Ministry of Education and Science of Ukraine
Ternopil Ivan Puluj National Technical University

Faculty of Computer Information Systems and Software Engineering
(full name of faculty)
Computer Science Department
(full name of department)

QUALIFYING PAPER

For the degree of

Bachelor's thesis
(degree name)

topic: Development of a software module for working with PDF files using the Qt framework

Submitted by: fourth year student VI , group ICH-43
specialty 122 "Computer science"
(code and name of specialty)

		Yusuf Kamal Adebowale
	(signature)	(surname and initials)
Supervisor		Roman Zolotyi
	(signature)	(surname and initials)
Standards verified by		Oleksandr Matsiuk
	(signature)	(surname and initials)
Head of Department		Ihor Bodnarchuk
	(signature)	(surname and initials)
Reviewer		
	(signature)	(surname and initials)

Ternopil
2023

Ministry of Education and Science of Ukraine
Ternopil Ivan Puluj National Technical University

Faculty Faculty of Computer Information Systems and Software Engineering
(full name of faculty)

Department Computer Science Department
(full name of department)

APPROVED BY
Head of Department
_____ Ihor Bodnarchuk
(signature) (surname and initial0073)
«____» _____ 2023 p.

ASSIGNMENT
for QUALIFYING PAPER

for the degree of Bachelor's thesis
(degree name)

specialty 122 "Computer science"
(шифр і назва спеціальності)

student Yusuf Kamal Adebowale
(code and name of the specialty)

1. Paper topic Development of a software module for working with PDF files using the Qt framework

Paper supervisor Ph.D., Assoc. Prof Roman Zolotyi
(surname, name, patronymic, scientific degree, academic rank)

Approved by university order as of «_14_» _12_____ 2022 № _*4/7-1010*_____

2. Student's paper submission deadline 20.01.2023

3. Initial data for the paper Technical requirements for system

4. Paper contents (list of issues to be developed)
   1. JUSTIFICATION OF THE RELEVANCE OF THE DEVELOPMENT
   2. DESIGN COMPONENTS. REQUIREMENTS ANALYSIS AND SOFTWARE SELECTION
   3. TESTING AND DEVELOPMENT OF THE COMPONENT
   4. OCCUPATIONAL HEALTH AND EMERGENCY SAFETY

5. List of graphic material (with exact number of required drawings, slides)
   1. Title slide
   2. Actuality
   3. Purpose of work
   4. Main part
   5. Conclusion

6. Advisors of paper chapters

Chapter	Advisor's surname, initials and position	Signature, date	
		assignment was given by	assignment was received by
LIFE SAFETY,			
BASICS OF LABOR PROTECTION			

7. Date of receiving the assignment _____

TIME SCHEDULE

LN	Paper stages	Paper stages deadlines	**Notes**

Student _____ Yusuf Kamal Adebowale
(signature) (surname and initials)

Paper supervisor _____ Roman Zolotyi
(signature) (surname and initials)

ANNOTATION

Development of a software module for working with PDF files using the Qt framework // Yusuf Kamal Adebowale // Ternopil National Technical University named after Ivan Pulyuy, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, ICH-43 group // Ternopil, 2023 // P. ___, fig. - ___, tables - ___, chair. - ___, annexes - ___, references. - ___.

Key words: component, pdf, programming, viewer, analysis.

This thesis is devoted to the development of a software module for working with pdf files. During the implementation of this thesis, the main concepts of developing software modules using the C++ and Qt programming languages, the disadvantages and advantages of this language were considered.

An overview of general information about existing solutions used for working with PDF files was also reviewed, the module development process was described, and testing was conducted. In addition, the explanatory note describes the operating systems and module settings for these systems.

The result of the work can be used by any user who is familiar with programming.

LIST OF SYMBOLS, SYMBOLS, UNITS, ABBREVIATIONS AND TERMS

MOC - Meta Object Code

UIC - User Interface Compiler,

RC - Resource Compiler

VS – Visual Studio.

CONTENT

INTRODUCTION

In the era, as we known, of modern technologies, information plays a very important role. The development, as we see, of scientific and technological progress has led, as rule, to the fact that the many organizations activities and, as usually, institutions are now related closely to the storage of a amount large, as rule, of information. To process and store information, structures began to use information systems, a mandatory databases component.

Nowadays, the number, as usually, of many users with personal computers has increased significantly. It is not surprising, because they not only speed up, as usually, the time of work, but also allow you to save information resources. These resources also include electronic books and various documents. These resources are of great interest from different points of view of society.

On computers, tablets, e-readers - everywhere we come across various software for viewing electronic documents. This software has not just become popular in the world. In particular, a large amount of this provision can be singled out. Each of them is characterized by its advantages and disadvantages.

It is worth noting, as usually, that computer equipment is found in the possession of almost everyone who uses the functions that they provide us. This is both mobile communication and the exchange of resources, both graphical and textual, etc. We are, as usually, trying to expand the functionality of the technology we use. In this way, systems were, as rule, created that allowed a specially trained person to add to the functionality exactly what is necessary for him, and only then give the opportunity, as usually, to use new developments to those who need it.

1 JUSTIFICATION OF THE RELEVANCE OF THE DEVELOPMENT

1.1 Description of the informatization object

In today's world, for efficient operation of enterprises, companies and other institutions, it is necessary able for operate electronic documents, books and documentation well. We have many reasons for this, and the main one of them is that employees need to use information that greatly facilitates their work, no matter in what field, while working, without having original documents or other means of information with them. For example, programmers need to use documentation for languages of programming type and rapid tools for development. This is, as usually, a necessity, because only when starting work with a new system you need to familiarize yourself with, as usually, it first of all.

There are quite, as rule, a few formats for electronic storing documentation and resources other. In particular, as we known, the most popular PDF are, ePub, eBook, fb2, doc, and others types. Each formats, usually, of electronic documents has an electronic viewer, which is installed by the user on the workstation. Viewers can be both official, that is, produced by the format developers themselves, and personal, which are generally supported and developed by a programmers group.

The question of the need to automate the management of document management has long since moved into the practical realm, and more and more Ukrainian enterprises are implementing systems, allowing organizations to assess the benefits of new technology for working with documents on their own experience. However, even for those few who consider automation to be a passed stage, it may soon be necessary to rethink the choice made and plunge again into the problem, as usually, of improving the document management efficiency.

This is due to a change in the market situation, the growth of organizations, as rule, that create crises of the "transitional age" and which, as usually, leads to

the need for restructuring, as well as the information development and communication technologies, on the one hand, providing new opportunities for doing business, on the other - forcing in step with the times to keep up with the competition.

Today, various organizations see the need for automation of document management in different ways: some - in increasing the efficiency of organizational and administrative document flow, others - in increasing the efficiency of the work of functional specialists who create documents and in their daily work use them, and only a few pay, as usually attention to both aspects. Such a division, as rule, of points of view is determined by the different role and significance of the themselves documents in the organization's activities, which on the size of the organization depends, management style, industry, general level of technological maturity, as rule, and many other factors. Therefore, for some, a document can be a basic management tool, and, as rule, for others - a means and product of production.

Now we have many thinking about how, as rule, to improve the process of sharing and reviewing documents. This leads to the purchase of expensive and systems of complex type that are not always clear to employees and require, as rule, them to spend time studying them. Thus, heads of organizations and enterprises think about how, as rule, to make the process of transition to one or another system the easiest, so as not to waste time on its development. Therefore, most choose the process of developing a new corporate solution and its further support.

We should not forget about those who use their developments for profit. So many enterprises more, as rule, focused on the development of a system that will allow to bring the product to the market quite conveniently and, as usuallu, quickly in the future. These can be e-books or even documentation for tools that do not have it. Publishers are more interested in this issue, because nowadays paper books are losing demand, and electronic books are replacing them, which are quite

convenient and do not take up a lot of space, and at the same, as rule, time contain quite a lot of information.

Currently, the electronic format of PDF, as rule, documents has become widely popular. This is an open, as usually, file format that was once developed, as rule, by Adobe System and continues, as we known, to be supported by it. Format is intended for electronic presentation of printing products. This format give you possibilities to save the necessary fonts, vector and raster images, forms and multimedia inserts. Supports almost all color schemes. In addition, it has its own technical formats for printing. Contains a mechanism of electronic signatures for document protection and verification. A large documentation number and books in this format are distributed.

The simplest protection provided by a PDF file is to set a password on the file. When setting a password, the file is, as rule, encoded, and at the same time, the first bytes of information are changed, which signal that the PDF document uses encoding. In addition, the document supports data editing protection, which is quite necessary when sending legal and other types of documents.

A more complex and already expensive protection for PDF documents is DRM protection. For this type of documents, this type of protection is slightly modified and is called E-DRM (Enterprise digital rights management). The main purpose of E-DR M is the prevention of unauthorized use of internal documents outside the enterprise, primarily industrial espionage. Protection technologies are typically combined with electronic management document systems.

A PDF document is most often a combination of text with raster or vector graphics, less often text with forms, JavaScript scripts, 3D graphics and other types of elements.

The information volumes of two PDF documents that look the same on the screen can differ significantly depending on fonts and multimedia, the resolution of bitmap images, and the used document compression mechanism. To create the minimum volume of the document, it is necessary to use vector graphics and safe

fonts. Since 1993, the PDF format has gained quite a lot of popularity and today it is used almost everywhere.

Publishing houses that do not have the capacity to print books publish them in electronic versions. This is how Packt Publishing distributes its books. Packt Publishing has been distributing books and other materials since 2004. During this time, it increased its profits several dozen times, only due to the production of electronic products. Packt distributes PDF versions of all their books for download, and it was only in 2010 that it began introducing a new format for ePub and Mobi sales. However, PDF remains their only primary format.

The PDF format is no longer owned by Adobe Systems, but is still supported by the company. In 2007, the International Organization for Standardization decided to make the format an international standard. The PDF standard is intended for cross-platform presentation of electronic documents.

Thus, today, the PDF format is used almost everywhere and is supported by many systems.

1.2 Analysis of existing solutions

When developing a software application or library, it is not uncommon for a programmer to pay attention to already existing solutions. Thus, when designing and further programming a tool for handling PDF files, it is necessary to analyze already similar implementations. After all, in the future, these implementations can significantly affect the final product, which will already be delivered directly to the user.

When analyzing various Internet resources, we will consider the existing solutions, which are given today, as rule, to us by free developers community and which interested us the most. Among for working with PostScript popular most libraries are:

- Haru PDF;
- PoDoFo;
- MuPDF;
- Xpdf;
- Poppler.

Xpdf

Xpdf is an open source program for viewing Portable Document Format (PDF) files (see Figure 1.1). The Xpdf project also includes PDF text extraction utilities, a PDF to PostScript converter, and various other utilities.

Xpdf runs under the X Window System on UNIX, VMS, and OS/2. The console components (pdftops, PDF to Text translation, etc.) can also run on Win32 systems and should run on most systems equipped with a C++ compiler.

Xpdf is designed to be a small but effective program. It can use Type 1, TrueType, or standard X fonts.

Xpdf need to work on most systems with libraries running Unix-like (POSIX) and X11. To successfully compile XPDF you need, as rule, ANSI C++ and C compilers.

If any security feature is enabled by the creator, as usually, of the PDF document, the PDF preprint files will be encrypted. These security features allow the author to prohibit printing, copying text/graphics, editing and/or adding annotations.

Starting with version 0.91, Xpdf includes decryption.

Since version 2.00, Xpdf includes a "native" LZW decoder.

Lempel-Ziv-Welch (LZW) is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch. It was published by Welch in 1984 as an improved implementation of the LZ78 algorithm published by Lempel and Ziv in 1978.

Xpdf also exists for AmigaOS. The Amiga version requires a limited version of X11 called Cygnix to be loaded on the system.

MuPDF

MuPDF is developed by Artifex, which is also the author and owner of GhostScript.

MuPDF is currently designed as a package of console utilities for working with the PDF format. MuPDF works in conjunction with the Fitz graphics library. MuPDF does the parsing, and Fitz does the rendering. It is fully open source, as rule, and licensed under the GPL or higher. This can be compiled under Windows and Linux.

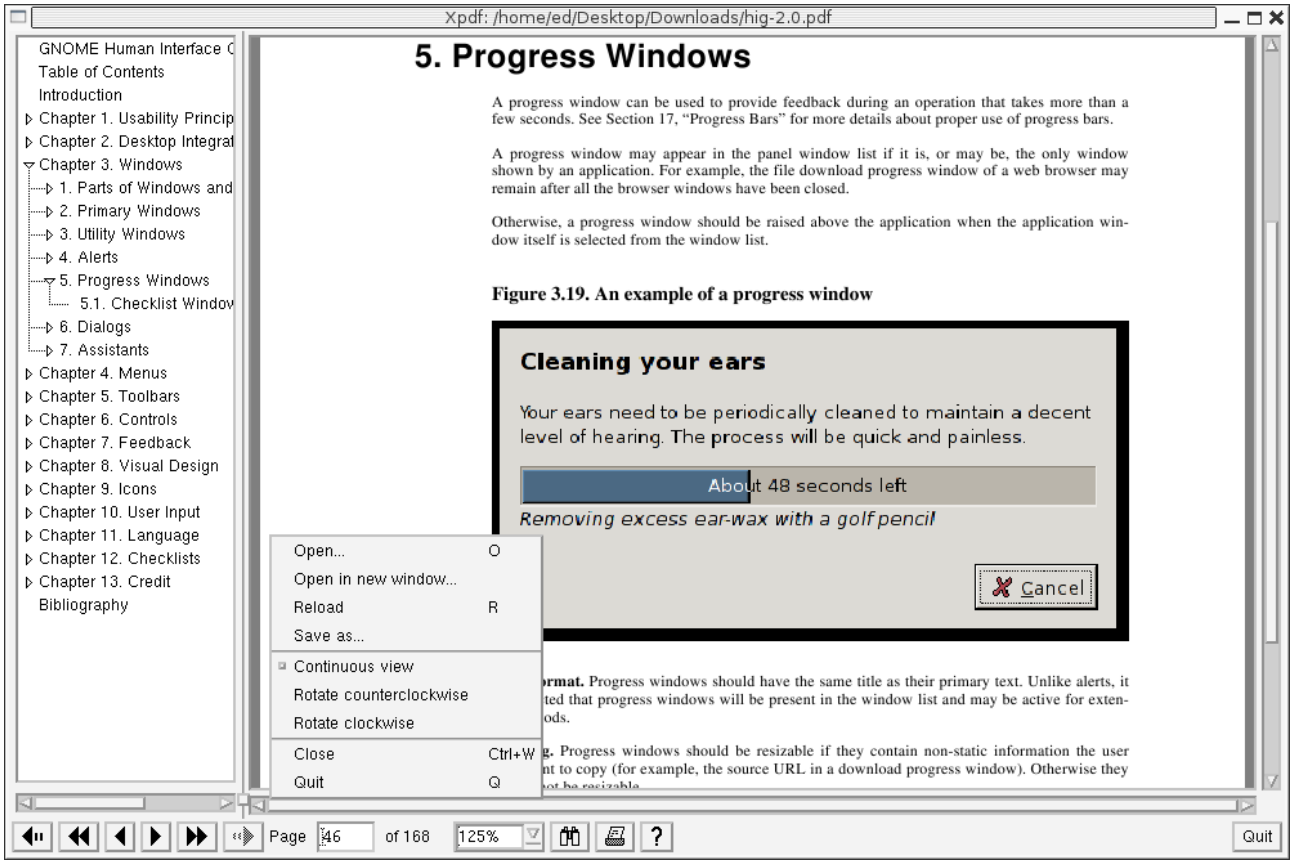

Figure 1.1 – Software for viewing files using Xpdf by PDF type

This library is usually new development of Artifex, which began to be created, as rule, to replace GhostScript. GhostScript was created in 1986, and it is possible that its architecture is so outdated that Artifex decided to start creating a completely new PDF library "from scratch" (to replace GhostScript with it over time).

It is also claimed that MuPDF is 5 times faster than GhostScript and uses the revolutionary Fitz graphics library, as rule, for rendering (fast, high-quality and tree-based), and the latest PDF standard supports - version 1.7. MuPDF is very compact in size (especially compared to GhostScript).

However, the current situation with the development of MuPDF looks the best. The impression general is that the Artifex company, for unknown reasons, practically abandoned of this library development. It is officially supported by Tor Andersson. The complete absence, as rule, of any documentation looks extremely strange for the library. There is only, as we see, a description of the line command syntax for each console utility.

Such "external underdevelopment" to the fact led, as we see, that the article about MuPDF from Wikipedia was removed with the wording "minor library".

MuPDF's promise was captured by the author of the SumatraPDF PDF viewer (see Figure 1.2) by implementing it in his program. It is extremely interesting that SumatraPDF was previously only used by Poppler.

SumatraPDF's switch from Poppler to MuPDF is not accidental - it most likely confirms MuPDF's strategic advantage over Poppler. here is the fact an important point that the creator of MuPDF is the same company that created GhostScript, the most popular and most developed free PDF-tool of today, and it created all the shortcomings of GhostScript and to replace it.

However, not everything is so perfect: the author of SumatraPDF himself made a special page of MuPDF shortcomings, where he examines in detail specific examples of MuPDF's incorrect operation.

As mentioned above, MuPDF does not have any documentation at all. It is worth reading the source files of MuPDF to how it works understand.

PoDoFo

This is a library, as usually, for electronic PDF documents working with. For working with documents some utilities are already included in the package library.

This is portable C++ library, free library, that includes, as rule, classes for PDF parsing files and contents modifying their in memory. On a physical medium changes can be recorded.

The analyzer can also be used to extract information from a PDF file. In addition to parsing, PoDoFo also includes very simple classes for creating your own PDF files. All classes are described so that it is very easy to start writing your own programs using PoDoFo.

PoDoFo is written in C++ and the source code of the library has been successfully compiled under Unix, Mac OS X and Windows.

PoDoFo tools are easy to use. These tools are primarily examples of how to use PoDoFo libraries in your projects. They also provide functions for working with PDF files. With each new release, existing tools get more features.

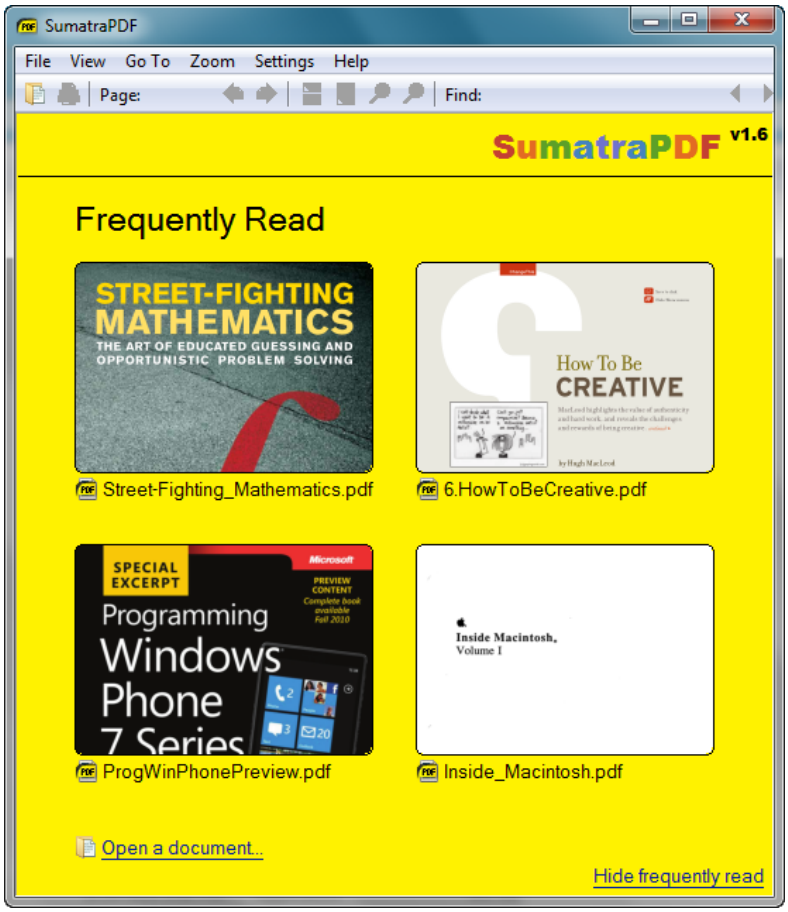

Figure 1.2 - Sumatra viewer PDF software

The tools following exist:

• Pencrypt – allows you PDF protection install and to encrypts any PDF;

• Pimgextract – from a given PDF all images extracts;

• Pimpose – from one or more files of PDF type replaces pages and other, applying positioning and scaling;

• Pmerge – connect into one two PDF files;

• Ppdfinfo – provides some basic information about PDF metadata, number of pages, details, etc.;

• Ptxt2pdf – will convert a text file into a PDF;

• Ptxextract – a tool that extracts all text from a PDF file. Works only for plain PDF files;

• Puncompress – removes all compression filters from a PDF file. This is useful for debugging existing PDF files.

In addition, there is an external tool, PoDoFoBrowser (see Figure 1.3), which is not included in the standard utilities package, but can be downloaded from the PoDoFo web page. PoDoFoBrowser is a Qt application for viewing and modifying objects in PDF files. This is very useful if you want to look at the internal structure of PDF files.

The structure of PDF files is displayed as PostScript code.

HaruPDF

HARU is a free, cross-platform, open source PDF generation library.

It supports the following features:

•

• Creation with text, lines, image files of PDF type.

• Annotation link, outline, annotation text,.

• With decoding compression of documents.

• JPEG, PNG images attaching.

• TrueType fonts and Type1attachment.

• Creation of encrypted files PDF.

• Use of different character sets.

• Encoding and fonts for CJK support.

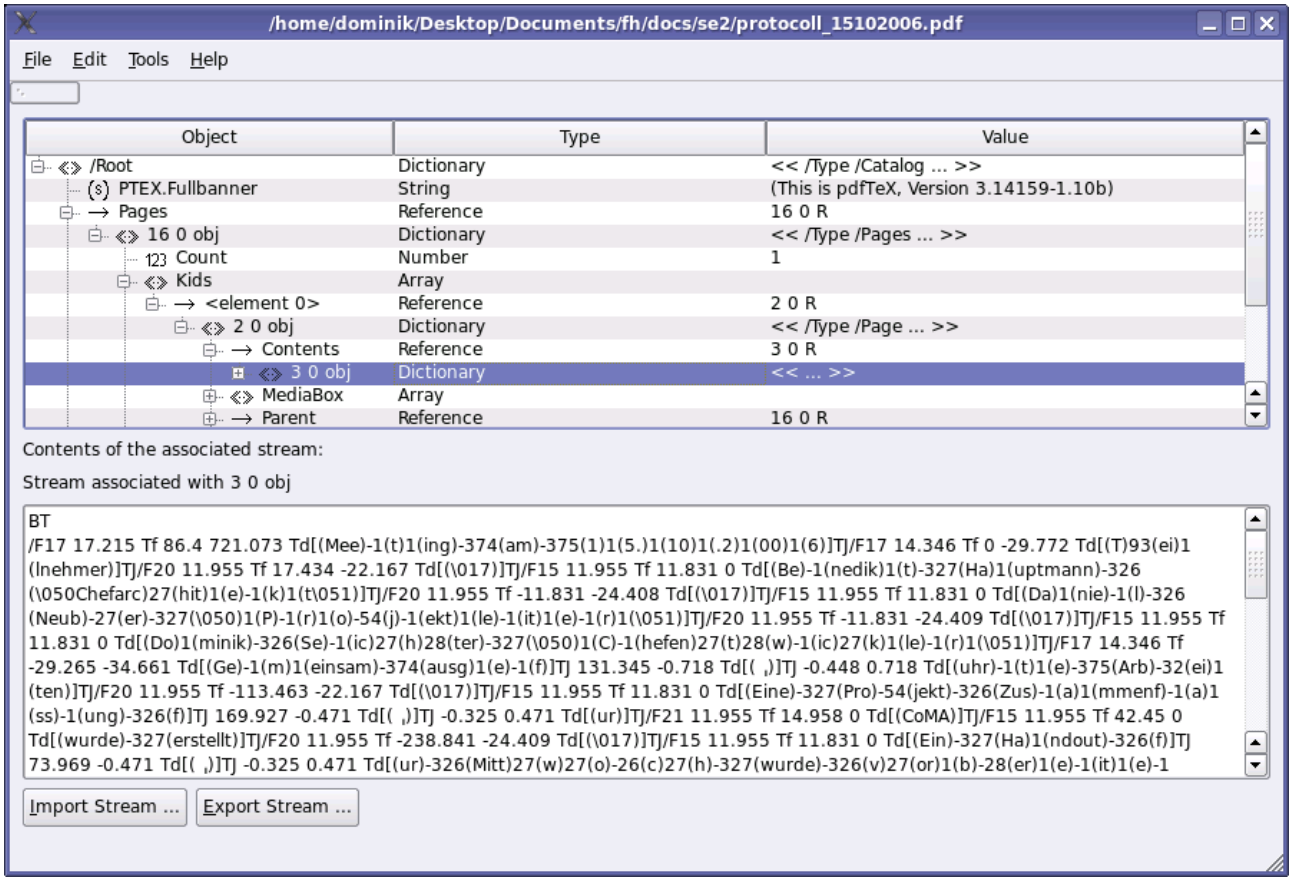

Figure 1.3 - Appearance of PoDoFo Browser

HARU is written in ANSI-C and is fairly easy to compile with any compatible C compiler.

Haru has been tested in various environments and can be assembled using the following means:

1. Cygwin + GCC (Microsoft Windows).
2. Cygwin + MinGW (Microsoft Windows).
3. Microsoft VC++ (Microsoft Windows).
4. Borland C++ (Microsoft Windows).
5. GCC (Linux, FreeBSD, NetBSD ...).

Also, HARU can easily be built on other platforms except above.

In addition, you must have the ZLIB and PNGLIB libraries for compilation if you need to use the compression and embedding functions of PNG images.

Poppler (or libpoppler) is a free software, as rule, library used to display documents of PDF type. Its development is supported by the freedesktop.org community. It is the most frequently used library on GNU/Linux systems, and is used most often as a PDF viewer in systems with GNOME and KDE interfaces.

The project with two goals was started:

1. A free library to make functionality PDF rendering a so that it can be used almost everywhere.

2. Integrate functionality by modern operating systems also provided and to go the goals beyond of Xpdf.

Poppler itself is a wrapper around Xpdf -3.0.

Poppler can use two methods for rendering PDFs, Cairo and Splash. One implementation of the library is based on Qt4 and is called "Artur. There is currently library support for Qt3 and Qt4 , which provide an interface to Poppler. Although Qt3 and Qt4 only support the Splash interface, there is a set of patches that add support for the server version of Cairo to Qt4, but the Poppler project currently does not want to include this feature in the library.

Some backend features include:

• Cairo – smoothing of vector graphics and transparency of objects. Cairo does not smooth images from scanned documents. It also has no dependency on the X Window System, so Poppler can run on platforms that do not use an X server, such as Windows or Mac OS.

• Splash – Supports bitmap filtering.

The Okular software tool (see Figure 1.4) uses the Poppler library to display PDF documents.

1.1 Formulation of the problem

The task of the diploma project is the development of a software complex for working, as rule, with PDF files, as rule, using the Qt Framework, as well as

the creation of a QML component for quick and convenient use of the final, as rule, product in an integrated, as rule, environment.

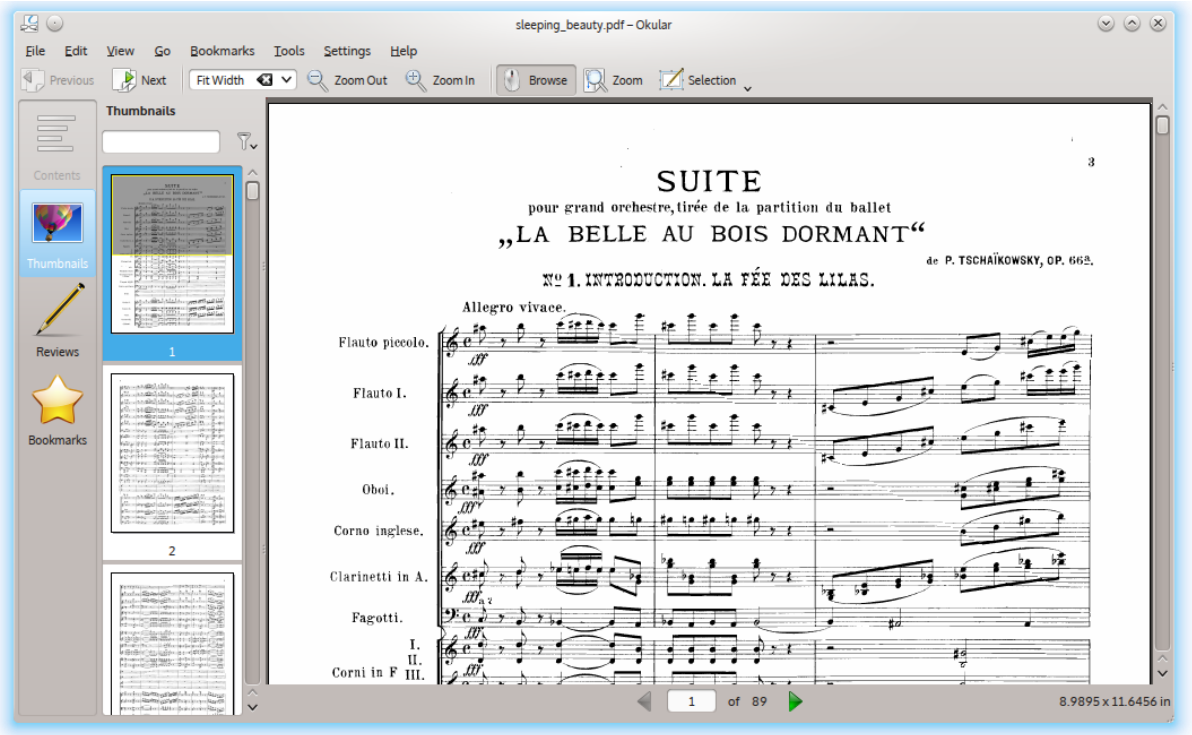

Figure 1.4 – Okular (Poppler) software interface

Qt Framework is a cross-platform, as rule,` software development toolkit in the C++ programming language. Allows software written with it to run, as rule, on most modern, as rule, operating systems by simply compiling, as rule, the program text for, as rule, each OS without changing the source, as rule, code. Includes all the, as rule, basic classes you might need in application development, from GUI elements to networking, database, OpenGL, SVG, and XML classes. The library allows for threading, networking, and cross-platform file access.

QML (Qt Meta Language) is a declarative programming language based on JavaScript and intended for developing, as rule, applications that place the main emphasis on the user, as rule, interface. It is part of, as rule, Qt Quick, a user interface development, as rule, environment distributed with Qt. It is mainly used to create applications aimed at mobile devices with touch control, as well as desktop solutions.

The library, as rule, for working with files PDF must perform the following relevant functionality assigned to it, namely:

• Change the input to display file. The file must meet certain requirements so that the program does issue not exceptional situations and errors do not occur, which would lead to the program exiting. If the file, as rule, is not supported by the library, it should emit a signal with a message that w`ill signal this.

• Display the current page and additional information, as rule, about it. Each page is assigned a sequence number, as rule, as well as a page title, if present. The library should be able to change, as rule, the current page. The change, as rule, algorithm is implemented based on the vector data type, that is, it, as rule, should be possible to change, as rule, the page in any direction. If the page does not exist, the library should generate an appropriate error.

• Zoom the page, display information, as rule, about the current zoom. The library must provide the maximum allowable zoom of the document, as well as the maximum minimum zoom available for viewing.

• Output the original page size. Designed to allow the window to be resized to fit the page and scale properly to fit the display. When, as rule, the page is loaded, it should display all the information it contains, regardless, as rule, of the display size.

• Total number, as rule, of pages. This information, as rule, is only necessary to limit the user and ensure that he does not go beyond the memory, as rule, , as rule, into which the document, as rule, is loaded.

• Save the page. Designed to allow the user, as rule, to save the page he is interested in. However, this function should be disabled if the file is encrypted or has another protection method.

• Saving a fragment of a page. It is designed so that the user, as rule, can use a manipulator or a sensor to save a fragment of the page for further use.

• Bookmark the page. This is a label that will allow the user to instantly switch to it if necessary. The label should have, as rule, a name and page number to make it easier, as rule, to find if the user wants, as rule, to jump to the label.

• Support for extensions in the form of plugins. The functionality, as rule, should be expanded not only by rewriting the component, as rule, code, but also by downloading the plug-in from, as rule, the appropriate directory.

These are only mandatory requirements that must, as rule, be present in the final product. All other, as rule, functionality of the library, as rule, will be added with each release of a new product release. You should also not forget about project documentation and the product debugging mechanism for correcting errors that users may encounter.

2 DESIGN COMPONENTS. REQUIREMENTS ANALYSIS AND SOFTWARE SELECTION

2.1 Description of the subject area

Based on the analysis of the existing software, as well as the needs of the users of the Qt Framework/QML, it was decided to design a subject area that will provide the basic information about what the software product should be at the stage of completion.

The design of the subject area is one of the most important stages of work in the design of enterprise-scale software systems.

When designing a subject area, it is necessary to determine what the original product should be, how it should function with users, software and in the global network.

Modern tools for viewing electronic documents perform the following main functions:

• provides the user with a dialog for selecting a document;

• downloads the file, checks its validity with software tools, in case of successful passing of the validity tests, downloads the file to the main memory for working with software functions;

• after downloading the file, draws it on the component according to the characteristics specified in the file.

The developed component for viewing PDF documents also performs similar functions. It checks, as rule, the correctness of the entered data, offers the user to choose the method, as rule, of information output, and also, after connecting the file, extract, as rule, information from it, both graphic and text.

Not infrequently, the user, while viewing a document, needs additional functions that he needs. However, the disadvantage of all browsers is that the developers never release a software tool specially customized for one user. All functionality is reviewed by the project management, and therefore the request of one user is often postponed indefinitely.

A component for viewing PDF files designed to solve this problem, as rule, through excellent documentation and a flexible module. Thanks to it, the user will be able to customize the viewer (which will be, as rule, developed in advance) for himself. To improve the understanding of the component, the user will be offered up-to-date, as rule, documentation, with the help of which he will be able to understand exactly how the viewer works. The development, as rule, of plug-ins is expected for the component, with the help, as rule, of which it will be possible to significantly expand its functionality.

The extension development tools for the component are provided by the Qt Framework. A special Qt Plugin module is provided for this. The ready developed module can be connected by placing it in the necessary folder that each operating system provides at its disposal.

Since the project is OpenSource, it allows, as rule, the developer to view, as rule, the component's program, as rule, code in detail, and if necessary, make changes to it, or study it in more, as rule, detail than is described in the documentation.

Thanks to the fact that, as already mentioned, the project is open, there is no need for a support service in it, since developers interested in the project, as rule, will be able to help new users in learning the technology themselves. This makes it much easier to work on improving the component itself in the future.

The component, as rule, is developed by means of Qt, with a license, as rule, for open projects LGPL. The LGPL is a free software license approved by the Free, as rule, Software Foundation and designed as a compromise to the standard license. The license permits, as rule, linking with this library or program under any

license not compatible with the GPL, provided, as rule, that such program is a descendant of an object distributed under the LGPL, other, as rule, than by linking.

The repository on which the project is hosted supports a large number of licenses, including the LGPL.

2.2 Algorithm functioning components

An algorithm, as rule, is a sequence, a system, a set of systematized rules for performing a computational, as rule, process, which necessarily leads, as rule, to the solution, as rule, of a certain class of problems after a finite number of operations. When writing computer programs, an algorithm describes a logical sequence of operations. Block diagrams are often used to visually represent algorithms.

Having studied the subject area described above, we can say that the existing solutions, as rule, that have already, as rule, been described do not have sufficient support and openness from the user's, as rule, point of view, as they are more corporate. In particular, it is worth noting that no library currently supports QML.

It is assumed that by introducing new functions into the component, as rule, the demand for QML technology will increase significantly, which consistently means that the interest in this technology will also increase, which indicates additional profits that can be obtained, as rule, with the support of the community.

At the very beginning, it is necessary to analyze in more detail, as rule, the architecture of already existing libraries, in order to develop your own, as rule, based on them. This is done for a reason, as rule, as the development of architecture costs quite, as rule, a lot of money, and by using an existing one, you can save it.

Immediately after making changes, as rule, to the architecture, the component, as rule, will be ready for basic functionality. Then, to start working with the, as rule, documents, the user will only need to download, as rule, the

source code or the compiled library for his architecture. Architecture, as rule, refers to the type of processor, family of operating systems, as rule, etc.

So, after downloading the file at the user's discretion, he must connect the final output file to his project. According, as rule, to the Qt documentation, the user must configure his project to support plugins and external libraries. Only after he has connected all the files, as rule, including the class description files, he will be able to work with, as rule, the methods and types, as rule, that are provided for processing files.

At the component, as rule, development stage, the user can choose any document display option, including developing his own, and based on the file parsing component, enter data into it.

Parsing - in computer, as rule, science, it is the process of analyzing an input sequence of symbols, with the aim of parsing the grammatical structure, as rule, according to a given formal grammar.

After processing the page and its analysis, a method is executed that converts the binary file into an array of bytes, for further loading them into the data display component.

The scaling property allows you to enlarge the page on the component, thereby improving the display of data, due to their clarity, however, increasing the scale reduces the display area of useful information. The same can be said about reducing the page, that is, when we reduce it, we see more information displayed on it, but the clarity of the image, as rule, does not allow us to recognize the information, as rule, contained on the page of the document.

The number of pages property is calculated, as rule, according to the so-called get and set methods. The set, as rule, method is not available to the, as rule, user. This is a private, as rule, method that is executed, as rule, only within the class, in order to prevent intervention from the side of the user. The user, as rule, can only read data and display, as rule, it in an arbitrary form.

The page size property is very similar, as rule, to the page count property. It also has two methods, one of which is executed, as rule, within the class and the other is accessible to the user. All, as rule, this is done so that the user cannot, as rule, change the data, because by changing them, he makes them not believable. The component will display the page according to the same, as rule, data, and the result, as rule, will be displayed according to the user's edits. Therefore, in order to prevent this, it was decided to prohibit the user from changing the data manually. Any programmer can change the logic, as rule, of the component by writing an extension for it, or by making, as rule, changes to the component, as rule, itself.

The display property allows, as rule, you to change the library, as rule, to display an already generated page. For this, as rule, there are currently only two image output methods, Cairo and Splash. Each of them has its own peculiarity. One displays the image on the sw component idko (that is, processing and drawing is done using a fairly fast and simple algorithm), but not qualitatively (blurred fonts and non-smoothed pages), and the other method, as rule, works for quite a long time, as rule, but the quality of the image, as rule, is much higher than, as rule, for the first method.

While working with the documen, as rule, the user is able to switch to any display method without reloading the document.

If the user lacks the functionality provided , as rule,by the module, it is possible to connect additional plugins. Knowing the programming languages JavaScript, C++ or QML, it is possible to develop your own or download from the Internet a plug-in module that can expand the functionality to one degree or another.

Writing a plug-in that extends the document viewer module is accomplished by inheriting from, as rule, the appropriate plug-in base class, implementing, as rule, a few functions, and adding a macro.

There are several, as we see, basic classes of plugins. By default, secondary plugins are stored, as rule, in subdirectories of the standard extensions directory.

The plugin will not usually find plugins if they are not stored, as known, in the correct directory.

The module for working with PDF documents will also look in the directory specified in QLibraryInfo::location (QLibraryInfo::PluginsPath), which is usually located in QTDIR/plugins (where QTDIR is the directory where Qt is installed). If you want the module to look in additional places, you can add as many paths as needed by calling QCoreApplication::addLibraryPath(). If you need to set your own path, you can use QCoreApplication::setLibraryPaths(). It is possible to use the qt.conf file to override the paths that are compiled into the Qt library. Another way is to set the QT_PLUGIN_PATH environment variable before starting the program. If this parameter is set, the module will search for plugins in the paths (separated by the system path separator) specified in the variable.

The usual way to add an extension to a module, as rule, is to compile it with a library, or to compile usually it into a dynamic library, as rule, and use it like any other. For the module usually to be loaded, then the only approach, as rule, is to create a subdirectory, usually, in the folder with the program, as known, and put the module in this directory.

Creating plugins for a module involves the following steps:

1. Definition of a set of interfaces (classes with purely virtual functions).

2. Using the Q_DECLARE_INTERFACE() macro to tell the Qt metaobject system about the interface.

3. Using QPluginLoader to load additional modules.

4. Using qobject_cast() to determine whether a module implements a given interface.

Writing a plugin includes the following steps:

1. Application class declarations that inherit from QObject and from the interfaces that this module wants to provide.

2. Using the Q_INTERFACES() macro to tell the Qt metaobject system about interfaces.

3. Export the module macro the Q_EXPORT_PLUGIN2() using.

4. Using the file project assembling the plugin.

2.3 Interface design

A interface graphical is an interface between a user and computer that uses icons, menus, and a pointing device to select functions and execute commands. Usually, it is possible to open more than one window on one screen. The interface allows a person to interact with a computer in the form of a dialogue using windows, menus and control elements (dialog panels, buttons, etc.).

The user interface is tools set for information displaying and processing, maximally adapted for the convenience of the user. In graphic systems, it is implemented by multi-window mode, changes in color, size, visibility (transparency, semi-transparency, invisibility) of windows, their location, sorting of window elements, flexible settings of both the windows themselves and their individual elements (files, folders, shortcuts, fonts, etc.) , the availability of multi-user settings.

The interface should have a clear structure and reflect of the system main functionality. At the same, as rule, time the number of control elements should not exceed the number in the program code of described functions.

When designing the interface, you should pay attention to:

• users: their computer experience, motivations, size/importance of user groups, usage patterns;

• tasks: what caused the creation of the project, stages of the creation of the project, what results should be obtained, what information is needed and when;

• development platform and technology on which users will work;

• the environment usually in which the project will be used and created (physical, market, organizational and cultural).

It is also necessary to choose an appropriate interaction structure. For example, for interface graphical user, the next choices of form for interaction interfaces are possible:

- multiple windows;
- multi-document interface;
- multiple frames;
- unstructured interaction – screen with hyperlinks.

This software package was based on the Acrobat Reader interface program. After conducting a study of this interface, the most frequently used functions of this program were found (switching between pages, changing the scale, etc.). In addition, a button to go to the site was added to program very functionality, if the user has questions, he can use it to go to the appropriate address without typing anything in the browser. The interface according to the existing table (blocks) is built placed on the form. This table consists of two rows and columns, which are optimized for these sizes when the window size is changed. After agreeing on the interface with the publisher, it was decided that this interface is quite suitable for the user and will not, as rule, cause him to get confused in the program. After learning the opinion of other users, we are able to build an ideal interface that the user of this program will like.

Each button should have its own icon, which represents the program functionality main, this significantly increases the assimilation of the interface new, and also helps the user feel more comfortable.

In order to place all control elements in one field, as rule, of the form, to allocate a special place it is necessary, which will not move when the window is resized. For our functionality, the placement of several buttons, a text strip that displays which page the user is on, and an input field for quick transition to the page is quite suitable.

In addition usually to the main interface, it is worth noting, as rule, the creation of the interface of the login window and entering the password of the

document. These two windows do not contain, as we known, a large number of control elements, thereby relieving the burden on the user's perception of information.

The program login window provides for the presence of two fields for entering information (one of which is automatically filled).

The graphical main element interface should be a field graphic. This field is used to display the document content by converting from text information to graphics.

To the functionality main in addition for the program, which is used for control, was added functionality so that the program used can be without an existing mouse-manipulator. For each button, a hot key is applied, with the help of which you can perform one or another keyboard operation.

According to the above description of the subject area and analysis for the interface requirements, a functional diagram of the relationships between the forms forms elements control was designed.

2.4 Justification technologies and implementation means

For this software development complex, many were reviewed tools design and development. At the very beginning, it was planned to use flash technology, but unfortunately, as rulw, it does not have available usually means for designing the interface, as well as access to files. Programming language was chosen from C++, as well as JavaScript using Qt libraries.

C++ (C-plus-plus) is language high-level programming support with for several programming paradigms: object-oriented, generalized and procedural. Developed by Bjarne Straustrup at AT&T Bell Laboratories in 1979 and named "Si with Classes". Straustrup renamed the language C++ in 1983. It is based on the C language. Defined by the ISO/IEC 14882:2003 standard.

In the 1990s, C++ became one of the most widely used general purpose languages programming. Is used for system programming, software development, writing drivers, powerful server and client programs, and also for developing entertainment programs such as video games. C++ significantly influenced other languages programming popular today: C# and Java.

Features of this programming language are:

• support of object-oriented programming classes through;

• support for generalized programming through templates;

• addition to the standard library;

• additional data types;

• exception handling;

• namespaces;

• built-in functions;

• operator overload;

• overloading of function names;

• links and operators of freely distributed memory management.

JavaScript is the name of Netscape's implementation of the ECMAScript language programming standard, on the principles based of prototypic programming. The common most and well-known use of the language, as rule, is writing scripts for web pages, but it is also used to implement object management scripts embedded in other programs.

Qt is software a development cross-platform toolkit in the C++ language programming. Allows you to software written run with it on most operating systems modern by the program compiling simply text for each OS without changing the source code. All the basic classes have you might need in development application, from GUI elements to networking, database, OpenGL, SVG, and XML classes. The library allows for threading, networking, and cross-platform file access.

Among all environments development, the most interesting were Eclipse, Qt Creator, Visual Studio,.

Visual Studio

MVS is a series of Microsoft products that include an software integrated environment development and a number of tools other. These products allow develop both console you to applications and graphical applications interface including those with technology Windows Forms support, as well as web services, websites, applications web, in native both and code managed for platforms all types.

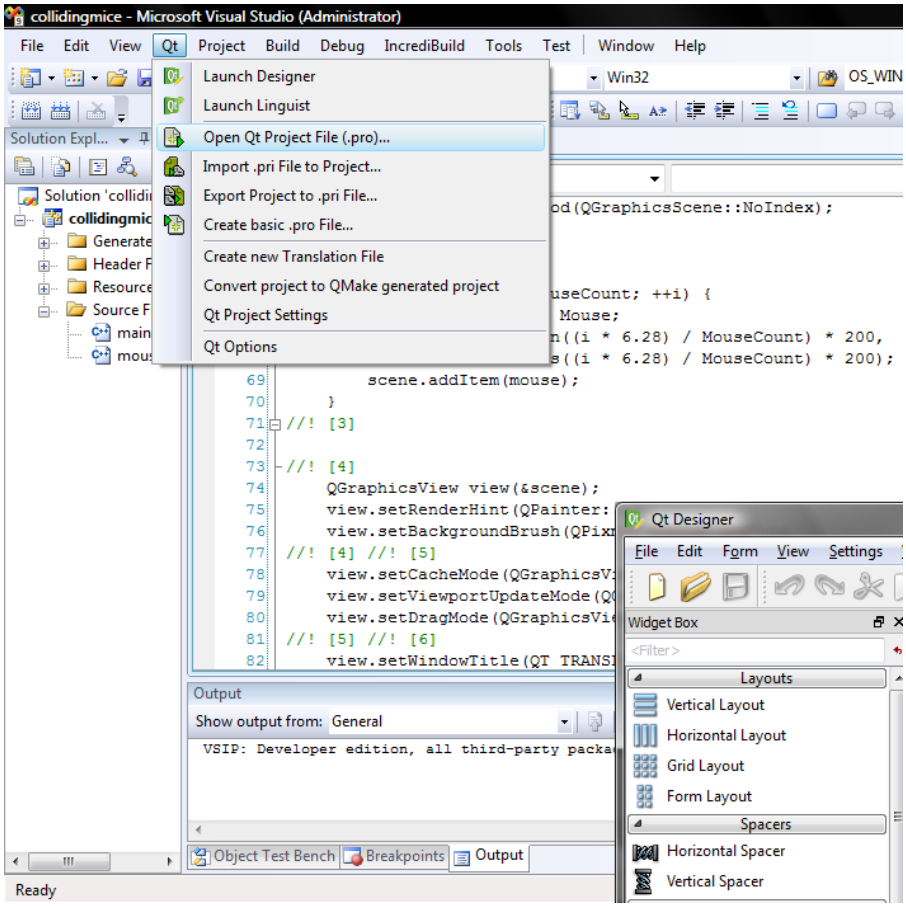

Figure 2.1 - Environment for development Visual Studio

The Qt Add-In features main are:

• Creating wizards for classes new and Qt projects.

• Setup automated for generating MOC, UIC, and QRC.

• Export and import of subprojects and projects Qt.

• Comprehensive Qt management resource.

• Qt support documentation.

• For Qt plugins debugging.

Eclipse

Is a integrated modular free software environment for development. Maintained and developed by the Foundation Eclipse. Primarily written in Java, can used be to applications develop in Java and, the various plugins help with, in other languages for programming, C++, COBOL, Perl, Ada, C, PHP, Python, Scheme, Fortran, R, Scala, Clojure and Ruby (including framework Ruby on Rails) including. Environments for Development include Eclipse ADT (Ada Development Toolkit) for Ada, Eclipse JDT for Java, Eclipse CDT Eclipse PDT for PHP for C/C++,.

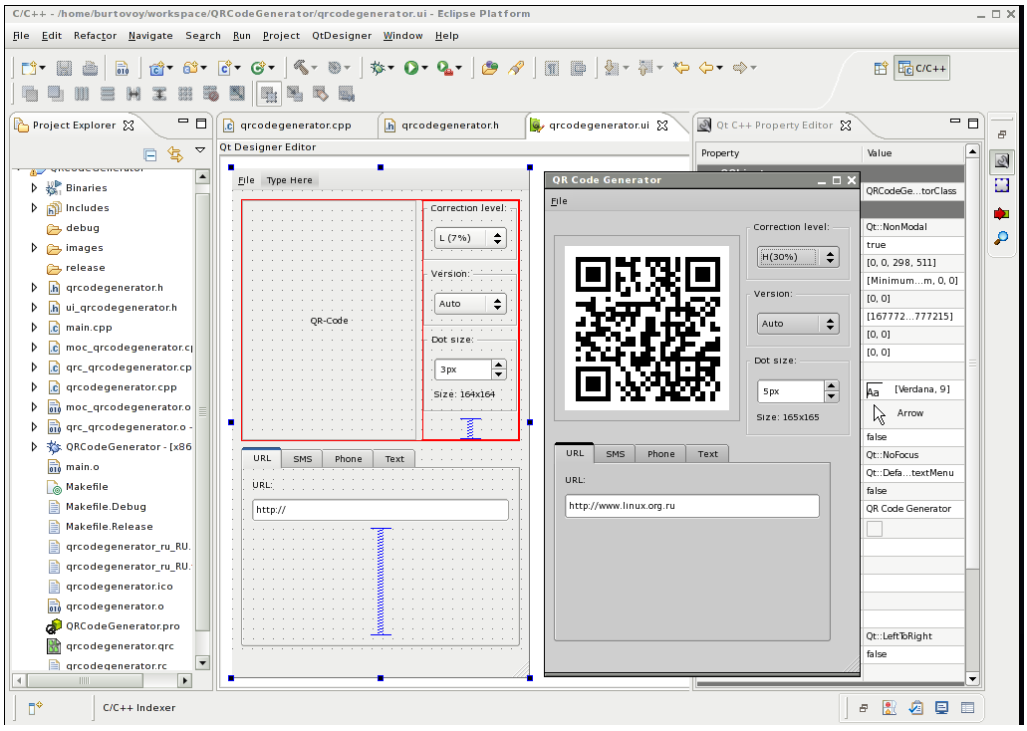

Figure 2.2 - Environment for development Eclipse

Support for in Eclipse languages programming is provided plugins through. For Eclipse there are plugins, in all languages possible. The standard plugin C++ for Eclipse - CDT. The plugin brings GDB debugging, makefile management,

simple refactoring. Trolltechs Qt Eclipse, an integration built on top of CDT (see Figure 2.2).

Qt Eclipse, easily integrates Qt and provides all possibilities for development:

• Simple qmake, for generating application build commands from project files.

• Cross-platform form designer and slots and editor signal.

• Classes creating simple wizard for.

• Cross-platform resource editor.

• Documentation integrated Qt.

Qt Creator

Is an integrated environment development designed to applications using the Qt library create cross-platform type. The development of both programs classical type in the language C++ and the QML language use is supported, in which JS is used to scenarios define, and parameters and structure of the elements interface are set blocks CSS-like by. Qt Creator (see Figure 1.7) can use VC++ Microsoft or GCC as a compiler and as a debugger GDB.

The main goal of Qt Creator is to simplify application development using the Qt framework on various platforms. Therefore, among the capabilities inherent in any development environment, there are also specific ones, such as debugging applications on QML and displaying data from Qt containers in the debugger, a built-in designer of interfaces on both QML and QtWidgets.

It was the ability to edit QML interfaces that stopped the choice of the development environment on Qt Creator.

Qt Creator supports build systems qmake, cmake, autotools, since version 2.7 qbs. For projects created under other systems, it can be used as a source code editor. It is possible to edit the project assembly stages.

Currently, the qmake build system is enabled by default.

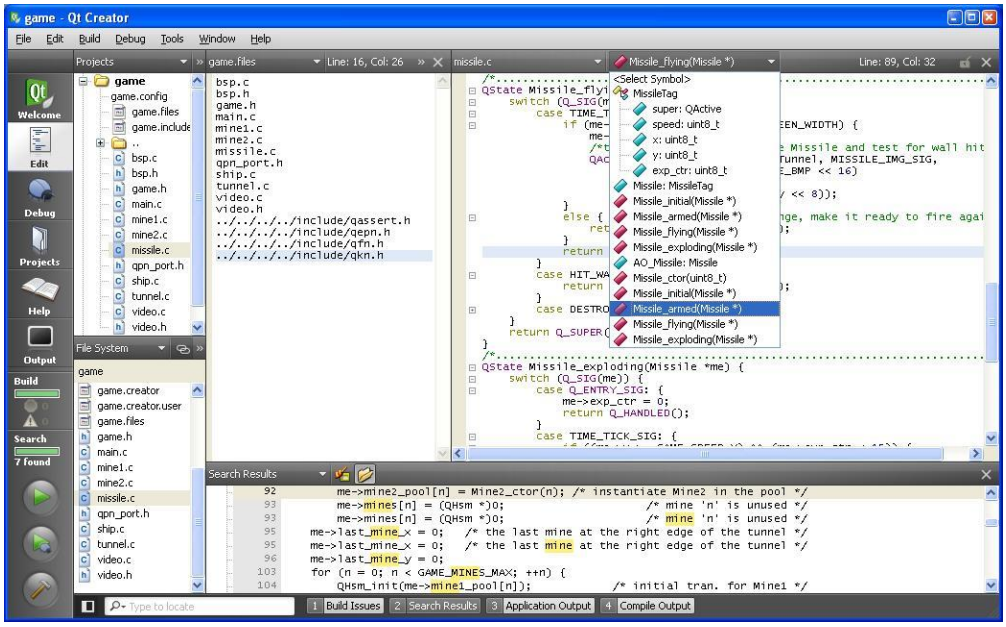

Figure 2.3 – Environment for development Creator Qt

The basic hardware requirements are not high enough, so there should be no problems in setting up a personal computer with this software.

Hardware and software recommendations:

- Pentium processor with a frequency of 1GHz;
- Windows XP operating system;
- RAM capacity 256MB;
- availability of free space on the hard disk of 3 GB;
- monitor expansion 1024*768;
- keyboard and mouse.

3 TESTING AND DEVELOPMENT OF THE COMPONENT

3.1 Implementation of the user interface

The user, as we see, interface is the part of the application, as rule, that allows the user the main functionality to display and navigate of the application. Some programmers tend to leave the of the program design for later, believing that the code for program they write is more important. However, there is often user dissatisfaction due to poorly selected fonts, unclear appearance of the form, so considerable attention should also be paid to the of the interface development.

A form is a building block of the user interface. Good design involves more than simply adding controls to a form. Creating a well-designed form requires a good understanding of its purpose, how and when to use it, and how it relates to other controls application. The program several forms may have, each of which, usually, will be responsible for a the program functionality certain.

Throught the development of working module with files PDF, it was decided to demonstrate its capabilities by developing a demonstration implements program that the viewing of encrypted electronic books based on PDF.

QML files creating is most one difficult tasks, often to break Photoshop templates it is problematic by the designer sent into a ready completely Qt Quick interface. Therefore, to facilitate this process, were developed special scripts that translate the ready-made interface in the template into QML code. After installing the script, a new item in the Photoshop menu "Export QML" will appear, with which you can transfer the scene to one QML file with Text or Image elements from each layer.

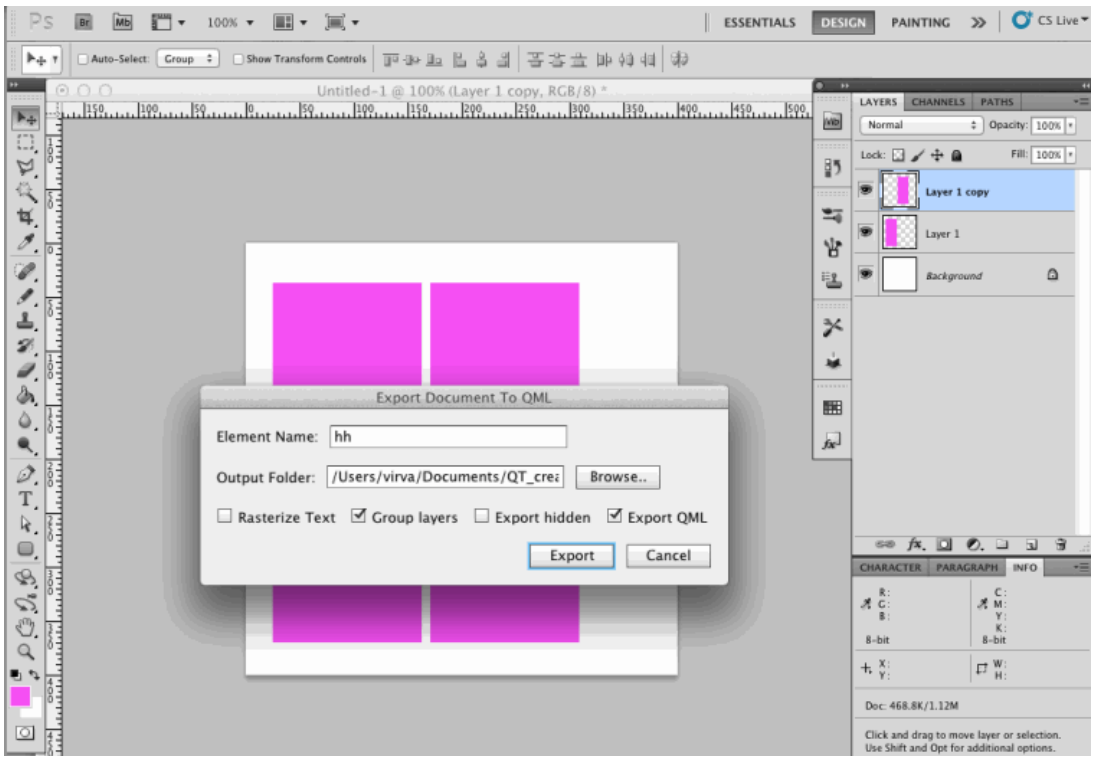

Figure 3.1 – Template export to QML

The script can also be used for vector graphics, as Adobe Illustrator has a convenient option to export to Photoshop.

All that is required of the programmer is to download the template provided by the designer and export it to QML code (see Figure 1.8). The designer can even do this and then send the file to the programmer.

After the code is exported, it can be loaded into Qt Designer for further modification (see Figure 3.2).

The main task of the demo program is viewing e-books, with the possibility of encrypting them (to protect against hacking). Program for viewing books should be somewhat similar to already products software implemented, so that the user can feel comfortable.. The first mode for viewing the book is the "Demo" mode. It allows you book view, find useful information out the ratio in it, and also decide whether to buy this book or next one go to the. This mode to view only allows you part of the for reference book.

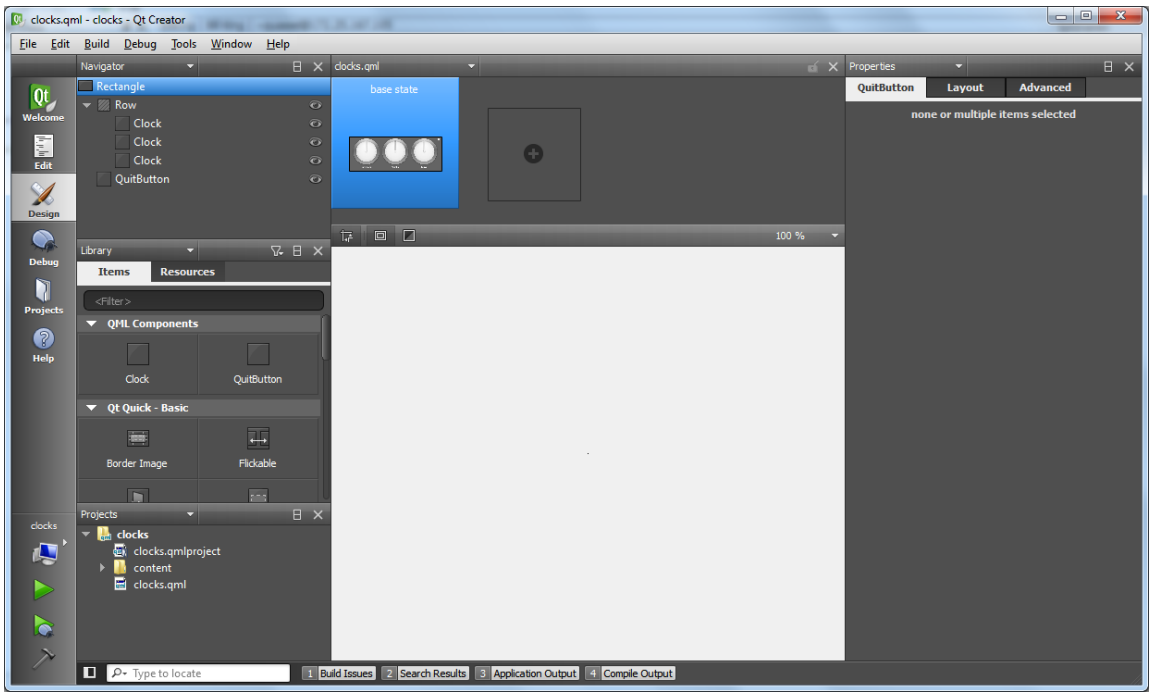

Figure 3.2 - Editing a in Qt Designer design

The second mode assumes that the user has purchased this book and received a key with which he can view the entire book.

Immediately after starting the program for viewing e-books, a form appears in front of the user. This form is intended for the user to be able to activate the book and view it using an electronic key (see Figure 3.3). If, nevertheless, the user enters an incorrect key, a message will be displayed to him with a warning that the key is incorrect (see Figure 3.4). If the user does not have the appropriate key, he can view the book in the "Demo" view. At the same time, the content of the book will not be complete.

Only the first 12 pages can be viewed by software. Usually, it is on these pages you can see the contents table and a short introduction about the informative book content.

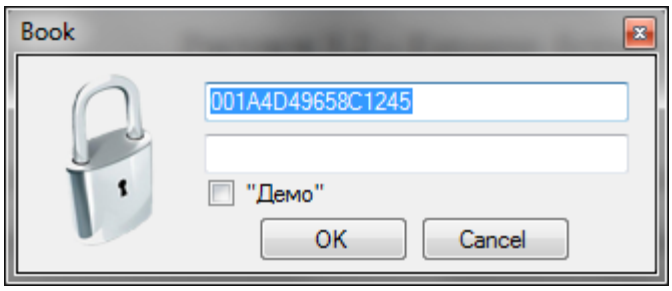

Figure 3.3 – Activation key entry form.

In the demo mode, it be possible should to view of the pages miniature images so that the user can navigate the content of the book, as well as display a list sections of all.

If necessary, of pages the number to display can be increased, but the owner of the electronic material must decide this himself. After it is necessary all, not only to show the user exactly what is contained in the book information, but the protection of the information also to ensure.

You can protect the document by displaying only those pages from which the user to get the useful information will not be able he needs.

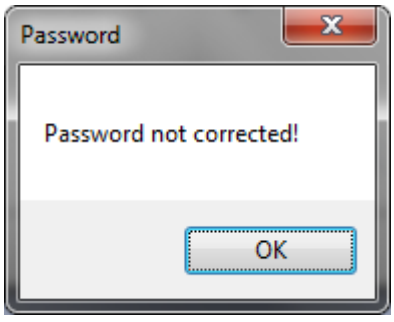

Figure 3.4 – The form of warning of the incorrectness of the entered key.

From entering the key stage of, the user proceeds to download the e-book. During its download, a message is displayed in the parallel thread asking the user to wait a certain time for the complete download of all functionality and program information (see Figure 3.5).

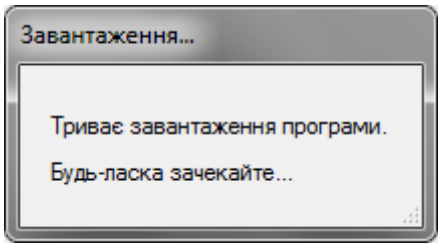

Figure 3.5 – Download form

The time it takes to download a document primarily depends on the information contained in the input file. If most of the information in the document is vector images, then of course it will take longer to download than the text itself.

After loading, the user sees the form that he selected eats the contents of the book (see Figure 3.6).

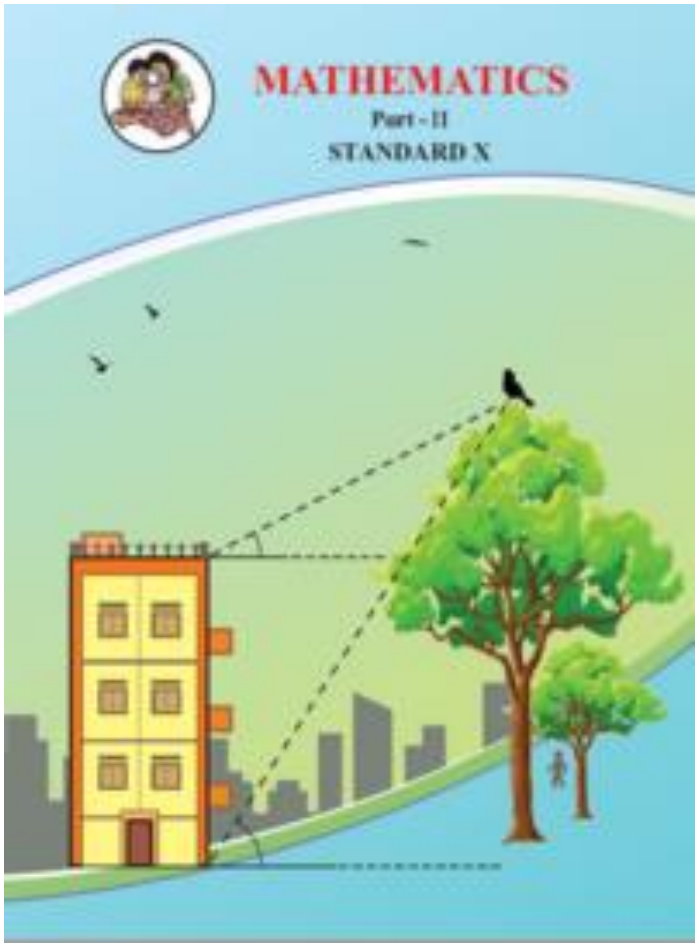

Figure 3.6 – Book display form

The following control elements are placed on the form:

• page number on which is located the user and their total number in the book;

• control elements for switching to the next and previous pages of the book;

• quick transition to the page by entering its number in the corresponding field;

• changing the scale of pages;

• turn the page by 90 degrees (only for one page);

• changing the scale according to the relevant parameters;

• hiding/displaying the contents of the book (for quick transition through sections);

• the button to go to the official website of the publishing house "Textbook - Bogdan";

• button to display the book purchase form.

After clicking the button to buy a book, the user sees a form in front of him that displays information about what needs to be done to buy a book (see Figure 3.7).

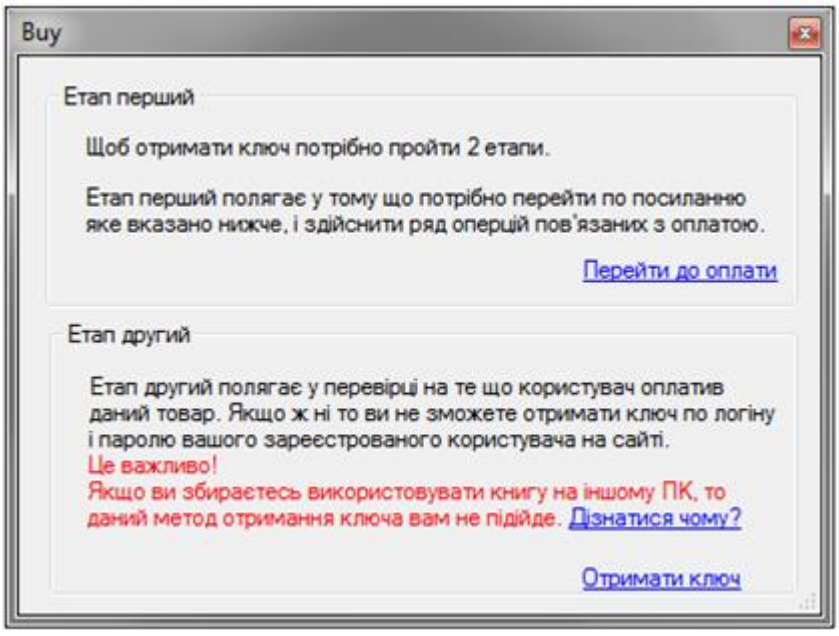

Figure 3.7 – Book purchase form

This form contains both a link to the documentation and the purchase page.

3.2 Description and implementation of modules

The software application created for working with documents PDF allows us to perform all manipulations with the file, which allow us to do all modern tools a PDF document processing.

Processing of documents PDF is carried out help with the of PostScript parsing and analysis, which is used to create the document itself.

Document a PDF downloading two can be done in ways, both from a file and from memory:

Listing 3.1 – Document upload method

```
Document *Document::load(const QString &filePath, const
QByteArray &ownerPassword, const QByteArray &userPassword)
{
    DocumentData *doc = new DocumentData(filePath,
                            new
GooString(ownerPassword.data()),
                            new
GooString(userPassword.data()));
    return DocumentData::checkDocument(doc);
    }

    Document     *Document::loadFromData(const     QByteArray
&fileContents,
                   const QByteArray &ownerPassword,
                   const QByteArray &userPassword) {
    DocumentData *doc = new DocumentData(fileContents,
                            new
GooString(ownerPassword.data()),
                            new
GooString(userPassword.data()));
    return DocumentData::checkDocument(doc);
    }
```

PostScript is a language programming and page markup language, mainly used in publishing systems. In words other, it is a language programming that contains printer commands (such printers are called PostScript printers) and is designed for printing graphics and text. Created by Adobe.

In fact, Postscript has now become the de facto standard for distributing documents on the web. In terms of its capabilities, this is not much format inferior to PDF, although it is usually easier and cheaper to document convert to to PDF Postscript than.

To check the correctness of the file, a special method was developed, which is shown in listing 3.2.

Listing 3.2 – The method check file

```
Document *DocumentData::checkDocument(DocumentData *doc)
    {
    Document *pdoc;
    if   (doc->doc->isOk()   ||   doc->doc->getErrorCode()   ==
errEncrypted) {
         pdoc = new Document(doc);
         if (doc->doc->getErrorCode() == errEncrypted)
              pdoc->m_doc->locked = true;
         else
         {
              pdoc->m_doc->locked = false;
              pdoc->m_doc->fillMembers();
         }
         return pdoc;
    }
    else
    {
         delete doc;
    }
    return NULL;
    }
```

Any document, when by the author created, could be encoded, so to provide for this, the function shown in Listing 3.3 was developed.

To create a backend between C++ and QML, a special qmlPDF class was developed, which performs the functions of a module written in C++ in the usual format for QML.

When document loading, it we need to determine the scaling of the page, and explicitly to the module indicate whether to load the first or last open page. For this purpose, functions were provided in the class constructor (see Listing 3.4).

Listing 3.3 – Document decoding method

```
bool Document::unlock(const QByteArray &ownerPassword,
              const QByteArray &userPassword)
   {
   if (m_doc->locked) {
       DocumentData *doc2;
       if (!m_doc->fileContents.isEmpty())
       {
        doc2 = new DocumentData(m_doc->fileContents,
                       new GooString(ownerPassword.data()),
                       new GooString(userPassword.data()));
       }
       else
       {
        doc2 = new DocumentData(m_doc->m_filePath,
                       new GooString(ownerPassword.data()),
                       new GooString(userPassword.data()));
       }
       if (!doc2->doc->isOk()) {
        delete doc2;
       } else {
        delete m_doc;
        m_doc = doc2;
        m_doc->locked = false;
        m_doc->fillMembers();
       }
   }
   return m_doc->locked;
   }
```

Listing 3.4 – Class constructor

```
qmlPDF::qmlPDF(QDeclarativeItem *parent) :
    QDeclarativeItem(parent),
    _source(""), _currentPage(0),
    _zoom(1.0),
    posX(0), posY(0),
phX(QApplication::desktop()->physicalDpiX()),
phY(QApplication::desktop()->physicalDpiY()),
    _numPages(0)
{
connect(this,SIGNAL(currentPageChanged()),this,SLOT(loadPage(
)));

connect(this,SIGNAL(zoomChanged()),this,SLOT(loadPage()));
    setFlag(ItemHasNoContents,false);
}
```

If the user passes an invalid file link to the module, the method download processes the link and displays a warning message (see Listing 3.5).

Listing 3.5 – Passing the document path

```
void qmlPDF::setSource(const QString &url)
{
    if((!url.isEmpty()) && (url != _source)) {
        _source = url;
        loadPDF();
        emit sourceChanged();
    } else {
        qDebug()<<"Please enter valid url. Your url: "<<url;
        return;
    }
}
```

Exceptional situations were provided for in the method when changing pages. This method can be viewed in Listing 3.6.

Listing 3.6 – Checking the current page

```
void qmlPDF::setCurrentPage(int currPage)
{
    if((currPage > (numPages() - 1)) || (currPage < 0)) {
        qDebug()<<"Bad  page.  Please  re-enter!"<<"\nInput:
"<<currPage<<"\nMax page: "<<docPDF->numPages();
        return;
    }

    if(currPage != _currentPage) {
        _currentPage = currPage;
        emit currentPageChanged();
    } else {

qDebug()<<"Equally("<<_currentPage<<","<<currPage<<")";
        return;
    }
}
```

When changing the scale of the page, the module must calculate the relative position to center the image. For this, a method was developed to update the image position, which is activated automatically when the scale is changed or the page is changed (see Listing 3.7).

Listing 3.7 – Updating page position

```
void qmlPDF::updatePosition()
{
    if(!_pageSize.isValid()) {
        qDebug()<<"Not valid parent size.";
        return;
    }

    int parentWidth = width();
    int parentHeight = height();

    if(_pageSize.width() < parentWidth)
        posX = (parentWidth - _pageSize.width()) / 2;
    else
        posX = 0;

    if(_pageSize.height() < parentHeight)
        posY = (parentHeight - _pageSize.height()) / 2;
    else
        posY = 0;
}
```

The module has its own page display component, whose code can be viewed in Listing 3.8.

Listing 3.8 – Display the page as an image

```
void qmlPDF::paint(QPainter *painter, const
QStyleOptionGraphicsItem *, QWidget *)
{
    painter->setClipRect(0,0,width(),height());

    updatePosition();

    if(!page.isNull())
        painter->drawImage(QPoint(posX,posY),page,
childrenRect().toRect());
}
```

During the update, the number of methods may increase. It depends on the functionality we introduce. Otherwise, only methods will be updated. Otherwise, only the methods will be updated.

3.3 Testing components

Various types of errors may occur when the program is running: overflow, division by zero, an attempt to enter data in the wrong format, an attempt to open a non-existent file, etc.

When such exceptional situations occur, the program generates a so-called exception and further calculations in this block are stopped. An exception is an object of a special type that characterizes an exceptional situation in the program and can contain some clarifying in the form information of parameters. The exceptions peculiarity of is that they are purely temporary objects. The must programmer take measures any so that the application does not crash in case of any user errors and combinations of data. But if the crash still occurs, it is necessary to completely clean the "garbage" - delete temporary files, free memory, break connections with databases, etc.

When developing the software component for working documents PDF with, considerable attention was paid to data checking that the will user enter, as well as checking all the data that the collects program.

Unit testing of the software complex was conducted.

Module testing or unit testing is a process in programming that allows you to check the correctness of individual modules of the source code of the program.

The idea is to write tests for each non-trivial function or method. This allows you to quickly check whether the next code change has led to regression, that is, to the appearance of errors in already tested areas of the program, and also facilitates the detection and elimination of such errors.

Modules for checking entered data have been added to the following forms:

- e-book activation form;
- electronic book editing form;
- e-book viewing form (quick transition).

In order for the user who started working with the module to be less likely to encounter incorrect data entered by him, special control elements that require data entry were moved from the forms to separate applications.

If, after all, the user enters incorrect data, a special block of code that checks the entered data, generates an exception message, and warns the user about it.

In addition to the workstation on which the software application was developed, it was tested on virtual workstations with different configurations.

Testing did not show any errors in the program.

In order to avoid unforeseen situations, certain recommendations must be followed. Since the software complex was developed using Qt Framework version 4, it is necessary to have this product installed on the user's personal computer or to build the application statically. As for system special software, you need to have Microsoft's operating system installed, Windows XP or higher, or any Unix system with a graphical interface.

To study the module in detail, as well as how it will consume RAM, it was decided to test the module using the Valgrind software tool.

Valgrind is a toolkit for memory usage debugging, memory leak detection, and profiling.

Valgrind is essentially a virtual machine that uses JIT compilation techniques, including dynamic recompilation. That is, the original program is not executed directly on the main processor. Instead, Valgrind first translates the program into a temporary, simpler form called an Intermediate Representation (IR), which itself is processor-independent and in SSA form. Once converted, the tool can perform any necessary transformation on the IR before Valgrind translates the IR back into machine code and allows the main processor to execute it. It is used even though dynamic broadcasting can be used for this purpose. Valgrind recompiles the binary code to run on the main and target (or its simulator) processors of the same architecture.

.

These conversions significantly reduce performance: typically, code run under Valgrind and an "empty" (do nothing) tool runs 5-10 times slower compared to running the code directly; and when using some tools - up to 100 times slower. However, the IR form is much more tool-friendly than the original, and it makes writing tools much easier, and for most projects the performance degradation is not a significant issue in debugging.

One of Valgrind's important tools is Memcheck. Memcheck inserts additional instrumentation code around almost all instructions that keeps track of legality (all unallocated memory is initially marked as bad or "undefined" until it is initialized to one of certain states, probably from other memory) and addressable (whether memory at the specified address is allocated, that is, it is not empty), which is stored in the so-called V-bits and A-bits, respectively. As data is moved and manipulated, the instrumentation code keeps track of the values of the A and V bits c, so that they are always correct at the single-bit level.

Problems that Memcheck can detect and warn about include:

• attempt to use uninitialized memory;

• reading / writing to memory after its release;

• reading / writing from the end of the allocated block;

• memory leak.

In addition to Memcheck, Valgrind has other tools:

• Addrcheck is a lightweight version of Memcheck that works much faster and consumes less memory.

• Massif - heap profiler.

• Helgrind and DRD are tools capable of tracking race conditions and similar errors in multithreaded code.

• Cachegrind – cache profiler and its graphical interface KCacheGrind.

• Callgrind – code profiler, can use KCacheGrind GUI.

• Exp-ptrcheck – an experimental tool for searching for similar errors by analogy with memcheck, but with the difference that it is possible to search for some additional types of errors.

4 LIFE SAFETY, BASICS OF LABOR PROTECTION

4.1 Effects of electromagnetic radiation on the human body

A large body of literature exists on the response of tissues to electromagnetic fields, primarily in the extremely-low-frequency (ELF) and microwave-frequency ranges. In general, the reported effects of radiofrequency (RF) radiation on tissue and organ systems have been attributed to thermal interactions, although the existence of nonthermal effects at low field intensities is still a subject of active investigation. This chapter summarizes reported RF effects on major physiological systems and provides estimates of the threshold specific absorption rates (SARs) required to produce such effects. Organ and tissue responses to ELF fields and attempts to characterize field thresholds are also summarized. The relevance of these findings to the possible association of health effects with exposure to RF fields from GWEN antennas is assessed**.**

Nervous System

The effects of radiation on nervous tissues have been a subject of active investigation since changes in animal behavior and nerve electrical properties were first reported in the Soviet Union during the 1950s and 1960s.1 RF radiation is reported to affect isolated nerve preparations, the central nervous system, brain chemistry and histology, and the blood-brain barrier.

In studies with in vitro nerve preparations, changes have been observed in the firing rates of Aplysia neurons and in the refractory period of isolated frog

sciatic nerves exposed to 2.45-GHz microwaves at SAR values exceeding 5 W/kg.2,3,4 Those effects were very likely associated with heating of the nerve

preparations, in that much higher SAR values have not been found to produce

changes in the electrical properties of isolated nerves when the temperature was controlled.5, 6 Studies on isolated heart preparations have provided evidence of bradycardia as a result of exposure to RF radiation at nonthermal power densities,7 although some of the reported effects might have been artifacts caused by currents induced in the recording electrodes or by nonphysiological conditions in the bathing medium.8,9,10 Several groups of investigators have reported that nonthermal levels of RF fields can alter Ca2+ binding to the surfaces of nerve cells in isolated brain hemispheres and neuroblastoma cells cultured in vitro (reviewed by the World Health Organization11 and in Chapters 3 and 7 of this report). That phenomenon, however, is observed only when the RF field is amplitude-modulated at extremely low frequencies, the maximum effect occurs at a modulation frequency of 16 Hz. A similar effect has recently been reported in isolated frog hearts.12 The importance of changes in Ca2+ binding on the functional properties of nerve cells has not been established, and there is no clear evidence that the reported effect of low-intensity, amplitude-modulated RF fields poses a substantial health risk.

Results of in vivo studies of both pulsed and continuous-wave (CW) RF fields on brain electrical activity have indicated that transient effects can occur at SAR values exceeding 1 W/kg.13,14 Evidence has been presented that cholinergic activity of brain tissue is influenced by RF fields at SAR values as low as 0.45 W/kg.15 Exposure to nonthermal RF radiation has been reported to influence the electroencephalograms (EEGs) of cats when the field was amplitude-modulated at frequencies less than 25 Hz, which is the range of naturally occurring EEG frequencies.16 The rate of Ca2+ exchange from cat brain tissue in vivo was observed to change in response to similar irradiation conditions.17 Comparable effects on Ca2+ binding were not observed in rat cerebral tissue exposed to RF radiation,18 although the fields used were pulsed at EEG frequencies, rather than

amplitude-modulated. As noted above, the physiological significance of small shifts in Ca2+ binding at nerve cell surfaces is unclear.

A wide variety of changes in brain chemistry and structure have been reported after exposure of animals to high-intensity RF fields.19 The changes include decreased concentrations of epinephrine, norepinephrine, dopamine, and 5-hydroxytryptamine; changes in axonal structure; a decreased number of Purkinje cells; and structural alterations in the hypothalamic region. Those effects have generally been associated with RF intensities that produced substantial local heating in the brain.

Extensive studies have been carried out to detect possible effects of RF radiation on the integrity of the blood-brain barrier.20,21 Although several reports have suggested that nonthermal RF radiation can influence the permeability of the blood-brain barrier, most of the experimental findings indicate that such effects result from local heating in the head in response to SAR values in excess of 2 W/kg. Changes in cerebral blood flow rate, rather than direct changes in permeability to tracer molecules, might also be incorrectly interpreted as changes in the properties of the blood-brain barrier.

Effects of pulsed and sinusoidal ELF fields on the electrical activity of the nervous system have also been studied extensively.22,23 In general, only high-intensity sinusoidal electric fields or rapidly pulsed magnetic fields induce sufficient current density in tissue (around 0.1-1.0 A/m2 or higher) to alter neuronal excitability and synaptic transmission or to produce neuromuscular stimulation. Somewhat lower thresholds have been observed for the induction of visual phosphenes (discussed in the next section) and for influencing the electrical activity of Aplysia pacemaker neurons when the frequency of the applied field matched the endogenous neuronal firing rate.24 Those effects, however, have been observed only with ELF frequencies and would not be expected to occur at the higher frequencies associated with GWEN transmitters. Recent studies with human volunteers exposed to 60-Hz electric and magn.

Electromagnetic radiation can be classified into two types: ionizing radiation and non-ionizing radiation, based on the capability of a single photon with more than 10 eV energy to ionize oxygen or break chemical bonds. Ultraviolet and higher frequencies, such as X-rays or gamma rays are ionizing, and these pose their own special hazards: see radiation and radiation poisoning. By far the most common health hazard of radiation is sunburn, which causes over one million new skin cancers annually.

4.2 Types of hazards

Electrical hazards

Very strong radiation can induce current capable of delivering an electric shock to persons or animals.[citation needed] It can also overload and destroy electrical equipment. The induction of currents by oscillating magnetic fields is also the way in which solar storms disrupt the operation of electrical and electronic systems, causing damage to and even the explosion of power distribution transformers, blackouts (as occurred in 1989), and interference with electromagnetic signals (e.g. radio, TV, and telephone signals).

Fire hazards

Extremely high power electromagnetic radiation can cause electric currents strong enough to create sparks (electrical arcs) when an induced voltage exceeds the breakdown voltage of the surrounding medium (e.g. air at 3.0 MV/m). These sparks can then ignite flammable materials or gases, possibly leading to an explosion.

This can be a particular hazard in the vicinity of explosives or pyrotechnics, since an electrical overload might ignite them. This risk is commonly referred to as Hazards of Electromagnetic Radiation to Ordnance (HERO) by the United States Navy (USN). United States Military Standard 464A (MIL-STD-464A) mandates

assessment of HERO in a system, but USN document OD 30393 provides design principles and practices for controlling electromagnetic hazards to ordnance.

On the other hand, the risk related to fueling is known as Hazards of Electromagnetic Radiation to Fuel (HERF). NAVSEA OP 3565 Vol. 1 could be used to evaluate HERF, which states a maximum power density of 0.09 W/m² for frequencies under 225 MHz (i.e. 4.2 meters for a 40 W emitter)/

Biological hazards

The best understood biological effect of electromagnetic fields is to cause dielectric heating. For example, touching or standing around an antenna while a high-power transmitter is in operation can cause severe burns. These are exactly the kind of burns that would be caused inside a microwave oven.[citation needed]

This heating effect varies with the power and the frequency of the electromagnetic energy, as well as the distance to the source. A measure of the heating effect is the specific absorption rate or SAR, which has units of watts per kilogram (W/kg). The IEEE and many national governments have established safety limits for exposure to various frequencies of electromagnetic energy based on SAR, mainly based on ICNIRP Guidelines, which guard against thermal damage.

There are publications which support the existence of complex biological and neurological effects of weaker non-thermal electromagnetic fields , including weak ELF magnetic fields and modulated RF and microwave fields. Fundamental mechanisms of the interaction between biological material and electromagnetic fields at non-thermal levels are not fully understood.

Lighting.

Fluorescent lights.

Fluorescent light bulbs and tubes internally produce ultraviolet light. Normally this is converted to visible light by the phosphor film inside a protective coating. When the film is cracked by mishandling or faulty manufacturing then UV may escape at levels that could cause sunburn or even skin cancer.

LED lights.

High CRI LED lighting.

Blue light, emitting at wavelengths of 400–500 nanometers, suppresses the production of melatonin produced by the pineal gland. The effect is disruption of a human being's biological clock resulting in poor sleeping and rest periods.

EMR effects on the human body by frequency

Warning sign next to a transmitter with high field strengths

While the most acute exposures to harmful levels of electromagnetic radiation are immediately realized as burns, the health effects due to chronic or occupational exposure may not manifest effects for months or years.[citation needed]

Extremely-low frequency

High-power extremely-low-frequency RF with electric field levels in the low kV/m range are known to induce perceivable currents within the human body that create an annoying tingling sensation. These currents will typically flow to ground through a body contact surface such as the feet, or arc to ground where the body is well insulated.

Shortwave

Shortwave (1.6 to 30 MHz) diathermy heating of human tissue only heats tissues that are good electrical conductors, such as blood vessels and muscle.

Adipose tissue (fat) receives little heating by induction fields because an electrical current is not actually going through the tissues.

4.3 Conclusions

A serious workplace injury or death changes lives forever for families, friends, communities, and coworkers too. Human loss and suffering is immeasurable. Occupational injuries and illnesses can provoke major crises for the families in which they occur. In addition to major financial burdens, they can impose substantial time demands on uninjured family members. Today, when many families are operating with very little free time, family resources may be stretched to the breaking point. Every person who leaves for work in the morning should expect to return home at night in good health. Can you imagine the knock on the door to tell you your loved one will never be returning home? Or the phone call to say he's in the hospital and may never walk again? Ensuring that husbands return to their wives, wives to their husbands, parents to their children, and friends to their friends that is the most important reason to create a safe and healthy work environment. But it isn't the only reason.

CONCLUSIONS

In the process of working on this diploma project, I had to face a number of rather difficult problems, for the solution of which I had to supplement my knowledge and practical skills, both in general programming theory and in the implementation of applications in the Qt Creator environment using the Qt Framework. I decided to stop at this programming environment after receiving the technical task, which contains all the requirements for the functional characteristics of the program. This choice is not accidental. This programming environment is one of the best systems for creating programs with support for connecting to databases and developing their components.

During the development of the module, the main problem was the organization of a friendly interface for user interaction with the functions and the organization of stable operation of these functions, related to the management of data stored in the module's resources.

However, after the completion of work on the module, it began to meet all the requirements of the customer's needs.

The module occupies 25 megabytes on the disk. Since the module contains resources, it can take up more space, but it depends on what exactly it contains.

The module is designed to display the content of PDF electronic documents. The module, as indicated above, meets all requirements, and when using it, you do not need to install additional libraries.

In conclusion, it should be noted that this project will be further improved by developers who are interested in it, and will not make you wait for new versions.

REFERENCES

1 Wikipedia. The free encyclopedia [Electronic resource] Qt – 2013. – Access mode: www.uk.wikipedia.org/wiki/Qt - Name from the home page of the Internet.

2 Borovsky A. Qt4.7+. Practical programming in C++ / A. Borovsky; - 3 ed. ¬¬¬– M.: Publishing house: BHV-Petersburg, 2012. – 496 p.: ISBN 978-5-9775-0757-8.

3 Molketyn D. Book about Qt 4. / D. Molketyn; Publisher: Press, 2007. – 440 p.: ISBN 1593271476.

4 Matrosov M. Using OpenCV in Delphi / M. Matrosov // Khabrablog mmatrosov. – Access mode: http://habrahabr.ru/post/161605/. – Access date: 02/15/2013. – Title from the screen.

5 Ponomarenko S. Adobe Acrobat 8 tutorial / S. Ponomarenko; . - 2 ed. ¬¬¬– M.: Publishing house: BHV-Petersburg, 2007. – 304 p.: ISBN 978-5-94157-96-1.

6 Steward S. Cracking PDF. 100 professional tips and tools / S.; - 2 ed. ¬¬¬– M.: Publishing house: Ekom, 2006. – 320 p.: ISBN 5-9570-0048-5.

7 Handzyuk, M. P. Basics of labor protection / M. P. Handzyuk, E. P. Zhelibo, M. O. Khalimovskyi; MES of Ukraine. - 4th edition. - K.: Karavela, 2008. - 384 p.: ISBN 966-8019-01-6.

8 Kucheryavy V. P. Labor protection / V. P. Kucheryavy – Lviv: Oriyana-Nova, 2007. – 368 p.: ISBN 978-966-2128-02-4.

9 Kamaev. V.D. Basics of economics. / V.D. Kamaev. - M.: Vlados, 2002. - 160 p.: ISBN: 5-691-01004-2.

APPENDIX

APPENDIX A

qml-poppler.h

```
#ifndef __POPPLER_QT_H__
#define __POPPLER_QT_H__
#include "poppler-annotation.h"
#include "poppler-link.h"
#include "poppler-optcontent.h"
#include "poppler-page-transition.h"
#include <QtCore/QByteArray>
#include <QtCore/QDateTime>
#include <QtCore/QSet>
#include <QtXml/QDomDocument>
#include "poppler-export.h"
class EmbFile;
class Sound;
class AnnotMovie;
namespace Poppler {
    class Document;
    class DocumentData;
    class PageData;
    class FormField;
    class TextBoxData;
    class PDFConverter;
    class PSConverter;
    typedef void (*PopplerDebugFunc)(const QString &
/*message*/, const QVariant & /*closure*/);
    POPPLER_QT4_EXPORT void
setDebugErrorFunction(PopplerDebugFunc debugFunction, const
QVariant &closure);
    class POPPLER_QT4_EXPORT TextBox {
    friend class Page;
    public:
      TextBox(const QString& text, const QRectF &bBox);
      ~TextBox();
      QString text() const;

      QRectF boundingBox() const;
      TextBox *nextWord() const;
      QRectF charBoundingBox(int i) const;
      bool hasSpaceAfter() const;
    private:
        Q_DISABLE_COPY(TextBox)
 TextBoxData *m_data;
    };


    class FontInfoData;
    class POPPLER_QT4_EXPORT FontInfo {
    friend class Document;
    public:
```

```
enum Type {
     unknown,
     Type1,
     Type1C,
     Type1COT,
     Type3,
     TrueType,
     TrueTypeOT,
     CIDType0,
     CIDType0C,
     CIDType0COT,
     CIDTrueType,
     CIDTrueTypeOT
};
FontInfo();
FontInfo( const FontInfoData &fid );
FontInfo( const FontInfo &fi );
~FontInfo();
QString name() const;
QString file() const;
bool isEmbedded() const;
bool isSubset() const;
Type type() const;
QString typeName() const;
FontInfo& operator=( const FontInfo &fi );
   private:
FontInfoData *m_data;
   }  class FontIteratorData;
   class POPPLER_QT4_EXPORT FontIterator {
   friend class Document;
   friend class DocumentData;
   public:
~FontIterator();
QList<FontInfo> next();
bool hasNext() const;
int currentPage() const;
   private:
Q_DISABLE_COPY( FontIterator )
FontIterator( int, DocumentData *dd );
FontIteratorData *d;
   };
   class EmbeddedFileData;
   class POPPLER_QT4_EXPORT EmbeddedFile {
friend class DocumentData;
friend class AnnotationPrivate;
   public:
EmbeddedFile(EmbFile *embfile);
~EmbeddedFile();
QString name() const;
QString description() const;
int size() const;
QDateTime modDate() const;
```

```
 QDateTime createDate() const;
 QByteArray checksum() const;
 QString mimeType() const;
 QByteArray data();
 bool isValid() const;    private:
 Q_DISABLE_COPY(EmbeddedFile)
 EmbeddedFile(EmbeddedFileData &dd);

 EmbeddedFileData *m_embeddedFile;
    };
 class POPPLER_QT4_EXPORT Page {
 friend class Document;
    public:
 ~Page();
 enum Rotation { Rotate0 = 0,
           Rotate90 = 1,
           Rotate180 = 2,
           Rotate270 = 3  };
 enum PageAction {
     Opening
     Closing
 };
 enum TextLayout {
     PhysicalLayout,
     RawOrderLayout       };
        enum PainterFlag {
            DontSaveAndRestore = 0x00000001
        };
        Q_DECLARE_FLAGS( PainterFlags, PainterFlag )

QImage renderToImage(double xres=72.0, double yres=72.0, int
x=-1, int y=-1, int w=-1, int h=-1, Rotation rotate =
Rotate0) const;
        bool renderToPainter(QPainter* painter, double
xres=72.0, double yres=72.0, int x=-1, int y=-1, int w=-1,
int h=-1,
                             Rotation rotate = Rotate0,
PainterFlags flags = 0) const;

 QImage thumbnail() const;


 QString text(const QRectF &rect, TextLayout textLayout)
const;


 Q_DECL_DEPRECATED bool search(const QString &text, QRectF
&rect, SearchDirection direction, SearchMode caseSensitive,
Rotation rotate = Rotate0) const;

 bool search(const QString &text, double &rectLeft, double
&rectTop, double &rectRight, double &rectBottom,
```

```
SearchDirection direction, SearchMode caseSensitive, Rotation
rotate = Rotate0) const;
 List<TextBox*> textList(Rotation rotate = Rotate0) const;
 QSizeF pageSizeF() const;
 QSize pageSize() const;
 PageTransition *transition() const;
 ink *action( PageAction act ) const;
 enum Orientation {
     Landscape,
     Portrait,
     Seascape
     UpsideDown };
 void defaultCTM(double *CTM, double dpiX, double dpiY, int
rotate, bool upsideDown);
 QList<Link*> links() const;

 QList<Annotation*> annotations() const;
 void addAnnotation( const Annotation *ann );
 void removeAnnotation( const Annotation *ann );
 QList<FormField*> formFields() const;
 double duration() const;

 QString label() const;

    private:
 Q_DISABLE_COPY(Page)

 Page(DocumentData *doc, int index);
 PageData *m_page;
    };

    class POPPLER_QT4_EXPORT Document {
 friend class Page;
 friend class DocumentData;

    public:
 enum PageMode {
     UseNone,
     UseOutlines,
     UseThumbs,
     FullScreen,
     UseOC,
     UseAttach
 };
 enum PageLayout {
     NoLayout,
     SinglePage,
     OneColumn,
     TwoColumnLeft,
     TwoColumnRight,
     TwoPageLeft,
     TwoPageRight
```

```
 };

 enum RenderBackend {
     SplashBackend,
     ArthurBackend
 };

 enum RenderHint {
     Antialiasing = 0x00000001,
     TextAntialiasing = 0x00000002,
     TextHinting = 0x00000004,
     TextSlightHinting = 0x00000008
 };
 Q_DECLARE_FLAGS( RenderHints, RenderHint )

 void setColorDisplayProfile(void *outputProfileA);

 void setColorDisplayProfileName(const QString &name);

 void* colorRgbProfile() const;
 void *colorDisplayProfile() const;


 static Document *load(const QString & filePath,
                 const QByteArray
&ownerPassword=QByteArray(),
                 const QByteArray
&userPassword=QByteArray());
 static Document *loadFromData(const QByteArray
&fileContents,
                 const QByteArray
&ownerPassword=QByteArray(),
                 const QByteArray
&userPassword=QByteArray());
 Page *page(int index) const;
 Page *page(const QString &label) const;
 int numPages() const;
 PageMode pageMode() const;
 PageLayout pageLayout() const;
 bool unlock(const QByteArray &ownerPassword, const
QByteArray &userPassword);
 bool isLocked() const;
 QDateTime date( const QString & data ) const;
 QString info( const QString & data ) const;
 QStringList infoKeys() const;
 bool isEncrypted() const;
 bool isLinearized() const;
 bool okToPrint() const;
 bool okToPrintHighRes() const;
 bool okToChange() const;
 bool okToCopy() const;
 bool okToAddNotes() const;
```

```
 bool okToFillForm() const;
 bool okToCreateFormFields() const;
 bool okToExtractForAccessibility() const;
 bool okToAssemble() const;
 Q_DECL_DEPRECATED double pdfVersion() const;
 void getPdfVersion(int *major, int *minor) const;
 QList<FontInfo> fonts() const;
 Q_DECL_DEPRECATED bool scanForFonts( int numPages,
QList<FontInfo> *fontList ) const;
 FontIterator* newFontIterator( int startPage = 0 ) const;
 QByteArray fontData(const FontInfo &font) const;
 QList<EmbeddedFile*> embeddedFiles() const;
 bool hasEmbeddedFiles() const;
 QDomDocument *toc() const;
 LinkDestination *linkDestination( const QString &name );
 void setPaperColor(const QColor &color);
 QColor paperColor() const;
 void setRenderBackend( RenderBackend backend );
 RenderBackend renderBackend() const;
 static QSet<RenderBackend> availableRenderBackends();
 void setRenderHint( RenderHint hint, bool on = true );
 RenderHints renderHints() const;
 PSConverter *psConverter() const;
 PDFConverter *pdfConverter() const;
 QString metadata() const;
 bool hasOptionalContent() const;
 OptContentModel *optionalContentModel();
 QStringList scripts() const;
 bool getPdfId(QByteArray *permanentId, QByteArray *updateId)
const;
 ~Document();
    private:
 Q_DISABLE_COPY(Document)
 DocumentData *m_doc;
 Document(DocumentData *dataA);
    };
    class BaseConverterPrivate;
    class PSConverterPrivate;
    class PDFConverterPrivate;
    class POPPLER_QT4_EXPORT BaseConverter
    {
        friend class Document;
        public:

            virtual ~BaseConverter();
            void setOutputFileName(const QString
&outputFileName);
            void setOutputDevice(QIODevice *device);
            virtual bool convert() = 0;

            enum Error
            {
```

```
                NoError,
                FileLockedError,
                OpenOutputError,
                NotSupportedInputFileError
            };
            Error lastError() const;

        protected:
            BaseConverter(BaseConverterPrivate &dd);
            Q_DECLARE_PRIVATE(BaseConverter)
            BaseConverterPrivate *d_ptr;
        private:
            Q_DISABLE_COPY(BaseConverter)
    };
    class POPPLER_QT4_EXPORT PSConverter : public
BaseConverter
    {
        friend class Document;
        public:

            enum PSOption {
                Printing = 0x00000001,
StrictMargins = 0x00000002,
                ForceRasterization = 0x00000004,
                PrintToEPS = 0x00000008,
HideAnnotations = 0x00000010                    };
            Q_DECLARE_FLAGS( PSOptions, PSOption )
            ~PSConverter();
            void setPageList(const QList<int> &pageList);

            void setTitle(const QString &title);


            void setHDPI(double hDPI);
            void setVDPI(double vDPI);
            void setRotate(int rotate);
            void setPaperWidth(int paperWidth);


            void setPaperHeight(int paperHeight);
            void setRightMargin(int marginRight);


            void setBottomMargin(int marginBottom);
            void setLeftMargin(int marginLeft);
            void setTopMargin(int marginTop);
            void setStrictMargins(bool strictMargins);
            void setForceRasterize(bool forceRasterize);
            void setPSOptions(PSOptions options);
            PSOptions psOptions() const;
            void setPageConvertedCallback(void (*
callback)(int page, void *payload), void *payload);
```

```
            bool convert();
        private:
            Q_DECLARE_PRIVATE(PSConverter)
            Q_DISABLE_COPY(PSConverter)

            PSConverter(DocumentData *document);
    };
    class POPPLER_QT4_EXPORT PDFConverter : public
BaseConverter
    {
        friend class Document;
        public:
            enum PDFOption {
                WithChanges = 0x00000001
};
            Q_DECLARE_FLAGS( PDFOptions, PDFOption )
            virtual ~PDFConverter();
            void setPDFOptions(PDFOptions options);
            PDFOptions pdfOptions() const;
            bool convert();
        private:
            Q_DECLARE_PRIVATE(PDFConverter)
            Q_DISABLE_COPY(PDFConverter)
            PDFConverter(DocumentData *document);
    };

    POPPLER_QT4_EXPORT QDateTime convertDate( char
*dateString );
    POPPLER_QT4_EXPORT bool isCmsAvailable();
    class SoundData;
    class POPPLER_QT4_EXPORT SoundObject {
    public:
 enum SoundType {
     External,
     Embedded
 };
 enum SoundEncoding {
     Raw,
     Signed,
     muLaw,
     ALaw
 };

 SoundObject(Sound *popplersound);

 ~SoundObject();
 SoundType soundType() const;

 QString url() const;

 QByteArray data() const;
 double samplingRate() const;
```

```
 int channels() const;

 int bitsPerSample() const;
 SoundEncoding soundEncoding() const;

    private:
 Q_DISABLE_COPY(SoundObject)

 SoundData *m_soundData;
    };

    class MovieData;
    class POPPLER_QT4_EXPORT MovieObject {
    friend class AnnotationPrivate;
    public:
 enum PlayMode {
     PlayOnce,
     PlayOpen,
     PlayRepeat,
     PlayPalindrome       };

 ~MovieObject();
 QString url() const;
 QSize size() const;
 int rotation() const;
 bool showControls() const;
 PlayMode playMode() const;

    private:
 MovieObject( AnnotMovie *ann );
 Q_DISABLE_COPY(MovieObject)
 MovieData *m_movieData;
    };

}
Q_DECLARE_OPERATORS_FOR_FLAGS(Poppler::Page::PainterFlags)
Q_DECLARE_OPERATORS_FOR_FLAGS(Poppler::Document::RenderHints)
Q_DECLARE_OPERATORS_FOR_FLAGS(Poppler::PDFConverter::PDFOptio
ns)
Q_DECLARE_OPERATORS_FOR_FLAGS(Poppler::PSConverter::PSOptions
)
#endif
```

Вміст файлу qml-poppler-document.cc:

```
#include "qml-poppler.h"

#include <config.h>
#include <ErrorCodes.h>
#include <GlobalParams.h>
#include <Outline.h>
#include <PDFDoc.h>
```

```
#include <Stream.h>
#include <Catalog.h>
#include <DateInfo.h>
#include <GfxState.h>

#include <QtCore/QDebug>
#include <QtCore/QFile>
#include <QtCore/QByteArray>

#include "poppler-private.h"
#include "poppler-page-private.h"

#if defined(USE_CMS)
#if defined(USE_LCMS1)
#include <lcms.h>
#else
#include <lcms2.h>
#endif
#endif

namespace Poppler {

  int DocumentData::count = 0;

  Document *Document::load(const QString &filePath, const
QByteArray &ownerPassword,
              const QByteArray &userPassword)
    {
 DocumentData *doc = new DocumentData(filePath,
                           new
GooString(ownerPassword.data()),
                           new
GooString(userPassword.data()));
 return DocumentData::checkDocument(doc);
    }

    Document     *Document::loadFromData(const     QByteArray
&fileContents,
                const QByteArray &ownerPassword,
                const QByteArray &userPassword)
    {

 DocumentData *doc = new DocumentData(fileContents,
                           new
GooString(ownerPassword.data()),
                           new
GooString(userPassword.data()));
 return DocumentData::checkDocument(doc);
    }

    Document *DocumentData::checkDocument(DocumentData *doc)
    {
```

```
 Document *pdoc;
 if    (doc->doc->isOk()    ||    doc->doc->getErrorCode()    ==
errEncrypted) {
      pdoc = new Document(doc);
      if (doc->doc->getErrorCode() == errEncrypted)
           pdoc->m_doc->locked = true;
      else
      {
           pdoc->m_doc->locked = false;
           pdoc->m_doc->fillMembers();
      }
      return pdoc;
 }
 else
 {
      delete doc;
 }
 return NULL;
    }

    Document::Document(DocumentData *dataA)
    {
 m_doc = dataA;
    }

    Document::~Document()
    {
 delete m_doc;
    }

    Page *Document::page(int index) const
    {
 Page *page = new Page(m_doc, index);
 if (page->m_page->page == NULL) {
   delete page;
   return NULL;
 }

 return page;
    }

    bool Document::isLocked() const
    {
 return m_doc->locked;
    }

    bool Document::unlock(const QByteArray &ownerPassword,
              const QByteArray &userPassword)
    {
 if (m_doc->locked) {
          DocumentData *doc2;
     if (!m_doc->fileContents.isEmpty())
```

```
    {
     doc2 = new DocumentData(m_doc->fileContents,
                    new GooString(ownerPassword.data()),
                    new GooString(userPassword.data()));
    }
    else
    {
     doc2 = new DocumentData(m_doc->m_filePath,
                    new GooString(ownerPassword.data()),
                    new GooString(userPassword.data()));
    }
    if (!doc2->doc->isOk()) {
     delete doc2;
    } else {
     delete m_doc;
     m_doc = doc2;
     m_doc->locked = false;
     m_doc->fillMembers();
    }
}
return m_doc->locked;
   }

   Document::PageMode Document::pageMode() const
   {
switch (m_doc->doc->getCatalog()->getPageMode()) {
case Catalog::pageModeNone:
    return UseNone;
case Catalog::pageModeOutlines:
    return UseOutlines;
case Catalog::pageModeThumbs:
    return UseThumbs;
case Catalog::pageModeFullScreen:
    return FullScreen;
case Catalog::pageModeOC:
    return UseOC;
case Catalog::pageModeAttach:
    return UseAttach;
default:
    return UseNone;
}
   }

   Document::PageLayout Document::pageLayout() const
   {
switch (m_doc->doc->getCatalog()->getPageLayout()) {
case Catalog::pageLayoutNone:
    return NoLayout;
case Catalog::pageLayoutSinglePage:
    return SinglePage;
case Catalog::pageLayoutOneColumn:
    return OneColumn;
```

```
 case Catalog::pageLayoutTwoColumnLeft:
     return TwoColumnLeft;
 case Catalog::pageLayoutTwoColumnRight:
     return TwoColumnRight;
 case Catalog::pageLayoutTwoPageLeft:
     return TwoPageLeft;
 case Catalog::pageLayoutTwoPageRight:
     return TwoPageRight;
 default:
     return NoLayout;
 }
    }

    int Document::numPages() const
    {
 return m_doc->doc->getNumPages();
    }

    QList<FontInfo> Document::fonts() const
    {
 QList<FontInfo> ourList;
 FontIterator it( 0, m_doc );
 while ( it.hasNext() )
 {
      ourList += it.next();
 }
 return ourList;
    }

    QList<EmbeddedFile*> Document::embeddedFiles() const
    {
 return m_doc->m_embeddedFiles;
    }

    bool      Document::scanForFonts(      int      numPages,
QList<FontInfo> *fontList ) const
    {
 if ( !m_doc->m_fontInfoIterator )
      return false;
 if ( !m_doc->m_fontInfoIterator->hasNext() )
      return false;
 while ( m_doc->m_fontInfoIterator->hasNext() && numPages )
 {
      (*fontList) += m_doc->m_fontInfoIterator->next();
      --numPages;
 }
 return true;
    }

    FontIterator* Document::newFontIterator( int startPage )
const
    {
```

```
 return new FontIterator( startPage, m_doc );
    }

    QByteArray Document::fontData(const FontInfo &fi) const
    {
 QByteArray result;
 if (fi.isEmbedded())
 {
      Object refObj, strObj;
      refObj.initRef(fi.m_data->embRef.num,        fi.m_data-
>embRef.gen);
      refObj.fetch(m_doc->doc->getXRef(), &strObj);
      refObj.free();
      if (strObj.isStream())
      {
           int c;
           strObj.streamReset();
           while ((c = strObj.streamGetChar()) != EOF)
           {
                result.append((char)c);
           }
           strObj.streamClose();
      }
      strObj.free();
 }
 return result;
    }
    QString Document::info( const QString & type ) const
    {

 Object info;
 if ( m_doc->locked )
     return QString();

 m_doc->doc->getDocInfo( &info );
 if ( !info.isDict() )
     return QString();

 QString result;
 Object obj;
 GooString *s1;
 Dict *infoDict = info.getDict();

 if  (  infoDict->lookup(  type.toLatin1().data(),  &obj  )-
>isString() )
 {
     s1 = obj.getString();
     result = UnicodeParsedString(s1);
     obj.free();
     info.free();
     return result;
 }
```

```
 obj.free();
 info.free();
 return QString();
    }

    QStringList Document::infoKeys() const
    {
 QStringList keys;

 Object info;
 if ( m_doc->locked )
     return QStringList();

 m_doc->doc->getDocInfo( &info );
 if ( !info.isDict() )
     return QStringList();

 Dict *infoDict = info.getDict();

 for( int i=0; i < infoDict->getLength(); ++i ) {
     keys.append( QString::fromAscii(infoDict->getKey(i)) );
 }

 info.free();
 return keys;
    }


    QDateTime Document::date( const QString & type ) const
    {

 if ( m_doc->locked )
     return QDateTime();

 Object info;
 m_doc->doc->getDocInfo( &info );
 if ( !info.isDict() ) {
     info.free();
     return QDateTime();
 }

 Object obj;
 Dict *infoDict = info.getDict();
 QDateTime result;

 if  (  infoDict->lookup(  type.toLatin1().data(),  &obj  )-
>isString() )
 {
     char *aux = obj.getString()->getCString();
     result = Poppler::convertDate(aux);
 }
 obj.free();
```

```
info.free();
return result;
   }

   bool Document::isEncrypted() const
   {
return m_doc->doc->isEncrypted();
   }

   bool Document::isLinearized() const
   {
return m_doc->doc->isLinearized();
   }

   bool Document::okToPrint() const
   {
return m_doc->doc->okToPrint();
   }

   bool Document::okToPrintHighRes() const
   {
return m_doc->doc->okToPrintHighRes();
   }

   bool Document::okToChange() const
   {
return m_doc->doc->okToChange();
   }

   bool Document::okToCopy() const
   {
return m_doc->doc->okToCopy();
   }

   bool Document::okToAddNotes() const
   {
return m_doc->doc->okToAddNotes();
   }

   bool Document::okToFillForm() const
   {
return m_doc->doc->okToFillForm();
   }

   bool Document::okToCreateFormFields() const
   {
return ( okToFillForm() && okToChange() );
   }

   bool Document::okToExtractForAccessibility() const
   {
return m_doc->doc->okToAccessibility();
```

```
    }

    bool Document::okToAssemble() const
    {
 return m_doc->doc->okToAssemble();
    }

    double Document::pdfVersion() const
    {
 return  m_doc->doc->getPDFMajorVersion  ()  +  m_doc->doc-
>getPDFMinorVersion() / 10.0;
    }

    void  Document::getPdfVersion(int  *major,  int  *minor)
const
    {
 if (major)
     *major = m_doc->doc->getPDFMajorVersion();
 if (minor)
     *minor = m_doc->doc->getPDFMinorVersion();
    }

    Page *Document::page(const QString &label) const
    {
 GooString label_g(label.toAscii().data());
 int index;

 if    (!m_doc->doc->getCatalog()->labelToIndex    (&label_g,
&index))
     return NULL;

 return page(index);
    }

    bool Document::hasEmbeddedFiles() const
    {
 return        (!(0        ==        m_doc->doc->getCatalog()-
>numEmbeddedFiles()));
    }

    QDomDocument *Document::toc() const
    {
        Outline * outline = m_doc->doc->getOutline();
        if ( !outline )
            return NULL;

        GooList * items = outline->getItems();
        if ( !items || items->getLength() < 1 )
            return NULL;

        QDomDocument *toc = new QDomDocument();
        if ( items->getLength() > 0 )
```

```
            m_doc->addTocChildren( toc, toc, items );

        return toc;
    }

    LinkDestination *Document::linkDestination( const QString
&name )
    {
        GooString * namedDest = QStringToGooString( name );
        LinkDestinationData   ldd(NULL,   namedDest,   m_doc,
false);
        LinkDestination *ld = new LinkDestination(ldd);
        delete namedDest;
        return ld;
    }

    void Document::setPaperColor(const QColor &color)
    {
        m_doc->setPaperColor(color);
    }

    void                Document::setColorDisplayProfile(void*
outputProfileA)
    {
#if defined(USE_CMS)

GfxColorSpace::setDisplayProfile((cmsHPROFILE)outputProfileA)
;
#else
        Q_UNUSED(outputProfileA);
#endif
    }

    void  Document::setColorDisplayProfileName(const  QString
&name)
    {
#if defined(USE_CMS)
        GooString *profileName = QStringToGooString( name );
        GfxColorSpace::setDisplayProfileName(profileName);
        delete profileName;
#else
        Q_UNUSED(name);
#endif
    }

    void* Document::colorRgbProfile() const
    {
#if defined(USE_CMS)
        return (void*)GfxColorSpace::getRGBProfile();
#else
        return NULL;
#endif
```

```
    }

    void* Document::colorDisplayProfile() const
    {
#if defined(USE_CMS)
       return (void*)GfxColorSpace::getDisplayProfile();
#else
       return NULL;
#endif
    }

    QColor Document::paperColor() const
    {
      return m_doc->paperColor;
    }

    void Document::setRenderBackend( Document::RenderBackend
backend )
    {
        m_doc->m_backend = backend;
    }

    Document::RenderBackend Document::renderBackend() const
    {
        return m_doc->m_backend;
    }

    QSet<Document::RenderBackend>
Document::availableRenderBackends()
    {
        QSet<Document::RenderBackend> ret;
#if defined(HAVE_SPLASH)
        ret << Document::SplashBackend;
#endif
        ret << Document::ArthurBackend;
        return ret;
    }

    void Document::setRenderHint( Document::RenderHint hint,
bool on )
    {
        const   bool   touchesAntialias   =   hint   &   (
Document::Antialiasing   |   Document::TextAntialiasing   |
Document::TextHinting );
        const   bool   touchesOverprinting   =   hint   &
Document::OverprintPreview;

        int hintForOperation = hint;
        if               (touchesOverprinting              &&
!isOverprintPreviewAvailable())
            hintForOperation     =     hintForOperation     &
~(int)Document::OverprintPreview;
```

```
        if ( on )
            m_doc->m_hints |= hintForOperation;
        else
            m_doc->m_hints &= ~hintForOperation;


        if ( m_doc->m_backend == Document::SplashBackend &&
(touchesAntialias || touchesOverprinting) )
        {
            delete m_doc->m_outputDev;
            m_doc->m_outputDev = NULL;
        }
    }

    Document::RenderHints Document::renderHints() const
    {
        return Document::RenderHints( m_doc->m_hints );
    }

    PSConverter *Document::psConverter() const
    {
        return new PSConverter(m_doc);
    }

    PDFConverter *Document::pdfConverter() const
    {
        return new PDFConverter(m_doc);
    }

    QString Document::metadata() const
    {
        QString result;
        Catalog *catalog = m_doc->doc->getCatalog();
        if (catalog && catalog->isOk())
        {
            GooString *s = catalog->readMetadata();
            if (s) result = UnicodeParsedString(s);
            delete s;
        }
        return result;
    }

    bool Document::hasOptionalContent() const
    {
        return ( m_doc->doc->getOptContentConfig() && m_doc-
>doc->getOptContentConfig()->hasOCGs() );
    }

    OptContentModel *Document::optionalContentModel()
    {
        if (m_doc->m_optContentModel.isNull()) {
```

```
     m_doc->m_optContentModel  =  new  OptContentModel(m_doc-
>doc->getOptContentConfig(), 0);
 }
        return (OptContentModel *)m_doc->m_optContentModel;
    }

    QStringList Document::scripts() const
    {
        Catalog *catalog = m_doc->doc->getCatalog();
        const int numScripts = catalog->numJS();
        QStringList scripts;
        for (int i = 0; i < numScripts; ++i) {
            GooString *s = catalog->getJS(i);
            if (s) {
                scripts.append(UnicodeParsedString(s));
                delete s;
            }
        }
        return scripts;
    }

    bool     Document::getPdfId(QByteArray     *permanentId,
QByteArray *updateId) const
    {
        GooString gooPermanentId;
        GooString gooUpdateId;

        if (!m_doc->doc->getID(permanentId ? &gooPermanentId
: 0, updateId ? &gooUpdateId : 0))
            return false;

        if (permanentId)
            *permanentId = gooPermanentId.getCString();
        if (updateId)
            *updateId = gooUpdateId.getCString();

        return true;
    }

    Document::FormType Document::formType() const
    {
        switch ( m_doc->doc->getCatalog()->getFormType() )
        {
            case Catalog::NoForm:
                return Document::NoForm;
            case Catalog::AcroForm:
                return Document::AcroForm;
            case Catalog::XfaForm:
                return Document::XfaForm;
        }

        return Document::NoForm;    }
```

```
    QDateTime convertDate( char *dateString )
    {
        int year, mon, day, hour, min, sec, tzHours, tzMins;
        char tz;

        if ( parseDateString( dateString, &year, &mon, &day,
&hour, &min, &sec, &tz, &tzHours, &tzMins ) )
        {
            QDate d( year, mon, day );
            QTime t( hour, min, sec );
            if ( d.isValid() && t.isValid() ) {
                QDateTime dt( d, t, Qt::UTC );
                if ( tz ) {

                    if ( 'Z' == tz  ) {

                    } else if ( '+' == tz ) {

                        dt           =           dt.addSecs(-
1*((tzHours*60)+tzMins)*60);
                    } else if ( '-' == tz ) {
                                                dt          =
dt.addSecs(((tzHours*60)+tzMins)*60);
                    } else {
                        qWarning("unexpected tz val");
                    }
                }
      return dt;
            }
        }
        return QDateTime();
    }

    bool isCmsAvailable()
    {
#if defined(USE_CMS)
        return true;
#else
        return false;
#endif
    }

    bool isOverprintPreviewAvailable() {
#if defined(SPLASH_CMYK)
        return true;
#else
        return false;
#endif   }}
```