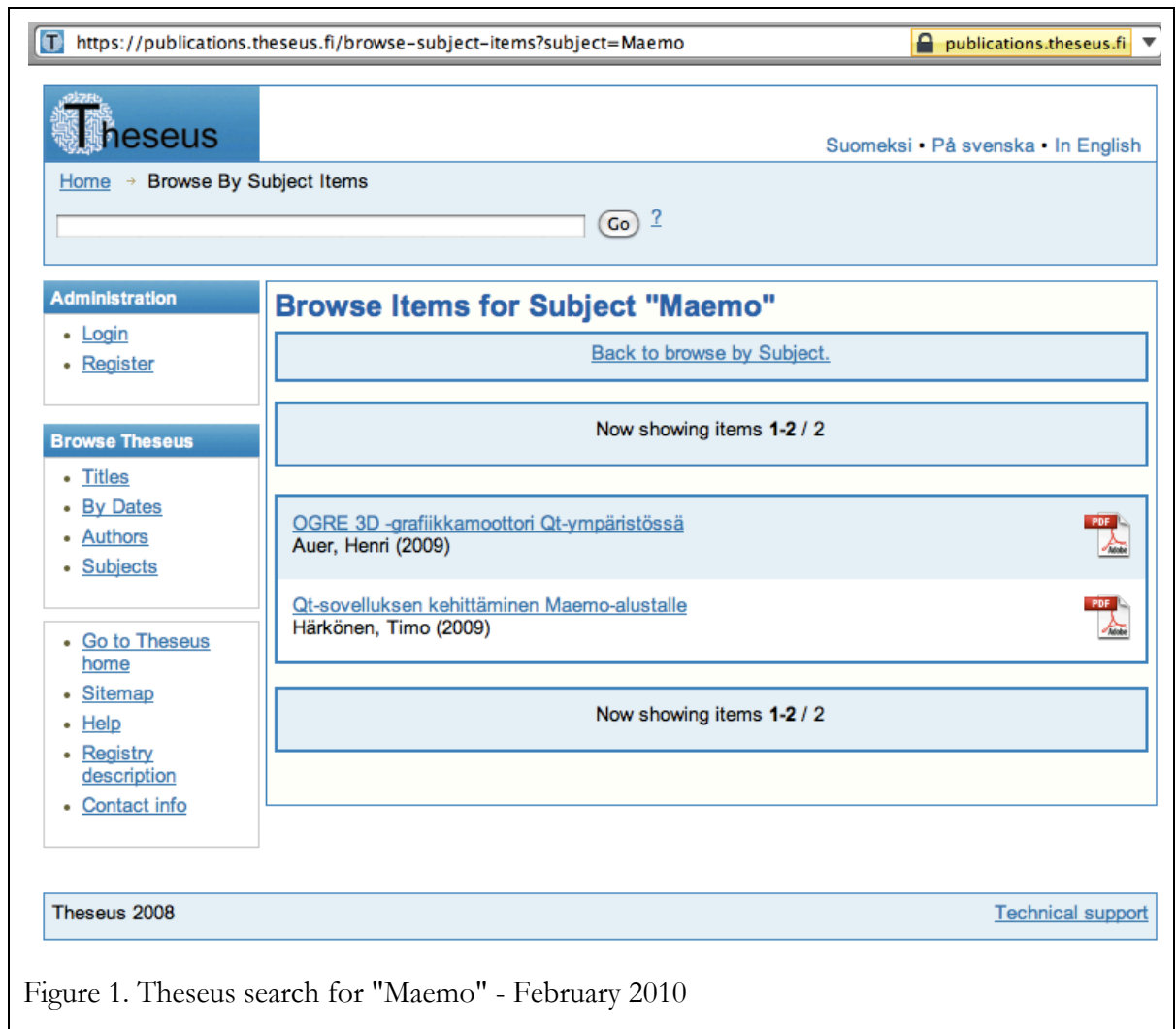


Figure 1. Theseus search for "Maemo" - February 2010

Due to the small presence of printed publications and the only recent rise in public awareness, most sources of Maemo are retrieved from the Internet. Nokia provides the public with thorough information on the platform and the Maemo community maintains credible and reliable sources, too. Additionally, Apple makes a large part of the iPhone OS documentation available online for free, which has been used extensively.

2.3 Source criticism

While attention has been paid to selecting recently published books, it cannot be ignored that the books have the main aim to teach and explain developing for the iPhone rather than to detail the iPhone platform and the Software Development Kit itself. This results in a good explanation of the iPhone OS' capabilities but does only little to explain the underlying configuration. Regarding the printed sources for Maemo, it must be said that most of them are already a little dated and not up-to-date with the recent developments.

The referenced online resources are extensive, both in width and depth. Apple's pages provide a clear and structured way to navigate all information regarding the iPhone OS. Furthermore they provide help, tutorials, guidelines and examples to ease the understanding and application of the information. The documentation is concise, complete and authoritative because Apple is the creator of the iPhone OS, but it should be kept in mind that this effort is nevertheless driven by a commercial entity.

The information available on the mentioned websites referring to Maemo is likewise quite detailed and thorough. The documentation can be considered complete and precise, illustrated with active presentations and -to some extent- structured in its appearance. However, as there are multiple actors in the realm of Maemo, information is spread over several different sites. The open-source nature of Maemo allows for active community contributions and makes the sites lively and dynamic, which at times can be confusing and challenging. New releases of software or updates to the SDK are often not documented at all prior to release and even afterwards it takes time for the community to provide proper documentation. This may result in a delay in obtaining data or in finding outdated text with old references and examples.

3 Software Development Kit Comparison: iPhone OS vs. Maemo

In order to create applications for any of the two platforms under review a desktop or laptop computer is required. It is not possible to develop or compile software on the iPhone and even though on the N900 it is possible (Maemo.org 2010a), it is not an out-of-the-box feature. Moreover, the obvious limitations in screen size, memory, computing power and the difficulties of text input on handheld devices make a full-sized computer the suitable choice for software development for both platforms (Mark & LaMarche 2009, 6).

To this end both Nokia and Apple provide Software Development Kits (SDK), which run on desktop computers and enable developers to create, compile and deploy software for the target environments. The common aspects of the SDKs are to provide a reproducible and consistent environment that provides the tools developers need. This usually includes an Integrated Development Environment (IDE), a compiler as well as tools and accessories to build, test, debug and install the created code on the device. Besides providing tools and services, SDKs themselves require certain hardware and software platforms in order to run. Each company employs its own approach in regards to SDKs, which shall be evaluated and compared in further detail.

Before being able to download the iPhone SDK Apple requires an obligatory registration to their services (Apple 2010a). Nokia on the other hand does not require such procedure (Nokia 2010a). This difference is as well reflected in the different licensing terms of the SDKs where Apple pertains to a proprietary licensing scheme for its SDK that inhibits user or community modifications to it (Apple 2010b), whereas Nokia built its SDK mainly on open-source software, which permits alterations, save for Nokia proprietary components (Maemo.org 2010b).

Apple provides an SDK that is very well integrated into its desktop operating system and provides no alternatives. Compilation of software for the iPhone has to be done using Apple's iPhone SDK. On the other hand, Nokia pursues a different approach and does not limit the developers' choices. With Linux at the core of the Maemo operating system it is possible to revert to (modified) development tools already used and proven in the Linux world and to "cross-compile" the code for the target hardware destination, i.e. the Nokia N900.

3.1 SDKs - Overview

iPhone SDK

The iPhone SDK itself is included in Apple's general development suite called Xcode, which is distributed together with the Mac OS X 10.6 installation discs. Alternatively, after registration with Apple it is downloadable for free from Apple's website. Its download is 3,05 GB in size, which should take around four hours to download at an average connection speed of 200kB/s (=1,6 MBit/s). Xcode is not installed on Apple computers by default however.

Xcode consists of Xcode IDE, Interface Builder, iPhone Simulator, Objective-C 2 programming language, compilers, debuggers as well as further tools and documentation to support programming projects of any size and scale (Apple 2010c). The Xcode IDE is a sophisticated development environment that can be used both for development for the Apple desktop operating system Mac OS X as well as for development for the iPhone OS. Upon creation of a project, template files and folders are created depending on the selected target and coding can start immediately. Compiling the source code is only the matter of a click as is testing the result in the iPhone Simulator, that exactly reproduces the iPhone's original behaviour, save for network and GPS connectivity.

The versions as of February 2010 are: Mac OS X 10.6.2, iPhone SDK 3.1.3 and Xcode 3.2.1.

Maemo SDK

Nokia provides the Maemo SDK and, alternatively, the developer community makes a fully preconfigured virtual machine image of the SDK available. Both forms are only available as a download, with the Maemo SDK being installed via scripts and downloaded on the fly (about 3GB, see appendix 3.), the compressed virtual machine image download file size is 1,6 GB.

The main difference between the two versions is that the installation package requires an existing Linux desktop environment based on x86 CPU architecture while the virtual machine image employs virtualization technology and is thus able to run on a more diverse spectrum of host machines including Microsoft Windows, Mac OS X and Linux.

Other development environments that could be considered as SDKs are a community effort named MADDE, i.e. Maemo Application Development and Deployment Environment, and Nokia's development platform named "Qt". However, MADDE is still in beta status and Qt integration has not yet been fully implemented with Maemo (as of March 2010) and will therefore not be taken into consideration.

Common to both versions taken into account is that the SDK provides a so-called "sandbox environment" for application development and compilation. This sandbox environment recreates or emulates the operating system of the device and provides the necessary development tools (Maemo.org 2010c).

Central to the SDKs is a software name "Scratchbox", which offers the possibility to cross-compile code for a certain target environment, meaning "compiling a program for a CPU architecture different from the one used on the machine where the compilation is done" (Scratchbox.org 2010). To this end the Maemo SDK provides two "toolchains", one for deployment on the device (ARMEL architecture) and for running in the application framework emulator of the SDK on the Linux computer (x86 architecture). The toolchains are used together with "rootstraps", the native system's root file system, to allow the compilation and the execution the compiled code in the Scratchbox environment.

Furthermore the SDK contains the Nokia proprietary binary packages as well as the necessary utilities to (cross-)compile applications and install them on the target device (Nokia 2010b).

The Maemo community maintains a plug-in for Eclipse in order to tailor the popular IDE for use in Maemo development (Fitzek & Reichert 2007, 195). Otherwise, developers may use

their IDE of choice and compile code via the command line. It is well worth mentioning that the virtual image SDK contains a fully preconfigured incarnation of the ESbox IDE (based on the Eclipse Ganymede IDE) with the necessary plug-ins for Maemo development, deployment and packaging, which greatly reduces user interaction via the command line to a minimum.

The versions as of February 2010 are: Maemo OS 5 3.2010.02-8, Maemo 5 Final SDK Update 5, Maemo SDK Virtual Image 1.0 (Ubuntu Desktop version)

3.2 Hardware requirements

iPhone SDK

Astonishingly, Apple's iPhone SDK has only one hardware requirement (iPhone SDK Readme file): A Mac with an Intel processor.

The absence of explicit hardware specifications may surprise, but an explanation behind this may be the fairly recent switch in 2007 by Apple to incorporate only processors from Intel in its hardware (Apple 2010d; Apple 2010e). The assumption presumably being that all hardware with Intel processors is powerful and recent enough to run the required software. An example of the lower end of the hardware that is supposed to support the iPhone SDK is the first MacMini with an Intel chipset that was clocked at 1,5 GHz and was equipped with 512 MB of RAM (introduced in February 2006). No reports about the viability of iPhone development on such a machine could be found. Despite the very brief minimum requirements description by Apple it should be assumed that disk space of around 5 GB is necessary for installation and that additional disk space in the order of 1GB should be available for execution.

Maemo SDK

In contrast to the iPhone SDK, the Maemo SDK has some more detailed hardware requirements and recommendations, but no specific brand or manufacturer prerequisites. These hardware requirements are mainly based on Scratchbox' requirements and are detailed as follows (Nokia 2010b):

- x86 platform.
 - For multiple users using Scratchbox on the same host a dual CPU machine or Hyper-Threading capable processor is recommended.
- 512 MB of memory.

- For multiple users using Scratchbox on the same host at least 1 GB is recommended.
- A full Scratchbox installation requires 1 GB of hard disk space, but at least 1 GB extra space should be reserved for each Scratchbox user,
 - For heavy development a minimum of 10 GB of space may be needed.
- Furthermore it is recommended, but not required, to have a working networking environment and a NFS server.

These requirements are as well not too detailed or restrictive and can most probably be met by hardware that is more than 4 years old.

3.3 Software requirements

iPhone SDK

Again, the iPhone SDK surprises with a very brief description of its software requirements, it basically states that it needs a computer “running Mac OS X Snow Leopard version 10.6.0 or later”.

This is a quite simple requirement in itself, however, the terms of Apple’s software license agreement do not allow installing its operating system on non-Apple hardware, turning the software requirement into a hardware requirement at the same time (Apple 2010e, 1).

Although precise measures are not available and it may largely depend on user preference, this software requirement may very well narrow the scope of viable computers for iPhone SDK installation and usage as old Apple computers with Intel processors may not be able to run the required “Mac OS X 10.6.0 or later” at acceptable speeds. But as mentioned before, no sources or reports were found to corroborate this assumption.

Maemo SDK

The Maemo SDK requires the Linux operating system, no further software requirements are stated (Maemo.org 2009d, 41). But while any Linux distribution may be able to run the SDK, it is advisable to employ a Debian-based Linux (Fitzek & Reichert 2007, 180). This stems from Maemo OS itself being based on Debian, so a higher degree of compatibility can be achieved as well as less customization of the SDK is necessary due to missing modules or packages in certain Linux variants. To this extent it should be pointed out that the “binfmt_misc”-module is vital, which is, for example, missing in RedHat Enterprise Linux 3 (Nokia 2010b).

A recent version of a Linux OS is recommended, but even the community-provided virtual image SDK employs Ubuntu 8.10 (Nokia 2010c), a version that, at the time of writing, has been superseded by three versions already.

3.4 Available programming languages and portability options

iPhone SDK

By default, the iPhone SDK supports the Objective-C 2.0 programming language as well as C/C++. Objective-C 2.0 is employed across Apple's entire portfolio and is used for desktop development, too. (Apple 2010f.) It is a superset of the C programming language and supports many similar features.

It is worth mentioning that there are efforts to enable the use of different programming languages and make them available to the iPhone SDK or at least to allow development in other languages and environments for iPhone programs (Wikipedia.org 2010). Additionally there are quite sophisticated frameworks available, which, to a certain extent, allow developing software for the iPhone outside of the iPhone SDK. More alternatives certainly exist, this is just a limited selection of viable alternatives:

PhoneGap

“PhoneGap (created by Nitobi) is an open source development framework for building cross-platform mobile apps. Build apps in HTML and JavaScript and still take advantage of core features in iPhone/iTouch, iPad, Google Android, Palm, Symbian and Blackberry SDKs.” (PhoneGap 2010). The iPhone APIs are abstracted and no code change is necessary, but the iPhone SDK is still needed for deployment (Stark 2010, 116).

RhoMobile

Its product “Rhodes” is an open source framework that enables the creation of native applications for a wide array of mobile phones, including the iPhone. It is enabling the use of HTML, JavaScript and Ruby for iPhone applications, but it is not free to use. (Rhomobile 2010.)

Titanium

Titanium Mobile is a free and open source application development platform that allows the creation of software for desktop and mobile systems. Source code in HTML, CSS, JavaScript, Ruby and Python is converted to native iPhone code. However, the iPhone SDK is required for compilation. (Appcelerator Inc. 2010.)

MonoTouch

As a branch of the Mono project MonoTouch provides the possibility to write C# and .NET applications that eventually run on the iPhone. It requires the MonoDevelop IDE and the iPhone SDK and is not a free product but available in three licensing models. (Mono Project 2010.)

Adobe Flash

The commercial product “Adobe Flash Professional CS 5” ships with a Packager for iPhone that will allow the translation of ActionScript projects as native iPhone applications. (Adobe Inc. 2010.)

Nevertheless, some of these approaches are mainly built around HTML and JavaScript usage and thus have only limited access to native functions of the iPhone. Plus, an Apple computer and the iPhone SDK are still required for compilation and deployment in most cases.

Furthermore, recent developments in the software license agreements from Apple point in the direction that applications for the iPhone must be written in C, C++, Objective-C or JavaScript, harshly limiting the use of third-party frameworks and software (Apple 2010g).

Maemo SDK

The official programming language for Maemo application development is C, but several other languages can be used with Maemo also (Maemo.org 2009d, 43). For example, the Virtual Image SDK comes with built-in support for C, C++ and Python. Support for other programming languages or portability options are as the list below. More language bindings are very probable to exist as community projects, this should be considered a selection of readily available alternatives:

C++

The “maemomm”-project created a C++ wrapper for Maemo. It helps developers familiar with Symbian and/or Windows Mobile .NET experience and en-

ables to use the C++ programming language on the Maemo platform. Extra packages have to be downloaded and installed both for the SDK and the device to enable C++ usage. (Maemo.org 2010g.)

Python - PyMaemo

The effort to port Python to the Maemo platform is partially funded by Nokia and community-supported and resulted in PyMaemo. PyMaemo is available for the SDK environment as well as for the device (Nokia N900), but has to be downloaded and installed manually. (Maemo.org 2010f.)

Qt

Nokia's own "cross-platform and application UI framework" will find significantly increased use within Maemo. It is intended that the next firmware release will fully incorporate Qt. Qt itself is available on platforms as diverse as Windows, Mac OS X, Linux, Windows Mobile, Symbian, Maemo and Embedded Linux and one of the key aims of Qt is, that code needs only to be written once and can then be deployed to all available platforms. It comes with a C++ class library but has bindings for many other languages including C#, Python, Perl, PHP, Ruby and others. Included as an IDE is "Qt Creator", which besides coding is as well suitable for GUI layout. (Nokia 2010d.)

MADDE – Maemo Application Development and Deployment Environment

MADDE is currently only a technology preview but as such interesting and promising for developing for Maemo on Windows, Mac OS X and Linux. Similar to the existing SDKs it provides cross-compilation functionality for several targets. (Maemo.org 2010e.)

3.5 Support resources

iPhone SDK

Apple provides extensive support resources, which are reachable via several paths. The complete Xcode and iPhone documentation is available online via the iPhone Developer Program. The same documentation is as well accessible in Xcode via the built-in "Help documentation" and can, at least partially, be viewed as PDFs, too.

These pages comprise a huge amount of knowledge ranging from introductory texts, platform and framework presentations, guidelines and coding examples to complete reference manuals of all iPhone Application Programming Interfaces (API). There is no shortage of help provided by Apple and it is complemented by individual technical support and discussion forums, albeit only available to members of the Apple iPhone Developer Program at 99,- USD per year .

Apart from official Apple resources there are many books, publications and websites aiming to provide support, either free or at a premium. However, it would be beyond the scope to discuss and analyze third-party offerings.

Maemo SDK

The available resources to guide developers are mainly gathered on the maemo.org-website, which is hosted by Nokia (Maemo.org 2010h). But due to the nature that Maemo itself consists or makes use of other (open source) software projects, which have been modified for use on Internet tablet devices, the parent websites of these projects often provide further and more detailed information (i.e. GTK+, GNOME, SQLite, Debian, Mozilla, Telepathy, etc.).

The maemo.org-website distinguishes between five major sections, with two of them attributable to support: Development and Talk. The Development-section caters to enabling people to develop programs for the Maemo platform and to this end provides documentation of the major development aspects like SDK download and installation, API documentation and reference guides. The open nature of Maemo is very visible in this section as it is organized as a Wiki, meaning that community contribution is possible and encouraged. One downside of this approach is that the quality of contributions varies greatly and that at times articles on certain topics are outdated, incomplete or, even worse, not existing at all. Plus, since the target audience is assumed to be proficient in program development in a Linux environment, some descriptions can be considered to be on a rather expert level and might be hard to understand for people just getting started with development.

The second section of the maemo.org website that accounts for support is the Talk-section, which is a user forum, divided into several sub-forums for better accessibility. This can be considered an active and lively forum with many exchanges on hardware, software, coding and troubleshooting taking place. It should be considered the prime location to get expert support within short delays. As of April 2010, the maemo.org-website had more than 33'000 registered users (see figure 2.).



Figure 2. Number of registered users on the maemo.org-website

A further alternative to find authoritative answers is provided by Nokia at http://www.forum.nokia.com/Technology_Topics/Device_Platforms/Maemo.xhtml. There, SDK downloads are available as well as screencasts explaining the Maemo platform and guiding through application development. A Wiki, discussion forums and a knowledge base are maintained there, too, however, they appear to be slightly more distant from the source than the maemo.org-website. And finally, just like Apple, Nokia provides technical support for paying customers with a varying range of services. Technical support for one case costs 170,- EUR, while a 12-month subscription is 5.100,- EUR (Nokia 2010g).

3.6 Devices

3.6.1 iPhone 3G S

Apple first sold the iPhone in June 2007 and has presented an updated version of its hardware every summer since. Very much in sync with the introduction and evolution of the hardware, Apple first presented the iPhone OS in 2007 and updated it accordingly in the following years. From the start it was conceived as a consumer-oriented “smartphone” and WLAN as well as cellular network connectivity were implemented tightly into the operating system and applications. Still, the iPhone evolved over the years, introducing new functionality with each iteration. The iPhone fully relies on tactile, on-screen input and does not provide a hardware keypad or keyboard. (Apple 2007h; Apple 2008i; Apple, 2009j.)

Currently (March 2010) the iPhone 3G S is the latest model, running iPhone OS 3.1.3.



The most relevant technical specifications are as follows:

Size:	Height x Width x Depth: 115.5 x 62.1 mm x 12.3 mm
Weight:	135 grams
Memory:	256 MB
Mass Memory:	16GB or 32GB flash
Processor:	ARM Cortex A8 600 MHz
Graphics:	PowerVR SGX535 graphics
Display:	3.5-inch widescreen TFT capacitive Multi-Touch display 480-by-320-pixel resolution at 163 ppi (Pixels Per Inch) 16 million colours
Input method:	QWERTY tactile keyboard, QWERTY onscreen keyboard
Cellular:	UMTS/HSDPA (850, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
Wireless:	Wi-Fi (802.11b/g) Bluetooth 2.1 + EDR wireless technology
Location:	Assisted GPS, Digital compass, Wi-Fi, Cellular
Video playback:	H.264 video, up to 1.5 Mbps, 640 by 480 pixels, 30 frames per second, MPEG-4 video, up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second
Audio playback:	Frequency response: 20Hz to 20,000Hz Audio formats supported: AAC, Protected AAC, HE-AAC, MP3, MP3 VBR, Audible, Apple Lossless, AIFF and WAV
Camera:	3 megapixels, Autofocus
Video:	Video recording, VGA up to 30 fps with audio
Connectors and input/output:	

30-pin dock-connector, 3,5mm stereo headphone jack
Sensors: Accelerometer, Proximity sensor, Ambient light sensor
Power and battery: Built-in rechargeable lithium-ion battery
Charging via USB to computer system or power adapter
Talk time: up to 12 hours on 2G, up to 5 hours on 3G
Standby time: up to 300 hours
Internet use: up to 5 hours on 3G, up to 9 hours on Wi-Fi
Video playback: up to 10 hours
Audio playback: up to 30 hours
(Apple 2009k; GSMArena.com, 2009.)

3.6.2 Nokia N900

In 2005 Nokia introduced the first “Internet Tablet”, the N770. This device used a Linux derivative called Maemo OS 2005, the first version of the current day Maemo 5 OS (Bosch 2006, 4). Further devices in this category were released, namely the N800, the N810, the N810 WiMax and the N900. Common to all devices was a touch-screen as the main input mechanism, assisted with various buttons. Only the models from the N810 onwards (October 2007) possessed a slide-out QWERTY keyboard, taking a first step in converting the device from a “one-way” surfing tool to a “two-way” communication device (Maemo.org 2008j, 15). The N810 WiMax model made a further step in adding mobile connectivity, but only the addition of GSM-based cellular phone capabilities in the N900 completed this transition (at the cost of WiMax connectivity, which was eventually removed from the device).

Development of the operating system continued in parallel with the device’s improvement and eventually lead to the introduction of Maemo 5 (also known as “Fremantle”) in late 2009.

The current N900 utilizes the Maemo 5 operating system, version number 3.2010.02-8.



Figure 5. Nokia N900 - © Nokia

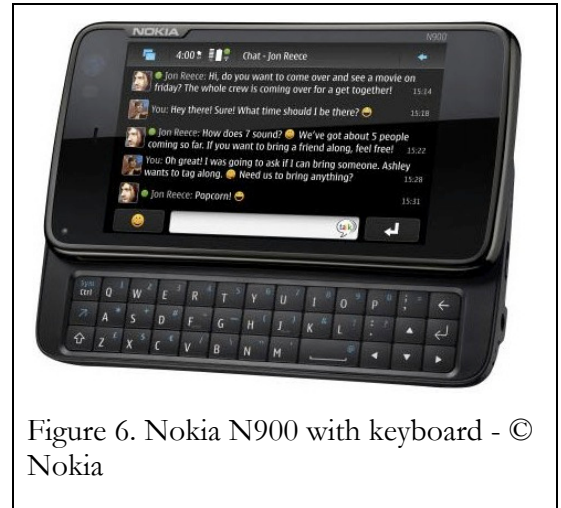


Figure 6. Nokia N900 with keyboard - © Nokia

The main technical specifications of the N900 are as follows:

- Size: 110.9 x 59.8 x 18 (19.55 at thickest part) mm
- Volume: Approx. 113cc
- Weight: Approx. 181g
- Memory: Up to 1GB of memory (256 MB RAM, 768 MB virtual memory)
- Mass memory: 32 GB internal storage, up to 16 GB of storage on microSD card
- Processor: TI OMAP 3430: ARM Cortex-A8 600 MHz
- Graphics: PowerVR SGX with OpenGL ES 2.0 support
- Display: 3.5 inch touch-sensitive, resistive widescreen display
800 x 480 pixel resolution at 225 ppi (Pixels Per Inch)
16 million colours
- Input method: QWERTY tactile keyboard, QWERTY onscreen keyboard
- Cellular: Quad-band GSM EDGE 850/900/1800/1900
WCDMA 900/1700/2100 MHz
- Wireless: WLAN IEEE 802.11b/g
Bluetooth 2.1 +EDR, Bluetooth Stereo Audio
Built-in FM transmitter
- Wired: Micro USB, UPnP, USB 2.0, USB Mass Storage
- Location: Assisted GPS, Cellular
- Video playback: .mp4, .avi, .wmv, .3gp; codecs: H.264, MPEG-4, Xvid, WMV, H.263
Wide aspect ratio 16:9 (WVGA)
- Audio playback: .wav, .mp3, .AAC, .eAAC, .wma, .m4a
- Camera: 5 megapixel camera (2584 x 1938 pixels), Dual LED flash
3x digital zoom, Autofocus with assist light, two-stage capture key
- Secondary camera: 0,3 megapixel (640 x 480 pixels)

Video: Video recording at up to 848 x 480 pixels (WVGA), up to 25fps
Video recording file format: .mp4; codec: MPEG-4

Connectors and input/output: Micro-USB (USB 2.0), 3.5mm stereo
headphone jack, TV out (PAL/NTSC) with Nokia Video Connec-
tivity Cable

Sensors: Accelerometer, Proximity sensor, Ambient light sensor

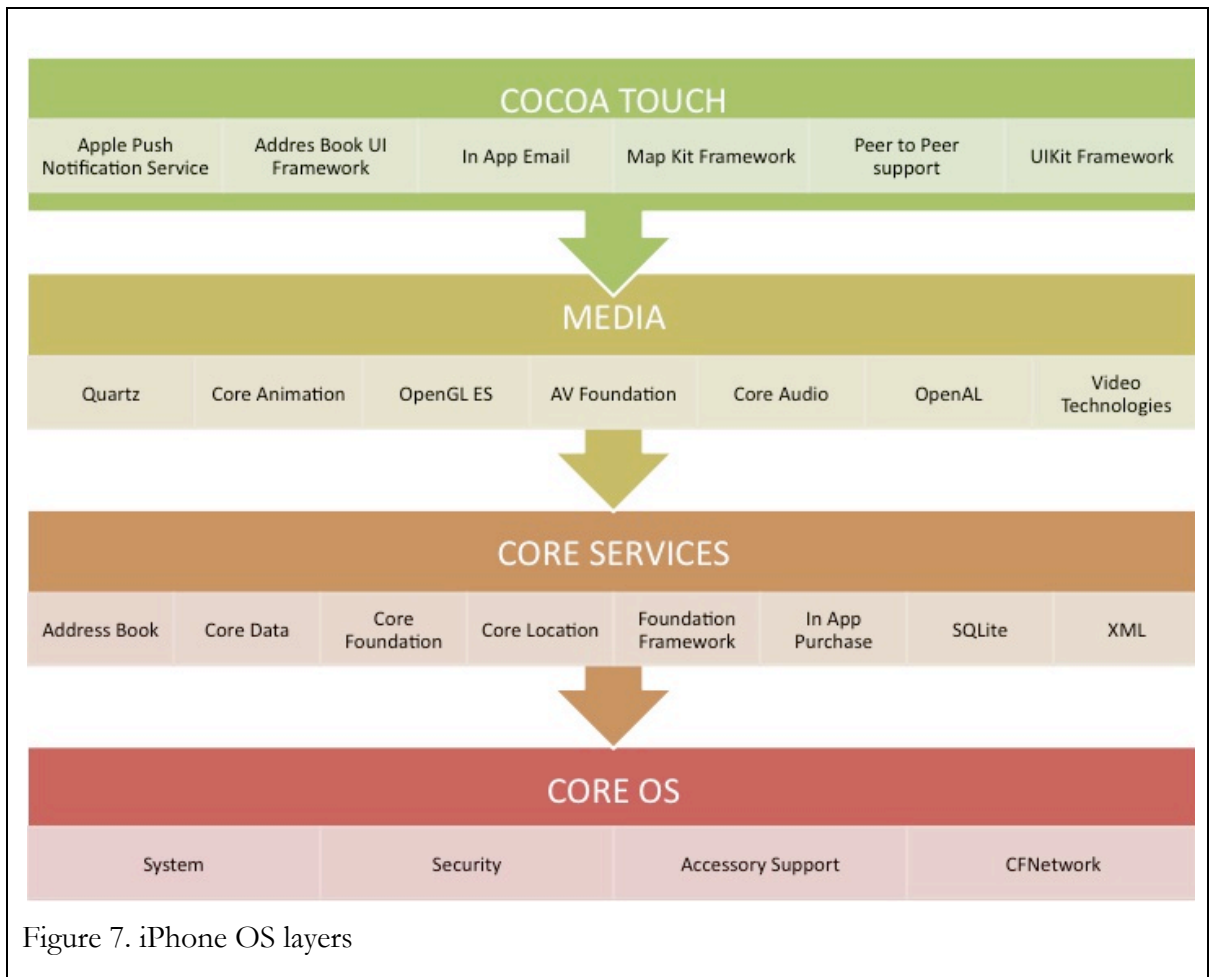
Battery: 1320mAh
Charging via USB to computer or power adapter
Talk time: up to 6,3 hours on 2G, up to 4,3 hours on 3G
Standby time: up to 288 hours
Internet use: up to 5,5 hours on 3G, up to 8 hours on Wi-Fi
Video playback: up to 5,6 hours
Audio playback: up to 24,5 hours

Operating system: Maemo 5 software on Linux
(Nokia 2010e; Nokia 2010f.)

3.7 Operating systems

iPhone OS

Apple's proprietary operating system for handheld devices was first released in 2007, together with the first generation iPhone (Apple 2007h). It has been continually updated and is available in version 3.1.3 as of March 2010. The underlying concepts of the iPhone OS are related to its desktop operating system Mac OS X, but altered and adopted to both suit a mobile device and a touch-only input method. iPhone OS basically consists of four layers, with the Core OS at the lowest level. On top of that follow Core Services, Media and finally Cocoa Touch (see figure 7.).



Core OS as lowest layer contains four frameworks that are vital to the functioning of the device. The System level is responsible for all aspects of the operating system as it includes the kernel and drivers. The kernel is based on Mach, a microkernel technology developed at the Carnegie Mellon University and used as well in Mac OS X' kernel, and it controls low-level tasks like memory allocation, networking, threading, file access and communication between processes. Three further frameworks at this layer are Security, Accessory and CFnetwork, which take care of access execution rights, handle hardware accessories attached to the device and manage networking and socket access, respectively.

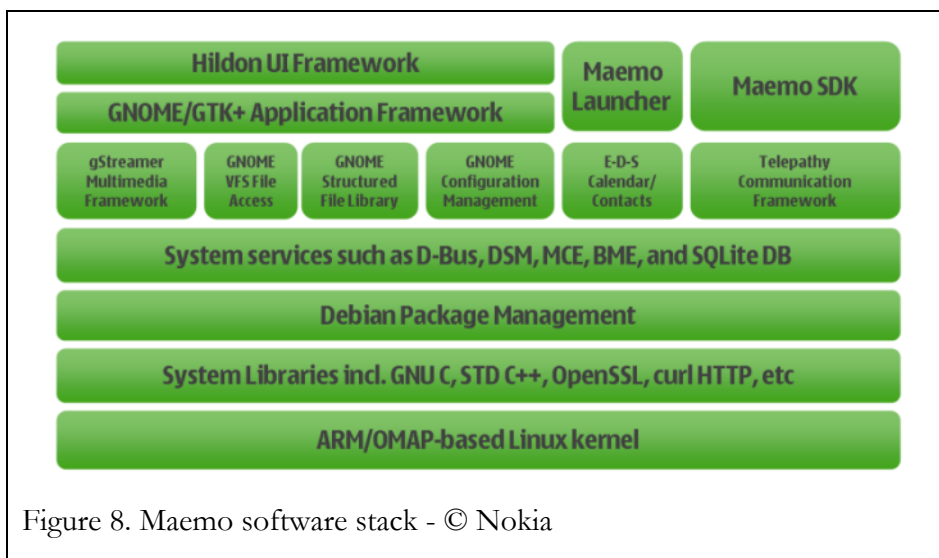
The second layer is Core Services, which is often used by applications. Most of the times the services in this layer are not contacted directly but via abstractions on the superior levels. This is meant to make the development process easier, but, if desired, these services can be used directly. Instead of providing functionality for the operation of the device like the kernel, the Core Services layer comprises functions that are vital for most applications.

Next in line is the Media layer, which provides access to multimedia services. This includes not only playback of audio and video files but as well manipulation of multimedia files and on

the fly rendering of graphics, for example for games. The layer is mainly divided into three sections, namely Graphics, Audio and Video. The Graphics section with the Quartz vector-based drawing engine greatly contributes to the look and feel of applications and seamlessly provides graphics and animations for functions of the superior level. The Audio section provides the functionality for the simultaneous playback of multiple audio sources and as well enables sound recording and processing. The third section, Video, manages the playback of a multitude of video formats on the iPhone display. On top of these three layers resides Cocoa Touch, with the very central UIKit framework. This serves as starting point for all applications and helps the developer with the implementation of many essential features, first and foremost the handing of touch input. Besides this, Cocoa Touch provides abstractions of low- and lower-level services and allows programming in Objective-C. (Apple 2009l.)

Maemo OS

The software architecture employed for Maemo 5 is largely based on Debian GNU/Linux, using a modified kernel 2.6, owing to handheld, single-person, touch-screen use (Tarkoma 2009, 56). Furthermore certain additions had to be made to accommodate cellular activity and implement power-saving features (Maemo.org 2009i). The open source nature of Linux is well reflected in Maemo, as it is open source, too, save for several proprietary binaries from Nokia. In parts, this approach aims at quick and easy porting of applications form desktop Linux systems to the Maemo platform and vice versa, with many applications developed as open source projects.



From the bottom up, the most important layers and services shall be briefly described (see figure 8.). The Kernel is one of the items that is loaded first during device start-up. Its main task is to deal with hardware events, handle process creation and manage memory allocation.

As the central part of the operating system, it is providing device drivers for hardware access and managing networking, too.

The next layer consists of system libraries, core services, core daemons, core utilities, BusyBox, Bluetooth stack, Hardware Abstraction Layer (HAL) and several other components to ensure a secure and smooth operation. The system libraries are, among others, made up of the GNU C library and a C++ library, given that C is the native programming language for Maemo. The core services provide a low-level access to functions of the operating system, like HTTP access for example, BusyBox is a compact replacement of the standard Linux shell and the Bluetooth stack handles, just as the name suggests, Bluetooth operations. HAL allows provides abstraction from the underlying hardware and in turn enables access to hardware features via an API. More utilities are located at this level, enhancing the functionality or security of the system.

One layer up is the Debian Package Management, which is controlling the installation and removal of software on the device. Built around the Advanced Packaging Tool (APT) this layer takes care of managing various software repositories and enables the user to install local files as well as from remote locations. One important aspect of APT is the handling of dependencies, which implements automatic retrieval of dependent packages.

Next in line are system services, with D-BUS probably the most important as it handles communication and interaction between the system and applications, but as well LibOSSO deserves mentioning as a means to register applications with D-BUS and to manage application state serialization for putting and retrieving apps “from the background” and for improved battery life. Other system services are Device State Management (DSM), Battery Management (BME), Mode Control (MCE), SQLite database.

On top of these system-wide layers are further specialized frameworks and services that control specific tasks, such as window management, User Interface handling, video and audio recording and playback, file system access, address book management, email and instant messaging and many more to complete the usage possibilities of a mobile device. Both the GNOME/GTK+ Application framework and the derived Hildon UI framework are contributing large parts to the look and feel of the device interface and to the user experience. (Maemo.org 2010k; Maemo.org 2008l; Fitzek & Reichert 2007, 187; Tarkoma 2009, 57).

3.8 Summary

The iPhone and the Maemo SDK are both encompassing development environments that provide vast resources to the interested developer. Be it the very integrated Xcode environment for the iPhone or the multitude of choices for Maemo, there are very little omissions, if any. From the powerful and versatile hardware to comprehensive software stacks and intuitive user interfaces, both phones produce an enticing user experience, but with differing flavours. The same can be said about the resources that are available to guide developers in their task of writing applications for either platform.

Both environments developed over time and given the recent progress and announcements it is clear that they will evolve to integrate further functionality and improve the existing base. The Apple environment proves to be more controlled with a higher integrity across functions whereas the Maemo platform offers the vast freedom and choice of open source software as well as (almost) unrestricted access to the phone's hard- and software.

3.9 Conclusion

It must be said that understanding, analyzing and presenting the realm of the Software Development Kits for the iPhone and for Maemo is a huge task and can hardly be reflected in one single document. Best efforts were made to cover important and relevant aspects but completeness could not be achieved and is not assumed.

Following is a list of differences and similarities between the Software Development Kits for Maemo OS and iPhone OS. The individual items of the list are explained in more detail.

Differences:

1. The SDKs need distinctive, different operating systems to run on.

While the iPhone SDK can only be run on the latest Apple operating system, the Maemo SDK is only restricted to a Linux operating system, better yet for compatibility, Debian Linux. However, it runs on current and previous releases of Linux OS.

2. The iPhone SDK needs distinctive, different hardware to run on.

The iPhone SDK can only be run on Apple computers with Intel processors. The Maemo SDK can only be run on x86-architecture computers (thus including Apple computers with Intel processors).

3. The SDKs have different native programming languages.

While Maemo natively supports the C programming language, the iPhone SDK has Objective-C 2.0 as native programming language. Nevertheless it appears that there are several options to use different programming languages for development purposes on both environments, providing developers with a choice.

4. The iPhone SDK includes a tool for creating Graphical User Interfaces, while similar functionality requires additional software for the Maemo SDK.
5. The iPhone SDK download requires registration with Apple, the Maemo SDK does not require registration.
6. Application deployment costs 99,- USD per year for the iPhone SDK, application deployment does not incur any fee for the Maemo SDK.
7. Documentation and support resources

The documentation of APIs and the presentation of support resources are very structured and clear for the iPhone SDK. Obviously, quality control is implemented before release. In comparison, the support resources for the Maemo SDK appear unorganized, outdated, incomplete, unstructured and are scattered over many pages or websites.

8. The software stacks differ on both phones.

Even though the phones perform very similar tasks at very similar execution speeds, the software platforms and their components differ.

9. Open source vs. closed source

The Maemo SDK and OS are in large parts open source software and rely only on little, special-purpose, proprietary Nokia software. This enables active user and community contributions to both the SDK and the OS. The iPhone SDK and OS are completely closed source software and prohibit user modification. All development is in the hands of Apple.

Similarities:

1. The SDKs are freely available at no cost.

The SDKs can be downloaded from the Internet. Both SDKs are more than 3GB in size.

2. The hardware of both devices is relatively similar.

Without going too much into detail regarding the hardware of the iPhone 3G S and the Nokia N900 and without sophisticated performance comparison, it should still be safe to state that they do not differ that much in terms of capa-

bilities. Both devices feature the same processor, working at identical speeds and both devices make use of similar hardware graphics acceleration. While the Nokia N900 features, just like the iPhone, 256MB RAM, it can be dynamically increased to run applications. Even though there are fundamental differences in the touch-screens (capacitive vs. resistive), both instances perform their task with precision and ease, each naturally with distinct advantages and disadvantages, but these are not of real concern in this discussion. The form factor does not differ much and even the differences in screen resolution can almost be neglected in everyday use, as the screen size is nevertheless identical. The above comparisons should be understood as very approximate and nowhere near scientific research. However, the main specifications are rather similar and can be considered to provide a close enough match to be comparable.

3. Both SDKs permit executing code in a simulator environment.

4 SDK installation

4.1 iPhone SDK

The latest release of the iPhone SDK can be downloaded here:

<https://developer.apple.com/iphone/index.action#downloads>

As mentioned before, accessing this page and downloading the iPhone SDK requires registration with Apple. Once the download has finished, the Apple Disk Image file (.dmg) needs to be opened and “mounted“. Its contents are an installation guide as well as a package file containing Xcode IDEs, the iPhone SDK and development utilities. Opening the package file will start the installation procedure, which graphically guides the user through the installation process. Besides confirming two software license agreements very limited input is necessary from the user: the desired installation packages can be chosen and the installation location can be selected, but it is a good choice to leave the default values for smooth operation. Once the installation has completed, the iPhone SDK and the iPhone simulator can be started via Xcode, the Integrated Development Environment for both Mac OS X and iPhone OS. (See figures 9 and 10.) Illustrations of the whole installation process can be found in appendix 1.

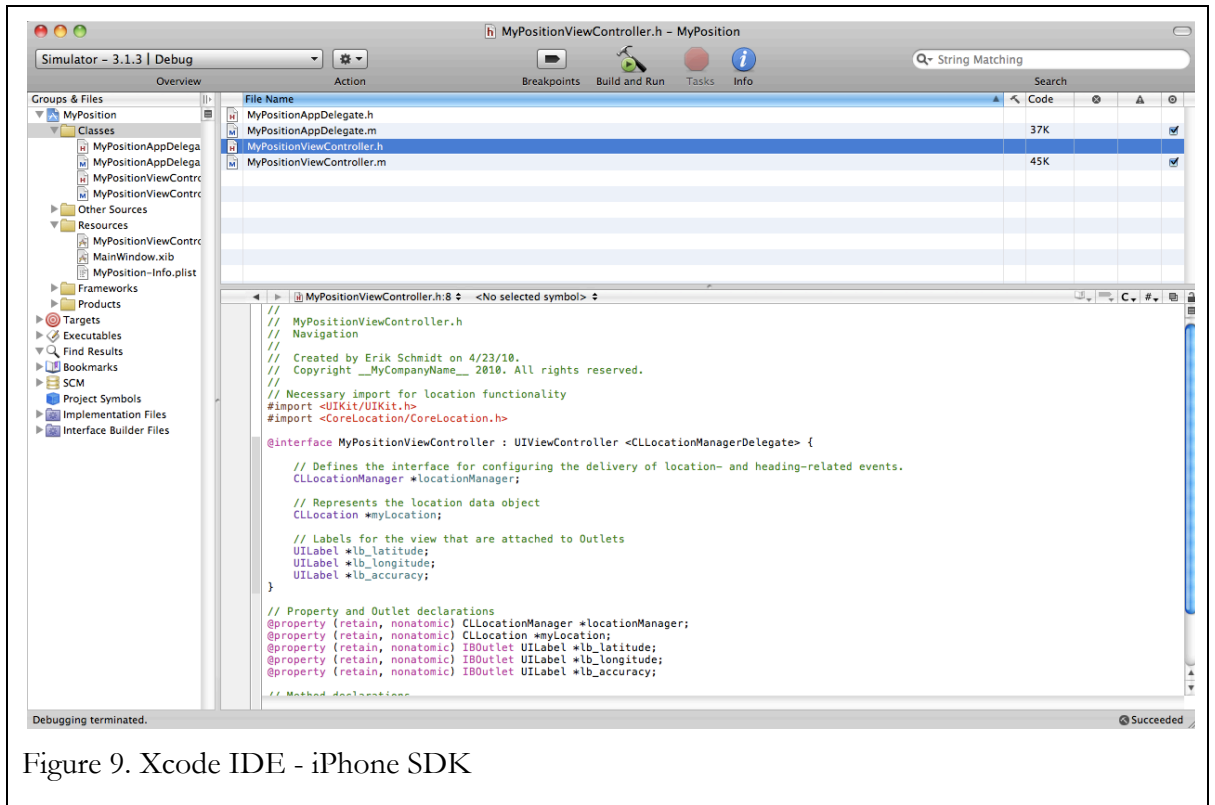


Figure 9. Xcode IDE - iPhone SDK



Figure 10. Xcode IDE - iPhone Simulator - Application example

4.2 Maemo SDK

There are probably several ways to install a fully functioning Maemo SDK, but the focus will lie on the installation scripts provided by Nokia and the Virtual Image Maemo SDK provided by the community and supported by Nokia. Both installation methods have almost nothing in common and as well lead to differing results. First, the installation scripts are presented.

4.2.1 Installation scripts

A prerequisite is a current and running (Debian-based) Linux installation, in this case Ubuntu 9.10 was selected, with all recent updates installed. Then, to obtain the installation scripts a browser needs to be directed to the following address:

http://www.forum.nokia.com/info/sw.nokia.com/id/c05693a1-265c-4c7f-a389-fc227db4c465/Maemo_5_SDK.html.

There are three scripts available, but only the GUI installer is currently of interest. This GUI installer Python script can alternatively be downloaded via the Terminal's command line:

```
$ wget http://repository.maemo.org/stable/5.0/maemo-sdk-install-  
wizard_5.0.py
```

After changing the access right of the script, it subsequently has to be executed in a Terminal using superuser rights,:

```
$ chmod a+x maemo-sdk-install-wizard_5.0.py  
$ sudo ./maemo-sdk-install-wizard_5.0.py
```

As the Python Qt4 bindings are missing, it is wise to agree to the suggested installation. Once the bindings are downloaded and installed, the GUI installer starts and guides the user through the installation process as per the figures in appendix 2. User interaction is minimal, reduced to clicking through the dialogs and along the way selecting the type of installation (standard or custom), accepting the Open Source License Agreement and whether Nokia binaries and Nokia applications shall be installed in the SDK. If the user chooses to install the Nokia binaries, a number displayed on screen has to be typed into a textbox for confirmation of accepting Nokia's End User License Agreement.

The figures in appendix 2 illustrate the installation process. Once the installation has been completed, a link on the desktop allows starting the Maemo application framework, an emulated version of the Maemo 5 operating system (see figures 29 and 30 in appendix 2.).

The application framework can as well be started via the command line, but a Xephyr Xwindows server instance must be started first, though:

```
$ Xephyr :2 -host-cursor -screen 800x480x16 -dpi 96 -ac -kb &
```

Following this command, the user needs to join the group “sbox” to be able to subsequently login to the Scratchbox environment:

```
$ newgrp sbox
$ /scratchbox/login
```

Once logged into Scratchbox, the user can start and stop the application framework, switch the targets between ARMEL (N900 device) and X86 (host computer), compile code or install software for the currently active.

Starting and stopping the application framework:

```
[sbox-FREEMANTLE_X86: ~] > af-sb-init.sh start
[sbox-FREEMANTLE_X86: ~] > af-sb-init.sh stop
```

Switching targets:

```
[sbox-FREEMANTLE_X86: ~] > sb-conf se FREEMANTLE_ARMEL
[sbox-FREEMANTLE_ARMEL: ~] > sb-conf se FREEMANTLE_X86
```

Compile source code:

```
[sbox-FREEMANTLE_X86: ~] >
gcc -Wall -g <source_file_name>.c `pkg-config --cflags --libs
gtk+-2.0` -o <output_file_name>
```

Package installation:

```
[sbox-FREEMANTLE_X86: ~] > fakeroot apt-get update
[sbox-FREEMANTLE_X86: ~] > fakeroot apt-get install <package_name>
```

A graphical configuration tool for the Scratchbox environment is available (see figure 11.):

```
[sbox-FREEMANTLE_X86: ~] > sb-menu
```

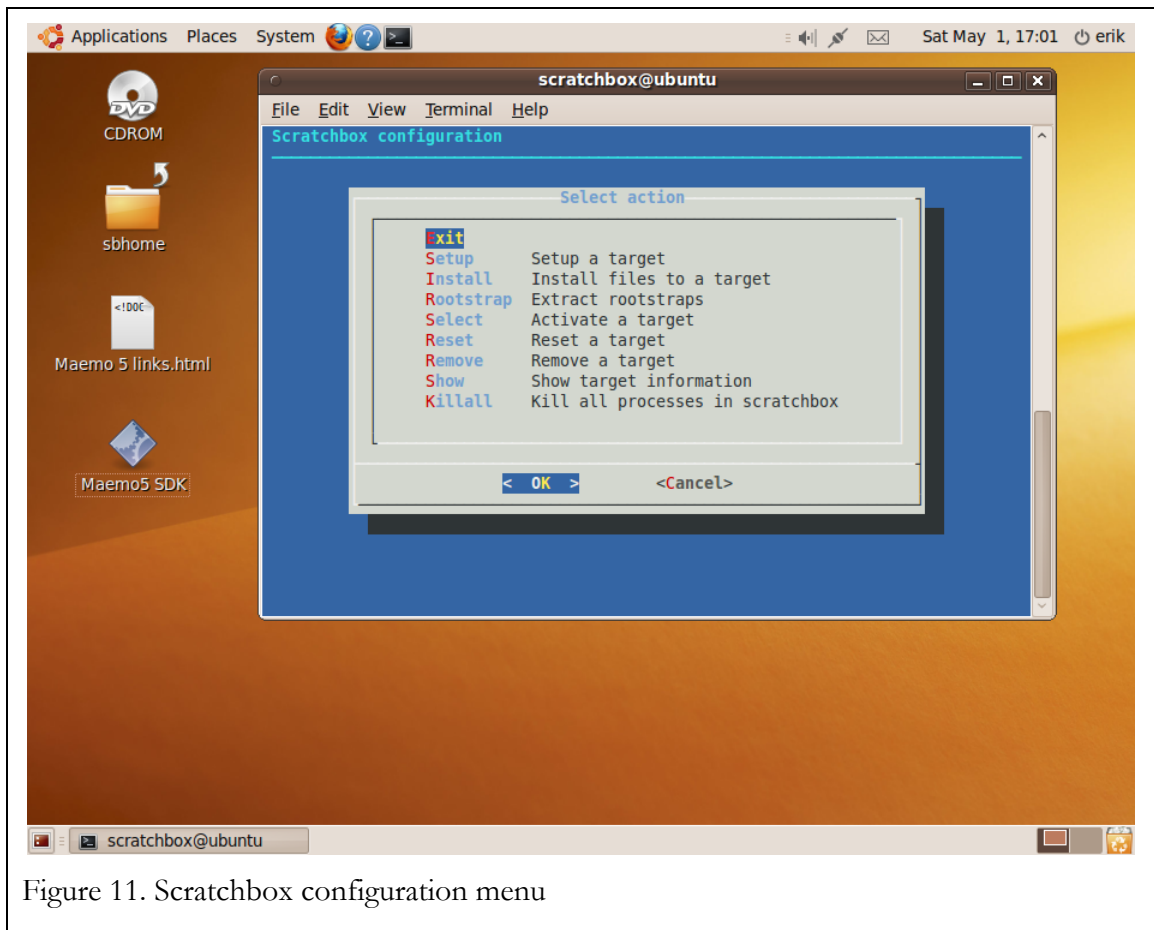


Figure 11. Scratchbox configuration menu

4.2.2 Virtual Image SDK

The Virtual Image SDK can be found at this location: <http://tablets-dev.nokia.com/maemo-dev-env-downloads.php>

After accepting the End User Software Agreement the following file can be downloaded: Maemo_Ubuntu_Intrepid_Desktop_SDK_Virtual_Image_Final.7z.

Once the 1,69 GB file is downloaded, it needs to be unpacked, resulting in a directory of 10.2 GB. The main contents of which are a virtual image disk and a virtual image configuration file that can be used with the following virtualization solutions: VMware Player (Windows & Linux), VMware Fusion (Mac OS X), QEMU (Windows, Linux & Mac OS X) and VirtualBox (Windows, Linux & Mac OS X).

The next steps are depending on the chosen virtualization environment, in this case VMware Fusion is employed. By simply opening the “maemosdk_desktop_intrepid-10-08.vmx” file, it is automatically associated with VMware Fusion and run in the virtualization environment.

The virtual machine boots and when finished loading the Ubuntu 8.10 desktop is ready (see figure 12.). (Note: in the VMware Fusion environment it is advisable to install the VMware tools for best performance and compatibility option.) The Maemo SDK can now be accessed via the ESbox IDE, which has a shortcut on the desktop or via the Terminal:

```
$ cd /scratchbox  
$ ./login
```

The ESbox IDE can be used for writing and compiling code and to run applications in the (emulated) target environment. (See figures 13 and 14.)



Figure 12. Virtual Image SDK – First run

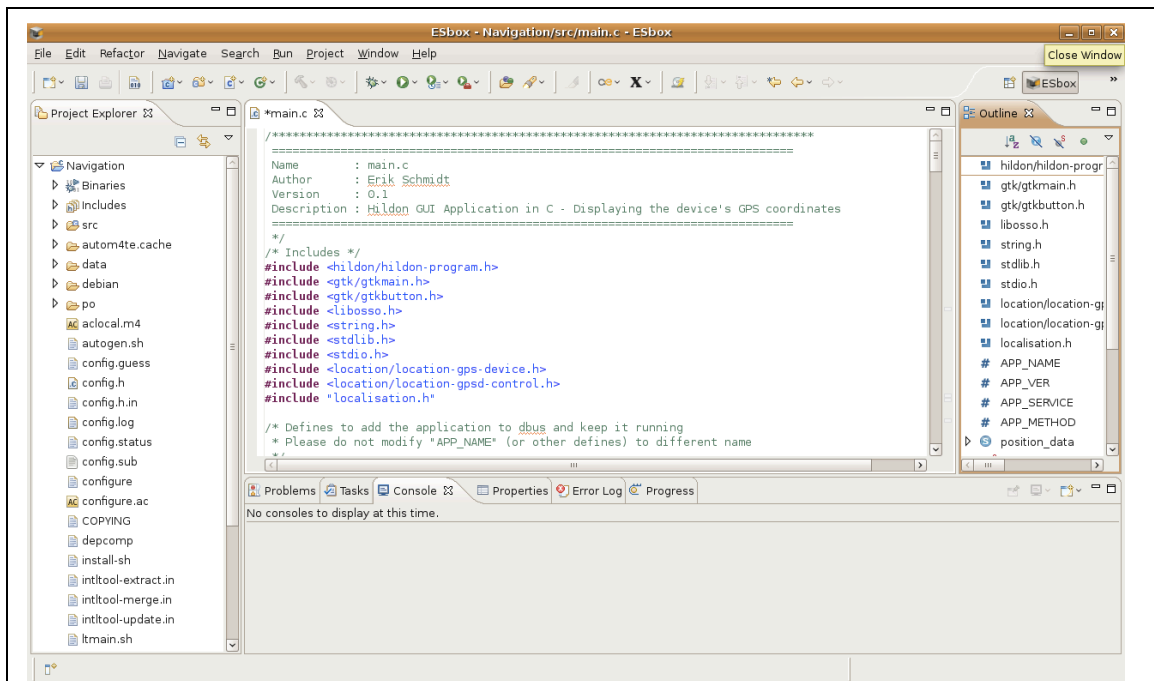


Figure 13. Virtual Image SDK – ESbox IDE

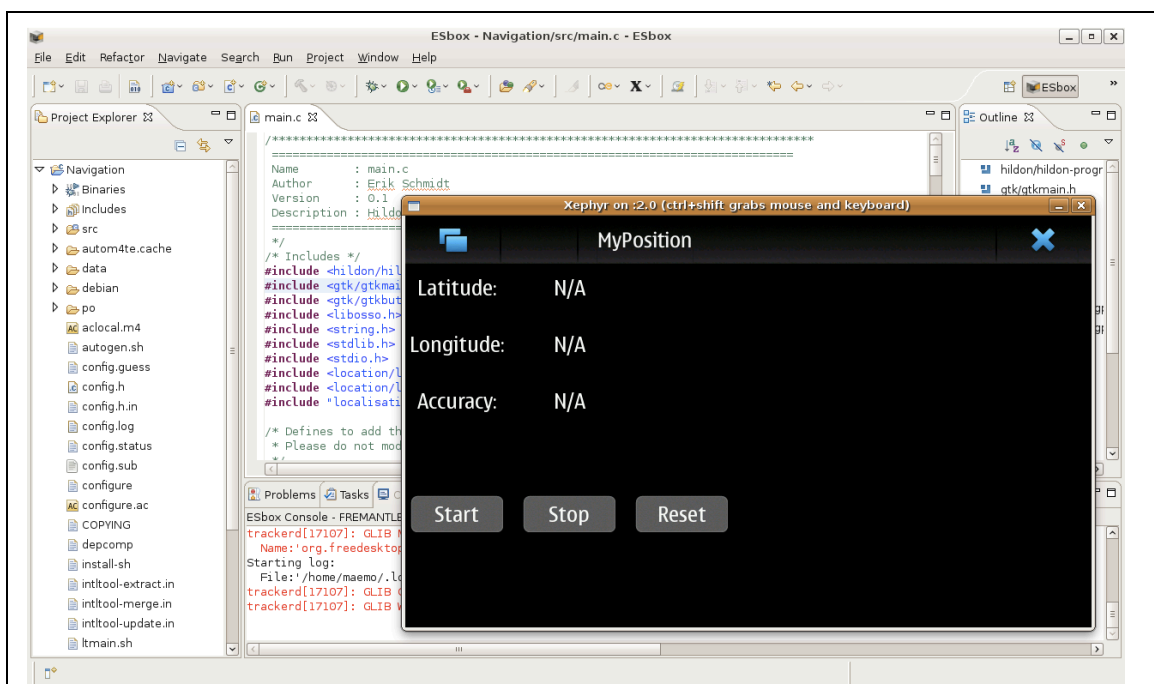


Figure 14. Virtual Image SDK – ESbox IDE – Maemo Application Framework example

5 Application implementation

With the Software Development Kits for iPhone and Maemo installed, application development can take place. The goal is to implement a basic and simple application that displays GPS data on screen. Both the iPhone 3G S and the N900 support GPS usage and it can be expected that the creation of similar applications is feasible. The output of the application shall just be the data of the current location (latitude and longitude) as well as the accuracy of the position data, which are all data provisioned by the relevant APIs of the devices. It must be stated that GPS usage is limited on the iPhone Simulator as only one, static location is given, and on the Maemo SDK it is not possible at all to use GPS functionality in simulation.

5.1 iPhone application

The Xcode IDE together with Interface Builder provide all that is necessary for implementing the desired application. First, a new project is created in Xcode, namely a “View-based application”. The IDE then automatically creates a template file and folder structure that corresponds to the compilation requirements. Cocoa touch development is largely based on the Model-View-Controller concept and the organization of the source code reflects this very well (Dave Mark, p. 32). So adequately named and organized default classes with header and implementation files as well as template contents are provided and coding can start right away. However, the `CoreLocation.framework` has to be added to the project manually. Building and compiling the source code is done by the click of a button, as is deployment to the iPhone Simulator and eventually to an iPhone device, if the Apple Developer Program membership fee of 99,- USD per year is paid (Goldstein 2010, 43).

For the application in question it is necessary to use the `CoreLocation` framework and to display its output to the view. The project’s “ViewController” classes take care of the main functionality, which means creating objects activating the GPS, obtaining and handling the GPS signal fix’ data while the corresponding “.xib”-file handles the visual presentation and layout of the view. It is in Interface Builder, the tool to build the graphical appearance of iPhone apps, where the labels and buttons are placed onto the view and connected to methods. The data is then stored in the corresponding “.xib”-file of the application package. The source code is available in appendix 3.

5.2 N900 application

The Virtual Image SDK comes with all necessary tools to create and deploy applications for the N900 and the included ESbox IDE can be considered a good choice for coding and project handling. Similar to the Xcode IDE, ESbox creates a template file and folder structure for new projects and provides one-click functionality for building and/or running applications in the device emulator or directly on a connected device. Nevertheless a manual entry has to be made to the `configure.ac`-file, as the “liblocation”-dependency must be added to be able to use the GPS functionality of the device.

The programming language C is used. The limited scope of the project allows maintaining all code within one single source file. Using the Hildon/GTK+-framework, the visual and layout aspects of the application are handled in the source code. This is sometimes a cumbersome method to create the User Interface, but in this example it is not too difficult and, if desired, graphical tools exist that allow the comfortable creation of graphical elements. Once the interface is ready, the signals (=clicks) of the buttons have to be assigned to the corresponding methods. These methods then activate or deactivate the GPS functionality and reset the view. Since the GPS functionality is a system service it keeps sending signals to the application that must be handled and assigned to methods, too. These methods take care of the three statuses: “position changed”, “service stopped” and “error”. The source code is available in appendix 4.

6 Summary and conclusion

6.1 Summary

The intended outcome of this thesis project was to complete a comparison of the iPhone and Maemo Software Development Environment and to produce a list of their differences and similarities. A secondary objective was to create an application for the iPhone 3G S and the Nokia N900 that displays the current GPS coordinates of the user.

It can be said that both goals have been met, as a thorough comparison list has been established and the applications have been completed on both platforms. To produce the list, the major aspects of the Software Development Kits for the iPhone and for Maemo have been exhaustively analyzed and a detailed list of differences and similarities has been elaborated. If there are items missing on the list, it is mainly due to intentional scope limitations and resource restrictions. Compromises had to be made and more thorough research or a wider ex-

tent would have required more time. The width and depth of the topic at hand presented some difficulties, meaning that almost every aspect that has been compared could be extended and intensified, but naturally resources were limited and a best effort was made to present a viable summary.

Once the theoretical foundation for application development had been laid by understanding more about the two platforms and comparing the development environments, the SDKs in question were installed and configured. But before any code could be written it was necessary to get familiar with the two different SDKs, which proved to be more time consuming than anticipated. Grasping the concepts of two new programming languages took additional time as well, because both presented distinctive features that were hard to comprehend initially. Even after overcoming these initial barriers it was still not a given that the applications would be completed in time or even completed at all. Eventually, the learning progressed well and it was possible to complete the secondary objective in good time and create the applications for the two different platforms.

However, it was not an easy task to acquire as much knowledge as possible in a limited amount of time in quite diverse fields, in order to first fully understand the topic and then be able to compare the two different approaches to it. This was challenging and one repeating intricacy of the process was the interwoven dependencies of items. Sometimes this situation inhibited progress on one subject until a depending subject was fully understood. As a result the research did not proceed as gradual as planned and the anticipated timeline could not always be followed strictly. Similarly, the theoretical understanding was often improved and corroborated by the practical application.

In this area, further research could be undertaken by comparing the development environments of other mobile platforms like Android or Symbian. Android has seen an extensive exposure thanks to its backing by Google and is probably going to be a widely used OS on mobile devices. Symbian currently is the most widely used mobile OS to date and is actively developed by the non-profit Symbian Foundation. Taking these two additional and relevant platforms into consideration could be a good continuation of this paper's topic.

Another interesting aspect for further research could be the area of marketing, distributing and monetizing applications on the two platforms (or on mobile platforms in general). Currently Apple leads the way with its App-Store, but it could be interesting what other companies do in this field and how these approaches compare both from the users' and the devel-

opers' perspective. After all, the possibility to generate income with the sale of applications is a major motivation for companies and individuals to engage in application development and having well-founded data and research in this field could be of good use.

Finally, if a recommendation could be made, it would be that a more rigorous quality control for the articles on maemo.org should be introduced, in order to better distinguish current and old material. Too often the bits and pieces for one item had to be gathered from several places, only to find that there is a newer version available. This did not happen with Apple's documentation and facilitated its usage tremendously, so it could be considered a necessary improvement to entice developers to engage with Maemo.

6.2 Conclusion

This thesis project has contributed very much to my understanding of the realm of mobile application development. Besides the two main objectives, this was as well a goal when presenting the thesis topic and I am glad that despite the limited prior knowledge of the details of this domain, I succeeded in acquiring sufficient knowledge and was able to complete applications in two programming languages previously unknown to me.

Certain aspects of this thesis could have been better, there is no doubt about that. But limited time, other obligations plus events beyond the control of humans put boundaries to what could be done. As a result of all that, the strains of project work, the necessity of planning and the importance of flexibility when plans fail are lessons learned. Even without the negative effects of external factors the project would not have been easy to complete: the sheer information available and the complexity of the topic provided far more material than could be covered. Without prior knowledge it was very difficult to distinguish between vital and superfluous information, resulting in additional time spent. At the same time this provided me with a more thorough understanding of matters and a more complete perspective.

Turning to the contents of this project, I can say that I have found a good entry to a subject of my interest and I definitely plan to continue in this direction in my next professional endeavours. It is a highly dynamic field, during the course of writing this paper, there were many announcements, changes and releases in the frame of the iPhone and Maemo, that are sure to keep developers, and users, busy and excited in the near future and beyond. From all the news, the iPhone OS 4 and the "MeeGo"-alliance of Nokia and Intel are probably going to have the biggest impact.

So after having learned much about and compared the two platforms, it is hard to tell which one is “better”. Owning and using both handsets evaluated in this paper, the N900 and the 3G S, it is obvious that there is a distinct feel to each. Similarly there is a clear distinction between the development environments, too. While it is difficult to pinpoint the distinction, an attempt in a few words could be “smooth, controlled and fast” for the iPhone and “raw, open and powerful” for Maemo. This surely does not do justice to all aspects of the platforms, but hints at my general perception. However, during the course of writing this thesis one of the most prominent mobile platforms, the iPhone, has been uncovered in detail and on the other hand, the power of Linux was felt. Consequently very much of an evaluation boils down to personal preference. It can even become a matter of philosophy, with the two side representing quite antagonistic approaches: open vs. closed source. And I have to say that I have not made up my mind on these issues and cannot yet judge comprehensively enough before obtaining more experience on both platforms.

The mobile technology environment remains very dynamic and I am glad that I have made a further step towards better understanding that sector.

Bibliography

Adobe Inc. 2010. Adobe Flash Professional CS5. URL:

<http://labs.adobe.com/technologies/flashcs5/> Quoted: 22 April 2010.

Appcelerator Inc. 2010. Titanium Cross Platform Application Development. URL:

<http://www.appcelerator.com/products/titanium-cross-platform-application-development/>
Quoted: 22 April 2010.

Apple 2010a. Apple Dev Center. URL: <http://developer.apple.com/iphone/index.action>

Quoted: 03 March 2010.

Apple 2010b. Apple iPhone SDK Agreement. URL:

http://developer.apple.com/iphone/download.action?path=%2Fiphone%2Fiphone_sdk_3.1.2__final%2Fipa0574_iphone_sdk.pdf Quoted: 03 March 2010.

Apple 2010c. Xcode - Developer Tools Technology Overview. URL:

<http://developer.apple.com/technologies/tools/xcode.html> Quoted: 03 March 2010.

Apple 2010d. Apple to use Intel microprocessors beginning in 2006. URL:

<http://www.apple.com/pr/library/2005/jun/06intel.html> Quoted: 13 April 2010.

Apple 2010f. iPhone OS Technology Overview. URL:

<http://developer.apple.com/technologies/iphone/> Quoted: 13 March 2010.

Apple 2010e. Software License Agreement for Mac OS X. URL:

<http://www.apple.com/legal/sla/docs/macosx106.pdf> Quoted: 09 March 2010.

Apple 2010g. iPhone SDK 4 Agreement. URL:

https://developer.apple.com/iphone/download.action?path=%2Fiphone%2Fiphone_sdk_3.4__final%2Fiphone_sdk_agreement.pdf Quoted: 29 April 2010.

Apple 2007h. Apple reinvents the phone with the iPhone. URL:

<http://www.apple.com/pr/library/2007/01/09iphone.html> Quoted: 21 April 2010.

Apple 2008i. iPhone 3G on sale tomorrow. URL:

<http://www.apple.com/pr/library/2008/07/10iphone.html> Quoted: 21 April 2010.

Apple 2009j. Apple announces the new iPhone 3G S. URL:
<http://www.apple.com/pr/library/2009/06/08iphone.html> Quoted: 21 April 2010.

Apple 2009k. iPhone 3G S - Technical Specifications. URL:
<http://support.apple.com/kb/SP565> Quoted: 14 April 2010.

Apple 2009l. iPhone OS Technology Overview. URL:
<http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html> Quoted: 21 April 2010.

Bosch, J. (2006). Software Product Families of Nokia. Rennes (FRA): Software Product Lines, 9th International Conference - Proceedings.

Fitzek, F., & Reichert, F. (2007). Mobile phone programming and its applications to wireless networking. New York, USA: Springer.

Goldstein, N. (2010). iPhone application development for dummies (2nd edition ed.). Hoboken (N.J.), USA: Wiley Publishing.

GSMarena.com 2009. Apple iPhone 3G S - Full phone specifications. URL:
http://www.gsmarena.com/apple_iphone_3gs-2826.php Quoted: 21 April 2010.

Maemo.org 2010a. Developing on the N900 itself ? URL:
<http://talk.maemo.org/showthread.php?t=32772> Quoted: 04 March 2010.

Maemo.org 2010b. Maemo 5 Developer Guide/Architecture/Top Level Architecture. URL:
http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/Architecture/Top_Level_Architecture Quoted: 03 March 2010.

Maemo.org 2010c. Maemo 5 Developer Guide/Development Environment/Maemo SDK. URL:
http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/Development_Environment/Maemo_SDK Quoted: 03 March 2010.

Maemo.org 2009d. Maemo Diablo Technology Overview. URL: http://maemo.org/midcom-serveattachmentguid-de0e0f0afdb811dd9642532f4bc4c627c627/Maemo_Diablo_Technology_Overview_Training_Material_for_maemo_4.1.pdf Quoted: 02 April 2010.

Maemo.org 2010e. MADDE / Frequently Asked Questions. URL: <http://wiki.maemo.org/MADDE/FAQ> Quoted: 28 April 2010.

Maemo.org 2010f. PyMaemo. URL: <http://wiki.maemo.org/PyMaemo> Quoted: 28 April 2010.

Maemo.org 2010g. Maemo for mobile developers. URL: <http://maemo4mobile.garage.maemo.org/> Quoted: 28 April 2010.

Maemo.org 2010h. Terms and Conditions of use. URL: http://maemo.org/legal/terms_of_use/ Quoted: 28 April 2010.

Maemo.org 2009i. Maemo Diablo Getting Started. URL: http://maemo.org/midcom-serveattachmentguid-9e0a60d4fdb811dd80aed97025304c974c97/Maemo_Diablo_Getting_Started_Training_Material_for_maemo_4.1.pdf Quoted: 26 April 2010.

Maemo.org 2008j. Maemo Platform Overview. URL: http://maemo.org/midcom-serveattachmentguid-340a4b94504311deaa00432fc84a748a748a/maemo_platform_overview.pdf Quoted: 28 April 2010.

Maemo.org 2010k. Intro: Software Platform. URL: <http://maemo.org/intro/platform/> Quoted: 29 April 2010.

Maemo.org 2008l. Maemo Platform Overview. URL: http://maemo.org/maemo_training_material/maemo4.x/html/maemo_Technology_Overview_Chinook/Chapter_03_maemo_Platform_Overview.html Quoted: 29 April 2010.

Mark, D., & Lamartine, J. (2009). Beginning iPhone 3 Development. New York, USA: Apress.

Mono Project 2010. MonoTouch from Novell. URL: <http://monotouch.net/> Quoted: 29 April 2010.

Nokia 2010a. Forum Nokia - Maemo 5 SDK. URL: http://www.forum.nokia.com/Tools_Docs_and_Code/Tools/Platforms/Maemo/ Quoted: 03 March 2010.

Nokia 2010b. Forum Nokia - Maemo 5 SDK Release Notes. URL: http://sw.nokia.com/id/4ddabfe0-8367-4400-a774-3ac970f486e6/Meamo_5_SDK_Release_Notes_v1_3_en.txt Quoted: 02 April 2010.

Nokia 2010c. Readme file for the Ubuntu Intrepid Desktop SDK Virtual Image Final. URL: http://tablets-dev.nokia.com/maemo-dev-env-downloads.php?f=Readme_Ubuntu_Intrepid_Desktop_SDK_Virtual_Image_Final.txt Quoted: 14 April 2010.

Nokia 2010d. Qt for Maemo Developer's Guide v0.5 Beta. URL: http://www.forum.nokia.com/piazza/wiki/images/7/7a/Qt_for_Maemo_Developers_Guide_v0_5_Beta.pdf Quoted: 28 April 2010.

Nokia 2010e. Device details - Nokia N900. URL: <http://www.forum.nokia.com/devices/N900/> Quoted: 29 April 2010.

Nokia 2010f. Nokia Europe - N900 - Specifications. URL: <http://europe.nokia.com/find-products/devices/nokia-n900/specifications> Quoted: 28 April 2010.

Nokia 2010g. Forum Nokia Developer Program e-Store. URL: https://pro.forum.nokia.com/productList.do?sortCol=NAME_TRANS&sortDirection=1&start=0 Quoted: 23 April 2010.

PhoneGap 2010. PhoneGap Homepage. URL: <http://www.phonegap.com> Quoted: 29 April 2010.

Rhomobile 2010. Rhodes. URL: <http://wiki.rhomobile.com//index.php?title=Rhodes> Quoted: 29 April 2010.

Scratchbox.org 2010. Introduction to Cross-development for Embedded Linux. URL: <http://www.scratchbox.org/documentation/general/tutorials/introduction.html> Quoted: 21 March 2010.

Stark, J. 2010. Building iPhone Apps with HTML, CSS, and JavaScript. Sebastopol, USA: O'Reilly Media.

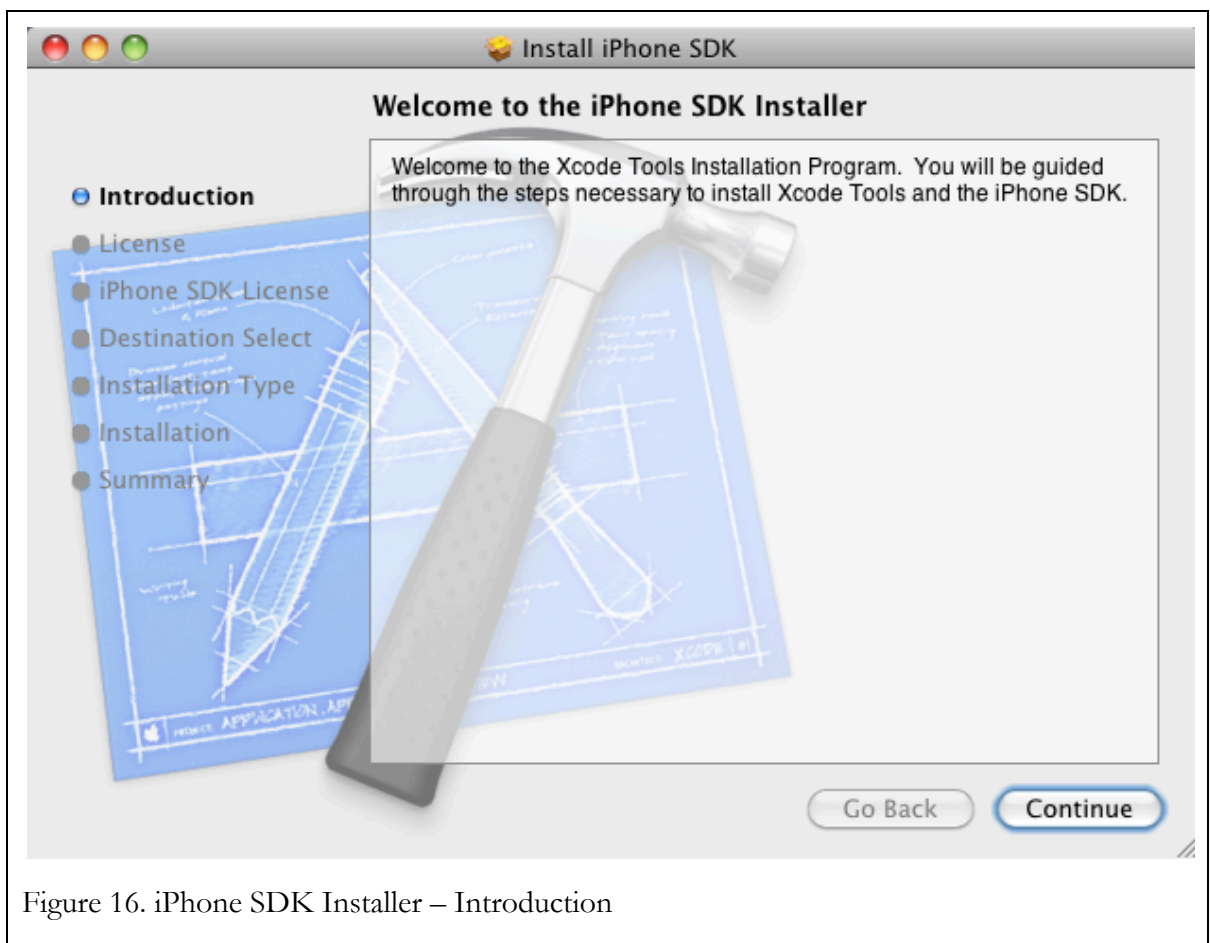
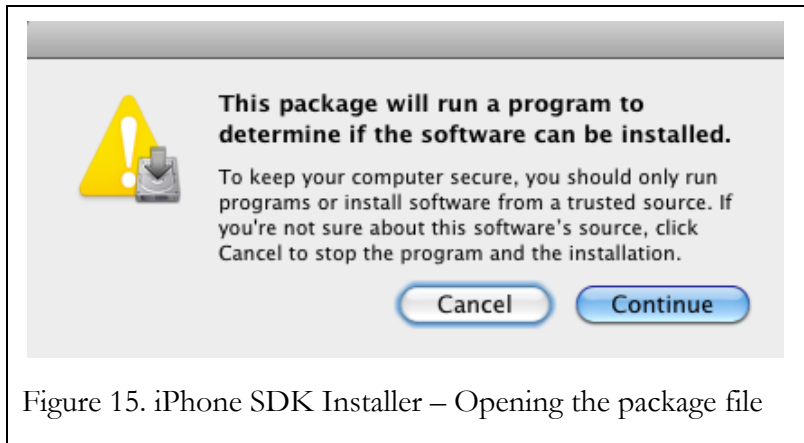
Steinbock, D. 2007. The Mobile Revolution: The Making of Mobile Services Worldwide. London, UK: Kogan Page Limited.

Tarkoma, S. 2009. Mobile Middleware: Supporting applications and Services. Chichester, UK: John Wiley & Sons.

Wikipedia.org 2010. iPhone SDK. URL: http://en.wikipedia.org/wiki/IPhone_SDK#.NET.2FCLI Quoted: 19 April 2010.

Appendices

Appendix 1. iPhone SDK installation procedure



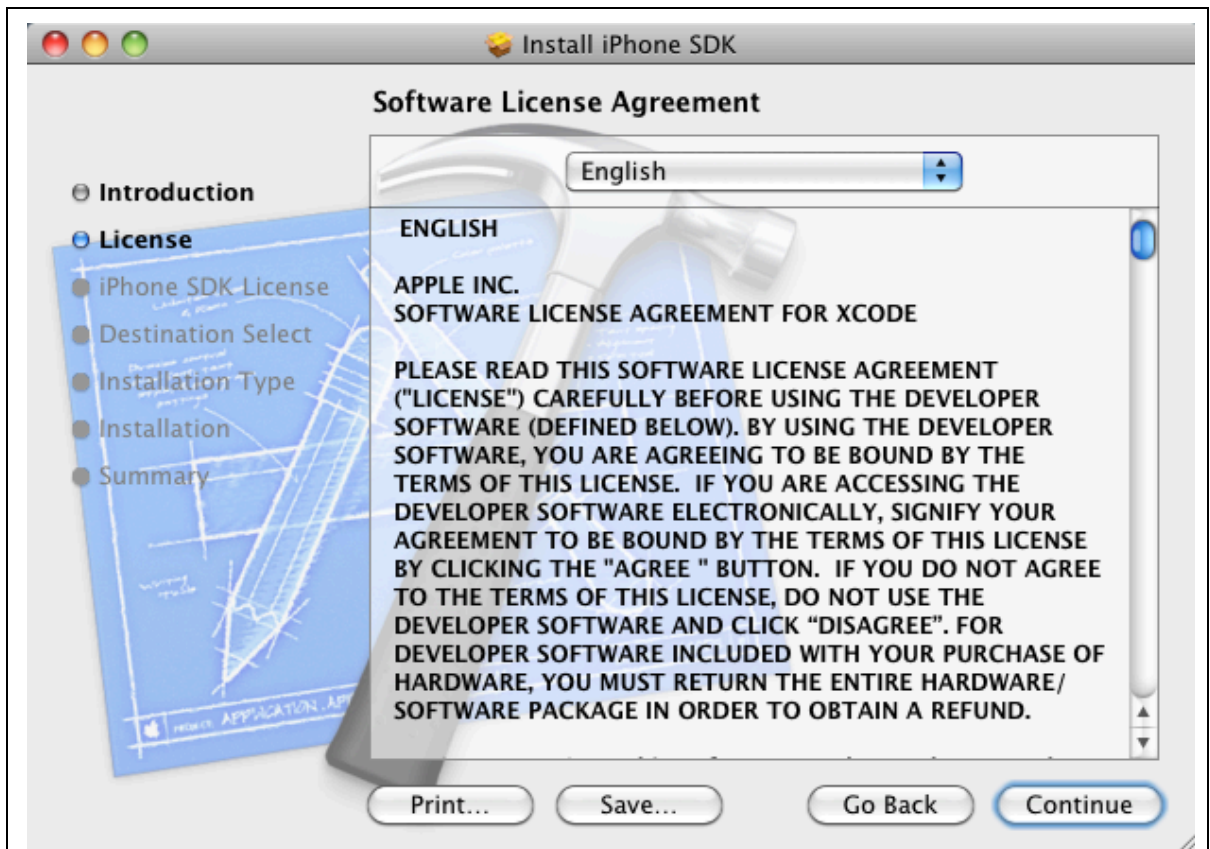


Figure 17. iPhone SDK Installer – Software License Agreement

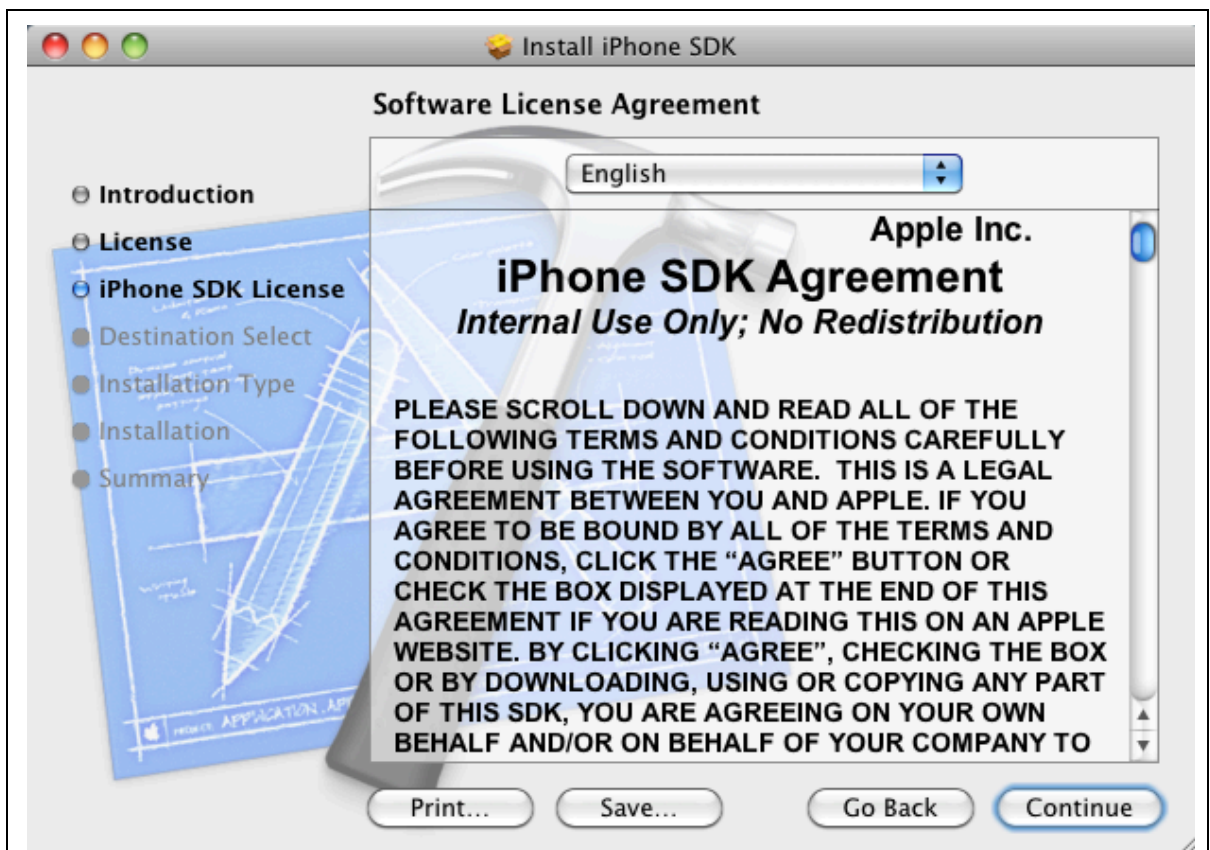


Figure 18. iPhone SDK Installer – iPhone SDK Agreement

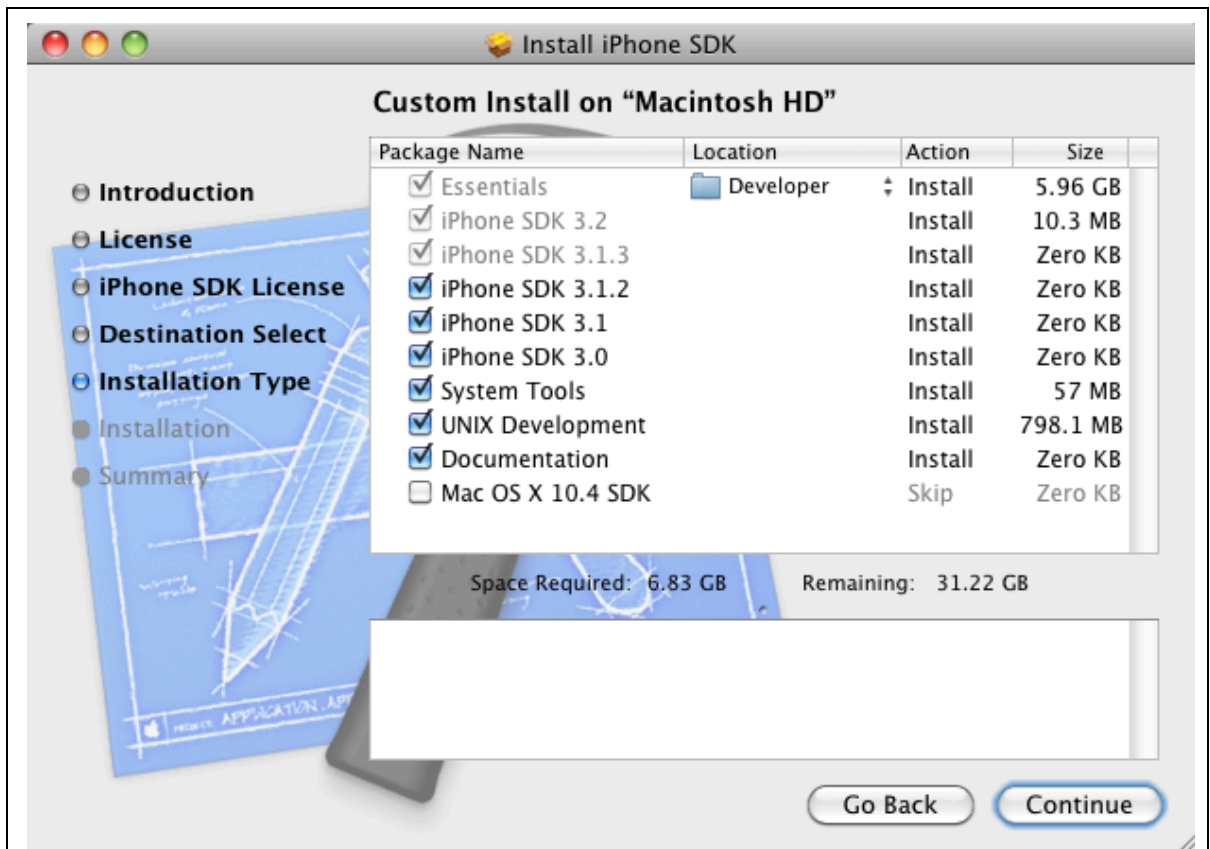


Figure 19. iPhone SDK Installer – Options selection

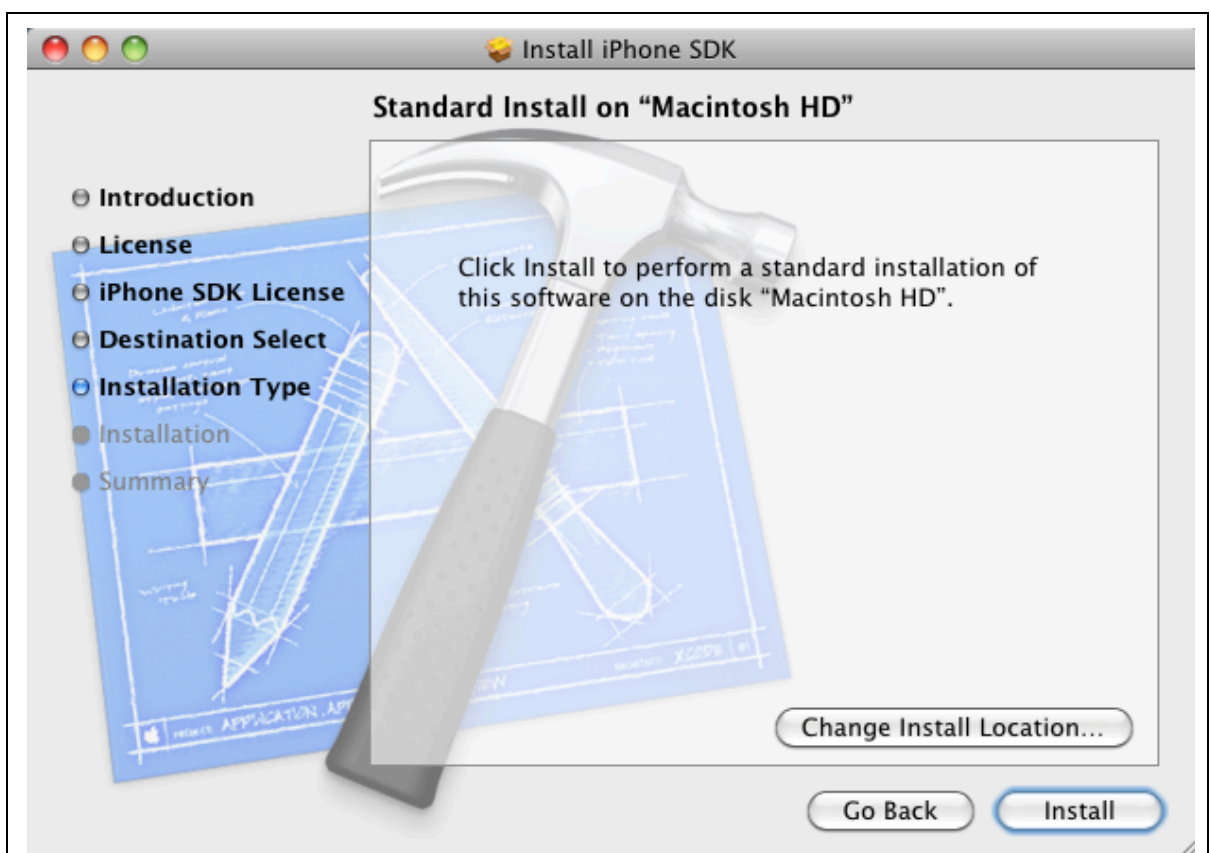


Figure 20. iPhone SDK Installer – Install location selection



Figure 21. iPhone SDK Installer – Installation

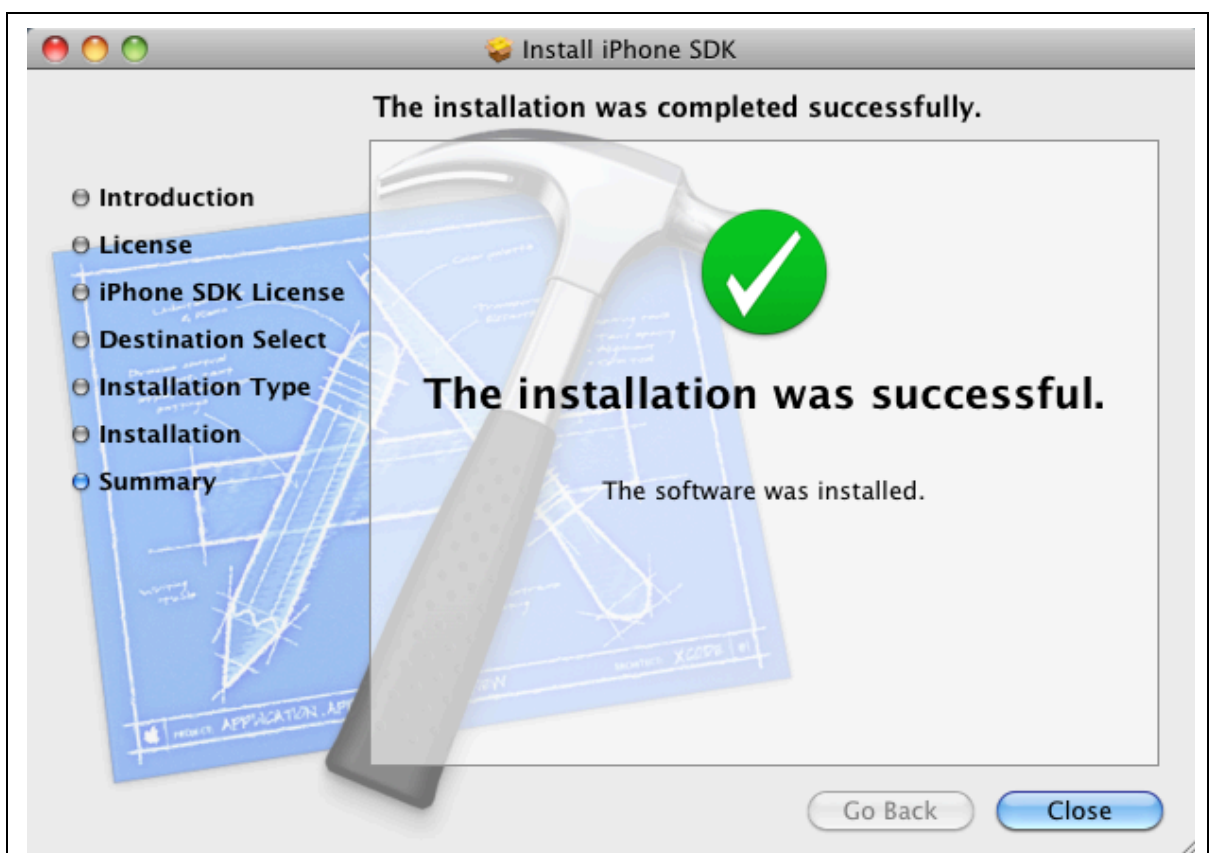


Figure 22. iPhone SDK Installer – Installation completed

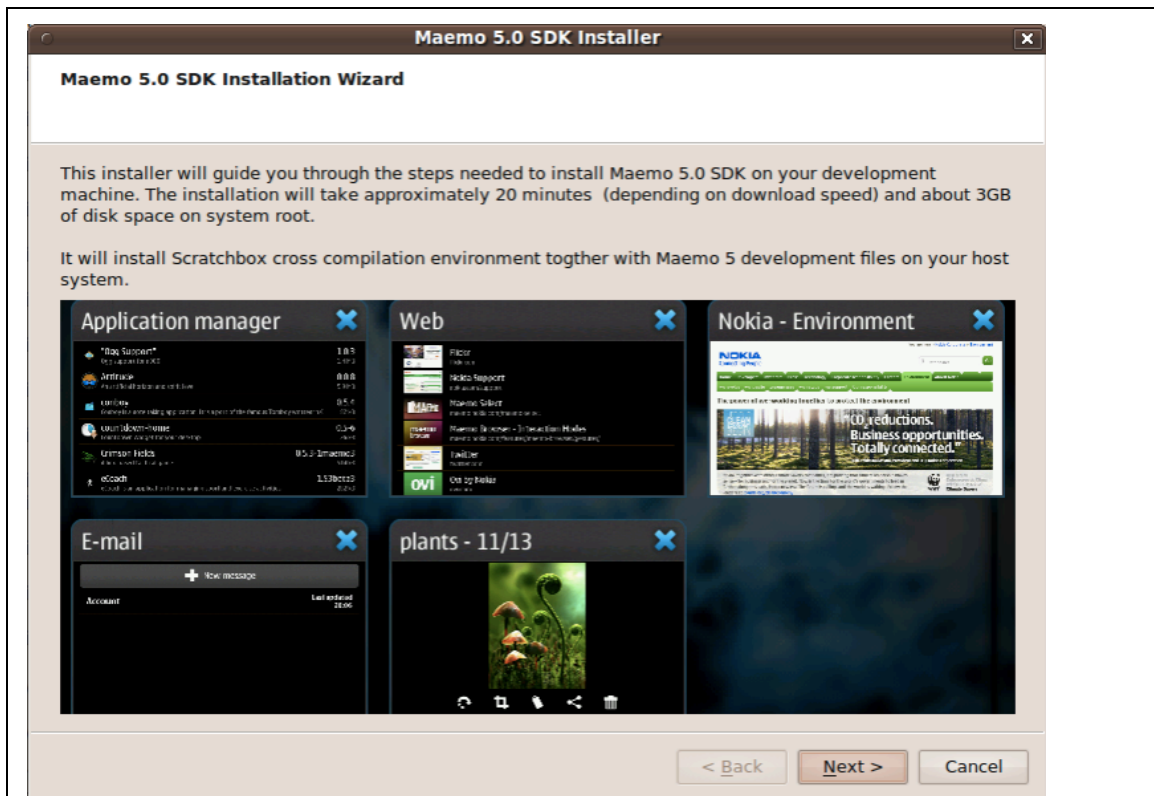


Figure 23. GUI Installer - Introduction

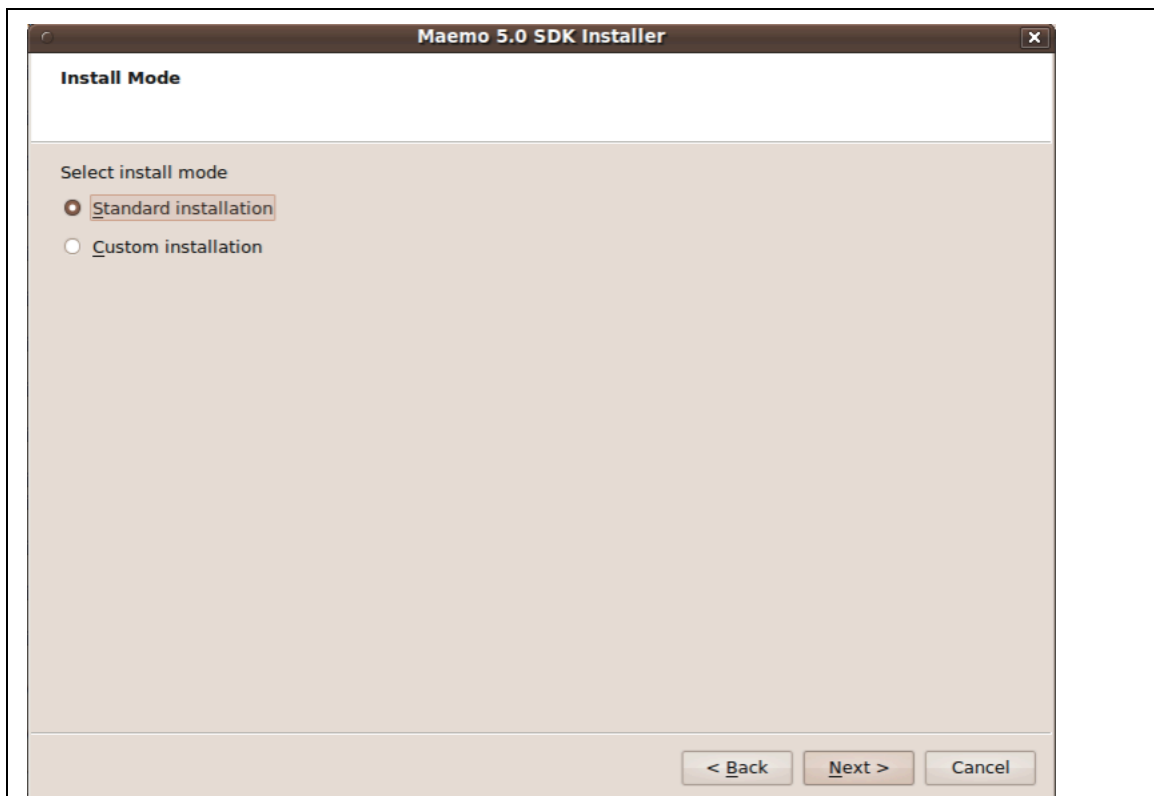


Figure 24. GUI Installer – Install mode selection

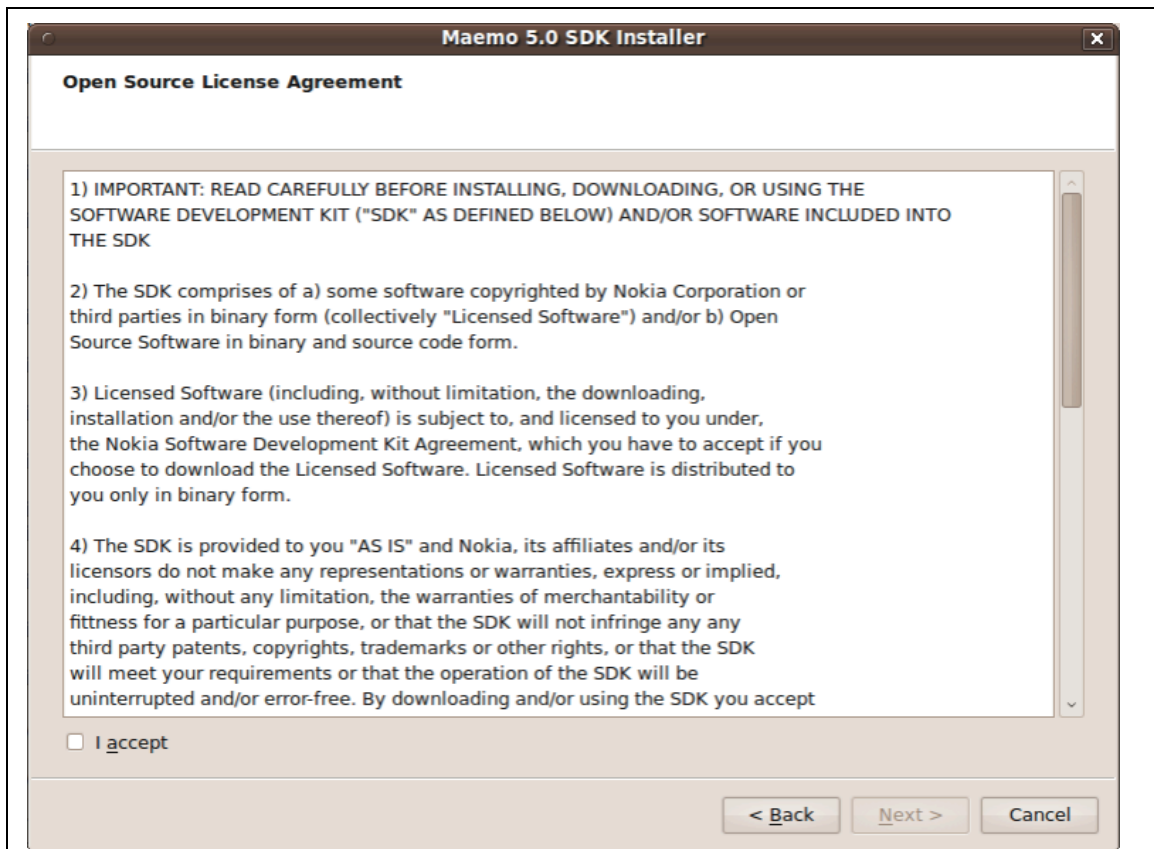


Figure 25. GUI Installer – Open Source License Agreement

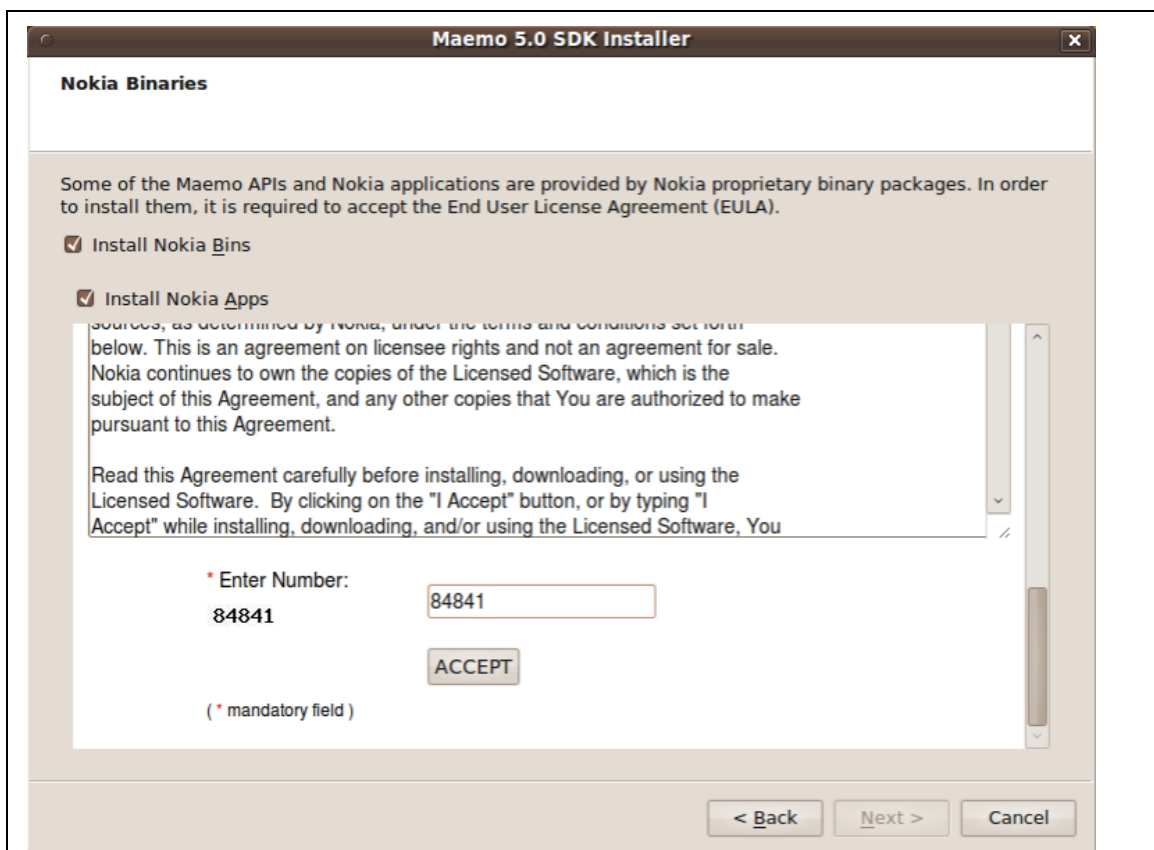


Figure 26. GUI Installer – Nokia binaries End User License Agreement

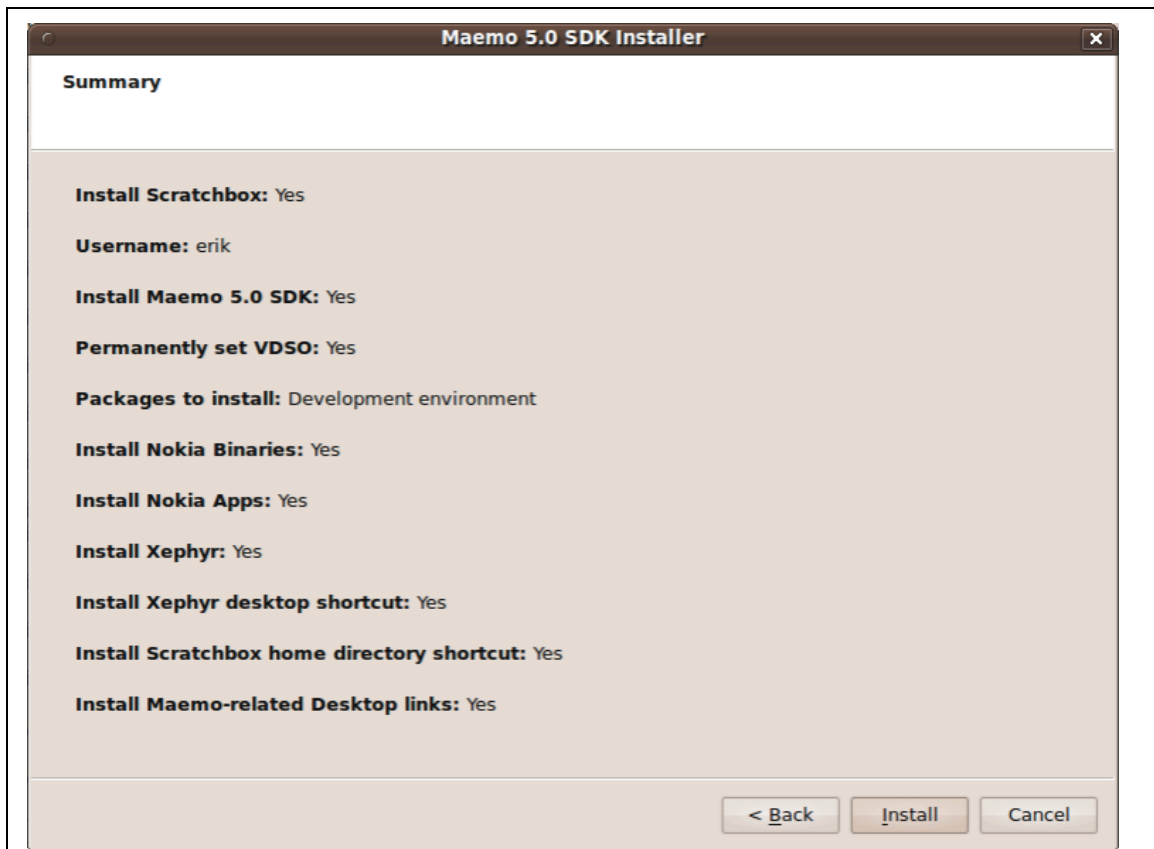


Figure 27. GUI Installer – Installation summary

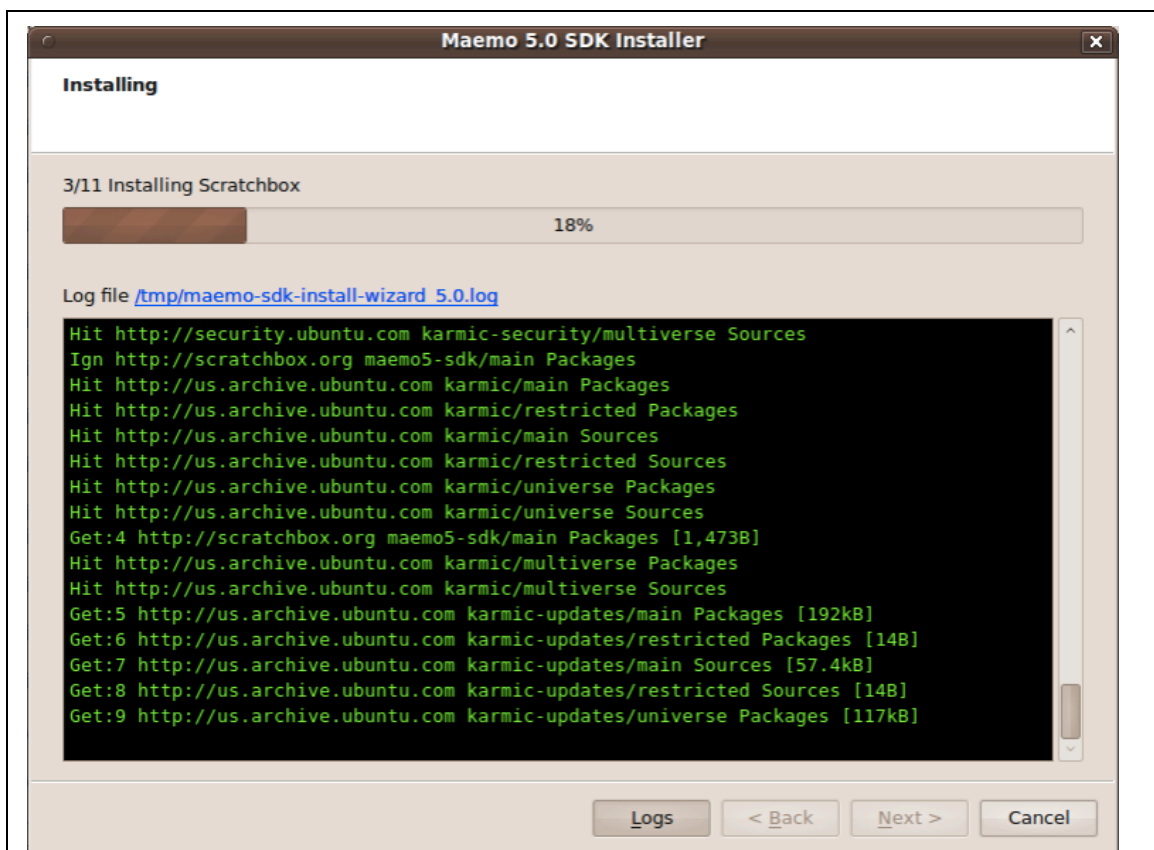


Figure 28. GUI Installer – Installation progress

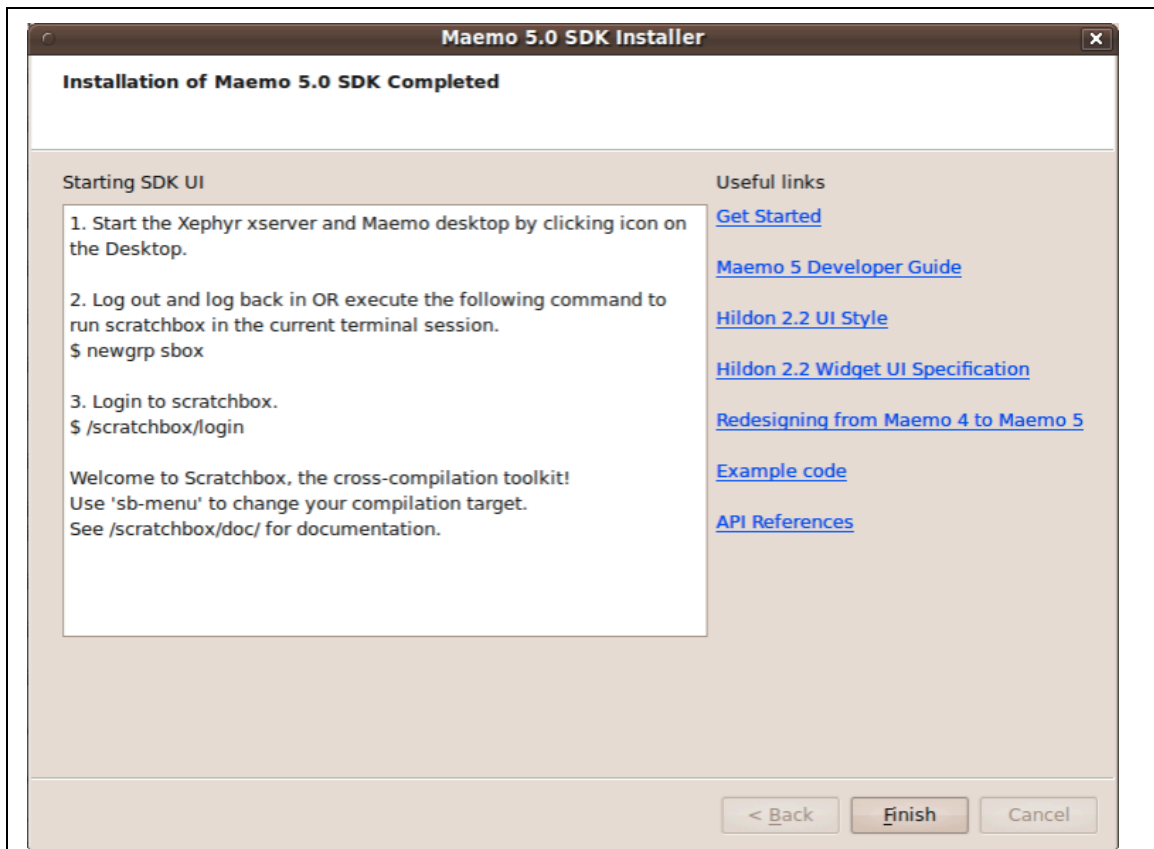


Figure 29. GUI Installer – Installation completed

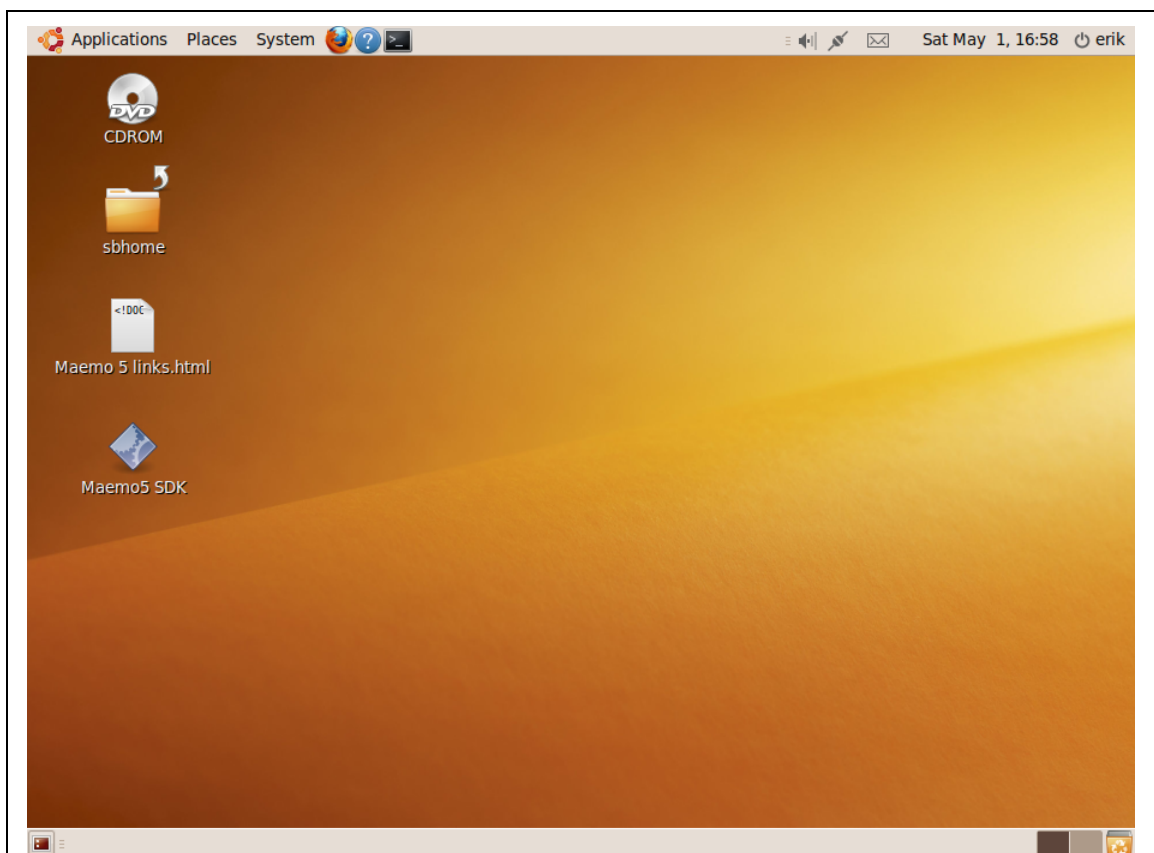


Figure 30. GUI Installer – Desktop after completion

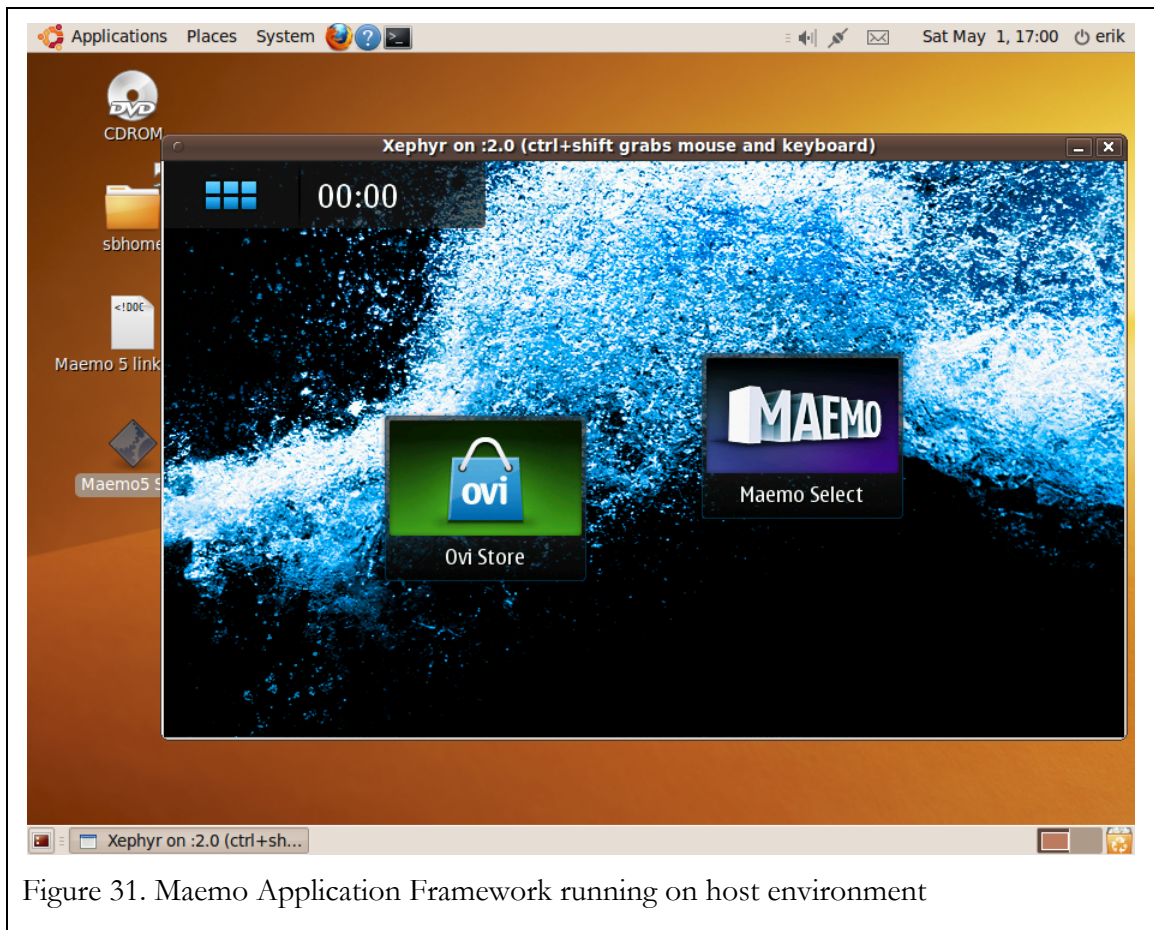


Figure 31. Maemo Application Framework running on host environment

MyPositionViewController.h

```

//
// MyPositionViewController.h
// Navigation
//
// Created by Erik Schmidt on 4/23/10.
// Copyright __MyCompanyName__ 2010. All rights reserved.
//
// Necessary import for location functionality
#import <UIKit/UIKit.h>
#import <CoreLocation/CoreLocation.h>

@interface MyPositionViewController : UIViewController <CLLocationManagerDelegate> {

    // Defines the interface for configuring the delivery of location- and heading-related events.
    CLLocationManager *locationManager;

    // Represents the location data object
    CLLocation *myLocation;

    // Labels for the view that are attached to Outlets
    UILabel *lb_latitude;
    UILabel *lb_longitude;
    UILabel *lb_accuracy;
}

// Property and Outlet declarations
@property (retain, nonatomic) CLLocationManager *locationManager;
@property (retain, nonatomic) CLLocation *myLocation;
@property (retain, nonatomic) IBOutlet UILabel *lb_latitude;
@property (retain, nonatomic) IBOutlet UILabel *lb_longitude;
@property (retain, nonatomic) IBOutlet UILabel *lb_accuracy;

```

```
// Method declarations
-(IBAction)start_GPS:(id) sender;
-(IBAction)stop_GPS:(id)sender;
-(IBAction)reset_GPS:(id)sender;
@end
```

MyPositionViewController.m

```
//
// MyPositionViewController.m
// Navigation
//
// Created by Erik Schmidt on 4/23/10.
// Copyright __MyCompanyName__ 2010. All rights reserved.
//
#import "MyPositionViewController.h"

@implementation MyPositionViewController

// Implement accessor methods (create setters and getters) for properties
@synthesize locationManager;
@synthesize myLocation;
@synthesize lb_latitude;
@synthesize lb_longitude;
@synthesize lb_accuracy;

// Method for the Start button
-(IBAction)start_GPS:(id)sender{
    [locationManager startUpdatingLocation];
}

// Method for the Stop button
-(IBAction)stop_GPS:(id)sender{
    [locationManager stopUpdatingLocation];
}
```

```

// Method for the Reset button
-(IBAction)reset_GPS:(id)sender{
    lb_latitude.text = @"N/A";
    lb_longitude.text = @"N/A";
    lb_accuracy.text = @"N/A";
}

// Additional setup after loading the view, typically from a nib.
// Initializing the locationManager and setting the location accuracy and distance filter.
- (void)viewDidLoad {
    self.locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate = self;
    locationManager.desiredAccuracy = kCLLocationAccuracyBest;
    locationManager.distanceFilter = kCLDistanceFilterNone;
}

// Releasing any retained subviews of the main view.
- (void)viewDidUnload {
    self.locationManager = nil;
    self.lb_latitude = nil;
    self.lb_longitude = nil;
    self.lb_accuracy = nil;
    [super viewDidUnload];
}

// Releasing the objects, freeing memory
- (void)dealloc {
    [locationManager release];
    [myLocation release];
    [lb_latitude release];
    [lb_longitude release];
    [lb_accuracy release];
    [super dealloc];
}

```

```

// Delegate method for handling location updates
// Updating the view with the location information received
-(void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation
        fromLocation:(CLLocation *)oldLocation {

    if(myLocation == nil)
        self.myLocation = newLocation;

    NSString *latitude = [[NSString alloc] initWithFormat:@"%g",
        newLocation.coordinate.latitude];
    lb_latitude.text = latitude;
    [latitude release];

    NSString *longitude = [[NSString alloc] initWithFormat:@"%g°",
        newLocation.coordinate.longitude];
    lb_longitude.text = longitude;
    [longitude release];

    NSString *accuracy = [[NSString alloc] initWithFormat:@"%gm",
        newLocation.horizontalAccuracy];
    lb_accuracy.text = accuracy;
    [accuracy release];
}

// Handling errors of GPS initialisation: No fix or denied by user
// Displaying error alert to the user. Not applicable in iPhone Simulator!
-(void)locationManager:(CLLocationManager *)manager
    didFailWithError:(NSError *)error {
    NSString *errorType = (error.code == kCLErrorDenied) ?
        @"Access Denied" : @"Unknown Error";

    UIAlertView *alert = [[UIAlertView alloc]
        initWithTitle:@"Error getting Location"
        message:errorType
        delegate:nil

```

```
cancelButtonTitle:@"Okay"  
otherButtonTitles:nil];  
[alert show];  
[alert release];  
}  
@end
```

main.c

```

/*****
=====

Name      : main.c
Author    : Erik Schmidt
Version   : 0.1
Description : Hildon GUI Application in C - Displaying the device's GPS coordinates
=====
*/

/* Includes */
#include <hildon/hildon-program.h>
#include <gtk/gtkmain.h>
#include <gtk/gtkbutton.h>
#include <libosso.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <location/location-gps-device.h>
#include <location/location-gpsd-control.h>
#include "localisation.h"

/* Defines to add the application to dbus and keep it running
 * Please do not modify "APP_NAME" (or other defines) to different name
 */
#define APP_NAME "navigation"
#define APP_VER "0.1"
#define APP_SERVICE "com.nokia.navigation"
#define APP_METHOD "/com/nokia/navigation"
/* end defines */

/* Structure to hold and pass the position data */
struct position_data {
    GtkWidget *la, *lo, *ac;
    LocationGPSControl *control;

```

```

        LocationGPSTDevice *device;

};

/* Function to start the GPS functionality */
static gboolean start_location(gpointer data){
    location_gpsd_control_start((LocationGPSControl *) data);
    return FALSE;
}

/* Implementing the functionality of the buttons */
/* Method assigned to the start button
 * Starting the GPS functionality */
void on_start_button (GtkWidget *widget, struct position_data *data){
    /* Update one label */
    gtk_label_set_text(GTK_LABEL(data->la), "Start");
    /* Add the GPS functionality to the process */
    g_idle_add(start_location, data->control);
}

/* Method assigned to the stop button, stopping the GPS functionality */
void on_stop_button (GtkWidget *widget, LocationGPSControl *control){
    location_gpsd_control_stop((LocationGPSControl *) control);
}

/* Method assigned to the reset button, clearing the display */
void on_reset_button (GtkWidget *widget, struct position_data *data){
    gtk_label_set_text(GTK_LABEL(data->la), "Reset");
    gtk_label_set_text(GTK_LABEL(data->lo), "Reset");
    gtk_label_set_text(GTK_LABEL(data->ac), "Reset");
    location_gps_device_reset_last_known(data->device);
}

/* Method to display the location data and info, if available */
static void on_changed(LocationGPSTDevice *device, struct position_data *data)
{

```

```

if (!device)

    return;

if (device->fix) {
    if (device->fix->fields & LOCATION_GPS_DEVICE_LATLONG_SET) {
        gchar buffer[8];
        gtk_label_set_text(GTK_LABEL(data->la), g_ascii_dtostr(buffer, 8, device-
        >fix->latitude));
        gtk_label_set_text(GTK_LABEL(data->lo), g_ascii_dtostr(buffer, 8, device-
        >fix->longitude));
        gtk_label_set_text(GTK_LABEL(data->ac), g_ascii_dtostr(buffer, 8, device-
        >fix->eph));
    }
}

/* Method to display a stop signal, when GPS is stopped */
static void on_stop(LocationGPSDControl *control, struct position_data *data)
{
    gtk_label_set_text(GTK_LABEL(data->la), "Stop");
    g_idle_remove_by_data(control);
}

/* Method to display an error signal, when GPS error occurs */
static void on_error(LocationGPSDControl *control, LocationGPSDControlError error,
    struct position_data *data)
{
    switch (error) {
    case LOCATION_ERROR_USER_REJECTED_DIALOG:
        gtk_label_set_text(GTK_LABEL(data->la), "User didn't enable requested methods");
        break;

    case LOCATION_ERROR_USER_REJECTED_SETTINGS:
        gtk_label_set_text(GTK_LABEL(data->la), "User changed settings, which disabled
        location");
        break;
    }
}

```

```

case LOCATION_ERROR_BT_GPS_NOT_AVAILABLE:
    gtk_label_set_text(GTK_LABEL(data->la), "Problems with BT GPS");
break;

case LOCATION_ERROR_METHOD_NOT_ALLOWED_IN_OFFLINE_MODE:
    gtk_label_set_text(GTK_LABEL(data->la), "Requested method is not allowed in offline
    mode");
break;

case LOCATION_ERROR_SYSTEM:
    gtk_label_set_text(GTK_LABEL(data->la), "System error");
break;

        }
    }

/* The main function */
int main( int argc, char* argv[] )
{
    /* Create needed objects/variables */
    HildonProgram *program;
    HildonWindow *window;
    GtkWidget *bt_start;
    GtkWidget *bt_stop;
    GtkWidget *bt_reset;
    GtkWidget *table;
    GtkWidget *lb_latitude;
    GtkWidget *lb_longitude;
    GtkWidget *lb_accuracy;
    GtkWidget *lb_timestamp;
    struct position_data label_data;
    LocationGPSSDControl *gPScontrol;
    LocationGPSDevice *gPSdevice;

    /* Initialize the GTK. */
    gtk_init( &argc, &argv );

```

```

g_type_init();
locale_init();

/* Creating the GPS device and controller objects */
gPScontrol = location_gpsd_control_get_default();
gPSdevice = g_object_new(LOCATION_TYPE_GPS_DEVICE, NULL);

/* Set the preferred GPS method and the interval */
g_object_set(G_OBJECT(gPScontrol), "preferred-method",
LOCATION_METHOD_USER_SELECTED, "preferred-interval",
LOCATION_INTERVAL_2S, NULL);

/* Connect the GPS signals (error, changed and stopped) to methods */
g_signal_connect(gPScontrol, "error", G_CALLBACK(on_error), &label_data);
g_signal_connect(gPSdevice, "changed", G_CALLBACK(on_changed), &label_data);
g_signal_connect(gPScontrol, "gpsd-stopped", G_CALLBACK(on_stop), &label_data);

/* Create the hildon program and setup the title */
program = HILDON_PROGRAM(hildon_program_get_instance());
g_set_application_name("MyPosition");

/* Create HildonWindow and set it to HildonProgram */
window = HILDON_WINDOW(hildon_window_new());
hildon_program_add_window(program, window);

/* Quit program when window is closed. */
g_signal_connect (G_OBJECT (window), "delete_event", G_CALLBACK (gtk_main_quit),
NULL);

/* Quit program when window is otherwise destroyed. */
g_signal_connect (G_OBJECT (window), "destroy", G_CALLBACK (gtk_main_quit),
NULL);

/* Create a 3x5 table to position the various labels and widgets
* and add the table to the window */
table = gtk_table_new(3, 5, TRUE);

```

```

gtk_container_add(GTK_CONTAINER(window), GTK_WIDGET(table));
gtk_table_set_row_spacings (GTK_TABLE (table), 10);
gtk_table_set_col_spacings (GTK_TABLE (table), 10);

/* Create the labels and add them to the table */
lb_latitude = gtk_label_new("Latitude:");
lb_longitude = gtk_label_new("Longitude:");
lb_accuracy = gtk_label_new("Accuracy:");
gtk_table_attach(GTK_TABLE(table), lb_latitude, 0, 1, 0, 1, GTK_FILL, GTK_FILL, 5, 5);
gtk_table_attach(GTK_TABLE(table), lb_longitude, 0, 1, 1, 2, GTK_FILL, GTK_FILL, 5,
5);
gtk_table_attach(GTK_TABLE(table), lb_accuracy, 0, 1, 2, 3, GTK_FILL, GTK_FILL, 5,
5);

/* Create 3 buttons and add them to the table */
bt_start = gtk_button_new_with_label("Start");
bt_stop = gtk_button_new_with_label ("Stop");
bt_reset = gtk_button_new_with_label("Reset");
gtk_table_attach(GTK_TABLE(table), bt_start, 0, 1, 4, 5, GTK_FILL, GTK_FILL, 5, 5);
gtk_table_attach(GTK_TABLE(table), bt_stop, 1, 2, 4, 5, GTK_FILL, GTK_FILL, 5, 5);
gtk_table_attach(GTK_TABLE(table), bt_reset, 2, 3, 4, 5, GTK_FILL, GTK_FILL, 5, 5);

/* Creating the labels that are dynamically updated and adding them to the table */
label_data.la = gtk_label_new("N/A");
label_data.lo = gtk_label_new("N/A");
label_data.ac = gtk_label_new("N/A");
label_data.control = gPScontrol;
label_data.device = gPSdevice;
gtk_table_attach(GTK_TABLE(table), label_data.la, 1, 2, 0, 1, GTK_FILL, GTK_FILL, 5,
5);
gtk_table_attach(GTK_TABLE(table), label_data.lo, 1, 2, 1, 2, GTK_FILL, GTK_FILL, 5,
5);
gtk_table_attach(GTK_TABLE(table), label_data.ac, 1, 2, 2, 3, GTK_FILL, GTK_FILL, 5,
5);

```

```

/* Connecting the signals (click of the buttons) to methods */
g_signal_connect (G_OBJECT (bt_start), "clicked", G_CALLBACK (on_start_button),
&label_data);
g_signal_connect (G_OBJECT (bt_stop), "clicked", G_CALLBACK (on_stop_button),
gPScontrol);
g_signal_connect (G_OBJECT (bt_reset), "clicked", G_CALLBACK (on_reset_button),
&label_data);

/* Begin the main application
 * Show the widgets */
gtk_widget_show_all ( GTK_WIDGET ( window ) );
gtk_main();

/* Exit */
return 0;
}

```