



VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Lam Le

ENTERPRISE SERVICE BOOK APPLICATION FOR MULTIPLE MOBILE PLATFORMS

Department of Technology and Communication
2010

ABSTRACT

Author	Lam Le
Title	Enterprise Service Book Application for Multiple Mobile Platforms
Year	2010
Language	English
Pages	114
Name of Supervisor	Ghodrat Moghadampour

The purpose of this thesis was to build an enterprise mobile service book application for Wärtsilä Oyj, a global company whose main office is located in Vaasa, Finland. Wärtsilä is one of the world leading companies in boat engines and power plants.

Wärtsilä has a user manual application for engine information and troubleshooting called Eldoc Server. It was built in ASP .Net. This application is widely used within Wärtsilä employees, especially mechanical engineers who are maintaining and fixing engines every day. In order to increase the portability, mobility and accessibility of this electronic manual, they have requested a mobile version for this Eldoc Server.

The idea of this thesis was to implement an application for multiple mobile platforms which implements the functionally now existing in Eldoc web server such as viewing documents, pictures, videos, etc. Furthermore, it should utilize advantages of mobile devices such as camera, voice recorder, and phone service.

The application was targeted for these platforms: Windows, Linux and Maemo. Qt was selected as a development framework for this purpose as it is a cross-platform framework, meaning that it is possible to code the application once and deploy them across different platforms. Nokia N900 with Maemo OS was the main targeted device thank to its capability, mobility, and reusability.

All objectives of the final thesis mentioned above have been met in the Qt based application associated with this project. Tests have been performed and indicated that all main features work perfectly. The application has been presented with Wärtsilä people and is waiting for their permission to change to production version.

Keywords	QT, programming, Maemo, eldoc, application
----------	--

ACKNOWLEDGEMENT

I would like to thank all the people who helped me and inspired me during the final thesis period.

At the beginning, I would like to give my honest thanks to my tutor, meanwhile, my thesis's supervisor, Ghodrat Moghadampour. He not only instructs me the academic knowledge, but also the way how to handle problems in life. When I encountered difficulties, his patience and professional skills give me a lot of power to overcome the adverse circumstances.

Secondly, I would like to thank Professor Petri Helo, who is a lecturer in University of Vaasa and a manager in my company. He has given me this interesting project and some basic background information which is the basement for my achievement.

In the end, I will thank my beloved parents that encourage me from another remote country.

Here are my deepest thanks again for all of you I mentioned above.

Contents

Contents	4
1 INTRODUCTION	9
1.1 Eldoc Server Description	10
2 TECHNOLOGY OVERVIEW	12
2.1 Darwin Information Typing Architecture (Dita).....	12
2.1.1 Dita	12
2.1.2 Dita main features.....	13
2.2 Qt Framework.....	15
2.2.1 Introduction about Qt framework.....	15
2.2.2 Qt features summary	16
2.3 Maemo Platform Overview	19
2.3.1 Maemo Main features	20
2.3.2 Software development on Maemo platform.....	22
3 ELDOC SERVICE BOOK	24
3.1 Requirement Analysis.....	24
3.2 Main Features Deployment	25
3.3 Main Functions Specification.....	26
3.4 Class Hierarchy	28
3.4.1 Manual related classes	28
3.4.2 Spare part related classes	31
3.4.3 Main UI classes	32
3.4.4 Optional features UI classes.....	36
3.4.5 Mbarcode project integration	37

3.4.6	Mbarcode additional plugins	42
3.4.7	Media player classes	44
3.5	Detailed Descriptions of Main Functions	45
3.5.1	Open document.....	45
3.5.2	View image	47
3.5.3	View video	48
3.5.4	Add note/comment	50
3.5.5	Take and attach picture	51
3.5.6	Search manual/spare part document	52
3.5.7	Create technical request	54
3.6	Component Diagram.....	55
3.7	Architectural Diagram	57
4	GUI DESIGN	58
4.1	Main Window	58
4.2	Request Window	59
4.3	Setting Window	60
5	IMPLEMENTATION.....	62
5.1	Dita Parser Function	62
5.2	Add New Note.....	65
5.3	Take and Attach Pictures	67
5.4	Image Viewer	69
5.5	Media Player	71
5.6	View pdf File.....	72
5.7	QLabelFingerExtension Class.....	74

5.8	Update Database	75
5.9	Search Engine.....	76
6	TESTING	78
6.1	Viewing Technical Documents	79
6.2	Adding Notes or Comments in Windows and Linux	81
6.3	Adding Notes and Pictures in Eldoc Maemo Version.....	81
6.4	Barcode Reader	83
6.5	Setting Window	84
6.6	Request Window	85
7	CONCLUSION	86
7.1	Future Development	86
8	REFERENCES	87
APPENDICES		

ABBREVIATION

API: Application Programming Interface

OS: Operating System

GUI: Graphical User Interface

UI: User Interface

JRE: Java Runtime Environment

SDK: Software Development Kit

IDE: Integrated Development Environment

DITA: Darwin Information Typing Architecture

PC: Personal Computer

MADDE: Maemo Application Development and Debugging Environment

HTML: HyperText Markup Language

URL: Uniform Resource Locator

LIST OF APPENDICES

APPENDIX 1. Qt Development Setup for Maemo Platform

APPENDIX 2. Madde – Maemo Development Tool on Windows

1 INTRODUCTION

Nowadays, mobile phones are playing an increasingly essential role in human life. According to a recent report of the mobile market, the usage of mobile phones is increasing by 4.2% per season, especially of the phones with intelligent systems. Together with the increasing capability and functionality of mobile phones, users not only utilize mobile phones in communication, but also treat them as multi-functional devices, such as a multimedia player, a video player, a navigator, a game box or an internet tablet, etc. With the rapid development of mobile phone hardware and software, a smart phone can have nearly the same capability with a desktop computer. Moreover, thank to its small size, mobility and portability are the big advantages of a mobile phone. Having all these notable advantages, it is one of the most widely used devices and can be found in everyone's pocket.

Seeing the advantages and future of mobile phones, companies all over the world are changing from desktop to mobile phone software development. Websites are designed to have the capability to recognize the mobile phone devices and change to suitable resolution to provide mobile users with the best look and view. Applications written Java, C++, .Net, etc which once targeted for desktop environment are modified in order to run in mobile devices. Game development is shifting to mobile game providing touch screen.

Follow the trend, Wärtsilä, one of the world leading companies in boat engines and power plants, has requested to convert one of its applications called Eldoc Server to a mobile phone application so that it can be utilized among Wärtsilä employees all around the world. Basically the mobile application should implement the functionality now existing in Eldoc web server such as viewing documents, pictures, videos, etc. Furthermore, it can utilize advantages of mobile devices such as camera, voice recorder, and phone service.

Moreover, they also want the application available for their other commonly used devices such as Rugs portable PC running Windows and Laptop running Linux. This creates the need that to have a software development framework that can be deployed to multiple platforms and operating systems. The application should be coded once and able to be deployed to many different devices.

1.1 Eldoc Server Description

This user manual application is mainly used by engineers for engine information and troubleshooting. The main objective of this application is to convert documents which are on DITA format (Darwin Information Typing Architecture) to readable html format and display in a web browser with normal web functionalities such as navigation links between pages, image viewer, and video viewer.

The user interface consists of multiple stacked selections. In the first step, user chooses one of three main products. Language is by default set as English, and then user can choose module which is either engine type or spare part type. After selection of document number, the document will be displayed.

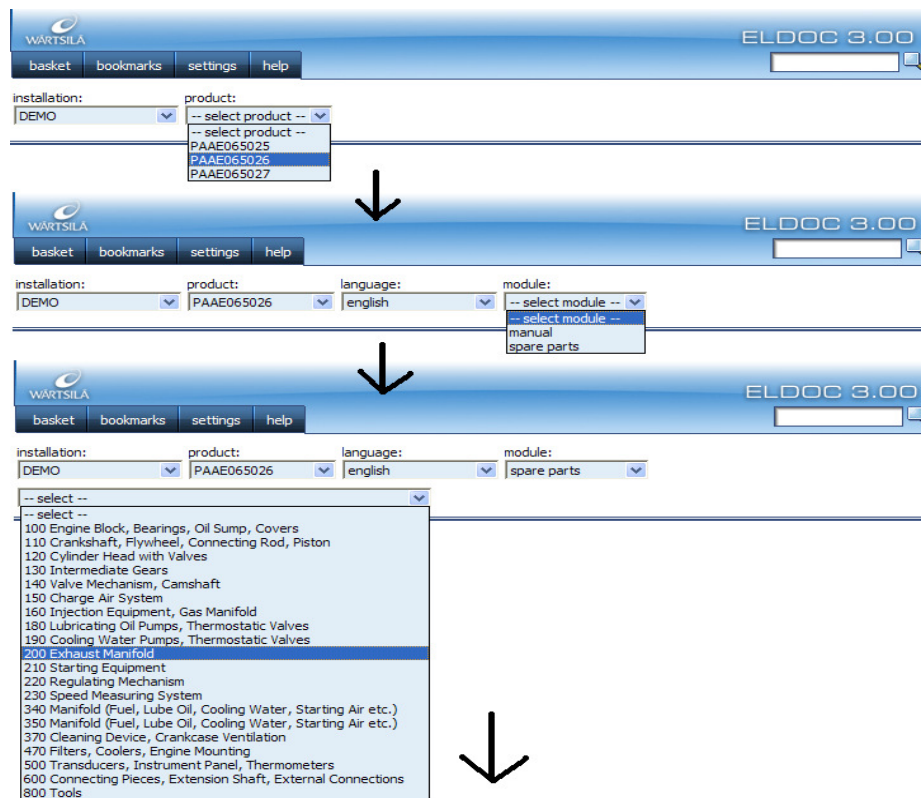


Figure 1.1. Eldoc server .Net version - browsing steps.

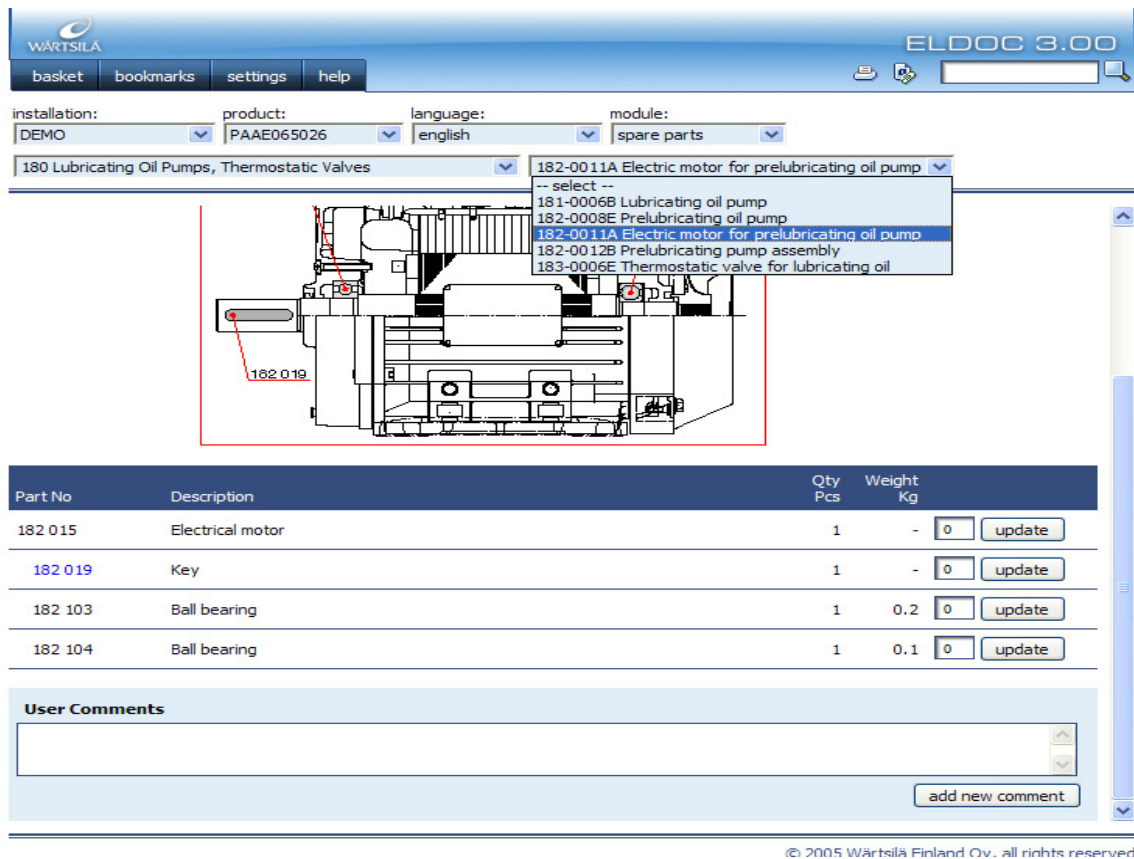


Figure 1.2. Eldoc server .Net version – final user interface.

There are two types of document: manual and spare part. Manuals are engines' related information and spare parts are technical details about individual parts of an engine. The document type can be selected in "module" selector mentioned above. Inside each document there are links to navigate between them and a collection of videos and pictures for visualizing guidance.

User can also add comments or notes on the document he/she is reading. Added notes/comments will be displayed together with the created date in the bottom of the document.

2 TECHNOLOGY OVERVIEW

Below is a short description of the development framework and platform used in this mobile application. There are three main parts: Dita document, Qt framework and Maemo platform. In each part there will be a definition and summary of the main features of the framework or platform.

2.1 Darwin Information Typing Architecture (Dita)

Dita is the format of the documents used as resources in this thesis application. It is also the main format of majority of Wärtsilä technical documents because of its advantages over XML documents.

2.1.1 Dita

The Darwin Information Typing Architecture (Dita) is an XML-based architecture for information exchange originally developed by IBM. The architecture applies the main characteristics in XML architecture such as modulation, content reusability and specialization. Dita is now one of OASIS standards. The main advantage of Dita format compared with XML is that its architecture follows a standard and unified format; therefore it can be converted to other format such as HTML, pdf, image using a convert engine. /1/

The structure of a Dita documents is basically similar which XML or HTML documents. It has one open tag and one close tag for every element. Each element can have attributes and single or complex content. As with HTML, any images, video files or other files which need to be displayed in output are included via reference. Thank to its simple format, any XML editor can be utilized to create or edit Dita content. Various editing tools which support Dita documents have been developed for instance XMLmind XML editor /3/. Below is a sample Dita document.

```
<h0>
<h0t label="man-06">Adjustments, Clearances and Wear
Limits</h0t>
<h1>
<h1t label="man-06.1">Adjustments</h1t>
<p>
<hp0>Valve timing</hp0>
</p>
```

```

<p>The valve timing is fixed and cannot be changed
individually, cylinder by cylinder.</p>
<fig label="fig-200601-low" caption="06-1">Valve
timing</fig>
<li>
<it>Inlet valve opens.</it>
<it>TDC.</it>
<it>Exhaust valve closes.</it>
<it>Exhaust valve opens.</it>
</li>
<p>
<hp0>Other set values:</hp0>
</p>
<li>
<it>Valve clearances, cold engine: inlet valves 0.4 mm,
exhaust valves 0.8 mm.</it>
<it>Fuel delivery commencement. See test records.</it>
<it>Opening pressure of fuel injection valve 450±10 bar</it>
</li>
</h1>
</h0>

```

2.1.2 Dita main features

- ✓ **Topic orientation:** Dita content is organized as modular topics. This is the highest standard structure in Dita. Topics are listed by a Dita map. This is a document contains links to different topics in the order that they appear in a finished document. A Dita map specifies table of contents for deliverables. Furthermore, Dita documents also have relationship table which defines the linking between different topics. Modular topics can be reused in different deliverables easily. /2/
- ✓ **Reusability:** Another advantage of Dita document is that it allows copying content from one place to another as a way of reusing content. Reusability in Dita documents occurs in two levels:
 - Topic reuse: As mentioned above, Dita topics have non-nesting structures; they are organized by maps and relationship tables. This allows topics to

be reused in any topic-like context. When a topic is reused in a new information model, the architecture will process it appropriately in its new context. /2/

- Content reuse: Dita inherits this feature from XML format and has some improvements on using the SGML method of declaring reusable global entities. Dita has developed a new SGML reuse technique that gives each element a “conref” attribute that can point to any other element in any topic. /2/
- ✓ **Specialization:** Dita has a mechanism engine to create a new element by extending existing element. When creating, its identifier will be added to the class attribute through its dtd. Because of this, the new element is always associated to its origin and the element hierarchy can be maintained easily.
- Topic specialization: Extends a general topic to new information types which can later be extended to other information structures. /2/
 - Domain specialization: Extends an element vocabulary into a new element which reflects a particular aspect of its parent element within a topic. This makes the whole vocabulary available throughout all the topics. For example, a keyword can be extended to be a nick name or a code. /2/
- ✓ **Property-based processing:** Dita topics can be filtered or associated by its metadata or attributes. There are many applications doing this job such as content management system, search engines, processing filters. Dita property-based processing has the following features:
- Extensive metadata: Dita metadata allows many different way of content management applied to its content, therefore searching topics is easier. /2/
 - Universal property: Almost all the elements in a topic contains a set of universal attributes which are very convenience for identification, content filter or content referencing infrastructure /2/
- ✓ **Using a set of HTML-like tags and tools:** Dita has built a set of tags which is widely accepted thank to its familiarity and compatibility with standard XML tools.

- Leverage popular language subsets: Dita borrows standard tags from other popular format for instance HTML or XHTML with familiar tag names like p, ul, ol, etc.. Moreover, Dita makes use of popular OASIS table model. /2/
- Leverage popular and well supported tools: Dita format can be easily translated to other popular tools thank to its class-based extension mechanism. It can be converted well to design featured XSLT or CSS style sheet language. /2/

2.2 Qt Framework

In order to achieve the customer requirement that the application should be able to be deployed to many different platforms, Qt was selected as a main development framework for this thesis application thank to its cross-platform compilation feature.

2.2.1 Introduction about Qt framework

Software development in mobile environment is diffused to many different platforms. Qt is one solution for this multi-platform problem. Qt is a cross-platform application and UI framework that allows developers to write applications that can be deployed to multiple platforms, from desktop to mobile phone or embedded system without the need to rewrite the source code. Qt is a superset of standard C++ so developers can use Qt or C++ data types or their combination. Nowadays Qt is being used by developers all over the world, and this notable framework is used as a key developing environment for many big projects for instance Linux KDE desktop, Google Earth and Skype. Migrating Qt application from one platform to others is often no more than a recompilation. Some operating systems can have their own custom Qt libraries.



Figure 2.1. Qt for multiple mobile platforms. /5/

2.2.2 Qt features summary

Qt contains a collection of widgets, which are equivalent to controls in Windows Form application. Like other GUI frameworks, Qt includes layout widgets such as QVBoxLayout or QHBoxLayout, component widgets such as QLabel, QPushButton, QTextEdit, etc... Based on these basic widgets, developer can build complex GUI applications. In order to control user interaction with the application, Qt introduces an innovative alternative for inter-object communication called “signals and slots” that replaces the old and unsafe callback technique used in many legacy frameworks. Qt also provides a conventional way to handle mouse clicks, key presses and other user inputs. Last but not least, Qt desktop integration features enables applications to be extended into the surrounding desktop environment by some of the services provided on each development platform. /5, 3-5/

Qt has its own visual editor called Qt Designer. It is a tool for graphically designing interfaces for UI applications simply by “drag and drop”. Users can use Qt Designer purely for GUI design, or creating entire applications with its support for integration with popular IDEs. /5, 11-12/

Qt has excellent support for 2D and 3D graphics by providing Qt wrapper for platform independent OpenGL programming. Qt’s painting system has a sophisticated canvas framework that enables developers to create interactive graphical applications taking advantages of high quality rendering across all supported platforms. Qt OpenGL also supports one of the newest 3D technology recently – stereoscopic 3D programming which allows users to see a likely 3D object with depth buffer via glasses and supported GPU and screen; for instance NVIDIA

glasses with NVIDIA Geforce graphics card, ViewSonic 3D Vision enabled screen. /5, 13-14/

Qt support connections with standard databases irrespective of platform. Native drivers for some popular databases such as Oracle, Microsoft SQL Server, PostgreSQL, MySQL... is included natively in Qt. Like Windows Form programming, Qt has database-specific widgets and any built-in or custom widget is data-awareness. /5, 24-26/

Qt applications have default look and feel of the operating system for which it is compiled. However, it is still possible to change the appearance of Qt widgets through Qt style sheets. Every built-in or custom widget has a method call “setStyleSheet” to define its custom style sheet. The format follows standard style sheet format.

```
aQLineEdit ->setStyleSheet("background-color: yellow");
```

Qt uses Unicode standard and English is the default language. Moreover, it has considerable support for internationalization by a translator named Qt Linguist. Application language can be translated to many other Unicode supported languages through separate translation file (*.ts). /5, 26-28/

Qt applications can use additional plug-ins or dynamic libraries to extend their functionalities. Additional libraries and plug-ins should be declared in project file (*.pro).

Qt offers an extensive set of tools for developing software. Qt Assistant is Qt’s documentation tool. It contains code snippets and a lot of helpful information on how to use the framework’s classes. There is also example code in each method’s description. Packed together with Qt Creator is a collection of samples demonstrating Qt key features: GUI, webkit, graphics view, scripting, OpenGL, XML, multimedia, database, network, unit test, and others. Developer can download more samples in Nokia Qt official website /4/.

✓ Platform Support for Qt /4/

- Windows
- Mac OS X
- Linux/X11 (KDE)
- Windows CE

- Embedded Linux
- ✓ Language Support for Qt /4/
- Python
 - Ada
 - Pascal
 - Perl
 - PHP
 - Ruby
 - Java (Qt Jambi)

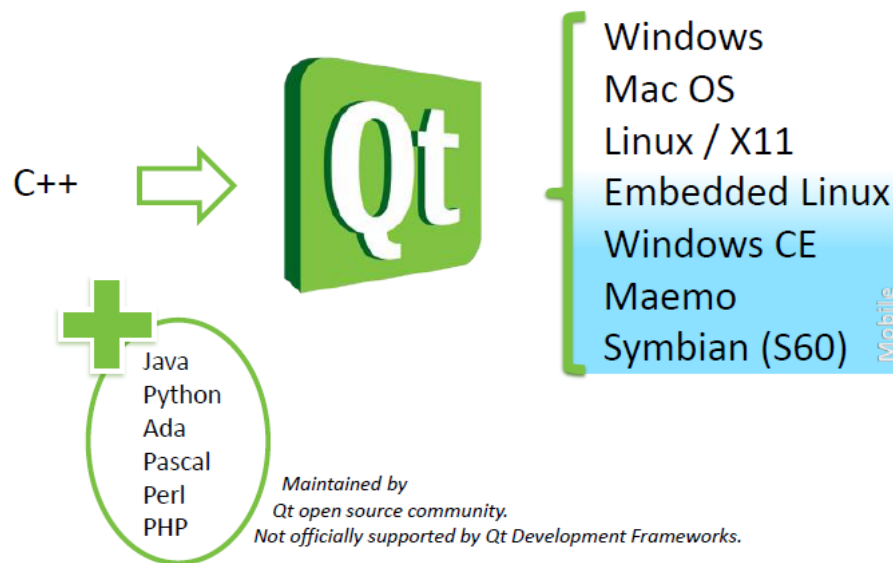


Figure 2.2. Qt supported platforms and languages. /4/

2.3 Maemo Platform Overview

Maemo is a software platform developed by Nokia for smart phones and internet Tablets. It is based on Debian Linux distribution. Maemo is mostly based on open source code, and has been developed by Nokia in collaboration with many open source projects such as the Linux kernel, Debian and GNOME. Maemo uses most of frameworks and libraries from GNOME project such as Matbox window manager, GTK-based Hildon GUI and application framework. Latest Maemo version is Maemo 5 with 4 desktops which can be customized with shortcuts and widgets. Nokia has published some smart phones having Maemo OS for example N810 and the latest phone is N900 which is one of the main target devices for this project. /6/



Figure 2.3. Maemo OS look and feel. /6/

There is also new operating system called MeeGo which is a co-operation between Maemo and Moblin OS of Intel but currently there is no phones using this operating system yet. Basically, majority of the Qt code for this new OS will be the same compared with Maemo, there might be some minor changes in UI components.

2.3.1 Maemo Main features

Below is the summary of this new platform's main features which are considered better compared with those in other Linux-based platforms.

2.3.1.1 Updating

Maemo devices can be updated either by a built-in software update tool via internet connection (seamless software update) or a simple flashing method with a computer via USB cable.

- With flashing the program will clean everything and install a fresh copy of newest maemo version, just like reinstalling an OS in desktop computers. However, personal data such as pictures and contacts which are kept in separate memory allocation remains. /6/
- Using built-in software update tool is very handy and easy to utilize. Whenever there are software updates, the phone will notify user and he can perform software updates just by few clicks. However, the limitation of this method is that it is not possible to perform full a update with major changes of the core of the OS. /6/

2.3.1.2 Security

Maemo security concentrates on preventing remote attacks via wireless network or Bluetooth or even 3G networks. Normal user has a default trivial role and limited access to the OS. However, developer can easily gain root privilege by installing a tool named "rootsh" and using the following command:

```
>>Sudo gainroot
```

2.3.1.3 Maemo components

Maemo is a modified light weight version of Debian Linux distribution targeted for mobile devices. It has a lot of changes to reduce the consumed resource and energy usage. It uses Xomap and Matchbox window manager which are X Window System-based graphical user interfaces. The GUI use GTK+ toolkit and Hildon user interface widgets. All installed widgets will have default Hildon look and feel.

Maemo replaces the GNU core Utilities used in Debian by a new software package for embedded and mobile devices called BusyBox in order to reduce memory and

storage usage. As a result it will have some limitations compared with GNU core utilities but that is minor and can be ignored.

Maemo uses GNOME ESD (Enlightened Sound Daemon) as the primary sound server and GStreamer to play back sound and movie. GStreamer is used in shipped media player and any many Qt media applications in maemo repository [7]. Like any Linux distribution, the format supported by GStreamer can be extended by installed additional plugins (gstreamer-plugins-bad, gstreamer-plugins-good).

Window management is controlled by Matchbox window manager, which displays only one window at a time due to limited screen of a phone. Other windows will be minimized and displayed in a summary screen which allows user to have easy navigation between them. This effect has improved the handheld usability of a mobile device with a small screen.



Figure 2.4. Maemo multitasking with multiple processes running.

2.3.1.4 Software

Maemo has a collection of built-in applications which provides mobile phone users with common features such as calling, messaging, web browser, multimedia, etc. Furthermore, user can install additional applications from a number of sources, including official and various community repositories through a built-in application called “Application Manager”. Programmers can also install applications through

command line apt-get just like any other Linux distributions. Some of the most commonly known sources for developers:

- Maemo official repository. /7/
- Maemo development repository. /8/

Third-party application: Thank to the free and open source nature of Linux and Maemo, porting applications to Maemo is a straightforward procedure. Therefore, Maemo can utilize a huge amount of existing open source projects developed for Linux distributions. Moreover, Nokia and community developers are also creating mobile featured applications for Maemo platform which are available for download from Nokia OVI store /9/. These advantages enable Maemo users to have a considerable repository compared with Iphone's or Android's.

2.3.2 Software development on Maemo platform

Software development for Maemo is possible using many programming languages such as C, C++, Java, Python, Rubi, Mono and others. Especially Qt is widely used as the main development framework for Maemo because of its usability, simplicity and platform independence. The development environment used in development process is called Maemo SDK which can be freely downloaded from Maemo website /10/. The Maemo SDK uses Scratchbox as a cross-compilation toolkit and a "Sanbox" which is designed for embedded Linux application development. Developer can download and install Scratchbox from its website /11/ or install Scratchbox together with Maemo 5 SDK and Nokia closed binaries using Maemo 5 SDK GUI installer. /10/

The Maemo SDK provides a development environment for creating software to smart phones and internet tablets under Linux OS. The SDK runs inside Scratchbox and includes all required compilers, tools, libraries and headers to develop software for the two target hardware architectures, Intel x86 and Armel. /10/

During development process, the software is testing in x86 environment, which also includes the Hildon desktop for running the applications on desktop computer using virtual X server called Xephyr. It acts as a simulator of Maemo OS and provides the application with UI and functionality of the real Maemo OS. Using desktop computer makes the application development easier, just like with normal Linux application development. Furthermore, Maemo SDK has integrated to Eclipse IDE in a tool called Esbox to speed up the development process radically. /10/

After preliminary debugging on desktop computer is finished, developer can do the cross platform compilation and package the application for Armel architecture using the Armel target of the Scratchbox. Then the application can be deployed directly to a Maemo phone using Phone-PC connectivity or indirectly using deb installation file. This phase insures that the application is working properly as there are differences between the SDK and real Maemo OS.

Development tools and resources:

- Scratchbox: a cross platform compilation toolkit for embedded Linux application development. It includes a collection of tools to cross compile an entire Linux distribution under x86 and Armel architecture. /10/
- Maemo SDK rootstraps: a target root file system image for Scratchbox that serves as a basis for development. Maemo SDK supports rootstraps for both x86 and Armel architecture. /10/
- Nokia binaries: closed libraries whose source codes are not available but may provide public API so that developers can utilize and integrate into their applications. Some examples are contact information import/export libraries, GPS libraries, address book libraries etc. /10/
- Maemo tools: In addition to basic tools, Maemo SDK provides developers with more sophisticated development tools such as code analysis, debugging, memory leaks, test automation. /10/
- Maemo repositories: maemo.org website has a collection of repositories targeted for standard Debian package installation tools. /10/
- Maemo documentation: documentation for Maemo software development provides developers with configuration manual, tutorials, API references, sample codes and many other guides, available from maemo.org website. /10/

3 ELDOC SERVICE BOOK

Below I will give a description about the general idea, the development outline and functional specification of the application. In this section user will have a general view how the development process was planned and conducted. Moreover, in the part description of main functions, important features of the application will be described in order to achieve user's overall understanding before the implementation part which will analyze the implementation code.

3.1 Requirement Analysis

This Eldoc Service Book is mainly used by Wärtsilä mechanical engineers who are maintaining the engines in every day. They need to read information of different types of engine and spare part in order to do the maintenance. Therefore, the application is made so that user can easily access the needed information with minimum interactions with the phone.

User first choose product name, default language English, document type, and then depending on document type user can choose manual or spare part number, a related document will be displayed. In the document there are links where user can click on it to view image, video or navigate to other documents. User can also add note, take pictures and attach to the document.

In order to improve the accessibility and simplicity of the application, it provides users the ability to have one-touch access based on engines' barcode. User clicks on search to open search window and scan engine's barcode, when the code is recognized, corresponding document will be opened.

If there is problem with an engine, user can create new request, take pictures and attach them in the request, after click 'send', an email will be sent to resolution team.

When the phone is connected to internet, the data on the phone and on server will be synchronized.

Application should be available in three platforms: Windows, Linux and Maemo which are on the following portable devices:

- Rugged PCs – Xplore, Panasonic Toughbook – running standard Microsoft Windows XP operating system and normal programs.
- Laptop running open source Linux operating system

- Nokia smart phone N900 running Linux based Maemo operating system

Theoretically the Qt code for these three platforms should have 95% similarity with only some minor platform independent libraries or code changes. Nokia N900 with Maemo OS is the main targeted device thank to its capability, mobility, and reusability.

3.2 Main Features Deployment

The main and most important feature of the application is to display the user manual. The application saves its own data on the machine and can be updated with the server through internet connection or CD room or USB etc.

In addition to these main features, the application provides users visualization of engines/spare parts by a collection of related images and videos.

In Maemo phone version, the application can utilize the camera and messaging system of the phone on implementing additional features including taking and attaching new pictures, barcode search engine, and sending technical support request.

Priority level:

- 1: Must-have features.
- 2: Should-have features.
- 3: Nice-to-have features.

Table 3.1. Main features and their priorities.

Task number	Name	Priority
1	Display User manual	1
2	Synchronization with server	1
3	View pictures	2
4	View videos	2
5	Add notes	2
6	Take and attach photos	3
7	Search by scanning engine code	3
8	Send technical request	3

3.3 Main Functions Specification

User who is Wärtsilä engineers can have two ways to access needed documents, either by browsing it or searching using engine's barcode. After opening a document, user can view pictures, videos, add notes or take and attach more pictures. If there is problem with a particular engine or spare part, he can create a technical request, take and attach some pictures to illustrate the problem and send to the resolution team.

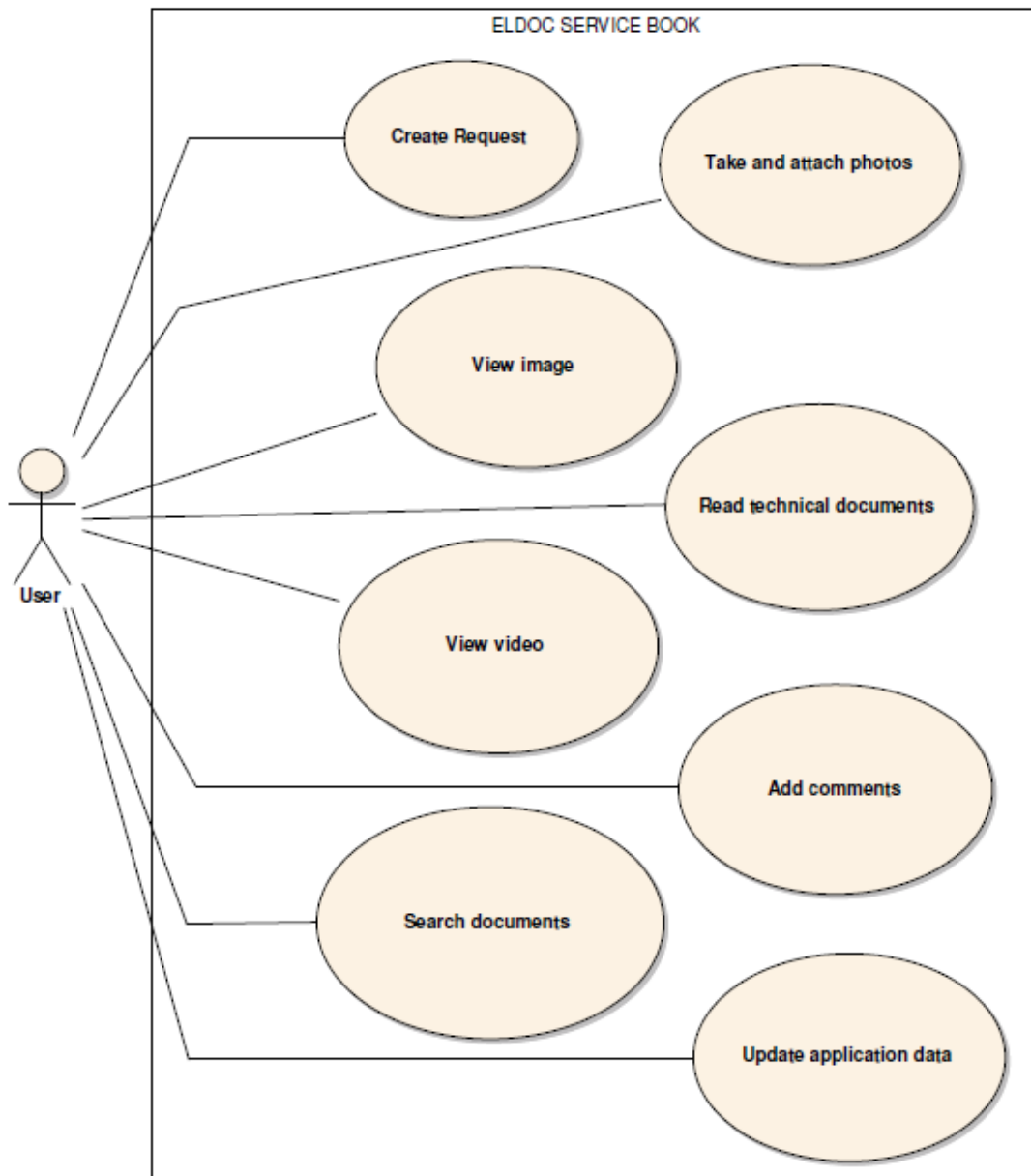


Figure 3.1. Application main functions.

The application data can be updated automatically when the internet connection with Wärtsilä's local intranet is created. However, user can still update the application manually using update option in the application menu. In case the phone does not have internet connection, user can also copy the newest application data to the phone using memory stick, CD room, etc.

3.4 Class Hierarchy

Documents include manuals and spare parts. Manuals are engines' related information and spare parts are technical details about individual parts of an engine. Depending on document types we have different class structures and parser methods. Some complex relations are cut into different parts to simplify the diagram.

3.4.1 Manual related classes

The structure of a manual contains many different child tags. First of all, in the very beginning of each manual is a tag named "Convinfo" where basic information of the document such as language, version, reference chapters, etc is listed. Like HTML document, the main content of a manual contains many heading tags h0; an h0 tag may embrace lower level tags h1, h2 ... An h* tag is followed by one paragraph (Paragraph). The content of each paragraph can include text, note (Note), figure (Fig), and unordered list (WP). A member of an unordered list can include note and figure. A note can also have figure inside. The structure of a manual document can be visualized in below class diagram.

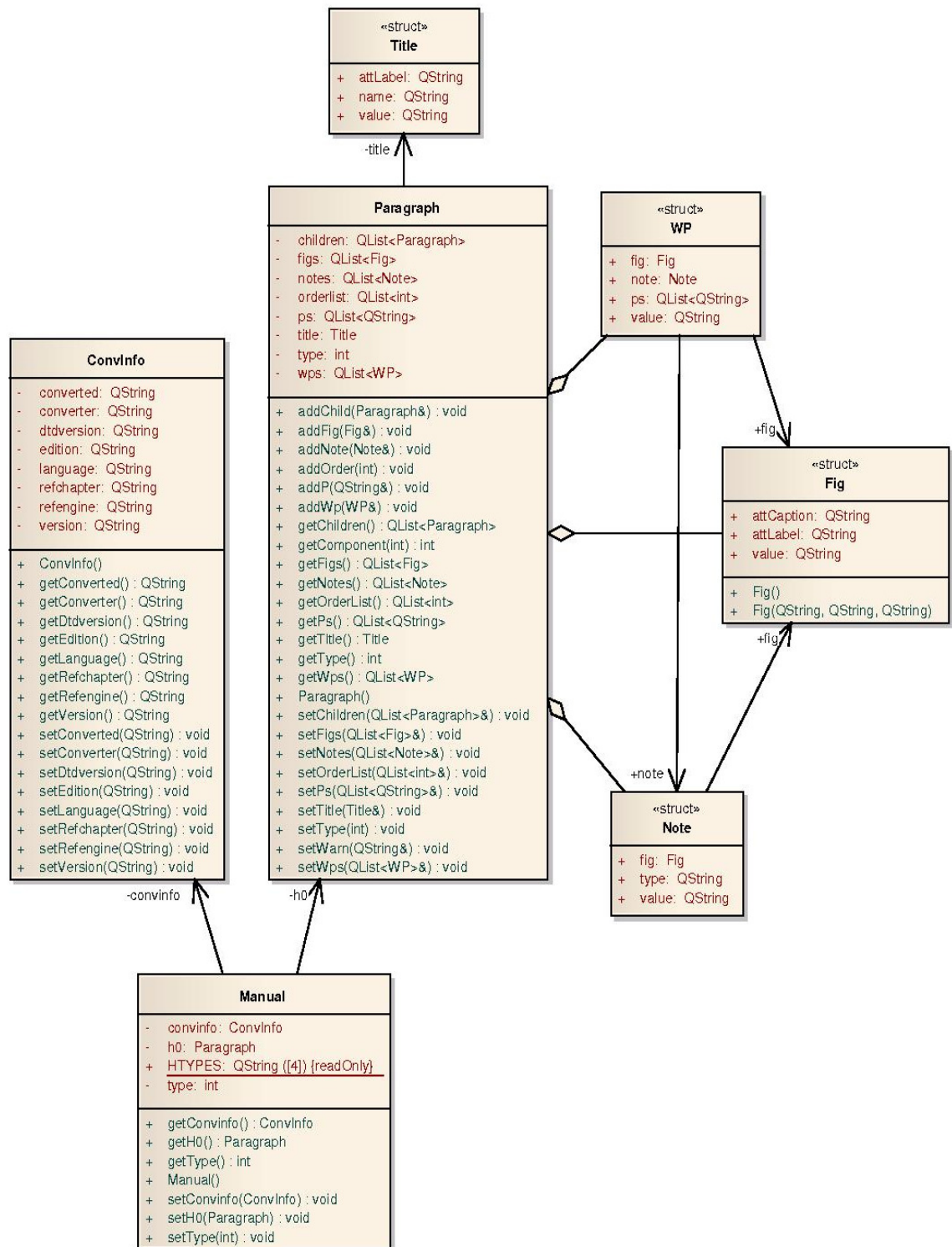


Figure 3.2. Structure of a manual document.

These are object classes with their getter and setter methods. Manual class represents an instance of manual document. It two main attributes:

- `convinfo`: This instance of class `Convinfo` includes basic information of the engine such as language, version, reference chapters, etc.
- `h0`: This `h0` tag which is an instance of class `Paragraph` includes the main content of a manual document. Each `h0` tag may include lower level tags `h1`, `h2`, etc which are also instances of class `Paragraph`.

Attributes of a `Paragraph` class:

- `children (QList<Paragraph>)`: As mention above, each `Paragraph` contains a list of other child `Paragraphs` of the same type. This interesting feature is not supported by C++ but Qt.
- `figs (QList<Figure>)`: a list of figures in this paragraph.
- `notes (QList<Note>)`: a list of notes in this paragraph.
- `ps (QList<QString>)`: a list of `<P>` tag in this paragraph.
- `wps (QList<WP>)`: an unordered list in this paragraph.
- `title (Title)`: title of the paragraph.
- `type (int)`: defines type of the paragraph such as `h0`, `h1`, `h2`, etc.
- `orderlist (QList<int>)`: defines the order that these above components will appear in the paragraph.

There are many other small tags. However, to simplify the code, the parser will parse them into normal string with HTML tag directly. This will be explained more in the parser code.

3.4.2 Spare part related classes

The content of a spare part document is much simpler compared with manual. A spare part document includes many pages. Each page has one image and a collection of portions. Each portion can contain other child portions. Normally a portion is a row in the table of description of that spare part. The structure of a spare part document can be visualized in below class diagram.

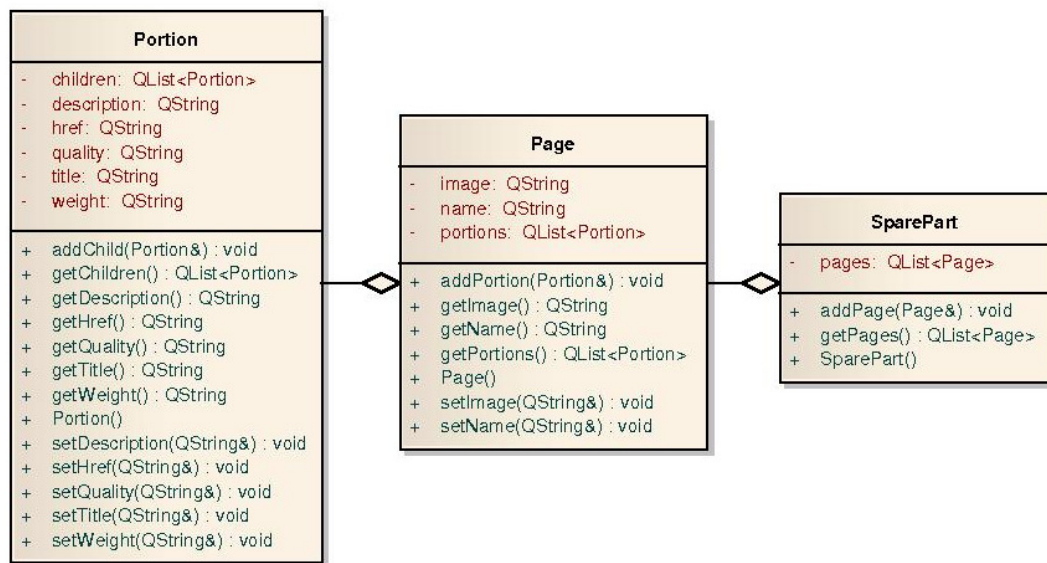


Figure 3.3. Structure of a spare part document.

These are object classes with their getter and setter methods. SparePart class represents an instance of spare part document. It contains a list of pages. Attributes of Page class:

- image (QString): the image in the page.
- name (QString): title of the page.
- portions (QList<Portion>): list of portions in the page.

Attributes of Portion class:

- children (QList<Portion>): a list of child portions.
- description (QString): content of the portion.
- href (QString): link to an image if available.
- quality (QString): number of portions in the engine. It is a string because it just needs to be displayed, no calculation.
- weight (QString): Weight of the portion.

3.4.3 Main UI classes

EldocServer is the main class of the application which receives user interactions and implements UI changes. Some of the most important functions' description:

- openDocument: open manual or spare part document.
- viewImage: view a specified image.
- viewVideo: view a specified video.
- uploadPicture: take and attach pictures to the document.
- addNote: add note to the document.
- openBarcodeReader: open the search engine.
- updateDatabase: synchronization with server.
- openRequest: open request form
- openSetting: open setting window.
- clearContent: clear the current content.
- goNextPage: move to next page when viewing pdf document.
- changeMenuVisibility: hide/show the menu.

The structure of these UI components can be visualized in below class diagram:

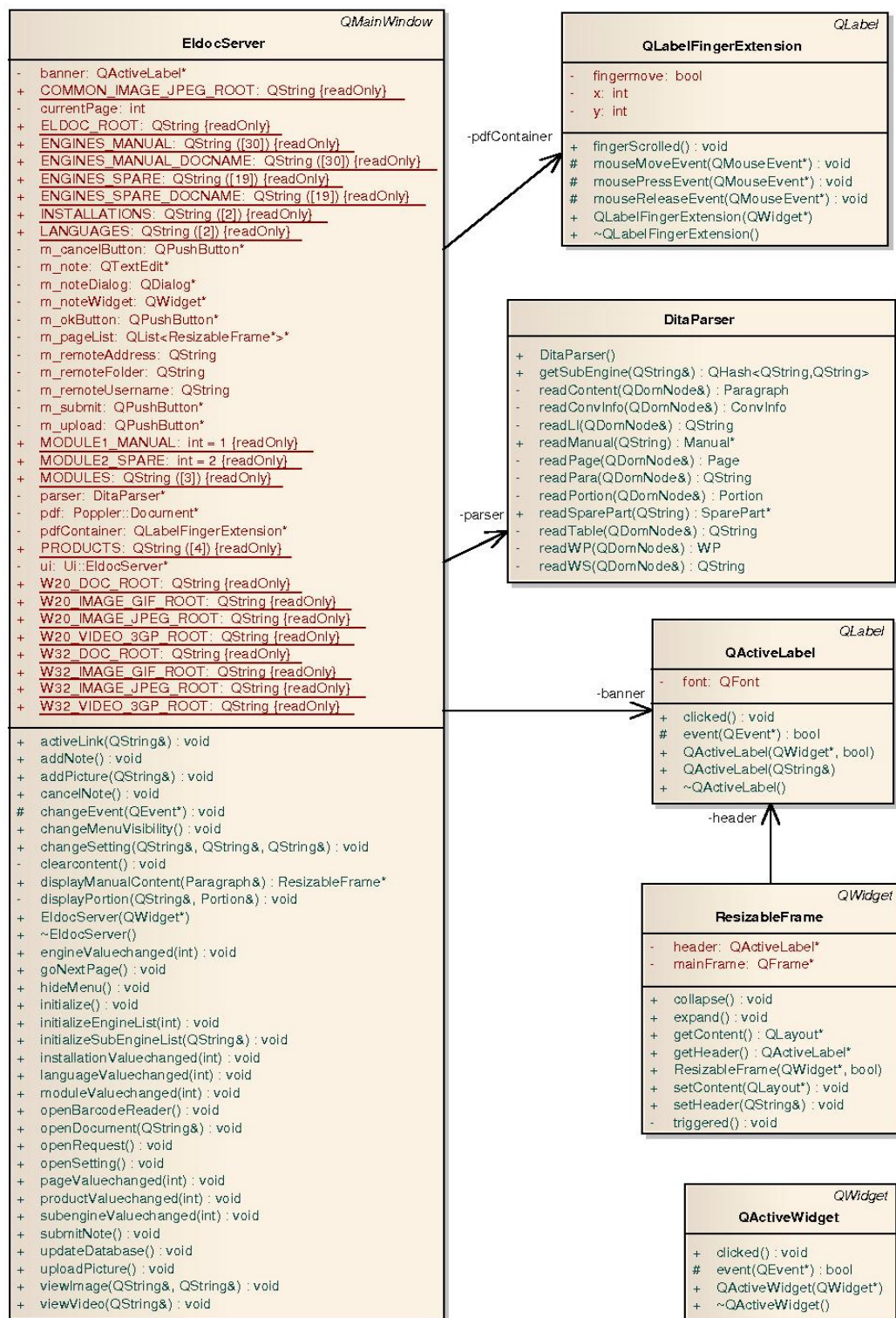


Figure 3.4. Eldoc Service Book main UI classes.

EldocServer class includes many constant attributes (attributes in capital) keeping application's global setting such as resource folder location, image folder location, manual document names, spare part document names, etc. Some of important attributes of EldocServer class:

- parser (DitaParser): parser class for converting documents from Dita format to HTML format.
- banner (QActiveLabel): banner on top of the application.
- m_noteDialog (QDialog): popup dialog for adding note. This widget contains m_note, m_okButton and m_cancelButton in its layout.
- ui (Ui::EldocServer): UI implementing class.
- pdf (Popler::Document*): pdf renderer.
- m_pageList (QList<ResizableFrame*>): a list keeping all the pages while displaying spare part document.

EldocServer class also includes other classes as helpers for its functionality:

✓ **DitaParser:** A parser converting documents from Dita format to HTML format. It has two main public functions “readManual” and “readSparePart” for converting the two types of document. Description of its main methods:

- getSubEngine: get sub-engine of the selected engine. This method is used when user is browsing a spare part document.
- readManual: read the content of a manual document.
- readConvinfo: read the information part of the document.
- readContent: read the main content part of a manual document.
- readSparepart: read the content of a spare part document.
- readPage: read the content of a page in the spare part document.

- ✓ **QActiveLabel:** An UI class implementing the Wärtsilä banner on top of the UI which has show/hide feature. This feature is also complemented by the class QActiveWidget. This class has an important custom signal:
 - clicked: a signal emitted when user clicks on the label. This is a user defined signal as normal QLabel does not have this signal.
- ✓ **ResizableFrame:** An UI class implementing a heading paragraph (h0, h1, h2...) which is the main content of a manual document. This class also has auto-hide feature when user clicks on the heading in order to reduce the display of unnecessary information. It has two main functions:
 - collapse: hide the content of the paragraph.
 - expand: show the content of the paragraph.
- ✓ **QLabelFingerExtension:** An UI class which extends QLabel class and implements finger swiping feature. It will emit a defined action when user swipes his/her finger across the screen. It has the following custom signals and events:
 - mouseClickEvent: signal is emitted when user click on the label.
 - mousePressEvent: signal is emitted when user hold the mouse press on the label.
 - mouseReleaseEvent: signal is emitted when user releases the mouse.
 - fingerScrolled: the combination of these three signals above under a specified circumstances will trigger this signal.

3.4.4 Optional features UI classes

✓ **TSCMobile** (Technical service calls mobile): is the UI class implementing sending technical request feature. Pictures can be taken and attached to the request by function “addPicture”. Request is sent by function “sendRequest”. Main attributes of the class:

- ui (Ui:TSCMobile): UI implementing class.
- parent (QWidget*): As this is the stacked window, it has a parent. This variable keeps the value of the parent window so that it can be called to display again when the class TSCMobile is destroyed.

Main functions of the class TSCMobile:

- addPicture: add taken picture to the request.
- viewPicture: view taken picture.
- sendRequest: send the request and go back to main view.
- cancelRequest: cancel request and go back to main view.



Figure 3.5. Request form and setting window.

✓ **SettingWindow:** settings for the connection between phone and server. There are three parameters in the EldocServer class which can be changed in this setting window:

- m_remoteAddress: the IP address of the host machine.
- m_remoteFolder: the folder in the host machine where data is saved.
- m_remoteUsername: username of the authenticated user.

3.4.5 Mbarcode project integration

The MBarcode open source project has been modified and integrated to the application. This project is using the two common open source libraries libdmtx and zbar to read barcode. In order to simplify the class diagram, it has been cut into 2 parts: barcode decoder classes and UI classes although there are relations between these two parts. The class structures will be described below.

3.4.5.1 Barcode decoder classes

This is the main decoder engine containing three classes. The main class is BarcodeDetector which reads the barcode from video widget and convert it to the real engine/spare part document number. This class contains two implementation classes DMTXDecoderThread and ZBarDecoderThread to read matrix data. Some of its main methods:

- start: start the pipeline and decoding.
- stop: stop the pipeline and decoding.
- analyseImage: analyze the captured image to find the data matrix.
- getLastImage: return the last attempted decode of either the zbar or dmtx thread.
- doFocus: auto focus on barcode in the image.
- dmtxCallback: callback for dmtx GStreamer element.
- zbarCallback: callback for ZBar GStreamer element.

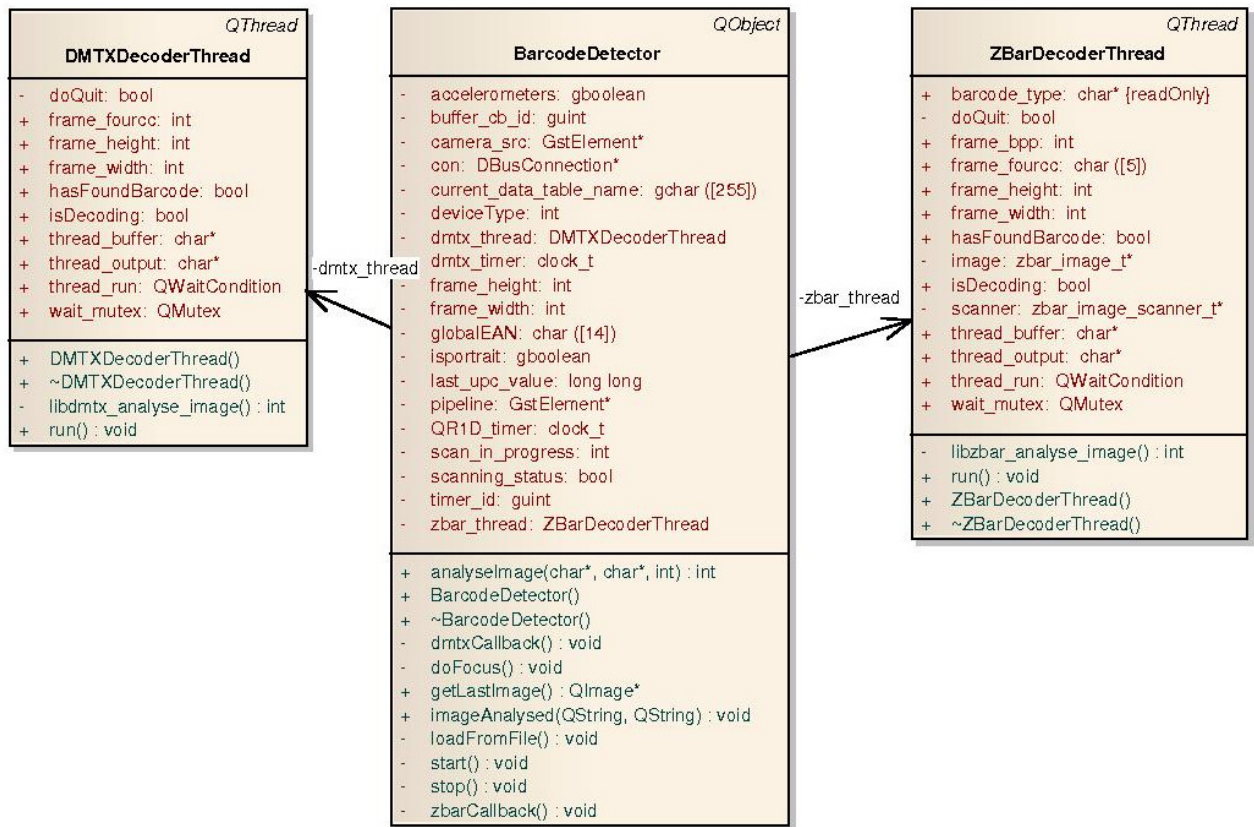


Figure 3.6. Barcode decoder classes.

The main barcode detector class is **BarcodeDetector**. It includes two child classes as helpers for its functionality: **DMTXDecoderThread** and **ZBarDecoderThread**. They are two independent threads using two different libraries and running simultaneously to read the barcode. The parent class **BarcodeDetector** will get the values of the thread finding the matrix data first.

DMTXDecoderThread class uses open source library **libdmtx** as its decoder and **ZBarDecoderThread** uses open source library **Zbar** as its decoder. Basically the two classes have the same attributes and methods; the difference is the way each method is implemented. Below is the description of these two important decoder classes.

- ✓ **DMTXDecoderThread**: The first class for reading data matrix barcode. It uses open source library libdmtx for the implementation. Main functions:

- run: start the thread.
- libdmtx_analyse_image: read data matrix from the image.

Main attributes:

- thread_buffer: Allocated buffer for the thread.
- thread_output: Output character of the thread.
- frame_width: width of a frame.
- frame_height: height of a frame.
- isDecoding: Indicate that the thread is running.
- wait_mutex: allow the caller to block the thread when data is not available.
- hasFoundBarcode: variable set if the decoder found something.

- ✓ **ZBarDecoderThread**: The second class for reading data matrix barcode. It uses open source library Zbar for the implementation. Main functions:

- run: start the thread.
- libdmtx_analyse_image: read data matrix from the image.

Main attributes:

- thread_buffer: Allocated buffer for the thread.
- thread_output: Output character of the thread.
- frame_width: width of a frame.
- frame_height: height of a frame.
- isDecoding: Indicate that the thread is running.

- `wait_mutex`: allow the caller to block the thread when data is not available.
- `hasFoundBarcode`: variable set if the decoder found something.

3.4.5.2 Mbarcode UI classes

In the below class diagram, `BarcodeDetector` is the class which has been mentioned in the previous chapter. The main UI class is `MainWindow` which is responsible for UI rendering. It contains a `VideoWidget` and a `BarcodeDetector` widget. When barcode is found, the result will be displayed in stacked window `ResultsWindow`.

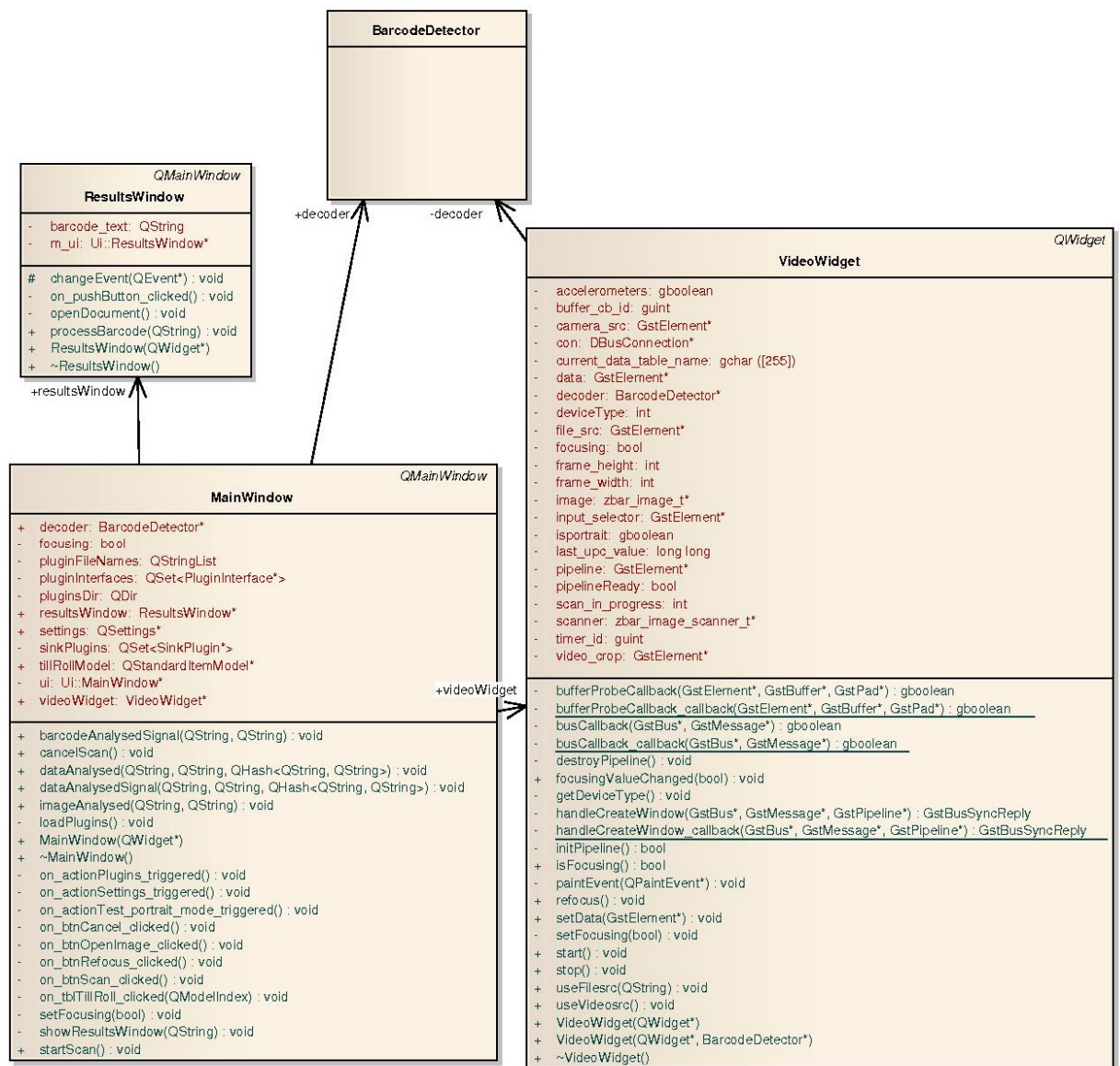


Figure 3.7. Mbarcode main UI classes.

Some of the main methods of MainWindow class:

- `startScan`: start scanning.
- `cancelScan`: stop scanning.
- `imageAnalysed`: Analyze the image to get barcode data.

- `dataAnalysed`: Analyze the barcode.
- `showResultsWindow`: show the result window when barcode has been found.
- `on_actionTest_portrait_mode_triggered`: change from landscape to portrait mode.
- `on_btnRefocus_clicked`: focus on the barcode in the image.
- `loadPlugins`: loading additional plug-ins.

Other important class is Video widget which is used to display the captured video. Below are some of its main methods:

- `start`: start the display.
- `Stop`: stop the display.
- `initPipeline`: initialize the pipeline.
- `destroyPipeline`: free the pipeline.
- `busCallback`: a callback function which retrieves any message from the GstBus.
- `getDeviceType`: get type of the device.
- `Refocus`: refocus on the image.

3.4.6 Mbarcode additional plugins

Mbarcode project also provides a collection of plug-ins in order to enhance its usability. Depending on format of the barcode, which can be a URL, a phone number, text, the plug-in will navigate the application to a browser; call a person, sms, etc. Because these features are not used in this application, I only give a short description. However, reader can still find more information in the description at the beginning of each class.

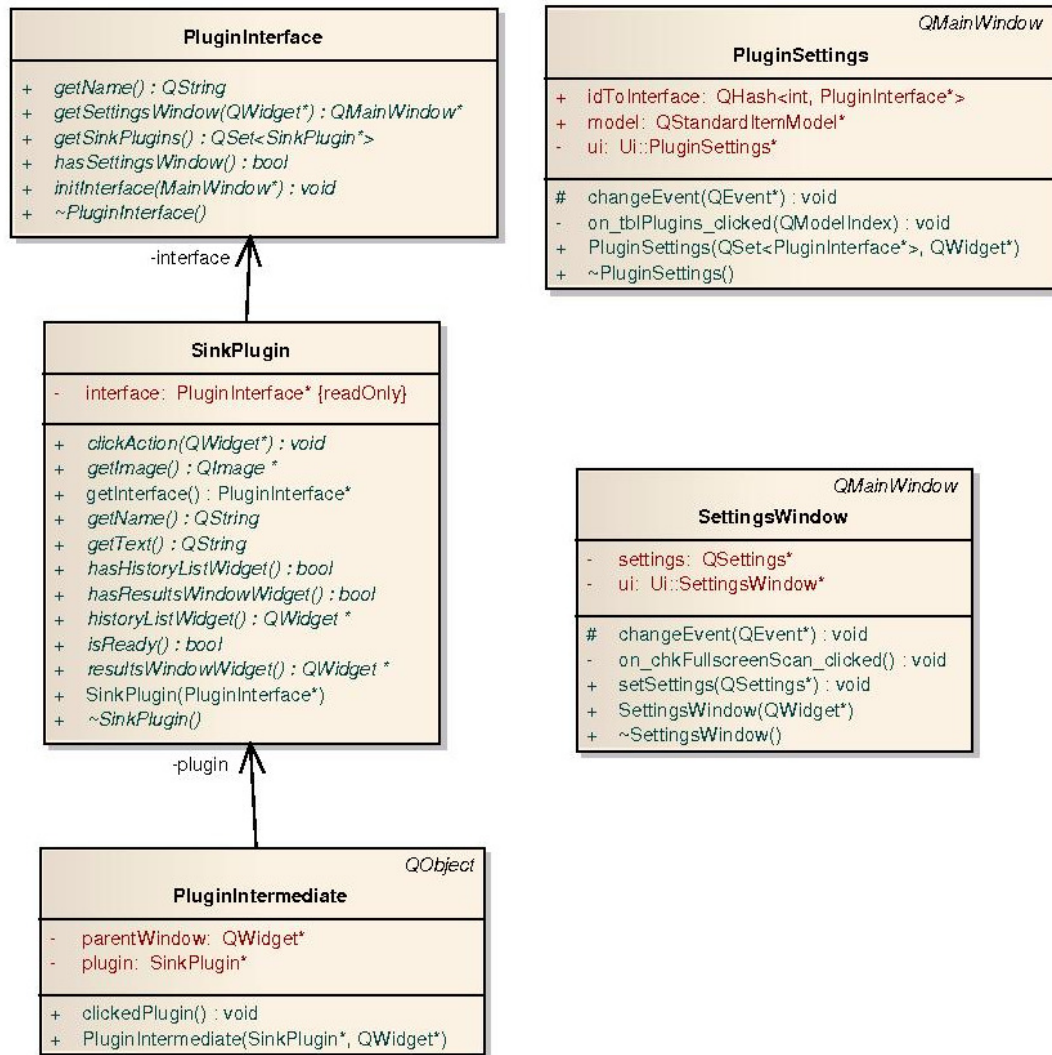


Figure 3.8. Mbarcode additional plug-in classes.

3.4.7 Media player classes

Below are three media classes including image viewer, picture capture and video viewer. These UI components are quite simple with a main widget to display video or image. They are all implemented as stacked windows.



Figure 3.9. Image viewer and picture capture.

- ✓ **ImageView:** display picture when user clicks on an image link in the document or reviews taken picture. Its methods:
 - initialize: initialize the image viewer.
 - setImage: set the URL of the image needs to be displayed.
- ✓ **PictureWindow:** Implement a thread for monitoring and getting taken pictures. Its methods:
 - initialize: initialize the picture window.
 - getTakenPictures: get taken pictures.
 - isIncluded: check whether the image is new or already existed.
 - closeEvent: event triggered when user closes this window to return to main window.

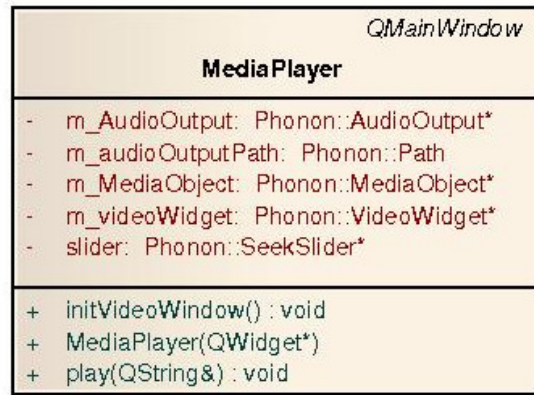


Figure 3.10. Media player.

- ✓ **MediaPlayer:** A class for displaying videos. It uses Qt phonon open source library for the display. Main components of this media player are a media object, a video widget and an audio widget which are linked together. Its methods:
- `initVideoWindow`: initialize the video widget.
 - `play`: play the requested video.

3.5 Detailed Descriptions of Main Functions

This part will give you the general idea of functionality of the application. All the functions listed in the use-case diagram mentioned above will be described in detail. The implementation codes of these functions will be analyzed in later chapters.

3.5.1 Open document

In order to access the correct document, user has to complete some nested selectors. User first chooses default installation Demo, product name, default language English, document type, engine number. If document type is spare part, there is one more selector for sub engine. Finally a related document will be displayed.

Because the structures of manual and spare part documents are different, depending on the product type manual or spare part there are different methods for reading and displaying document. The sequence diagram is described below.

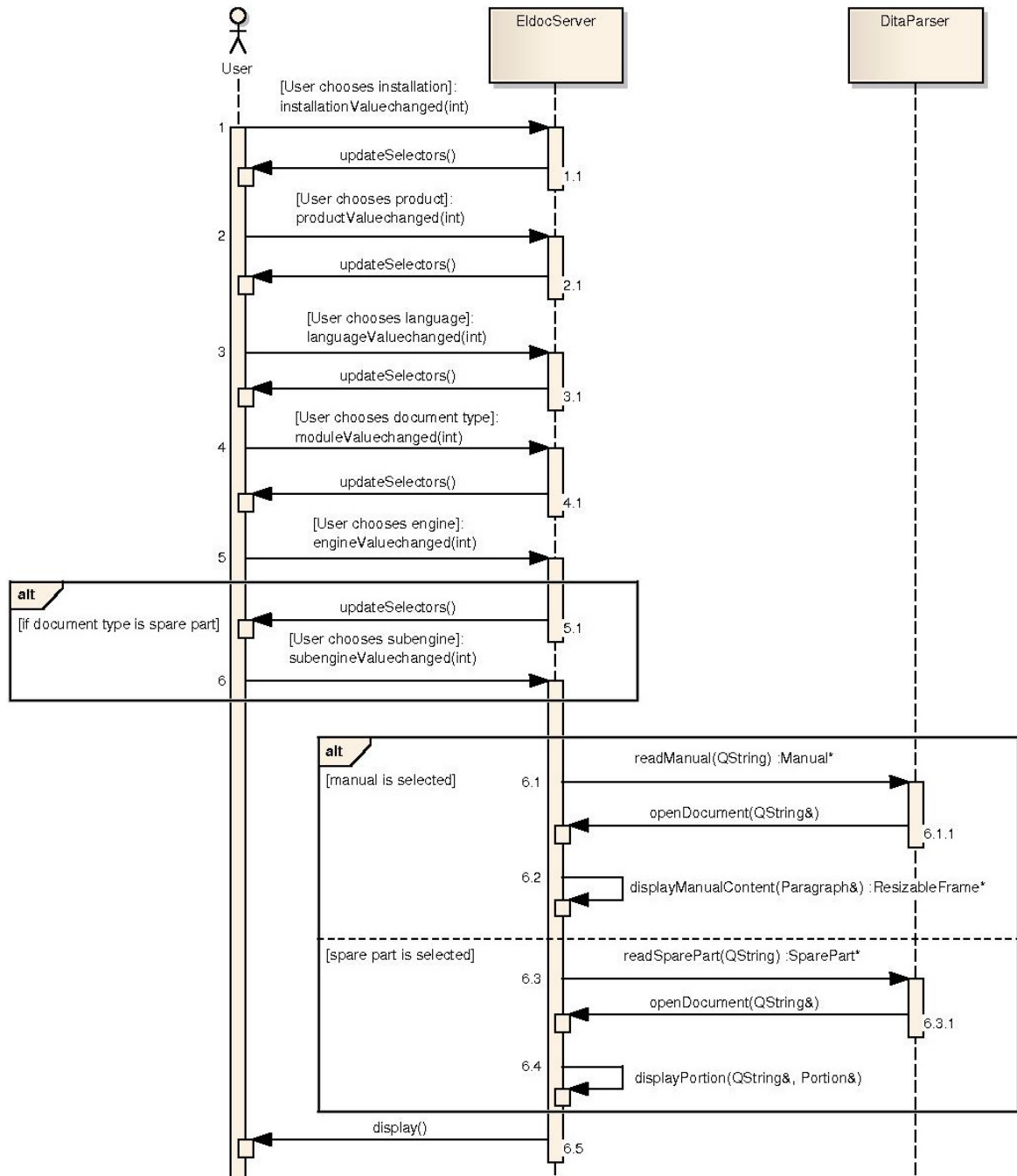


Figure 3.11. Sequence diagram - Open document.

The sequence can be described below.

- User chooses an installation; Eldoc Service Book will show an additional selector of products.
- User chooses a product; Eldoc Service Book will show an additional selector of language. English is selected as default language.
- User chooses a language; Eldoc Service Book will show an additional selector of document type (or module). There are two document types: manual or spare part.
- User chooses a document type; Eldoc Service Book will show an additional selector of documents.
- If the document type is spare part, there is one more selector for sub-engines.
- User chooses a document; depending on document type, Eldoc Service Book will have different methods to read the document and display it.

3.5.2 View image

When user clicks on an image link, Eldoc Service Book will open a new stacked window (ImageViewer) to display the image. After that, user can click on “Back” button to exit image viewer and return to the original window.

The image is automatically adjusted to fit the screen size. All the images are stored in image and common folder inside resource folder.

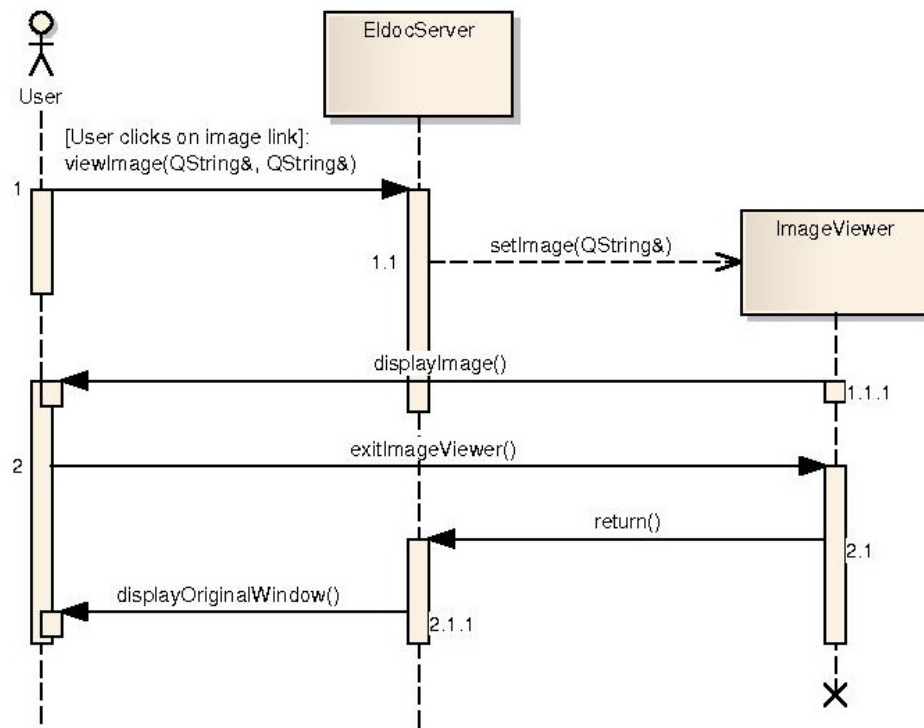


Figure 3.12. Sequence diagram – View image.

The sequence can be described below.

- User clicks on an image link; Eldoc Service Book opens a new instance of class `ImageViewer` as a stacked window and set the URL of the image needs to be displayed.
- `ImageViewer` displays the image.
- User clicks on the “Back” button, Eldoc Service Book will destroy the `ImageViewer` and restore original window.

3.5.3 View video

When user clicks on a video link, Eldoc Service Book will open a new stacked window (`MediaPlayer`) to display the video. After that, user can click on “Back” button to exit video viewer and return to the original window.

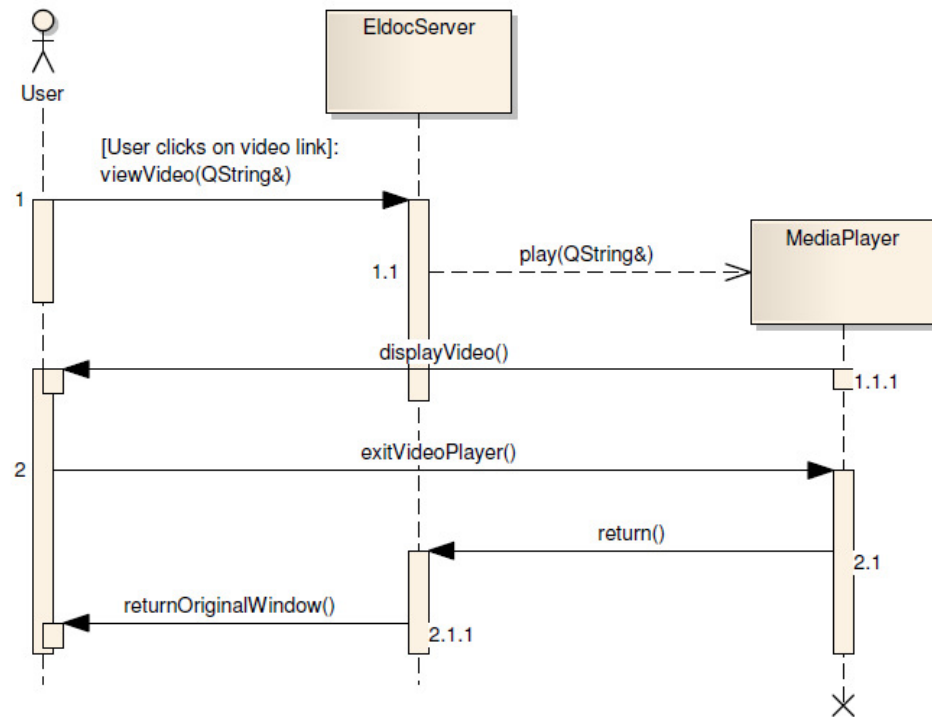


Figure 3.13. Sequence diagram – View video.

The sequence can be described below.

- User clicks on a video link; Eldoc Service Book opens a new instance of class MediaPlayer as a stacked window and set the URL of the video needs to be displayed.
- MediaPlayer displays the image.
- User clicks on the “Back” button, Eldoc Service Book will destroy the MediaPlayer and restore original window.

3.5.4 Add note/comment

User clicks on “Add Comment” button, a new comment dialog new be displayed. User enters the note/comment and click “OK” to submit note or “Cancel” to discard the note. New added note will be display at the end of the document, right after the “Add Comment” button.

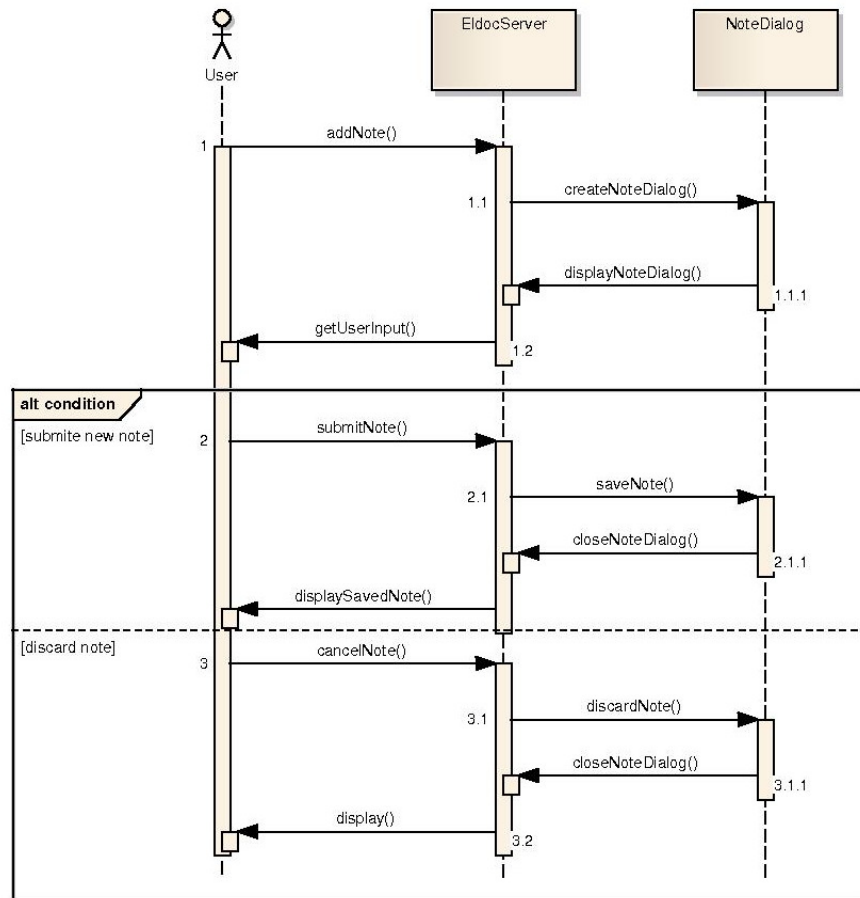


Figure 3.14. Sequence diagram – Add note/comment.

The sequence can be described below.

- User clicks on “Add Comment” button; Eldoc Service Book opens a new popup dialog for adding comment.

- User enters note/comment in the textbox.
- If user clicks on “Save” button to save the note, Eldoc Service Book will add the note to the end of the document and close the dialog.
- If user clicks on “Cancel” button to discard the note, Eldoc Service Book will close the dialog.

3.5.5 Take and attach picture

User clicks on “Attach picture” button, Eldoc Service Book will open a new stacked window (PictureWindow). This picture window UI class will create a thread monitoring taken pictures. When user takes a picture, it will be recorded and updated in the picture window. Finally user can click “Back” button to end the thread and attach taken pictures to the document. Like notes, taken pictures will be displayed at the end of the document.

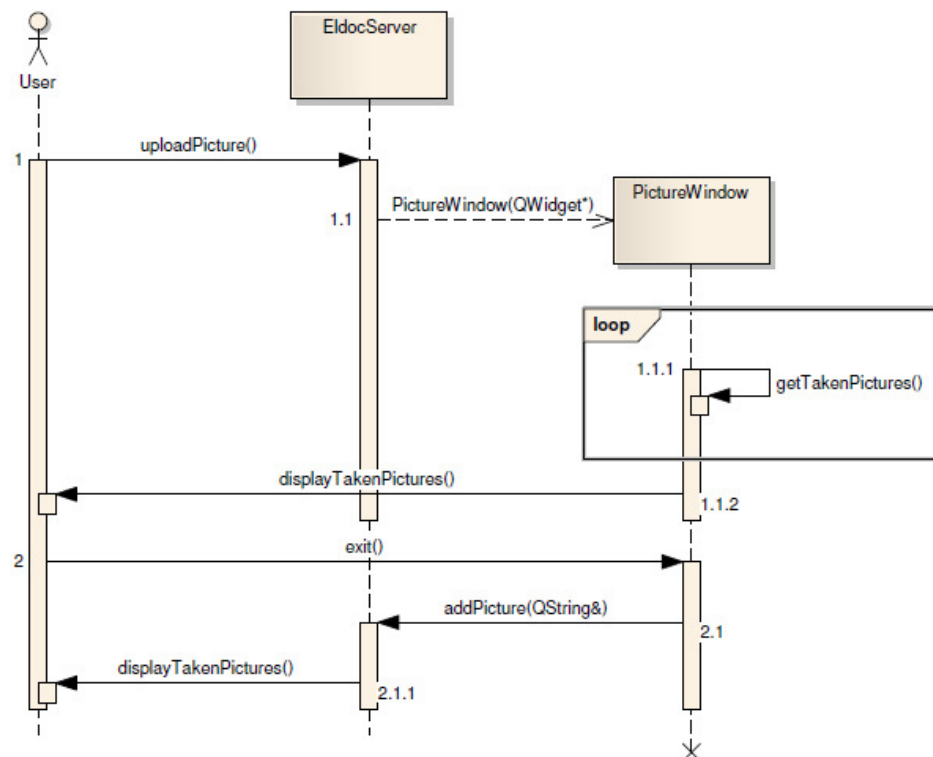


Figure 3.15. Sequence diagram – Take and attach picture.

The sequence can be described below.

- User clicks on “Attach picture” button; Eldoc Service Book opens a new instance of class PictureWindow as a stacked window for getting taken pictures.
- User takes pictures. The action getTakenPictures will be repeated every second to get taken pictures and display them.
- User clicks on the “Back” button, Eldoc Service Book will add taken pictures to the end of the document, destroy the PictureWindow and restore original window.

3.5.6 Search manual/spare part document

User clicks on Search menu item to open the search utility which is a barcode reader. It will be opened as a new stacked window. This utility will create a thread analyzing the data. When data has been analyzed and the barcode is recognized, it will be displayed in result window (ResultsWindow UI class). User can click on “Open Document” button to return to the main window and searched document is displayed.

The search engine is represented by BarcodeEngine in below figure. However the real class name is MainWindow in package mbarcode.

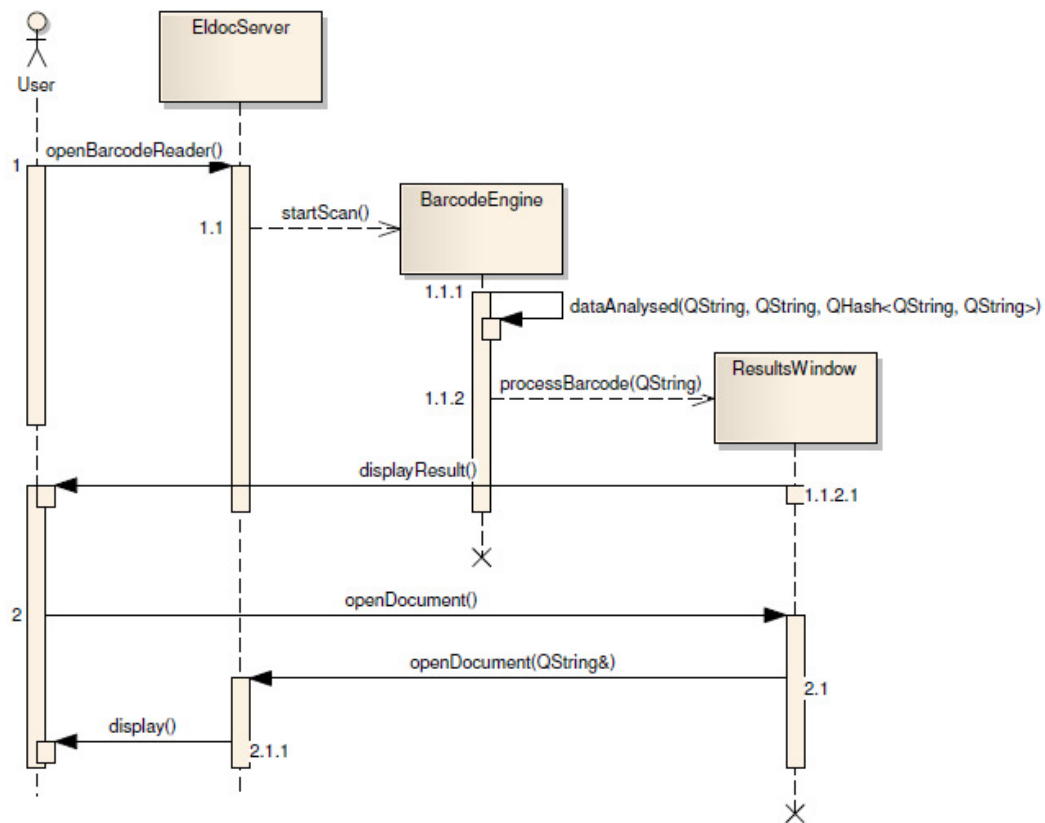


Figure 3.16. Sequence diagram – Search engine.

The sequence can be described below.

- User clicks on Search menu item to open the search utility; Eldoc Service Book opens BarcodeEngine for scanning barcode.
- BarcodeEngine creates a thread analyzing the data.
- When the barcode has been found, BarcodeEngine will create a new instance of ResultsWindow to display the result.
- User clicks on “View Document” button to view the searched document.

3.5.7 Create technical request

Whenever problem occurs, user can create a request to get technical support from resolution team. User clicks on Request menu item to create new technical service call (TSCMobile). In New Request window, user information such as name, telephone number will be automatically retrieved. The product name will also be fetched based on the document he is reading. User can write a description of the problem, take and attach pictures. These happen the same way with take and attach pictures described above.

Finally, user can click on “Send request” button to send the request to resolution team and return to main window. A notification will be displayed when the request has been sent or error occurs.

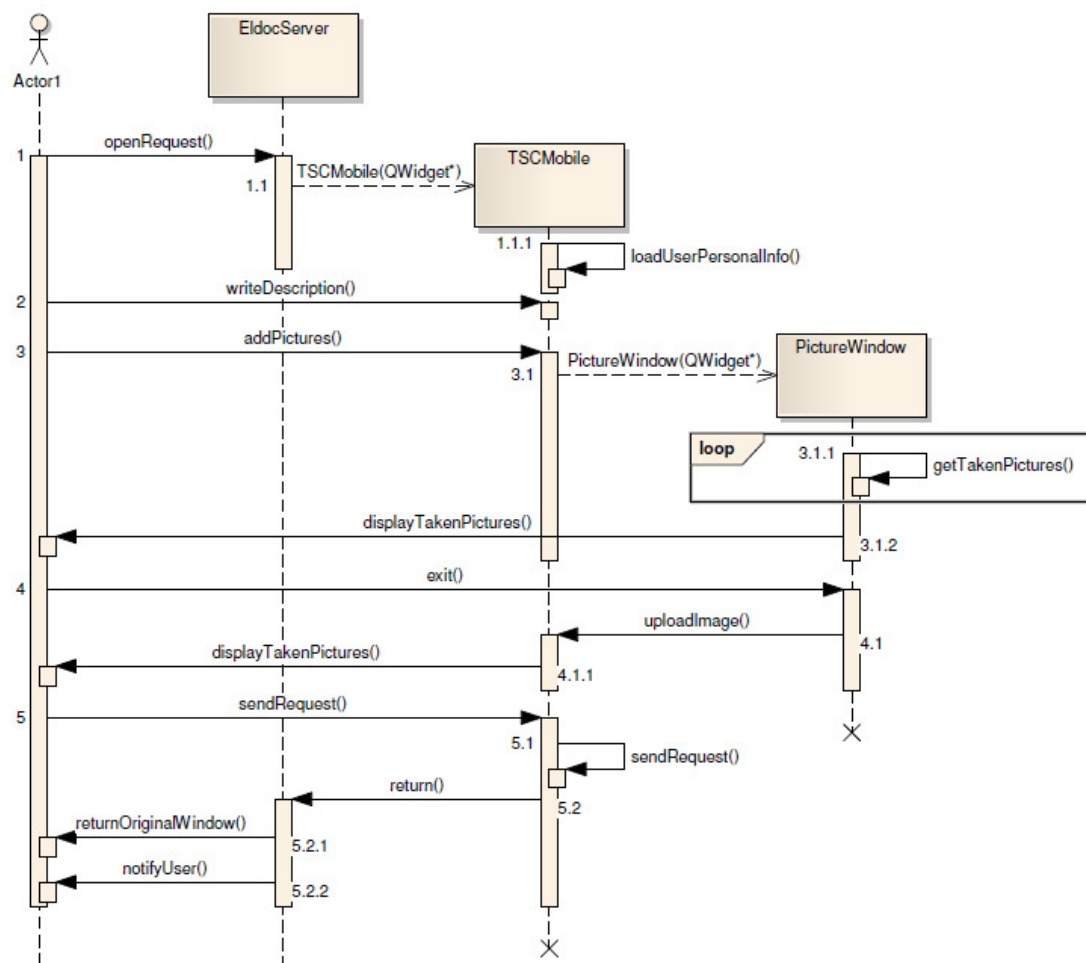


Figure 3.17. Sequence diagram – Create technical request.

The sequence can be described below.

- User clicks on Request menu item to create a new request. Eldoc Service Book will create a new instance of TSCMobile class as a stacked window.
- TSCMobile automatically adds user personal information saved on phone such as name, phone number. Product name will also be retrieved.
- User writes description of the problem.
- User can take and attach pictures to the request to illustrate the problem. This has been described in previous chapter.
- User clicks on “Send” to send the request and return to original window.
- Eldoc Service Book notifies user when the request has been sent or error occurs.

3.6 Component Diagram

There are four main packages in this application. All classes in package mediaplayer, image, and mbarcode are targeted for the main UI component in package main.

- Package main: dita parser and Eldoc server main UI component.
- Package mbarcode: barcode reader.
- Package mediaplayer: video player.
- Package image: responsible for taking new pictures and viewing pictures.

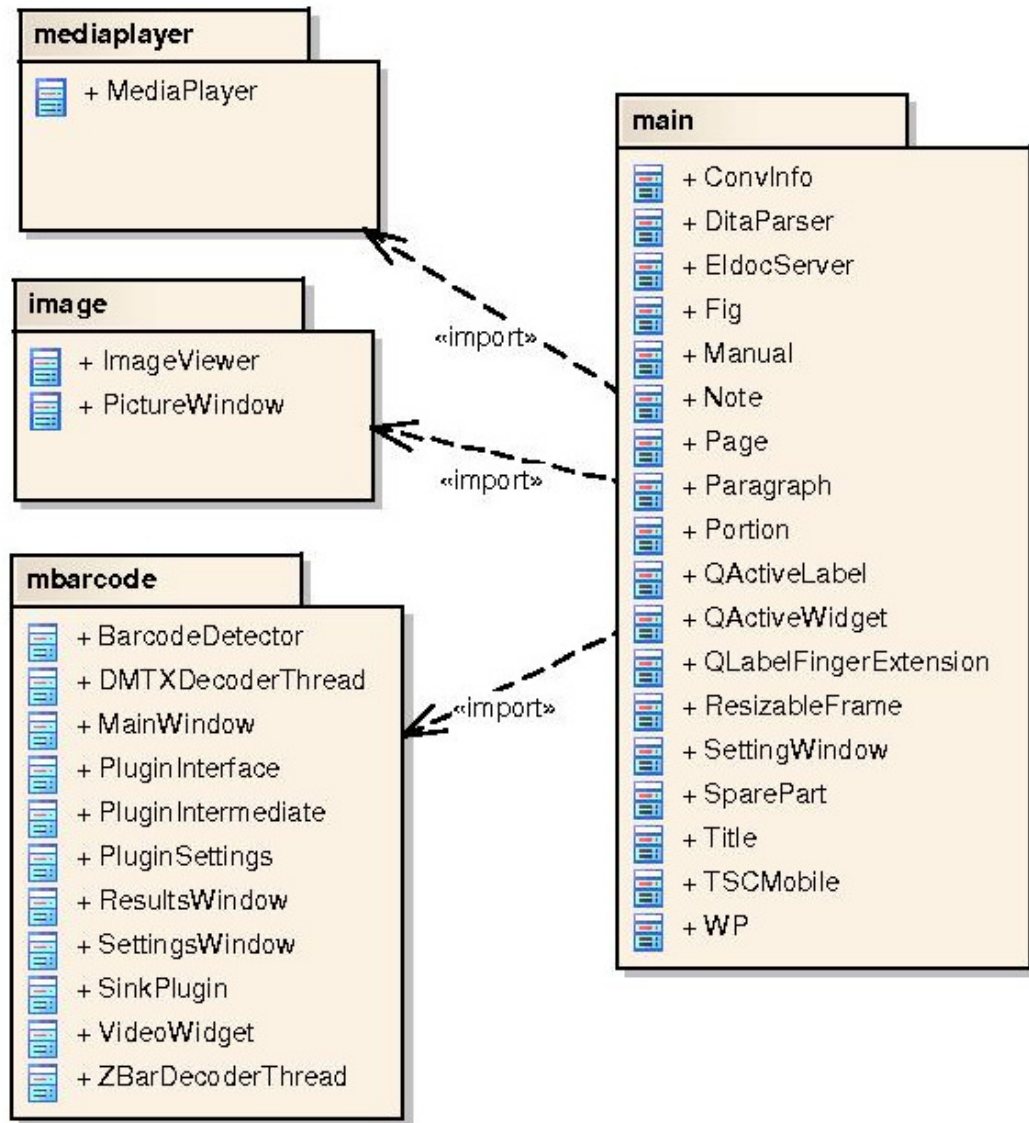


Figure 3.18. Component diagram.

The mbarcode package is the modified version of mbarcode open source project /20/. This project is mainly targeted for Maemo OS in Nokia N900.

3.7 Architectural Diagram

The architectural diagram of the application can be described below. Application data can be updated through many different ways.

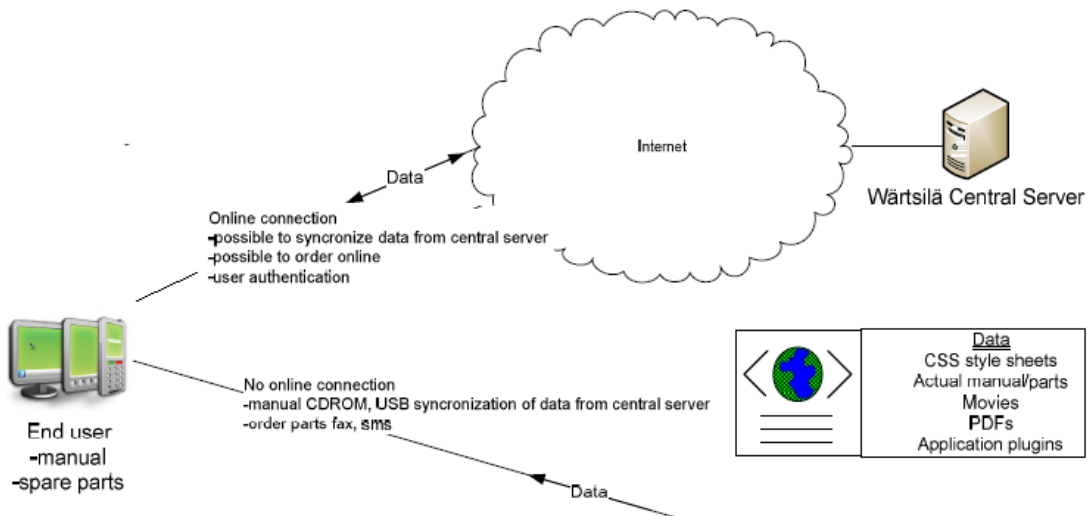


Figure 3.19. Architectural diagram.

In above picture the actual communication framework is described in high level. Basically the application needs Data. Data is located in a specified folder in the phone storage. Data consists of CSS style sheets, actual manual/spare parts (DITA format), movies, PDF or other attached files. Also the actual application updates can be seen as a data which can be updated to client. The data needs to be transferred to customer device somehow. If the device is connected to the Internet it is possible to do the updates automatically; in case no Internet connection the transfer needs to be done using other methods like USB memory stick, CD ROM. One way to connect to the server is to utilize the RAO communication which is another ongoing Wärtsilä project. Technically it is possible and it requires that the end user is strong authenticated. All the functionality needed to authenticate the user and for communicating with the site core exist in RAO project. When user is authenticated, it is possible to define which products the user has access, therefore all relevant manuals are downloaded.

4 GUI DESIGN

The core functionality of this application is Dita parser engine which reads documents in Dita format and saves information in objects so that it can be displayed later by Qt UI components.

In GUI design, there are two separate parts for Linux and Windows GUI design and Maemo GUI design. Technically the GUI for Linux and Windows are the same because they are implemented using common Qt UI components which are available in all platforms. The GUI for Maemo should be implemented separately as they use many Maemo 5 specific GUI components to take advantages of Maemo 5 look and feel.

The main GUI which displays documents is implemented the same for the three platforms. When deploying, depending on the OS it will have separate look and feel. The GUI for adding notes and photos, viewing pictures and videos will be implemented separately depending on the OS. Moreover, the version for Maemo will have additional functionalities and therefore additional GUI such as sending request, barcode reader, and client-server synchronization.

Below I will give description about some main parts of GUI design which are created by Qt Creator GUI designer.

4.1 Main Window

Before accessing the document, user has to complete some nested selector. In order to give more space for the content display, the selectors will be automatically hidden when user clicks on other parts rather than them and can be showed again when user clicks on Wäertsilä banner. Below is the GUI design made by Qt Creator.

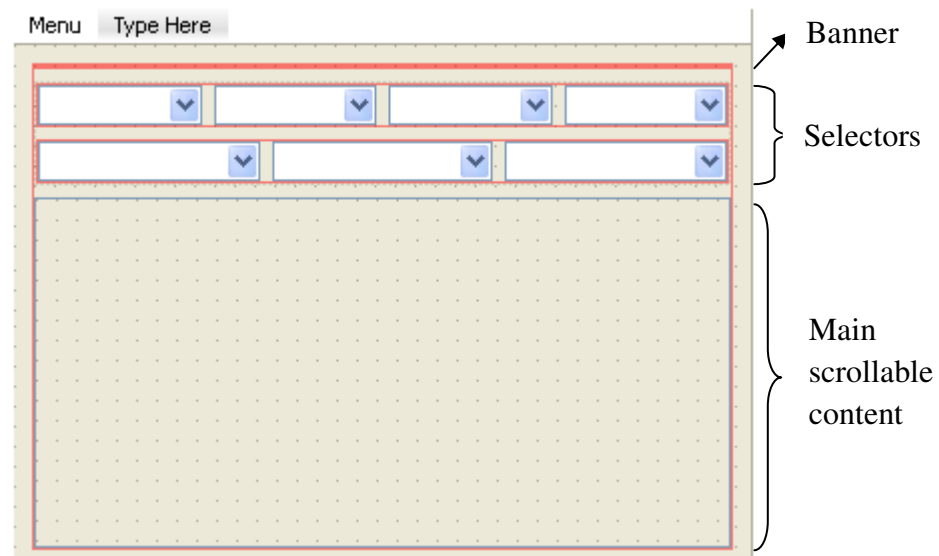


Figure 4.1. Main window design.

The banner will be filled with Wärtsilä icon when the application is loaded. In the above design, it is only a blank line. This banner is always showed in the screen so that user can click on it to show or hide the selectors.

All the selectors are placed inside a panel so that by showing or hiding the panel, the whole selectors will be showed or hidden. The widget displaying the main content has scrollable feature enabled in order to display document with long content.

4.2 Request Window

Request window is used to send request to resolution team for technical support. User can fill in detail of the problem he is having, take some pictures and attach to the request to illustrate the problem and send it to resolution team. This window is implemented as a stacked window.

A stacked window is a child window which will slide from the left side to the right side and overlap its parent window. When user clicks the Back button it will slide back and the parent window will be displayed again.

Below is the GUI design made by Qt Creator.

The image shows a Qt Designer window titled "Type Here". The window has a grid background. Inside, there are several input fields and a text area. The fields are labeled "Company", "Reporter", "Telephone", "Engine number", and "Description". The "Reporter" field contains the text "User", and the "Telephone" field contains "+358442725640". Below these fields is an "Attachments" label and a list box. At the bottom of the window are three buttons: "Attach Image", "Cancel", and "Send".

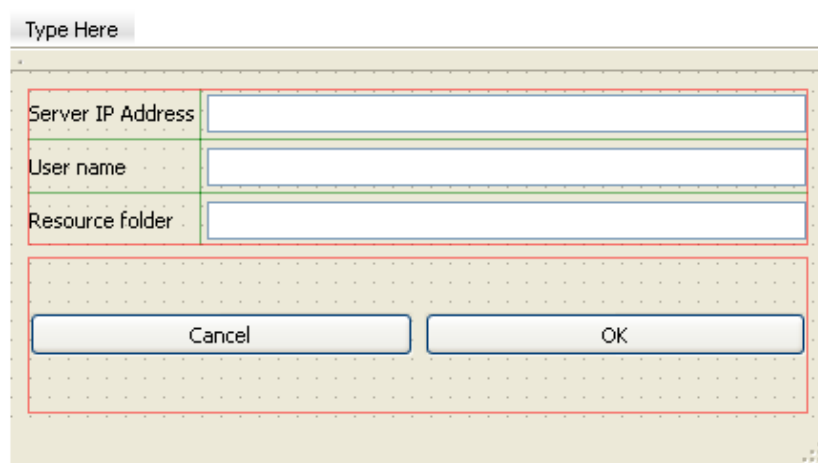
Figure 4.2. Request window design.

The “Attach Image” button is used to take and attach pictures to the request. Taken pictures will be displayed in “Attachments” area. User clicks on “Send” to send the request and return to original window. Eldoc Service Book will notify user when the request has been sent or error occurs.

4.3 Setting Window

Setting window is used to configure the connection between the phone and server in order to synchronize the data. The IP address of the server, user name of the authenticated user and resource folder are configurable attributes. This window is implemented as a stacked window.

Below is the GUI design made by Qt Creator.



Type Here

Server IP Address

User name

Resource folder

Cancel OK

Figure 4.3. Setting window design.

After user clicks on “OK” to exit this window and return to the main window, there will be a notification informing that the application setting has been changed.

5 IMPLEMENTATION

Because of the complexity of this application, hereby I will give a description or explanation of some important functions and classes. Details of each function can be found in the comments of the attached source code.

5.1 Dita Parser Function

In order to read the document in Dita XML format, Qt provides QDomDocument and its children classes in xml module. The conversion is made by class DitaParser. The class includes the following public methods:

- Manual* readManual(QString path);
- SparePart* readSparePart(QString path);

These methods are meant for reading manual and spare part document. Parameter path is the location of the converting Dita document. Because a Dita XML document is quite big, it is divided into multiple parts depending on the tag name. Each tag name will have its own private converting method. There might be nested tag, that one tag name contains others. Example of one tag name:

```
<li>
  <it>If the difference between exhaust gas temperatures of
  various cylinders is larger than 70°C at loads higher
  than 25 % the reason for this should be looked for.</it>
  <it>The charge air temperature should, in principle, be
  as low as possible at loads higher than 80 %, however,
  not so low that condensation occurs. See chapter 03,
  <ref xml:link="simple" inline="true"
  behavior="external" content-role="fig" href="fig-
  320352-low">Fig 03-1</ref> . At loads lower than 25 %
  it is favourable to have a charge air temperature as
  high as possible.
  </it>
</li>
```

Below is the method to read this unordered list.

```
QString DitaParser::readLI(const QDomNode &node){
  QString ul="<ul>";
```

```

//iterate through <li> tag list
for(int i=0;i<node.childNodes().size();i++){
    QDomNode it=node.childNodes().at(i);
    QString li = "<li>";
    //iterate through the child tags of every <li>
    //tag and convert the content to HTML
    for(int j=0;j<it.childNodes().size();j++){
        QDomNode itatt=it.childNodes().at(j);
        if(itatt.nodeName()=="#text"){
            // get the text content
            li +=itatt.nodeValue();
            //get the content of the link
        }else if(itatt.nodeName()=="ref"){
            //get the text content of the link
            QString displaytext=itatt.toElement().text();
            //get type of the link
            QString role=itatt.attributes().
                namedItem("content-role").nodeValue();
            //get href attribute of the link
            QString href=itatt.attributes().
                namedItem ("href").nodeValue();
            //convert to a normal HTML link
            QString link=" [<a href=\""+role+"\"+href+\" \"
                style=\"color:#B1FB17;\">"+displaytext+"</a>] ";
            li +=link;
        }
    }
    li += "</li>";
    ul+=li;
}
ul+="</ul>";
return ul;
}

```

This method will iterate through the tag list and read the content of each of them. When it reaches the <ref> tag, it will convert this tag to a normal link tag. Finally, the method will concatenate every part together to form the fully unordered list and return it.

The QDomNode node is the root tag of the paragraph (tag in above text). The *for* loop is used to iterate through all the <it> tag. Value of one <it> tag is like this:

```
QDomNode it=node.childNodes().at(i);
```

The second *for* loop is used to iterate through the content of each <it> tag. An <it> tag can have text and link. Value of one attribute of the <it> tag:

```
QDomNode itatt=it.childNodes().at(j);
```

If the content of the attribute is only text, read the text content:

```
if(itatt.nodeName()=="#text"){
    // get the text content
    li +=itatt.nodeValue();
}
```

If the attribute is a complex element containing other tags (in this case a ref tag), get the attributes “href” and “content-role” of this tag in order to form a link:

```
else if(itatt.nodeName()=="ref"){
    //get the text content of the link
    QString displaytext=itatt.toElement().text();
    //get type of the link
    QString role=itatt.attributes()
        .namedItem("content-role").nodeValue();
    //get href attribute of the link
    QString href=itatt.attributes()
        .namedItem("href").nodeValue();
    //convert to a normal HTML link
    QString link="[<a href=\""+role+"\";"+href+"\"
        style=\"color: #B1FB17;\" >"+displaytext+"</a>]";
    li +=link;
}
```

Finally, close the tag and concatenate it to the list.

```
li += "</li>";
ul+=li;
```


The return value of this function will be a converted HTML unordered list.

5.2 Add New Note

User clicks on “Add Comment” button, a new comment dialog new be displayed. User enters the note/comment and click “OK” to submit note or “Cancel” to discard the note. New added note will be display at the end of the document, right after the “Add Comment” button. Method code and explanation:

```
void EldocServer::addNote(){
    //if note is not already created, create one
    if(m_noteDialog == NULL){
        //display as a dialog
        m_noteDialog=new QDialog(this);
        //set auto orientation for this widget
        m_noteDialog->setAttribute(Qt::WA_Maemo5AutoOrientation,
            true);

        //start GUI design of a text box and 2 buttons OK and
        //Cancel
        QVBoxLayout *vlayout = new QVBoxLayout(m_noteDialog);
        m_noteDialog->setObjectName(QString::fromUtf8("New
        Note"));
        m_note = new QTextEdit();
        m_okButton =new QPushButton("OK");
        m_cancelButton = new QPushButton("Cancel");
        //horizontal layout
        QHBoxLayout *hlayout= new QHBoxLayout();
        hlayout->addWidget(m_cancelButton);
        hlayout->addWidget(m_okButton);
        vlayout->addWidget(m_note);
        vlayout->addLayout(hlayout);
        //end GUI design
        //Below is connect the two buttons with other action
        //submit or cancel note
        connect(m_okButton,SIGNAL(clicked()),this,
            SLOT(submitNote()));
        connect(m_cancelButton,SIGNAL(clicked()),this,
            SLOT(cancelNote()));
```

```

    }else{
        //clear the old content if note already exists
        m_note->setPlainText("");
    }
    // display the note dialog
    m_noteDialog->show();
}

```

In order to avoid unneeded UI creation, the note dialog, which is a global variable, is initialized once user wants to add note. First check if the note dialog has not already been created; if not, initialize it:

```

if(m_noteDialog == NULL){
    //display as a dialog
    m_noteDialog=new QDialog(this);
}

```

Set auto orientation feature for the note dialog:

```

m_noteDialog->setAttribute(Qt::WA_Maemo5AutoOrientation,
true);

```

Then, initialize the UI component inside the dialog, including a textbox, two buttons for OK and Cancel action.

```

m_note = new QTextEdit();
m_okButton =new QPushButton("OK");
m_cancelButton = new QPushButton("Cancel");

```

Connect the two buttons with the submitNote and cancelNote action:

```

//Below is connect the two buttons with other action
//submit or cancel note
connect(m_okButton,SIGNAL(clicked()),this,
        SLOT(submitNote()));
connect(m_cancelButton,SIGNAL(clicked()),this,
        SLOT(cancelNote()));

```

If the note dialog is already created and used before, then clear the old content:

```
m_note->setPlainText("");
```

Finally, display the note dialog:

```
m_noteDialog->show();
```

5.3 Take and Attach Pictures

User clicks on “Attach picture” button, Eldoc server will open a new stacked window (PictureWindow). This picture window UI class will create a thread monitoring taken pictures. When user takes a picture, it will be recorded and updated in the picture window. Finally user can click “Back” button to end the thread and attach taken pictures to the document. Like notes, taken pictures will be display at the end of the document.

Briefly description about implementation of this thread:

First of all, start a QTimer and set the time interval 1 second. Then, set the action will be executed when the QTimer is timeout (every 1 second).

```
//start a QTimer
QTimer timer = new QTimer(this);
//search for taken pictures after every 1 second
connect(timer, SIGNAL(timeout()), this,
        SLOT(getTakenPictures()));
//repeat after 1 second
timer->start(1000);
```

By doing this, the action `getTakenPictures` will be repeated every 1 second. Below is the detail of this function.

```
void PictureWindow::getTakenPictures(){
    //PATH: folder where pictures are saved
    QDir myDir(this->PATH);
    //filter for only image files
    QStringList filters;
    filters.append(tr("*.jpg"));
    filters.append(tr("*.jpeg"));
    QStringList currentlist=myDir.entryList
```

```

(filters,QDir::Files | QDir::NoSymLinks);
//iterate through current files to check if there
//is any new file
//filelist: old file list
foreach(QString file,currentlist){
    // if file is not included in the old file list
    if(!isIncluded(file,filelist)){
        QTableWidgetItem *fileNameItem =
        new QTableWidgetItem(file);
        fileNameItem->setIcon
        (QIcon(myDir.absoluteFilePath(file)));
        int row = filesTable->rowCount();
        filesTable->insertRow(row);
        //add the file to file table
        filesTable->setItem(row, 0, fileNameItem);
    }
}
//give filelist the value of current file list for the
//next check
filelist=currentlist;
}

```

This function will check if there is any new file compared with the file list it has previously. If there is a new file, it will be added to the taken picture table and displayed in the user interface.

Set the filter to monitor only image files in the image folder (PATH).

```

QStringList filters;
filters.append(tr("*.jpg"));
filters.append(tr("*.jpeg"));

```

Get all the image files in the image folder and save them in a list.

```

QStringList currentlist=myDir.entryList
(filters,QDir::Files | QDir::NoSymLinks);

```

Then iterate through every element in this list and check if each of them exists in the old file list. If not, add it to the table of taken pictures.

```

if(!isIncluded(file,filelist)){
    QTableWidgetItem *fileNameItem =
    new QTableWidgetItem(file);
    fileNameItem->setIcon
    (QIcon(myDir.absoluteFilePath(file)));
    int row = filesTable->rowCount();
    filesTable->insertRow(row);
    //add the file to file table
    filesTable->setItem(row, 0, fileNameItem);
}

```

Finally, give variable filelist the value of current file list for the next check.

```
filelist=currentlist;
```

5.4 Image Viewer

The image viewer window is implemented quite easily using a QLabel and set the image as a pixmap to that label. QLabel UI component can be used to display both text and picture. The code together with explanation:

```

//set as a stacked window
setAttribute(Qt::WA_Maemo5StackedWindow);
//set auto orientation
setAttribute(Qt::WA_Maemo5AutoOrientation, true);
setObjectName(QString::fromUtf8("Image Viewer"));
imageLabel = new QLabel;
//define custom attributes
imageLabel->setBackgroundRole(QPalette::Base);
imageLabel->setSizePolicy(QSizePolicy::Ignored,
QSizePolicy::Ignored);
imageLabel->setScaledContents(true);
//the main display window
scrollArea = new QScrollArea;
scrollArea->setBackgroundRole(QPalette::Dark);
//set the image label as the only component
scrollArea->setWidget(imageLabel);
setCentralWidget(scrollArea);

```

```
scrollArea->setWidgetResizable(true);
```

First of all, set basic attributes for this image viewer such as auto oriented, stacked window, and object name:

```
//set as a stacked window
setAttribute(Qt::WA_Maemo5StackedWindow);
//set auto orientation
setAttribute(Qt::WA_Maemo5AutoOrientation, true);
setObjectName(QString::fromUtf8("Image Viewer"));
```

Then, initialize the image label which is the main container to display the image.

```
imageLabel = new QLabel;
//define custom attributes
imageLabel->setBackgroundRole(QPalette::Base);
imageLabel->setSizePolicy(QSizePolicy::Ignored,
QSizePolicy::Ignored);
imageLabel->setScaledContents(true);
```

After initializing the image window, the application can call the function setImage multiple times to view different images:

```
void ImageViewer::setImage(const QString &path){
    //set pixmap for the image label which will display the
    //actual image.
    imageLabel->setPixmap(QPixmap(path));
}
```

This function will set the image as the pixmap content of the label in order to display it.

5.5 Media Player

The media player is created using Qt phonon library with gstreamer plugin to play 3gp media files.

Briefly description of the code for creating media player:

```
//set object name
this->setObjectName(QString::fromUtf8("Video"));
//set stacked window feature
setAttribute(Qt::WA_Maemo5StackedWindow);
//create new media object
m_MediaObject= new Phonon::MediaObject(this);

//create new video widget
m_videoWidget= new Phonon::VideoWidget(this);
// link the video to media
Phonon::createPath(m_MediaObject, m_videoWidget);
//create audio output
m_AudioOutput = new
Phonon::AudioOutput(Phonon::VideoCategory, this);
//link the audio to media
Phonon::createPath(m_MediaObject, m_AudioOutput);

//a slider to navigate through the video
slider = new Phonon::SeekSlider(this);
slider->setMediaObject(m_MediaObject);
// start GUI creation
QWidget *centralWidget=new QWidget(this);
this->setCentralWidget(centralWidget);
QVBoxLayout* layout=new QVBoxLayout(centralWidget);
layout->addWidget(m_videoWidget);
layout->addWidget(slider);
// end GUI creation
```

Phonon media player is a media object; it consists of a video widget which can be displayed in the user interface and an audio widget which outputs the sound. In order to create this media player, first create the media object, video widget and audio widget then link them together by the following codes:

```
// link the video to media
Phonon::createPath(m_MediaObject, m_videoWidget);
//link the audio to media
Phonon::createPath(m_MediaObject, m_AudioOutput);
```

The slider is used to navigate through the playing video. There are many other UI component with can be used to facilitate the video display such as next, back, pause button, etc.

```
//a slider to navigate through the video
slider = new Phonon::SeekSlider(this);
```

After creating the slider, it should be linked to the media object:

```
slider->setMediaObject(m_MediaObject);
```

Finally, place the video widget together with the slider in the main window:

```
// start GUI creation
QWidget *centralWidget=new QWidget(this);
this->setCentralWidget(centralWidget);
QVBoxLayout* layout=new QVBoxLayout(centralWidget);
layout->addWidget(m_videoWidget);
layout->addWidget(slider);
// end GUI creation
```

This will create a media player to play any media type.

5.6 View pdf File

The viewing pdf feature of the application is implemented by Qt Poppler library. This extended library is targeted for dealing with pdf document and can run in Linux and Windows platforms. Briefly description of the code:


```

//create new pdf renderer object
Poppler::Document *pdf = Poppler::Document::load(ELDOC_ROOT+
"attachment1.PDF");
//create a QLabel to contain the pdf file which is rendered
//as images.
QLabelFingerExtension pdfContainer =new
QLabelFingerExtension();
//set the content of pdfContainer to the first page of pdf
//file
pdfContainer->setPixmap(QPixmap::fromImage
(pdf->page(0)->renderToImage()));
//current page
currentPage=0;
//connect action to go to the next page
connect(pdfContainer,SIGNAL(fingerScrolled()),this,
SLOT(goNextPage()));
//add the pdf container to the main layout
layout->addWidget(pdfContainer);

```

First of all, create a new Poppler pdf renderer object:

```

//create new pdf renderer object
Poppler::Document *pdf = Poppler::Document::load(ELDOC_ROOT+
"attachment1.PDF");

```

Poppler will read the content of pdf file and convert it to a collection of images. The later part is just displaying the image using a QLabel. The below code show how to convert a page to image using poppler feature:

```

pdf->page(i)->renderToImage()

```

As normal QLabel does not recognize finger swiping action, a new class QLabelFingerExtension which inherits from QLabel and implements a custom signal to recognize finger interactions is created. Detail of the class will be explained later.

```

//create a QLabel to contain the pdf file which is rendered
//as images.
QLabelFingerExtension pdfContainer =new
QLabelFingerExtension();

```

Finally, connect the signal finger swiping to the action goNextPage.

```
connect(pdfContainer, SIGNAL(fingerScrolled()), this, SLOT(goNextPage()));
```

This will create a custom pdf viewer which supports finder swiping action.

5.7 QLabelFingerExtension Class

This class inherits from QLabel and implements a custom signal to recognize finger swiping action. Finger swiping is the action that user swipes his finger from the left side to the right side of the screen, normally to trigger an action to move to the next page. The class includes the following private member:

- x: x coordinate of the finger
- y: y coordinate of the finger
- fingermove (bool): indicate whether the finger is moving

It has three protected methods to track the movement of finger:

- mouseMoveEvent: finger is moving
- mousePressEvent: finger is touching the screen
- mouseReleaseEvent: finger is out

The action is recognized as finger swiping only when the finger has moved a distance along the X axis. When user touches the screen, mousePressEvent is triggered and does the following event:

```
//give x the value of current x coordinate of the finger
x=finger->x();
//give y the value of current y coordinate of the finger
y= finger ->y();
//indicate that the finger is moving
fingermove=true;
```

When user releases the finger from the touch screen, mouseReleaseEvent is triggered and does the following event:

```

if (abs(x-ev->x())>200&&abs(y-ev->y())<70&&fingermove){
    fingermove=false;
    emit fingerScrolled();
}

```

This code checks if the finger has moved far enough along the X axis and not exceeded the Y coordinate limitation. If the condition is satisfied, the signal `fingerScrolled` will be emitted:

```

emit fingerScrolled();

```

This case shows how a custom signal is defined in Qt.

5.8 Update Database

The synchronization between client and server is done using Linux command “rsync”. In order to use this command, rsync package should be installed both on Linux server and on the phone.

When user click on “Update” button, the application will create a separate process to execute this command, the new process is working independently with the application process. This works exactly the same way with daemon in Linux machine. Qt has the possibility to create new process using `QProcess`:

```

QProcess rsync;
rsync.startDetached("rsync",QStringList() << "r" <<
m_remoteUsername+ "@" +m_remoteAddress+": "+m_remoteFolder <<
W20_DOC_ROOT);

```

`m_remoteUsername`: authenticated user name.

`m_remoteAddress`: IP address of the server.

`m_remoteFolder`: location of the data folder in server machine.

`W20_DOC_ROOT`: location of data in local machine.

This command will start the independent process rsync to synchronize the files between host machine and local machine. The option “-r” to sync the whole folder.

After the process has completed and the application has been updated, a notification will be sent to user using Qt maemo5 information box:

```
QMaemo5InformationBox::information(this, "Update is
installed successfully",
QMaemo5InformationBox::DefaultTimeout);
```

If there is error in executing the command, user will be notified the same way:

```
QMaemo5InformationBox::information(this, "Error updating
database", QMaemo5InformationBox::DefaultTimeout);
```

5.9 Search Engine

The search engine is implemented using existing open source project mbarcode. The mbarcode application is modified and integrated into this application. It is changed so that after finding the code of the searching document, the barcode engine will send the information to the main window and trigger the action “openDocument” to display related document. Then the barcode engine will be destroyed and the original window with new information is restored.

```
//close itself
this->close();
//check if the result window has parent
if(this->parent()){
    //if it has parent then close it also
    MainWindow* m_barcode=(MainWindow*)this->parent();
    m_barcode->close();
    //get the main window EldocServer
    if(m_barcode->parent()){
        EldocServer* m_eldoc=((EldocServer*)
        m_barcode->parent());
        //open related document
        m_eldoc->openDocument(this->barcode_text);
    }
}
```

First of all, close the current window.

```
//close itself  
this->close();
```

Check if this result window has a parent window, and then close the parent.

```
if(this->parent()){  
    //if it has parent then close it also  
    MainWindow* m_barcode=(MainWindow*)this->parent();  
    m_barcode->close();
```

Then, get the instance of EldocServer class which is the parent of m_barcode.

```
EldocServer* m_eldoc=((EldocServer*)m_barcode->parent());
```

Finally, open the searched document.

```
m_eldoc->openDocument(this->barcode_text);
```

In this code, the EldocServer class is the parent of the MainWindow class which is the parent of this result window. In order to get the EldocServer class, it needs to call function this->parent() twice.

6 TESTING

According to the project plan described at the beginning of this document, all objectives of the final thesis have been achieved in the Qt based application associated with this project. The application has been deployed to the three platforms Maemo, Linux and Windows and showed to Wärtsilä people. All the required and extra features have been successfully implemented. The project is on its way to integrate with other ongoing Wärtsilä Qt projects.

When testing, there is a feedback from customer that there is few seconds delay before the actual document is showed. That is because the document is quite big (can be up to 2000 lines) it takes time for the application to convert it to HTML and display in the screen.

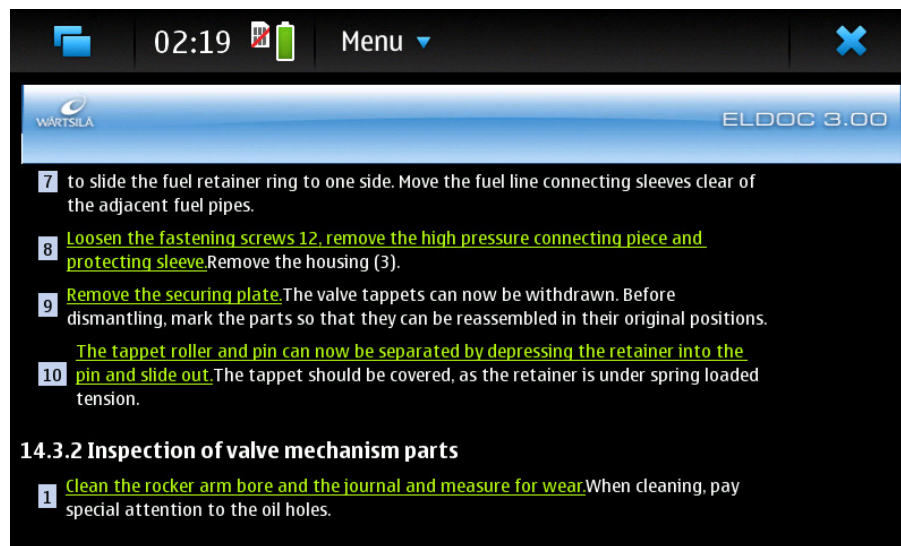


Figure 6.1. Eldoc service book look and feel.

Every implemented method was tested during coding and directly after it. Any found bug was immediately corrected. The applied testing approaches to methods are as follows:

- Updating when no internet connection.
- Adding comments with invalid characters.

- Sending request with invalid information.
- Scanning invalid barcode.
- Giving wrong server information such as wrong server IP address, wrong user name, etc.
- Taking too many pictures at the same time.

In addition, any place in the source code where there might occur an exception has been covered with try, catch clause. The GUI has been tested by simple performing operation. All feedbacks of the supervisor and customers have been taken into account and will be fixed when the application is moved to production version. The application is ready to be integrated with other ongoing Wärtsilä projects.

Below are the testing results and screenshots of some main features of the application.

6.1 Viewing Technical Documents

Before accessing the document, user has to complete some nested selector. In order to give more space for the content display, the selectors will be automatically hidden when user clicks on other parts rather than them and can be showed again when user clicks on the Wärtsilä banner. Below is the UI in Maemo, Windows and Linux platforms.

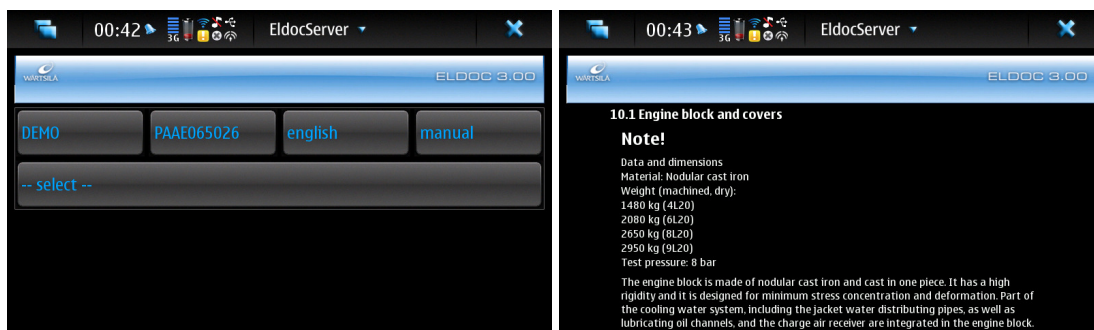


Figure 6.2. Eldoc Service Book main GUI in Maemo platform.

The main UIs on Windows and Linux have the same functionality compared with one on Maemo but the appearance is operating system native look and feel. Below is Eldoc Service Book in Windows platform.

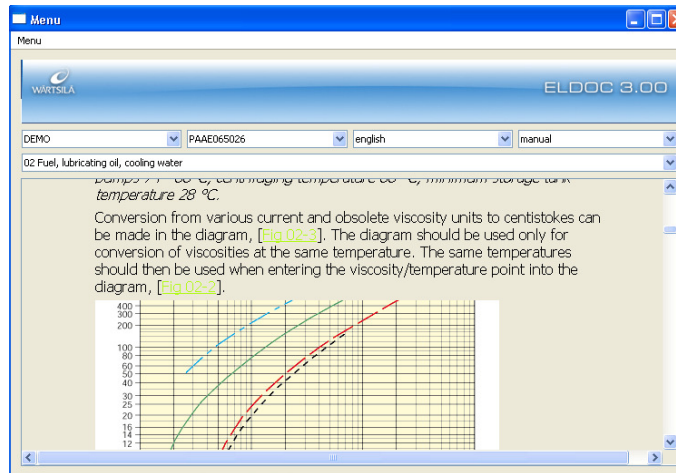


Figure 6.3. Eldoc Service Book main GUI in Windows platform.

Below is Eldoc Service Book Linux version with default Ubuntu Linux look and feel.

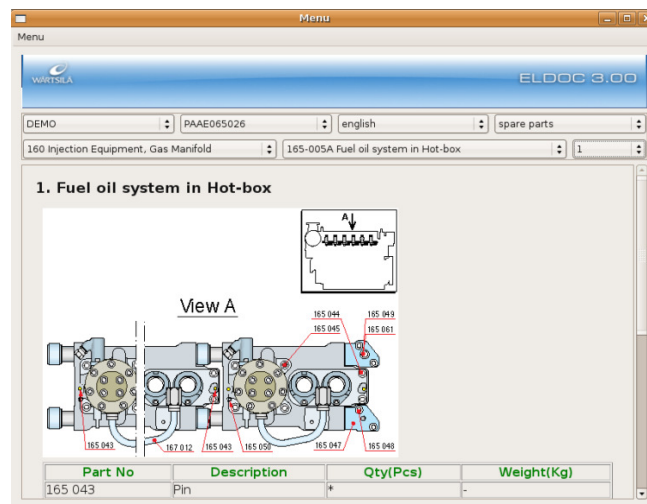


Figure 6.4. Eldoc Service Book main GUI in Linux platform.

The application can have the possibility to have the same style for every platform. In order to achieve that, the application look and feel should be changed using Qt style sheet for each widget.

6.2 Adding Notes or Comments in Windows and Linux

The same with .Net version, at the end of each specific document is the place where user can add notes/comments for it. The added note will be displayed directly below the text field

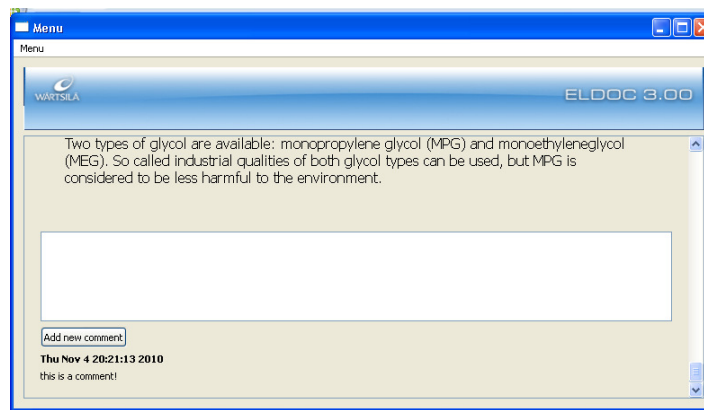


Figure 6.5. Adding notes/comments in Eldoc tablet version.

6.3 Adding Notes and Pictures in Eldoc Maemo Version.

In phone version, at the end of each specific document are two buttons for attaching comments or pictures. The two buttons are made big so that user can easily click on it using his fingers.

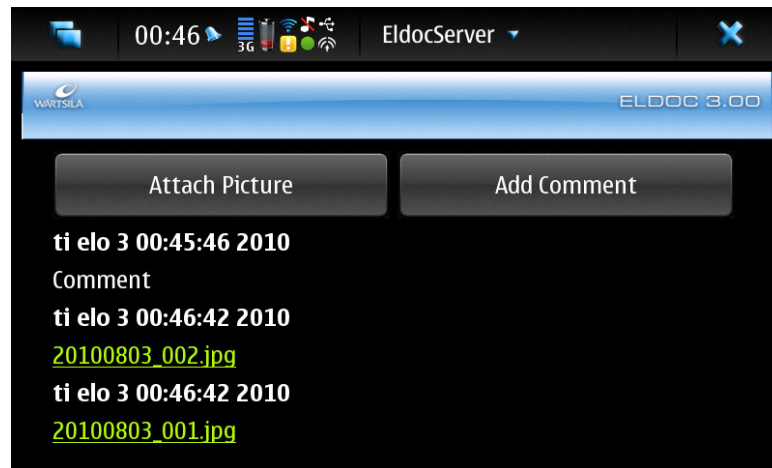


Figure 6.6. Adding notes/comments and pictures in Eldoc phone version.

The “Add Comment” action will be displayed as a popup dialog:

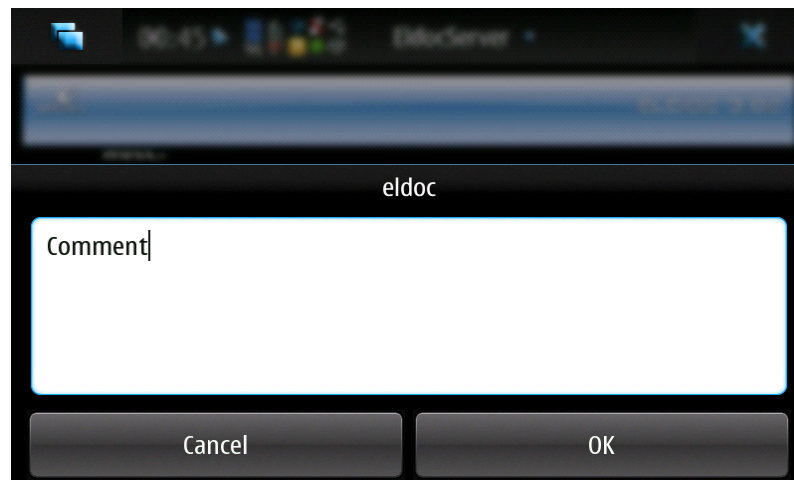


Figure 6.7. Adding notes/comments in Eldoc phone version.

The “Attach Picture” action will be displayed as a stacked window. A stacked window is a child window which will slide from the left side to the right side and overlap the parent window. When user clicks the Back button it will slide back and the parent window will be displayed again.

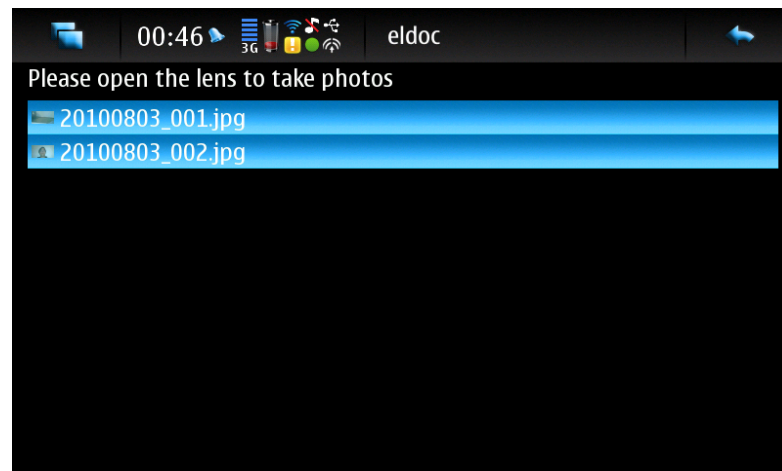


Figure 6.8. Attaching pictures in Eldoc phone version.

6.4 Barcode Reader

The barcode reader has the video display of what user is scanning. User can also have the option to input existing picture which has been taken before. At the bottom is two buttons to start scanning and open existing file.



Figure 6.9. Barcode reader.

When user click on “Scan Barcode”, the application will start scanning and there will be one more option for focusing the image.

6.5 Setting Window

Setting window is used to configure the connection between the phone and server in order to synchronize the data. The IP address of the server, user name of the authenticated user and resource folder are configurable attributes.



Figure 6.10. Setting window.

After user exit this window and return to the main window, there will be a notification informing that the application setting has been changed.

6.6 Request Window

The request window is also implemented as a stacked window. The name of the reporter and his telephone number is automatically retrieved based on his information saved on the phone.

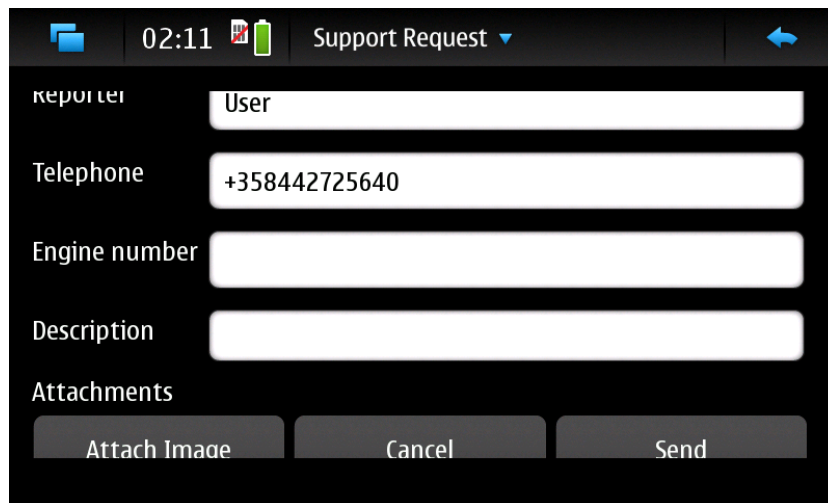
The image shows a mobile application interface for a 'Support Request' window. At the top, there is a status bar with a folder icon, the time '02:11', a battery icon, and the title 'Support Request' with a dropdown arrow. Below the status bar, the form contains several input fields: 'Reporter' with the value 'User', 'Telephone' with the value '+358442725640', 'Engine number' (empty), and 'Description' (empty). Below these fields is an 'Attachments' section. At the bottom of the form are three buttons: 'Attach Image', 'Cancel', and 'Send'.

Figure 6.11. Request window.

User can also click on “Attach Image” to take and attach pictures in the request to visualize the problem. This action will open the Picture Window which has been described in previous chapter.

User clicks on “Send” to send the request and return to original window. Eldoc Service Book will notify user when the request has been sent or error occurs.

7 CONCLUSION

This project has given an excellent example of the usability of Qt framework on multiple mobile platforms. Nowadays, being a key developing environment for many big projects for instance Linux KDE desktop, Google Earth and Skype, Qt is gradually accepted as a development framework among developers all over the world.

Together with Android, Maemo is becoming the future platform for mobile device. Maemo is mostly based on open source code, and has been developed by Nokia in collaboration with many open source projects such as the Linux kernel, Debian and GNOME. Because of its nature, all application for Maemo will be open source and can be reuse by other developers. This enables knowledge sharing between developers which is very important in software development that developer does not need to develop things others have done previously.

Mobile phones are playing an increasingly essential role in human life. Thank to the portability, usability and capability of modern mobile phones, in the near future, it will be the most important electric equipment which cannot be replace in our social life. It can be a TV, a multimedia player, a video player, a navigator, a game box or an internet tablet, etc.

All main and optional features of the application have been successfully implemented. The main function of the application is to view technical documents, images, and videos. Moreover, additional features of a phone have been utilized in taking pictures, sending technical request and updating the server. However, certain aspects of this thesis could have been better for example the time delays when displaying documents as stated in customer's feedback. These changes can be made when the application is moved to production version.

7.1 Future Development

Currently the application can be updated only via local wireless connection because of security reason. However, it can be integrated with another Wärtsilä project named RAO which provides secure communication; then it can be updated remotely via GPRS for instance.

8 REFERENCES

- /1/ Wikipedia - Darwin Information Typing Architecture (2010). [WWW]. [referred 1.10.2010] Available on the Internet: <URL:http://en.wikipedia.org/wiki/Darwin_Information_Typing_Architecture>
- /2/ Don Day, Mochael Priestlet, David Schell. IBM corporation (2005). Introduction to the Darwin Information Typing Architecture, [referred 1.10.2010] Available on the Internet: <URL: <http://www.ibm.com/developerworks/xml/library/x-dita1/>>
- /3/ XMLmind XML Editor (2010). [WWW]. [referred 15.10.2010] Available on the Internet: <URL: <http://www.xmlmind.com/xmleditor/>>
- /4/ Qt Nokia official website (2010). [WWW]. [referred 15.10.2010] Available on the Internet: <URL: <http://qt.nokia.com/>>
- /5/ Nokia corporation (2009). Qt 4.6 white paper, [referred 15.10.2010] Available on the Internet: <URL: <http://qt.nokia.com/products/files/pdf/qt-4.6-whitepaper>>
- /6/ Wikipedia - Maemo (2010). [WWW]. [referred 1.10.2010] Available on the Internet: <URL: <http://en.wikipedia.org/wiki/Maemo>>
- /7/ Maemo official repository (2006). [WWW]. [referred 1.07.2010] Available on the Internet: <URL: <http://repository.maemo.org/dev>>
- /8/ Maemo development repository (2006). [WWW]. [referred 1.07.2010] Available on the Internet: <URL: http://repository.maemo.org/extras_dev>
- /9/ Nokia OVI store (2006). [WWW]. [referred 1.07.2010] Available on the Internet: <URL: <https://store.ovi.com/>>
- /10/ Maemo official website (2007). [WWW]. [referred 1.07.2010] Available on the Internet: <URL: <http://www.maemo.org/>>
- /11/ Scratchbox (2009). [WWW]. [referred 1.07.2010] Available on the Internet: <URL: <http://www.scratchbox.org>>

- /12/ Maemo 5 SDK (2009). [WWW]. [referred 1.07.2010] Available on the Internet: <URL: http://repository.maemo.org/stable/fremantle/maemo-sdk-install-wizard_5.0.py>
- /13/ Maemo SDK virtual image (2009). [WWW]. [referred 1.06.2010] Available on the Internet: <URL:http://maemovmware.garage.maemo.org/2nd_edition/>
- /14/ VMWare player (2007). [WWW]. [referred 1.06.2010] Available on the Internet: <URL: <http://www.vmware.com/>>
- /15/ Java website (2000). [WWW]. [referred 1.06.2010] Available on the Internet: <URL: <http://www.java.com>>
- /16/ Esbox IDE (2009). [WWW]. [referred 1.06.2010] Available on the Internet: <URL: <http://esbox.garage.maemo.org>>
- /17/ MADDE technology overview (2009). [WWW]. [referred 1.06.2010] Available on the Internet: <URL: <http://wiki.maemo.org/MADDE>>
- /18/ Qt Creator nightly-build version (2009). [WWW]. [referred 1.06.2010] Available on the Internet: <URL: <http://qt.nokia.com/developer/qt-snapshots>>
- /19/ PC connectivity project (2009). [WWW]. [referred 1.06.2010] Available on the Internet: <URL: https://garage.maemo.org/frs/download.php/7104/PC-Connectivity_0.9.4.exe>
- /20/ Mbarcode open source project. [WWW]. [referred 1.06.2010] Available on the Internet: <<http://maemo.org/packages/view/mbarcode/>>

1 QT DEVELOPMENT SETUP FOR MAEMO PLATFORM

This chapter will explain the Qt development setup and configuration on Maemo platform. Because Maemo is a newly published environment, the development process is much more complex compared with other environment and Nokia will have to do a lot in order to make it easier for starter. This chapter will introduce completely all the needed setups and configurations to create a Qt Maemo application. This part will also include a sample Qt application running on Maemo platform.

1.1 Maemo SDK virtual image

Maemo community has been conducting a project called Maemo SDK Virtual Image to help the developing less painful. This project provides a fully programming environment for Maemo platform in one virtual image. All user has to do is to download and install a virtual machine player for instance VMWare together with this virtual image. This project is an important contribution since it offers the developer a complete Maemo programming environment with no need to spend time to download and configure correctly all tools which is a burden for starter. Below is installation procedure:

Download Maemo SDK virtual image from its website /13/ and extract it to one specified folder.

Download VMWare player from its website /14/, install it in your machine and start the program.

In VMWare Workstation window, click File -> Open then choose the location of the image file you have just extracted (maemosdk_desktop_intrepid-10-08.vmx).

In the main page, click on “Power on this virtual machine” to start Ubuntu machine with complete Maemo programming environment.



Figure 1. Maemo SDK virtual image.

This Linux operating system contains Esbox IDE and a complete Maemo SDK. Moreover, it also has some Qt Maemo sample applications which can be deployed and run directly to a Maemo OS.

1.2 Environment setup

Another alternative for advanced user is to have his own Linux machine and install maemo SDK on it. This is a preferable way as with Maemo virtual image there will be connectivity problem with your device through VMWare later. Another problem with VMWare is that you have to run double OSs at the same time; it takes a lot of computer resources. As a result, it will be much slower for developer to code and debug in Ubuntu OS running inside Windows OS. For the purpose of this thesis, I have installed Ubuntu 8.04 in my computer. In order to achieve a complete Maemo programming environment, user should install Scratchbox as a Maemo simulator, Maemo 5 SDK and esbox as an editor. The procedure will be described in the next chapter.

1.2.1 Maemo 5 SDK and Scratchbox installation

Maemo 5 SDK installation is now made easy with GUI installer. This installer will install Scratchbox and Maemo 5 SDK on Debian based systems. This installer will allow installation of Nokia closed packages and applications provided that EULA is accepted. The complete feature of this GUI installer:

- Install/Upgrade Scratchbox
- Installation of Maemo 5 SDK
- Installation of nokia-binaries
- Installation of nokia-apps
- Installation of Xephyr, if missing
- Can create a launcher for Xephyr on Desktop
- Can create a shortcut to Scratchbox home folder on Desktop
- Can create Maemo 5 info page on Desktop that contains useful links for developers

Download maemo5 SDK from its website /12/ and give the execution right to the file by the following command:

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

Start the GUI installer by the following command:

```
sudo ./maemo-sdk-install-wizard_5.0.py
```

And follow the instruction until you get the installation finish.

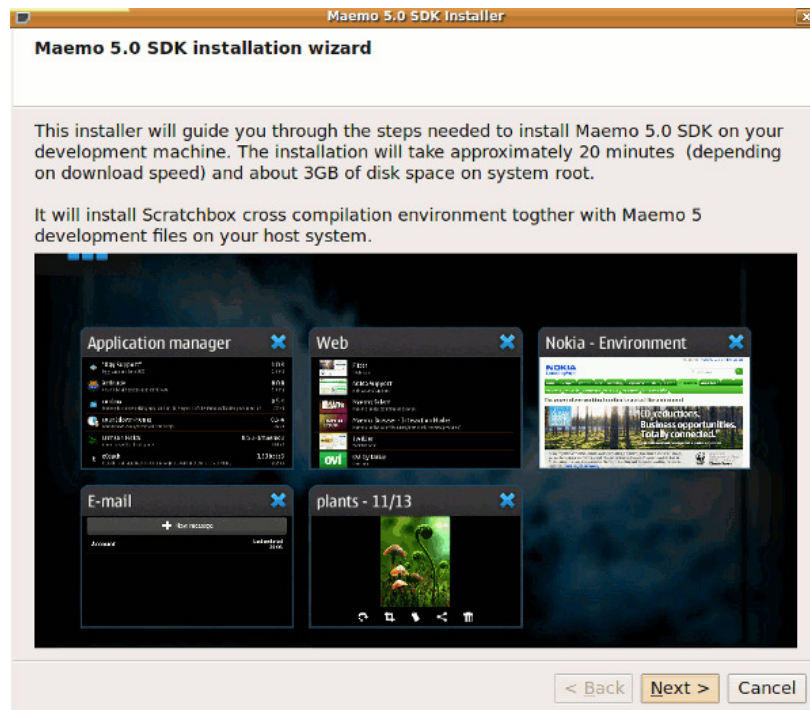


Figure 2. Maemo 5 SDK installation wizard.

Scratchbox does not support VSDO (Virtual Dynamic Shared Object) and you need to disable this feature. Edit the file `/etc/sysctl.conf` using your preferred editor and add the following line at the end of the file:

```
vm.vdso_enabled = 0
```

After that, you can start Scratchbox to check the installation. Open a terminal and type the following commands:

```
> Xephyr :2 -host-cursor -screen 800x480x16 -dpi 96 -ac -kb
&
> /scratchbox/login
[sbox-FREMANTLE_X86: ~] > export DISPLAY=:2
[sbox-FREMANTLE_X86: ~] > af-sb-init.sh start
```

As expected, you will see a simulator of maemo 5 operating system.



Figure 3. Maemo 5 simulator in Linux machine.

1.2.2 Esbox installation

ESbox is an Eclipse-based IDE. It is based on Eclipse Ganymede (3.4.2) and provides C/C++ and Python support, source editing, project building, run/debug/profiling, and Debian package deployment. ESbox communicates transparently with Scratchbox and one or more Maemo SDKs, available separately, to let you develop Maemo applications. ESbox can also update the image on your Maemo devices and communicate with them using the various network clients and servers available with the PC Connectivity project.

With ESbox, Scratchbox, Maemo SDK(s), and PC Connectivity installed, you can run, debug, and profile locally under emulator running in an X server, and also deploy the application to a Maemo device (N800, N810, N900) to run, debug, and profile on hardware.

Esbox needs Java environment to run. Download newest version of JRE from Java website /15/, choose Linux self-extracting file. You will get a *.bin file.

Change the permission of this file to executable by the following command:

```
chmod a+x jre-6u<version>-linux-i586.bin
```

Copy the file to a location you want to install, for example /usr/java, login as root and change current location to this location again.

```
sudo su
cd /usr/java
```

Run the self-extracting binary file:

```
./jre-6u<version>-linux-i586.bin
```

Accept the license agreement and follow the wizard instruction until it finishes. Java will be installed in folder /usr/java/jre1.6.0_<version>.

Download esbox from its website /16/. It includes 2 zip files:

- Common components
- Linux/GTK support

Extract the two files to the same folder you specify, for example /home/user/esbox. At this state, when you click on esbox icon it will not start because of linking to JRE problem of esbox. In order to overcome this problem, copy your JRE installation folder to esbox home folder and rename it to jre:

```
cp -r /usr/java/ jre1.6.0_<version> /home/user/esbox
mv /home/user/esbox/jre1.6.0_<version> /home/user/esbox/jre
```

Double click esbox.sh to start esbox with auto-linked Maemo OS simulator – Scratchbox and a full Maemo software development kit SDK.

1.3 Maemo PC connectivity

The Maemo PC Connectivity project aims to make an easy communication between a Maemo device and host PC through USB, wireless or Bluetooth connection. Developers can take advantages of this tool for deploying the application to the Maemo phone directly. The setup procedure is described below.

1.3.1 Introduction

The Maemo PC Connectivity project is targeted for the communication between a Maemo device and host PC through USB, wireless or Bluetooth connection. It provides developers with many tools such as connection sharing, remote accessing, file sharing and transfer, remote debugging and deployment.

Maemo PC connectivity supports Maemo Framantle and Diablo OS in mobile devices and Windows, Linux, Mac OS in host computer.

In developer point of view, Maemo PC connectivity enables developer to deploy the application directly to Maemo phone via USB, wireless or Bluetooth without the need to make installable deb file and install manually in the phone. The Maemo development environment consists of a Maemo device and host PC running Scratchbox and Maemo SDK integrated together with services provided by Maemo PC connectivity. With these tools up and running, developers can code the software in Esbox, test it using a simulator inside Scratchbox and finally deploy it directly to the phone.

Some basic services provided by Maemo PC connectivity between host computer and Maemo device include:

- Connection establishment: By using Maemo PC connectivity applets, a connection between Memo phone and host PC can be established via USB, WLAN or Bluetooth; user can also configure settings for allowed services between the two parties.
- Connection sharing: Once the connection between host PC and Maemo device has been established, it is possible to easily share the internet connection from one device to other.
- Secure Shell: Like other Linux distributions, user can take the advantage of securely sharing data between the host PC and remote mobile device using SSH command which creates a secure channel between the two devices and protects the connection from outside attacks.
- Scratchbox Remote Shell: SBHFS is a remote command execution system which is similar to rsh and ssh. It is designed with slow devices and Scratchbox's special requirements.
- Network File system: NFS allows file system sharing over a network.

- Samba: provides shared access to files, printers, serial ports and miscellaneous communications between nodes on a network. Most usages of Samba relate to host PC running Microsoft Windows.
- Virtual Network Computing: VNC is a graphical desktop sharing system that allows a host PC to remotely control a Maemo device.
- Rdesktop: Opposed to Virtual Network Computing, Rdesktop is a client for remotely accessing Windows desktops from a Maemo device.
- X tunneling over SSH: allows the forwarding of X11 desktop from a Maemo device to a host PC.
- Secure Copy: SCP is another way to transfer files between a local and a remote host PC using SSH
- Rsync: is a utility that provides file synchronization between host PC and mobile devices. It uses a protocol that transfers only the bytes inside files that have been changed since the previous transfer.

1.3.2 Installation and configuration

The Maemo PC connectivity product consists of two parts, installation of necessary services and configuration tools to Maemo device, and installation of necessary clients to the host PC which is used to access Maemo PC connectivity services from the Maemo device. The Maemo PC connectivity product includes many standard services available for Linux PCs by default. Most of the services require user expertise to be able to install and configure them correctly. Maemo PC connectivity aims to provide easy installation and configuration in both parties to make use of these services. For the purpose of this thesis, I use a host machine running Linux OS to communicate with the Maemo device. The procedure can be described below.

1.3.2.1 Installation in Maemo device

In order to install Maemo PC connectivity on Maemo device, the Maemo extras-devel repository should be added. This repository which is targeted for Maemo developers includes a lot of community software. The applications in this repository are being developed and can have bugs. Therefore, it is not added by default. To add this repository, open the “Application Manager”, then click “Repository catalogs” click “Add” and fill all the fields like the following screenshot:

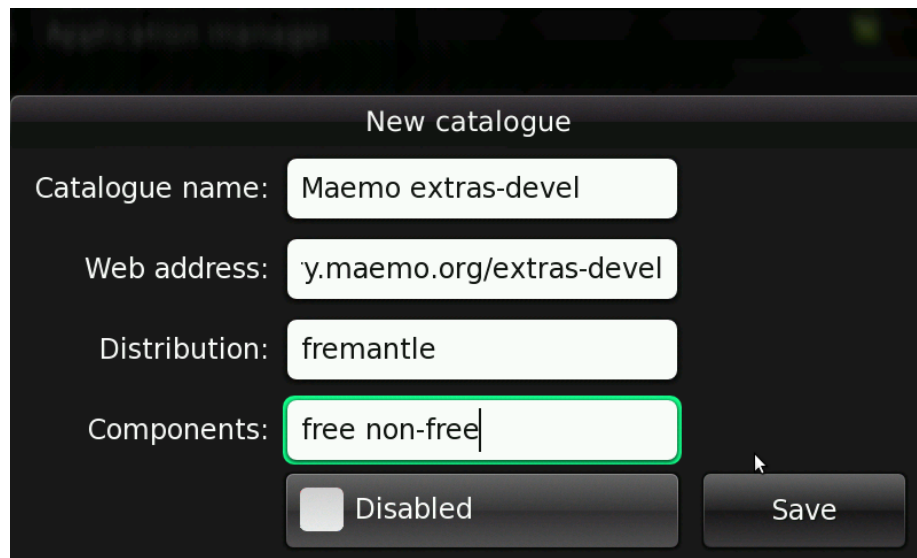


Figure 4. Adding new repository in Maemo device.

After that, refresh the application list and the package named `maemo-pc-connectivity` should be available. Installation of this package will install all the necessary Maemo PC connectivity components for this mobile device.

1.3.2.2 Installation in Linux host machine

Start Linux machine, and click System -> Administration -> Synaptic Package manager to open the Synaptic application manager. In opened window, under “settings” menu, select “Repository” option.

Click on “Add” button and add the following repository to the collection:

```
http://pc-connectivity.garage.maemo.org/repository intrepid  
main
```

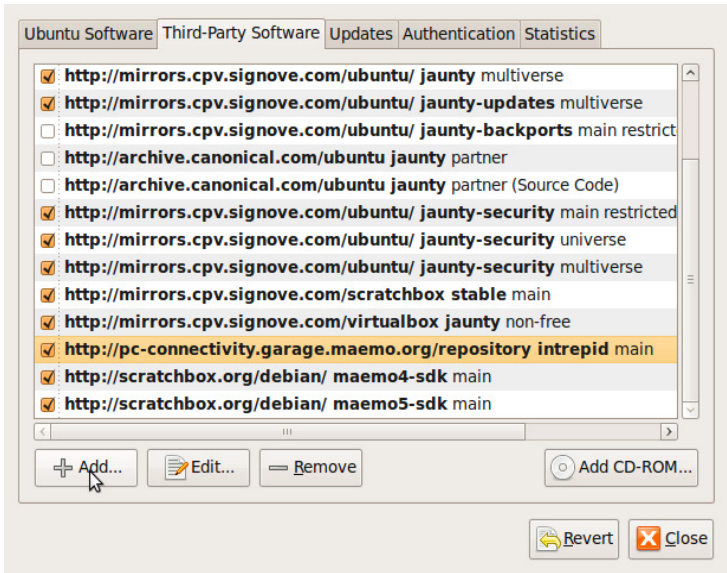


Figure 5. Adding new repository in Linux machine.

Click on “Close” button and reload the application list. After this, user should be able to search and install package named host-pc-connectivity.

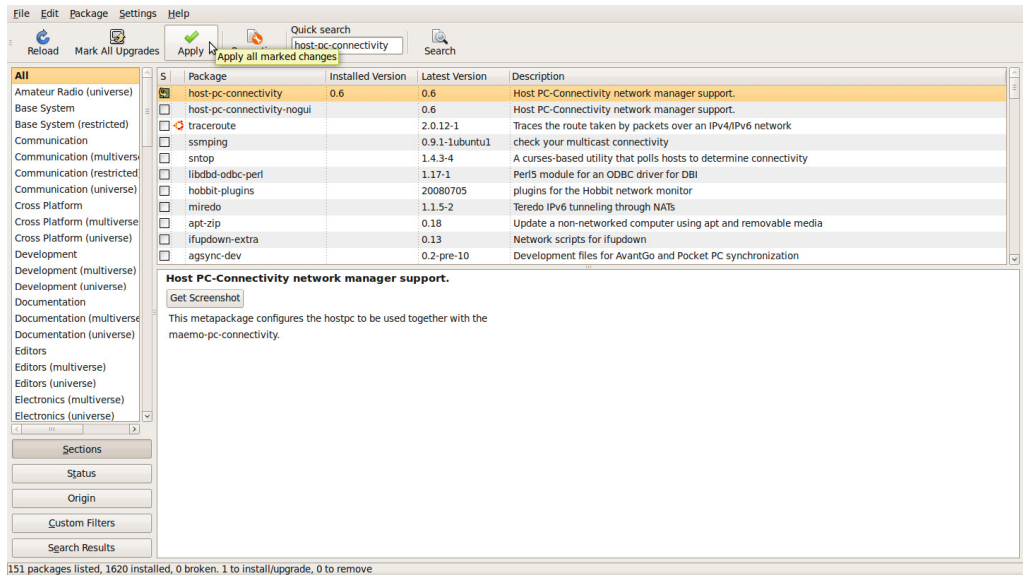


Figure 6. Host-pc-connectivity package installation.

1.3.2.3 Creating environments

With aemo PC connectivity, it can be used the idea of environment to contextualize the configuration of connections and tools. For instance, you can create an environment named "Home" that has the USB connection and the NFS tool configured and an environment named "Office" that has the WLAN connection and the SBRSH tool configured. When you select an environment, all connections and tools configured are applied.

In order to configure an environment on Maemo device, go to Settings -> Control Panel -> Connectivity -> PC connectivity manager. On PC connectivity manager window, click on "Advanced" button, then click on "New" button to add new environment. Set the environment name "Home" and click on "Ok". Click "Save" to save the configuration changes.

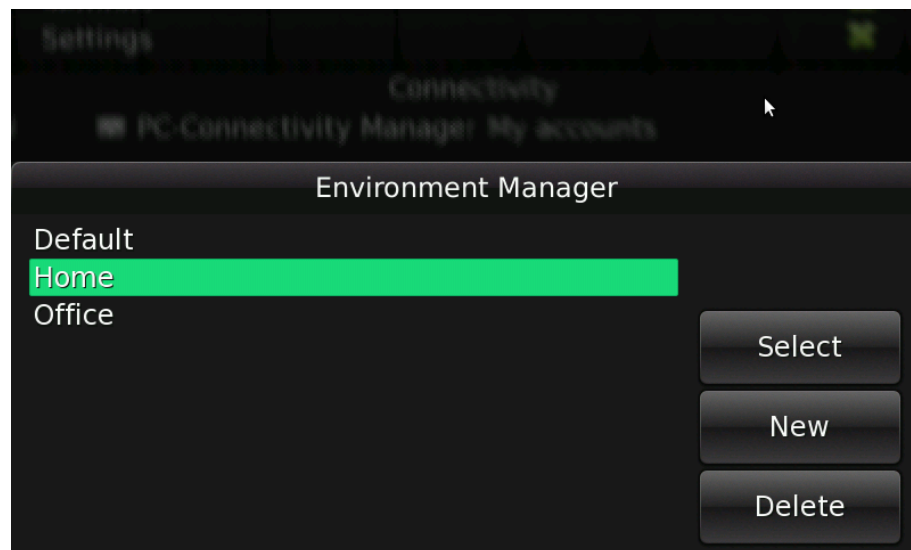


Figure 7. Maemo environment manager.

User can also configure connections and tools to be applied in that environment for instance USB networking.

After configuring your environments, user can switch between them by using the Connection Switcher applet which is available in the status bar:

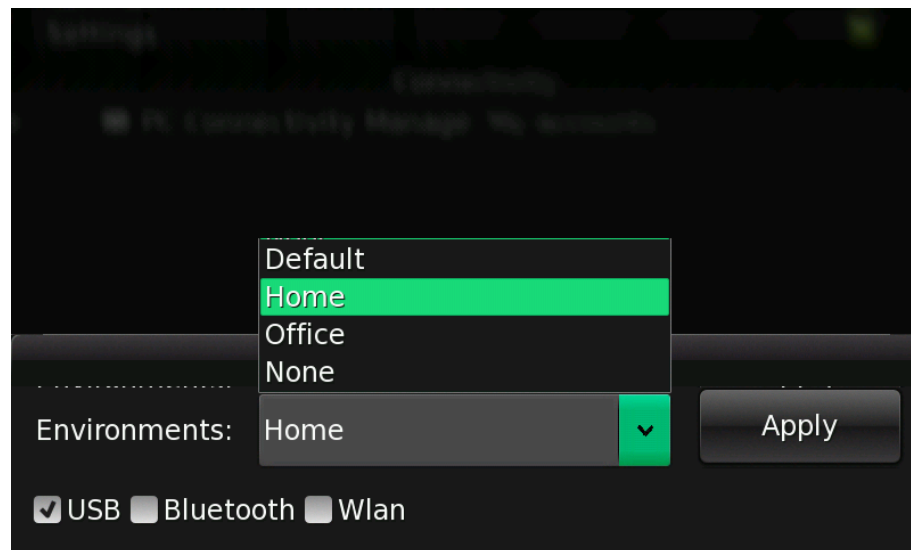


Figure 8. Maemo environment selector.

1.3.2.4 Connection establishment

On Maemo device, connection can be easily configured by using Maemo PC connectivity applets. User can have multiple options for the connection: USB, WLAN or Bluetooth. In my case, I use USB connection which is the easiest way. Connect the Maemo phone with the Linux machine. There will be a popup window asking to choose between Mass storage mode and PC suit mode. You should just click outside the popup to keep the mode previously set by Maemo PC connectivity. In order to set up correctly, follow the following steps:

- Disconnect the USB cable.
- Run Setting -> Control Panel -> Connectivity -> PC connectivity manager.
- Check “USB” option.
- Press “Apply” button.

The USB mode will be switched to USB networking mode with the following default values:

```
DHCP Server: enabled
IP address: 192.168.2.15
Gateway: 192.168.2.14
Netmask: 255.255.255.0
```

Connect the USB cable to the host PC and the Maemo PC connectivity has been successfully established.

1.4 Building and Debugging a QT Maemo application

Open Esbox. Point to File -> New and click "New Maemo 5 C++ Project". In Template Project Type window, choose "Maemo 5 Hello World" Project and click "Next". In Project Configuration windows, choose both configurations FREMANTLE_ARMEL and FREMANTLE_X86 as target architectures for your application. By doing this, the application can deploy and run in both x86 architecture which is a Linux machine (in testing phase) and Armel architecture which is Maemo machine (in final running version). Click Finish to leave other fields default values and create a new project.

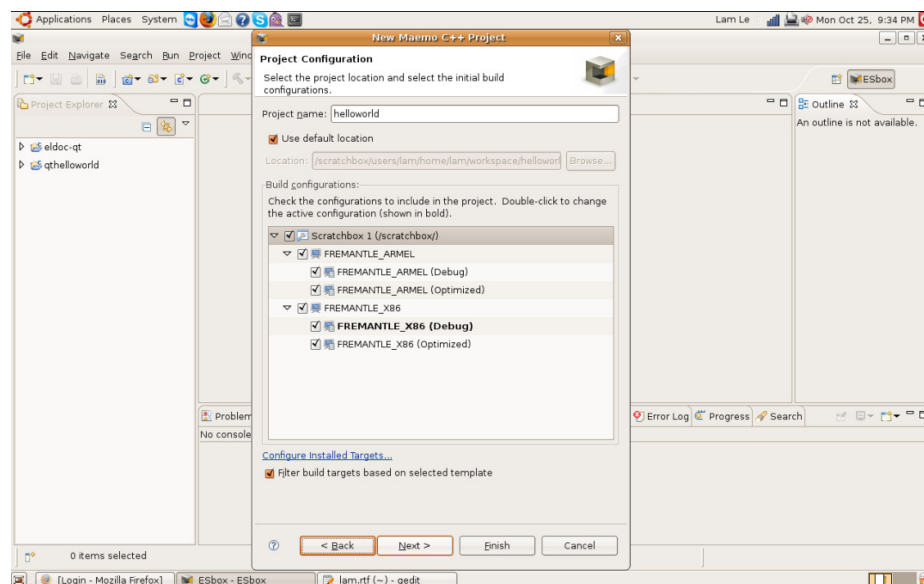


Figure 9. Qt Maemo project wizard.

If there is a dialog asking if you want to check for required build, runtime, and debug packages, Choose "Yes" to allow the application to get the latest build packages. In validate installed packages window, check both "Update package list" and "upgrade installed packages" and click "Finish".

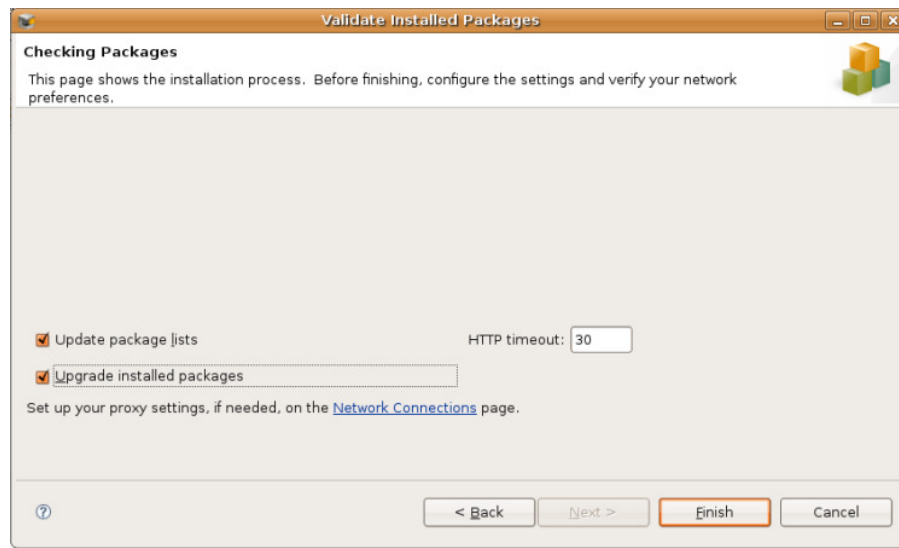


Figure 10. Nokia closed binary packages update.

After creating, the new project will have main.cpp and a class MyWindow where the UI is designed, just like normal QT application. Moreover, Esbox will automatically create some configuration files in debian folder which are needed in creating installable deb file. In these files, the file named "control" is the most important file defining the package information. You can also modify this file with your information

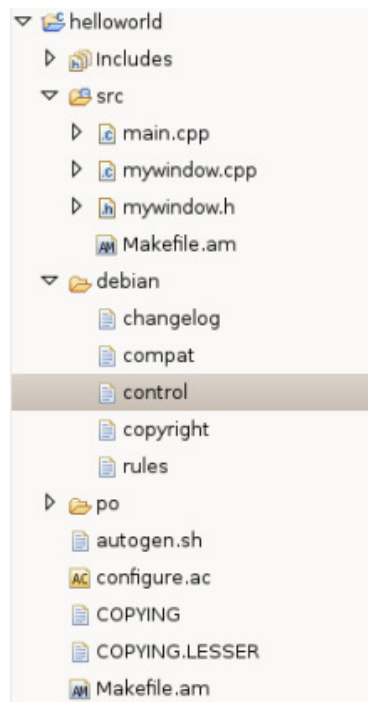


Figure 11. Qt Maemo project structure.

Having a look at file `mywindow.cpp`, you can see in the function `"on_menu_helloworld"`, it uses new QT component `Hildon::Note` which is included in library `hildonm.h`. This is one of additional UI libraries for creating maemo application special looking and feel. This specified sample component will have default MAemo 5 look and feel and specific behavior under Maemo OS.

```
void MyWindow::on_menu_helloworld()
{
    /* Create and show a "Hello World" dialog */
    Hildon::Note helloworld_note (Hildon::NOTE_TYPE_INFORMATION,
    ("Hello World!!!"));
    helloworld_note.run();
}
```

Start Scratchbox: In Esbox window, on the standard toolbar, click on "Start X server" button to start Xephyr server. After that, click on "Start Maemo application framework" to start Scratchbox.

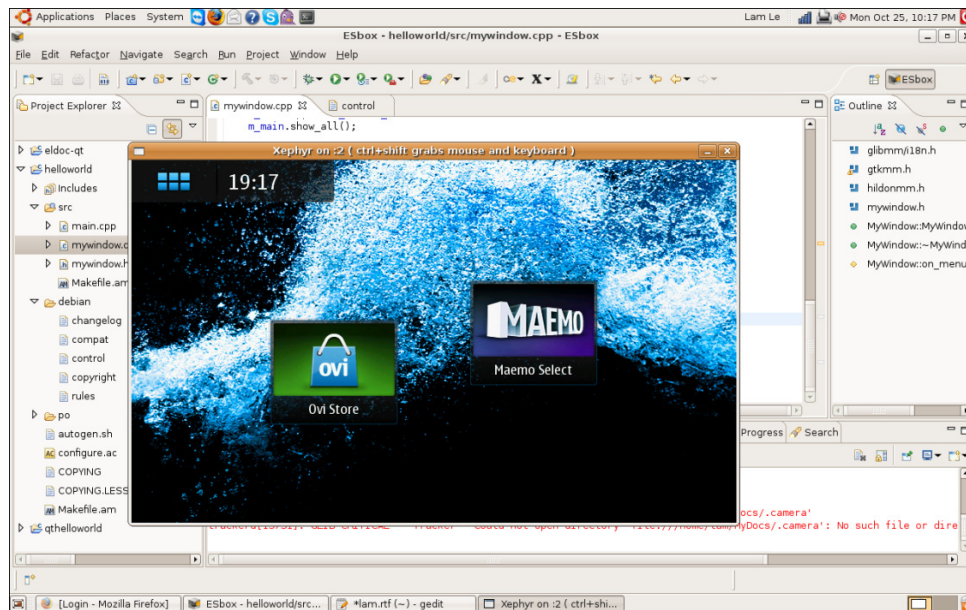


Figure 12. Starting Scratchbox inside Esbox IDE.

Run sample application: Right-click on the project and choose Build configurations -> Set active -> FREMANTLE_X86 to activate x86 mode. Right-click on the project again and choose Run as -> Maemo Local Application

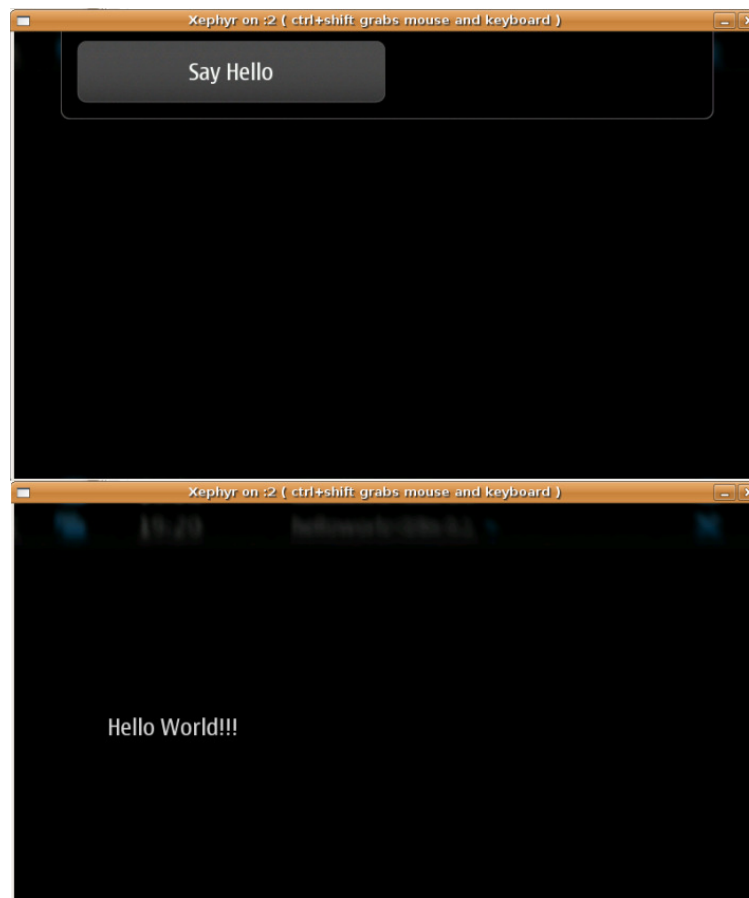


Figure 13. Sample Maemo application.

1.5 Creating debian package and installation process on Maemo phone

Like what is stated before, in order to create installable debian package to install in other Maemo device, developer has to create a "debian" folder containing some configuration files. These files contain basic information of the building process for instance software information, changelog, copy right, etc. When creating sample Maemo 5 C++ project, Esbox automatically adds those config files into the project. Developer can edit these files to update package information.

Among these file, developer can modify the file named "control" to edit package information like package name, description, architecture, etc. Most importantly, the build-Depends option specifies additional libraries needed prior to application deployment. Whenever software developer uses any additional library, it should be

listed here. When installing the deb file in Maemo OS, the system will check if all required libraries listed here are already installed in the system. If there is a library missing, installation will fail and user is notified about missing library. Below is one sample control file.

```
Source: helloworld18n
Section: unknown
Priority: extra
Maintainer: Lam Le <lamle@wapice.com>
Build-Depends: debhelper (>= 5), libhildonmm-dev ( >= 2.1.1
), libosso-dev ( >= 2.21 )
Standards-Version: 3.7.2

Package: helloworld18n
Architecture: any
Depends: ${shlibs:Depends} ${misc:Depends}
Description: Hello World C++
```

Creating debian package: Right-click on the project and choose Debian package -> Build debian package. On the Select Target window, choose FREMANTLE_ARMEL as the target architecture. After that select a folder to put generated debian file and click OK, the building process will start.

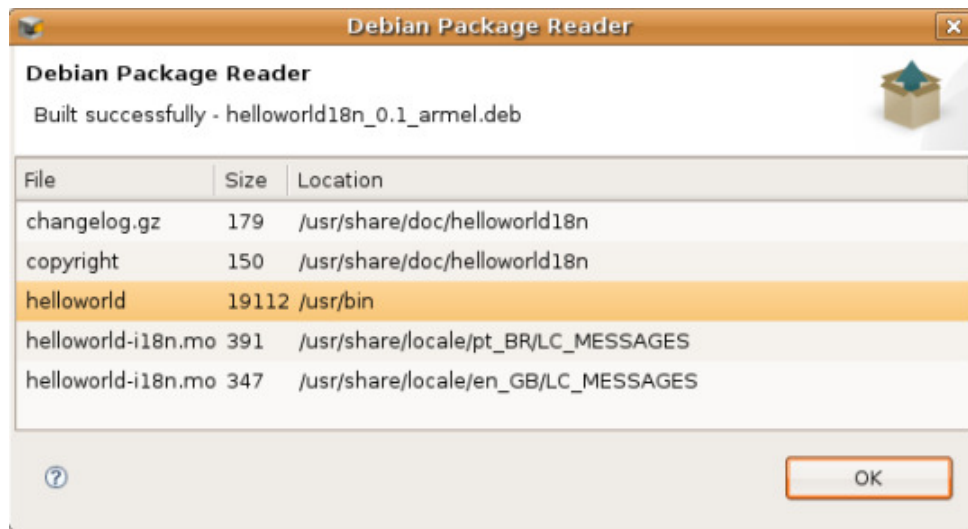


Figure 14. Creating debian package for Maemo device.

After the building process finishes and the debian file is generated, user can simply copy the file to Maemo device via USB cable and install the file in the phone by the following command:

```
sudo gainroot //get root previledge
apt-get -f install <filename>.deb
```

The -f option will allow the system automatically install missing packages if needed.

Using Esbox, developer can also have the option to install the application directly to the phone if Maemo PC connectivity has already been established. Right-click on the project and select Debian Package -> Install Debian Package On Target. In new open dialog, choose Maemo device USB or Bluetooth or WLAN ad-hoc depending on your connection and click Finish, the created debian installable file will be installed in the Maemo phone directly.

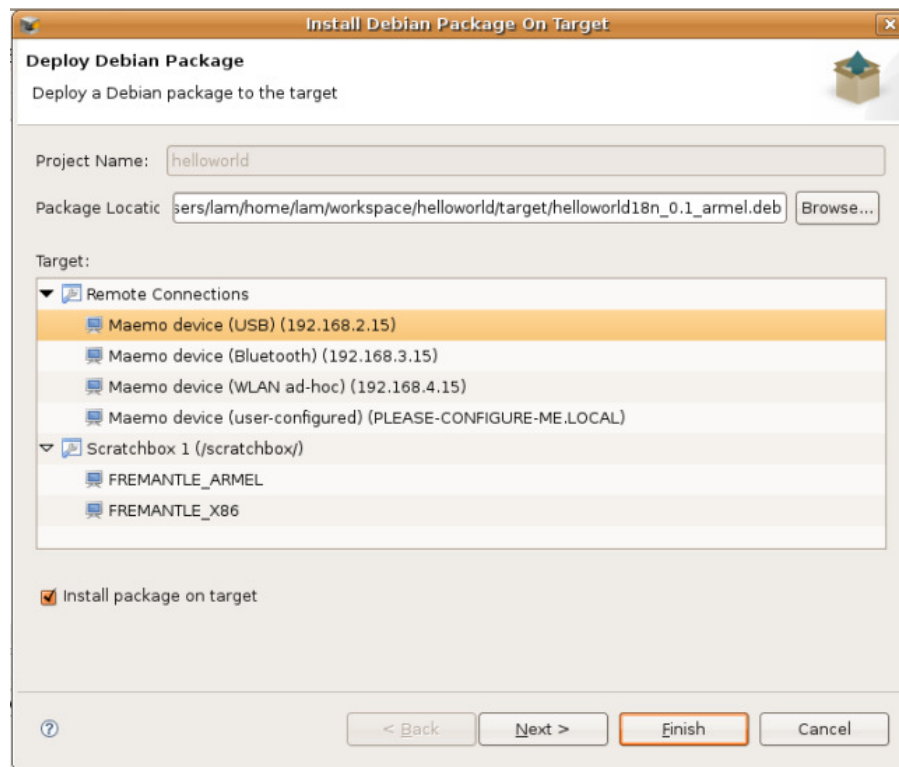


Figure 15. Deploying debian package directly to Maemo phone.

After this step, the application should be running in the phone. However, this kind of deployment is only targeted for developers to debug the application on the phone. When user closes the application, it will be removed from the phone. In order to have a perfectly running version of the application on the phone, developers have to create an installable debian file and install it in the phone. This procedure has been described in previous chapter.

2 MADDE – MAEMO DEVELOPMENT TOOL ON WINDOWS

Windows user can also experience Maemo development by a tool created by Nokia named Madde (Maemo Application Development and Debugging Environment). The purpose of this tool is to make the development less painful and easier for developer to start. Below I will give a description about development process using this tool.

2.1 Introduction

Madde is a Maemo application development tool in Windows platform. This tool can be integrated with developer preferred tool Qt creator. Developers can use Qt creator to code and compile the application either for Windows platform or Maemo platform. However, this tool is not officially published yet. Currently it is a ‘Technology Preview’ of a new development tool for Maemo. As a result, it is not so stable and there might be a case that some of the components do not work. Nevertheless, this is a quite interesting and easy tool for developer getting familiar with Maemo development, developers will be able to build project in Qt Creator for maemo device, and deploy, run and debug the application with a few mouse clicks.

Advantages of Madde:

- Windows development possibility.
- Include already-built tools which can generate installable deb file directly.
- Connect and deploy directly to maemo phone through an application on the phone named Mad Developer.

Disadvantages of Madde:

- Do not have maemo simulator. Consequencely, it is hard to debug the application on maemo platform.
- Installation of new library or new packages to the SDK is not supported. There are some workarounds for this but it is not easy and not always successful.

2.2 MADDE installation on Windows machine

Download and install Madde from here its official website /17/.

Download and install Qt Creator: As Madde is currently just an experimental project, it is not supported in the official release of Qt Creator. In order to get the version supports Madde, developer should download the nightly-build version of the latest Qt Creator from its website /18/. During installation, in addition to basic components, the components named "MinGW" and "Post mortem debugging" of Qt Creator should be also installed.

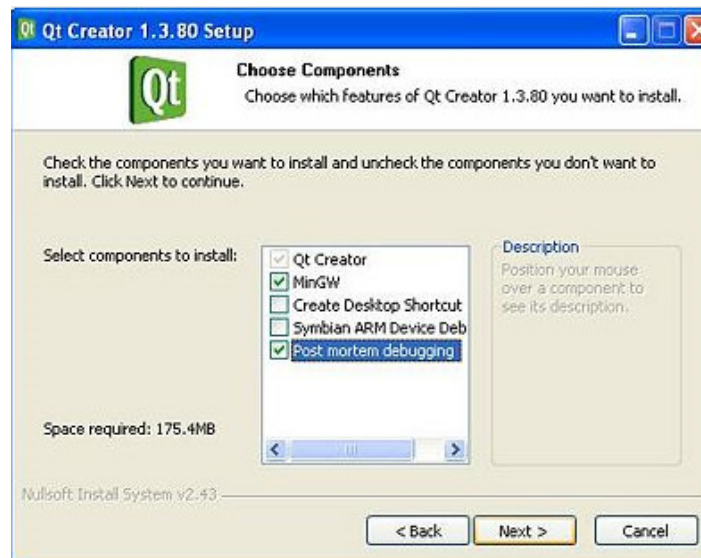


Figure 16. Qt Creator nightly-build version installation.

Configure Qt version in Qt Creator to use Madde library: In Qt Creator, click on Tools-> Option and in Options window, choose Qt4 -> Qt versions. Click on Add button to add new Qt version and specify the location of Madde qmake:

```
c:\madde\0.6.14\targets\fremantle-qt-0951\bin\qmake.exe
```

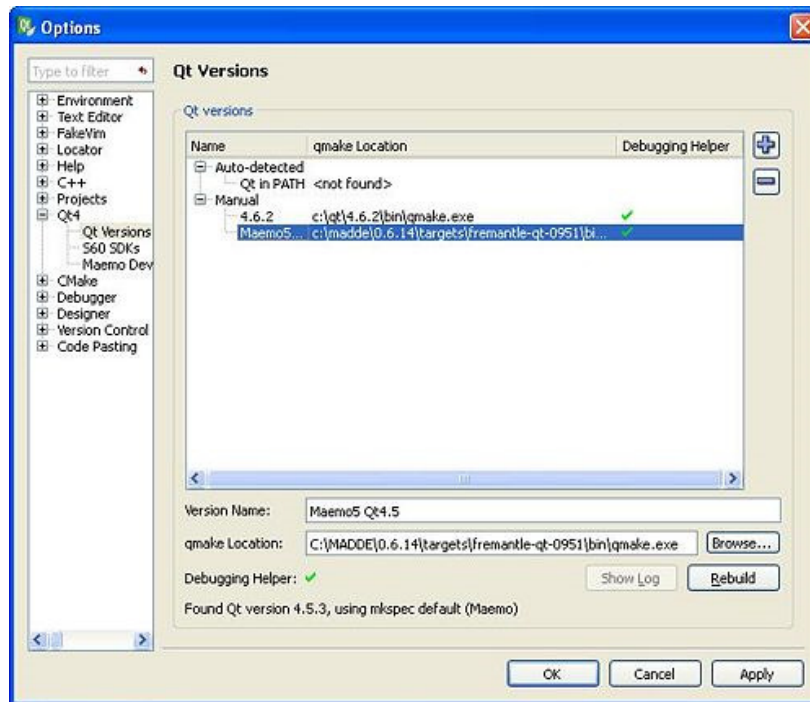


Figure 17. Adding MADDE qmake library.

Click OK to close Options window and your Qt Creator can deploy application in 2 platforms: Windows with standard qmake and Maemo with new madde maemo5 qmake.

2.3 Configure phone connectivity with Qt Creator

In order to deploy, test and debug the application on Maemo device, the connection between windows PC and the device should be established. The configuration is divided into 2 parts: Setting up Windows environment and installation of software in mobile device.

2.3.1 Device software installation

Open the application manager and enable the Extras repository if it is not already enabled.

Go to Download/Development/mad-developer and install the client.

Connect your phone with the PC by Usb cable, wireless connection or Bluetooth connection. In this case I use USB cable.

Start the client on maemo device. Click on “Mange Usb”. On new open tab, click on “Load g_ether” and then click “Close”.



Figure 18. Mad-developer.

After that you can set up the Usb settings. Click on “Edit” to open Network Configuration Settings tab. Click on “Configure” and your client is set up for the connection with your PC. In this case the IP address of the phone in the interface with the PC is 192.168.2.15.



Figure 19. IP address configuration.

2.3.2 Set up Windows environment:

Download PC-Connectivity from its website /19/ and install it to your computer. When installing, in addition to basic options, choose also the option “UsbNetworking”. Connect your device with the PC. The client should be configured as described above. Choose "PC suite mode" as connection method. User should see those configurations below:

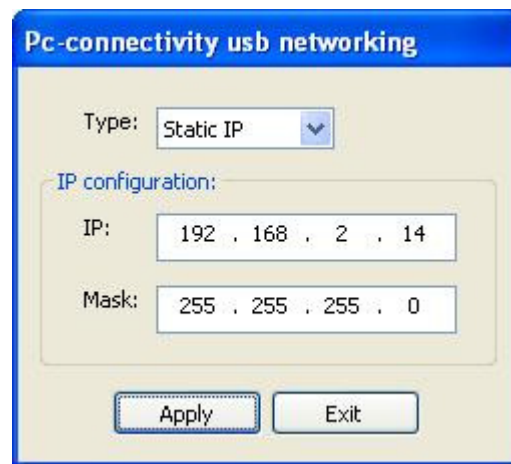


Figure 20. PC connectivity usb networking.

Click on “Apply” and you will see a notification that a new connection with your mobile phone has been established.

2.3.3 Qt Creator configuration

After the configuration between your computer and mobile device has been established, open Qt Creator to configure so that it can deploy the application on both platforms Windows and Maemo. The Windows platform is configured by default, we can add the Maemo platform in the configuration window. Click on Tools->Options, in Options window click on Qt4->Maemo Device. In Memo Device Configurations window, click on Add to add new device and fill all the information like the following screenshot.

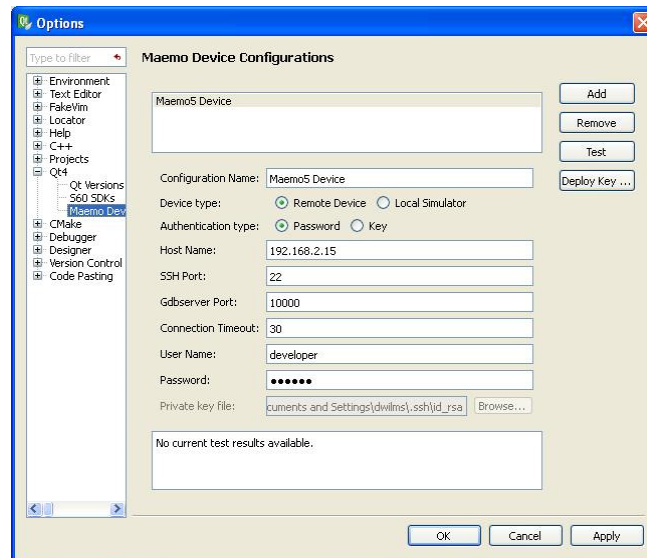


Figure 21. Qt Creator network configuration.

192.168.2.15 is the IP address of your Maemo phone. You can get the password from Mad developer on your Maemo phone. On Mad developer, click on 'Developer Password' to generate a password for Qt Creator. You should keep this window open as long as Qt Creator can connect and deploy to your phone. The password changes every time you open Mad Developer.

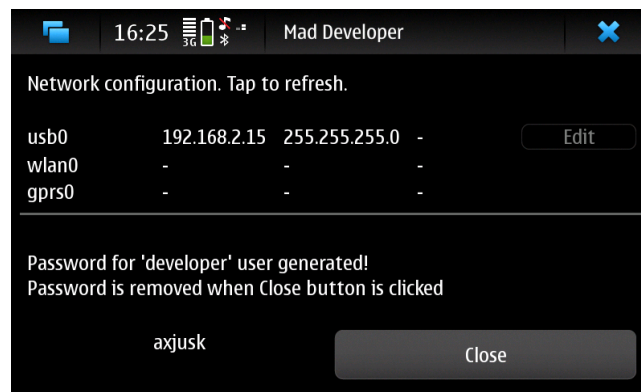


Figure 22. Mad developer temporary password.

Finally, Qt Creator can deploy application on both Desktop and Maemo Device.