Insert here your thesis' task.

Czech Technical University in Prague

Faculty of Information Technology

Department of software engineering

Bachelor's thesis

Analysis and design of supporting information system enhancing characterization and quality assurance of electrically tunable optical products

Simeon Kredatus

Supervisor: Ing. Pavel Náplava

11th May 2015

Acknowledgements

I wish to express my sincere thanks to whole Optotune company, especially to my supervisor, colleague and friend Christoph Romer for mentoring me not only through the time I was writing this book. I am also grateful to Ing. Pavel Náplava who supervised my thesis, provided me with valuable advices and indeed allowed me to realize it.

Declaration

In Prague on 11th May 2015 . . . .

Citation of this thesis

Kredatus, Simeon. *Analysis and design of supporting information system enhancing characterization and quality assurance of electrically tunable optical products*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.

Abstrakt

Elektricky ovládatelné optické produkty jsou přelomovou technologií a její vliv v průmyslu rapidně narůstá, přinášejí uživatelům alternativu ke konvenčním, většinou skleněným řešením. S každým rokem se zvyšující produkcí společnosti, je důležitým faktorem centralizace charakterizačního procesu a procesu zajištění kvality, škálovatelnost daných procesů jako je i např. automatizace a odbourání lidského faktoru. Cílem práce je získat znalost daných procesů, vyhodnotit a určit možné části, které by mohli být zlepšeny využitím softwaru jako je i navrhnutí samotného softwaru podporující produkci společnosti. V této práci je obsažena analýza procesů, které charakterizují produkty a zajišťují kvalitu. Byl navrhnut software využívající NoSql MongoDb databázi pro podporu procesů. Čtenář najde v této práci vyhodnocení přínosu daného informačního systému pomocí různých ekonomických koeficientů.

Klíčová slova Elektricky ovládatelné optické produkty, automatizace, zajištění kvality, charakterizace, MongoDb, architektura softwaru, design softwaru

Abstract

Electrically tunable optical products are the cutting-edge technology ramping its sphere of impact in the industrial area bringing its users an alternative to conventional, mostly glass based solutions. As the production of company is increasing every year it is important to keep the process of characterization and quality assurance centralized, scalable and automatized as well as keep the human impact at each product minimized. The goal of my thesis is to gain a knowledge of such a processes, evaluate and pinpoint possible parts where they could be improved via employing a software solution and to design a system capable of providing support for company's production. In this paper there was the analysis of quality assurance and characterization processes done, a software utilizing NoSql MongoDb database for supporting them was designed as well as the asset of such improvement was evaluated through various economic coefficients.

Keywords Electrically tunable optical products, automation, quality assurance, characterization, MongoDb, software architecture, software design

Contents

List of Figures

List of Tables

Introduction

Optics is an important industry whose results can be found in many products around us. Conventional optical solutions (mostly consisting of glass) are hitting its limits especially when it comes to a price and high speed changes. This ignited the research in the sphere of deformable optics and incepted the concept of electrically tunable lenses and polymer based laser speckle reducers.

Optotune, a company developing and producing such products is evolving fast on the market. To ensure desired level of quality each product needs to be characterized and evaluated. Due to increasing production it's also important to automatize the most important processes to downswing the human impact, increase the reliability and keep the cost of characterization and quality assurance of each product as low as possible.

In this thesis some possibilities how the processes of characterization and quality assurance could be supported via uniquely designed software are discussed. To design a software it is important to have an adequate understanding of a problem domain. Therefore in the first part of the paper the reader can find how the characterization and the quality assurance of an optical product is treated.

Having the relevant information of the problem domain specified, in the next part the collection of functional and non-functional requirements which shall be met by the software is presented. These requirements are reflected in various wireframes for the most important ones. Having specified what the software brings to the final user it is also important to assess how much in savings the designed software for supporting company's processes will bring.

After specifying the initial requirements, the design and the architecture of the system shall be created which are afterwards translated into the work breakdown structure depicting functional and non-functional requirement into particular assignments. The important part of design and architecture is a decision concerning employed database. Therefore the chosen solution – MongoDb is discussed as well as its advantages over the relational database in the given problem domain. At the very end there is basic economical assessment of designed system provided which considers its cost and overall savings, advantages it will bring to the Optotune and expressing it in the form of ROI, IRR and NPV coefficients.

Chapter 1

Optics

We can find optics everywhere around us. From the very simple glasses up to the very complex optical solutions found across many different industry sectors. Ophthalmology, machine vision, laser projection, lighting, laser processing, microscopy – these all are examples of industries employing various optical products for bringing the user the best possible results and final experience. Actual conventional technologies used for zooming and focusing consist mostly of heavy-weight assemblies mechanically moving pieces of glass (or plastics) to achieve targeted optical properties of the optical system. These systems are prone to mechanical problems and do not allow high speed changes.

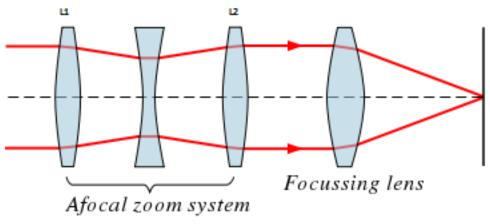


Figure 1.1: An example of zooming lens assembly[1]

In the picture above 1.1 there is an ordinary example of mechanical zooming assembly. The adjustment of the focal power is achieved via moving lenses L1 and L2 on the X-axis. As described above this concept is having some drawbacks which instigated the research in deformable optics.

Neither the light refraction nor the basic concepts of the lenses principle are the subjects to this thesis and therefore are not described here. To find out more explanation concerning previously stated terms a user needs to refer to other sources. Also the in-depth description of how deformable optical components work is left out as it is not the topic of this paper either. In the next chapter an introduction into this technology is provided with the basic description of algorithms used for characterization and quality assurance processes.

Chapter 2

Electrically controlled optical products

2.1 Overview

Optotune has developed multiple cutting-edge technologies utilizing polymers to simplify the construction and increase the performance of various optical products. There are 2 main categories of electrically controlled optical products developed by the company:

- Laser speckle reducers
- Tunable lenses

2.2 Laser speckle reducers

A speckle pattern is an intensity pattern produced by the mutual interference of a set of wavefronts.[6] The task of laser speckle reducers (LSR) is to lower the amount of speckle available in a projection. In its very basic principle it is the set of moving diffusers based on electroactive polymers which are used to generate an oscillating motion which causes the downswing of the speckle effect.

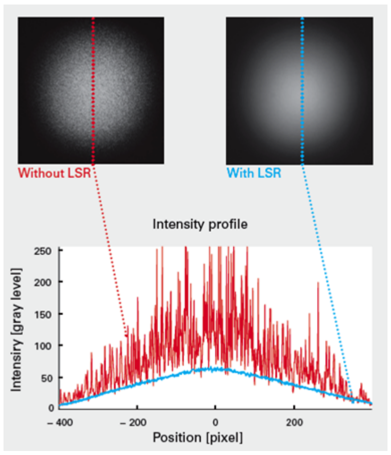


Figure 2.1: Laser speckle reduction[2]

Below we can see a picture 2.1 of a simple laser projection snapped before and after LSR was enabled.

2.3 Electrically tunable lenses

2.3.1 Overview

Electrically tunable lenses employ polymer membranes and optical liquids to achieve focal power adjustability while being controlled by an electrical signal. This allows having fewer components instead of utilizing industrially available lenses based on glass or plastics and avoids heavy-weight assemblies which are moving the components to achieve required optical properties. The sketch showing how the lens mechanism works is below - 2.2.

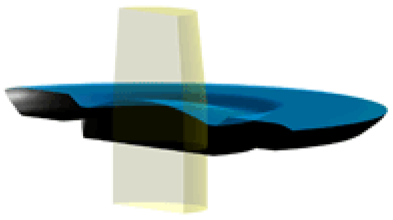


Figure 2.2: Lens principle[3]

In the picture above you can see multiple components:

- Polymer membrane with liquid inside
- Actuation zone

Each of electrically tunable lenses has a similar composition but it always differs in its optical properties and the way it is actuated– e.g. with type of liquid or different actuation technology. The actuator pushes or pulls on the membrane on a ring around the lens aperture. Depending on the actuation force (pull or push) the optical liquid is either pumped into the lens area or drawn from it. This results either in a more convex or more concave lens surface. Customers can integrate these lenses into their systems utilizing exposed pinouts. To allow a customer to use the lens in the so called focal power mode, a lookup table between actuator position and focal power must be known. This characterization data is stored in an EEPROM chip included within the lenses. Each type of lens is using different set of parameters to control the actuator. Some actuators use current, some may use voltage. Additionally there is a multitude of sensors present which can measure and report back the actual lens state.

2.3.2 Multiple lens types

Currently Optotune disposes with multiple electrically tunable lens types. Each of these types is controlled utilizing slightly different mechanisms. As the company is evolving fast, new types of electrically tunable (ET) lenses are released often. Therefore also their characterization and quality assurance mechanisms have to be scalable enough to reflect this trend. In the next section some of Optotune's lenses will be described in further detail.

2.3.2.1 EL-6-18

This lens is controlled either by current or a position feedback signal. Therefore its EEPROM (electrically erasable programmable read-only memory) chip has to contain information for both control modes. There is a picture of it attached below. In position controlled mode PID is driving the lens with different currents until proper position value is read-out from the sensor integrated in the lens.

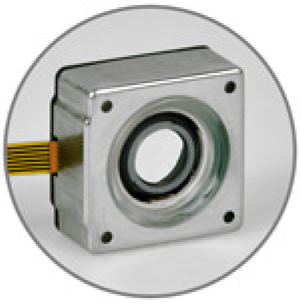


Figure 2.3: EL-6-18[4]

2.3.2.2 EL-10-30

Current controlled lens – lens is actuated by changing current provided to the lens.

Figure 2.4: EL-6-18[5]

2.4 Optical product characterization and quality assurance

Each optical product prior to being sent to the customer has to undergo particular quality assurance processes and some of them also require characterization. In the picture below there is the basic product's lifecycle (2.5) throughout the construction stated. The tasks marked with blue border are

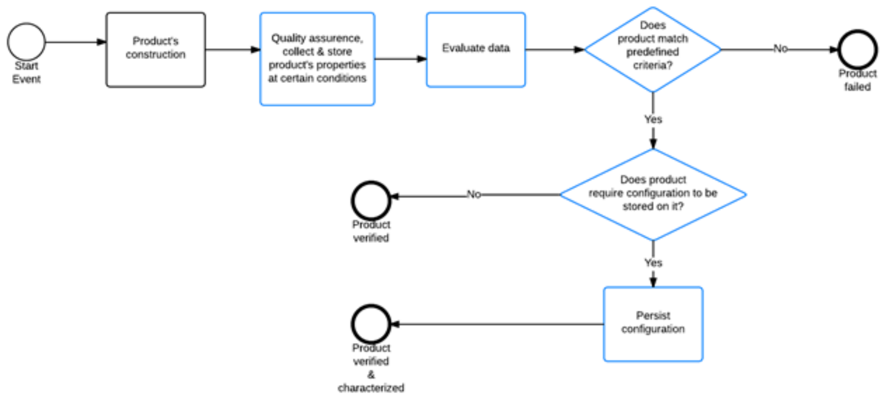

Start Event
Product's construction
Quality assurence, collect & store product's properties at certain conditions
Evaluate data
Does product match predefined criteria?
No
Product failed
Yes
Does product require configuration to be stored on it?
No
Product verified
Yes
Persist configuration
Product verified & characterized

Figure 2.5: Standard product lifecycle

duties which need to be carried out by software. Each product is utilizing many devices from which data are collected and uses different algorithms for acquiring them. Each product needs a different data evaluation algorithms to assess whether it matches a certain level of quality. As stated previously, electrically tunable lenses, in contrast with LSR, require configuration information to be stored in its memory chip. Therefore also “Persist configuration” task is included in BPM diagram above.

2.4.1 Electrically tunable lens characterization process

Each of the electrically tunable lens types undergoes a different characterization process throughout which various data about the lens are collected. Below there are the most important lens types and underlying procedures that need to be handled by the software discussed. Due to an existing NDA the description of characterization and quality assurance processes only reflects the algorithms from an abstracted manner. For the discussion of the software design this shall not pose a problem since the very particular details are not required to be known. During the characterization process various algorithms are employed to collect the data about the lens whereas it is measured at different configurations. The set of algorithms and configurations differ among product types. In general every characterization procedure consists of multiple characterization actions which need to be carried out. A characterization action is the very basic unit of a characterization procedure and in general encompasses setting the lens and its environment to a certain configuration and collecting the data from various external devices used for measuring it. The results of characterization action shall be assessed by preconfigured pass/fail criteria used to evaluate whether it is valid or not. For setting the product into particular configuration external devices are required to be used. Therefore the designed software needs to be able to interact with various devices and new ones have to be easily integrated into the system. In the diagram below (2.6) the general lifecycle of a single characterization action is depicted.

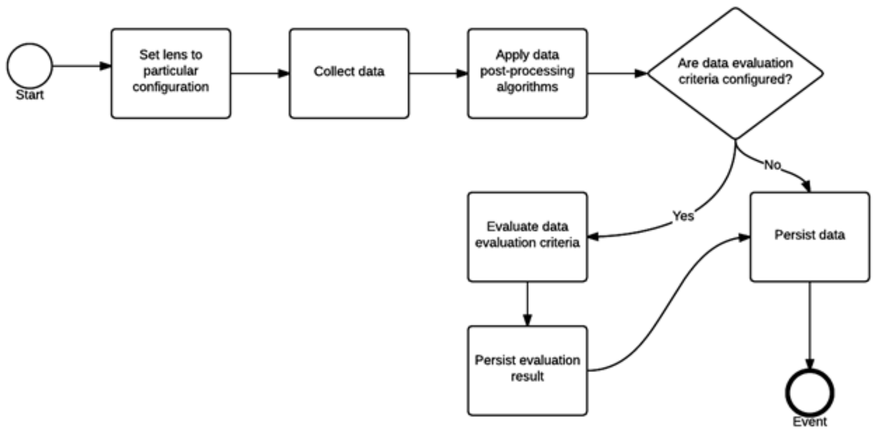

Start
Set lens to particular configuration
Collect data
Apply data post-processing algorithms
Are data evaluation criteria configured?
No
Yes
Evaluate data evaluation criteria
Persist data
Persist evaluation result
Event

Figure 2.6: Standard characterization action lifecycle

2.4.2 LSR characterization

LSR characterization consists of measuring the so called speckle contrast with and without LSR and cross-comparing the results. There needs to be a significant drop in speckle contrast when the LSR is powered on in contrast to when it is powered off. This measurement algorithm is handled by an external software. However, it is important to allow a future integration into the software because one of the most important goals for the company is to keep data centralized and to have a flexible system that can be extended very rapidly.

2.4.3 Quality assurance

Resulting from the description above it is clear that while executing most of the actions data has to be collected for further evaluation purposes. Each of the actions require either simple assertions above measured values or some more sophisticated logic to be executed. Both have to be incorporated within the designed unified system. As soon as one of the actions does not fit the evaluation criteria the execution of the rest is suspended and the product is marked as “Failed”. Of course this information has to be stored in a database for later post-processing and quality control. An example of a quality assurance criteria would be the maximal and minimal focal power the lens can be driven to (as they have been built for a certain focal power range). So having a position controlled EL-6-18 lens, this means at its maximum position value, focal power has to be below or equal to a certain level and for the minimum position value it has to exceed a certain threshold.

Chapter 3

Empowering processes via software – required features

As an output from the section above it is clear there are multiple products with various characterization procedures and different quality assurance specifications. Optotune needs a software solution which comes at an acceptable rate of ROI and providing support in following areas:

- Integration of the most important processes in one place
- Ensuring the reliability of each characterization process
- Downswing of human impact at each product validation
- Automation
- Backward data evaluation

3.1 Integration of the most important processes in one place

It is necessary to have all of the characterization and quality assurance processes integrated within one software solution to minimize the costs of maintainability. The fact is that company is evolving and new types of lenses come out each year and therefore new characterization and quality assurance algorithms need to be incorporated into the software very often. Having single centralized software will maximize the scalability possibilities as well as possible reusability of already existing features. From this criteria it is apparent that the best solution shall be the best-of-breed software.

3.2 Ensure the reliability of each characterization process

As the characterization itself depends on many external devices connected to a single PC station it is error prone. Whenever it comes to integration with hardware through exposed interface (e.g. RS232, TCP/IP) one has to deal with possible communication errors as hardware interfaces are not always stable and information may be lost as it is being transferred across the communication line.

3.3 Downswing of human impact at each product validation

Data are partially evaluated by engineers manually having a look at measured values or employing scripts which are not centralized and need manual triggering. This is a very error-prone approach as it depends on a single person. As the volume of the produced lenses is increasing this evaluation has to be covered by the software.

3.4 Automation

As there are many lenses being produced and therefore characterized and tested the software needs to integrate into an automated production environment which is feeding lenses to the measurement device. The software shall be capable of being integrated into such an assembly line. So it acts as a master where the logic for controlling the characterization part of the automation system is encapsulated.

3.5 Backward data evaluation

As the software is collecting data for each of the lens it shall be possible to have a look at this data, filter out the data user is interested in and allow the user to export it.

Chapter 4

Functional requirements

After going through the basic overview of what is required to be met by the software there is the specification of functional requirements provided in this chapter. Collecting of the functional and non-functional requirements as well as drawing of the use cases are the results of team work of Mr. Christoph Romer and me. The project leader (Mr. Christoph Romer) acts mostly as a product owner and HCI/UI designer very well aware of company's processes and needs which must be incorporated into the software. Together we have analyzed and designed whole software whereas my position was mainly to deal with non-functional requirements and information system design / architecture and afterwards preparation of the software development plan (WBS) and the final implementation itself.

- Product
    1. Product is uniquely characterized by serial number and batch
    2. Single product can be characterized multiple times
- Characterization process (session)
    1. Allow user to use single instance of a software to run characterization and quality assurance processes for multiple lenses, one at a time.
    2. Each characterization process can consist of as many steps as desired.
    3. Allow user to configure characterization process dynamically – as characterization process consists of multiple different steps which need to be carried out sometimes this step may be required to be retaken with a different set of parameters.
    4. Parametrized characterization steps – each characterization step can require some parameter input from the user.

5. Whole characterization process can be carried out by the software automatically – the assistance of operator is not required for executing the characterization process as soon as the initial configuration of characterization process parameters has been done.
6. Data export – it is possible to export the data throughout an ongoing characterization process (so called session)
7. It is possible to break the characterization process in between each of the steps and afterwards continue either with different characterization process or get back to the one originally running later on – e.g. after PC restart. This is referred to as well as session resume (consisting of session repopulation and re-running the session).
8. Each characterization process can have metadata (e.g. comment) assigned.
9. An operator shall have live data feed from product properties available so they would be aware of the product's status, e.g. polling of the sensor data from the lens.

- Data collection / evaluation / lookup

  1. Each characterization step (or action) can collect data and store it.
  2. Data can later on be accessed and filtered out by specific criteria
     a) Data needs to be filtered out by particular product they belong to
     b) Data needs to be filtered out by particular characterization process configuration utilized
     c) Data needs to be filtered out by quality assurance process result

- Quality assurance process – Pass/Fail tests

  1. Each characterization step (or action) may underlie configured data evaluation
  2. Each action may require very specific data evaluation logic
  3. In case of a failed characterization step, the characterization procedure is not allowed to proceed
  4. The data shall be evaluated on the vector and test assertion basis – i.e. the user can define vector of data above which they want to run particular test assertion evaluation
  5. Single test assertion can have multiple pass/fail tests mapped
  6. Pass/fail tests mapped within single test assertion are ordered
  7. Pass fail tests carry out the logic of the measured data evaluation

8. Single test assertion can have Boolean logic applied to evaluate the result of child pass/fail tests
    a) If there is not Boolean logic specified, if each of the test assertion's pass/fail tests passes test assertion does as well.
    b) If there is Boolean logic specified the final result of test assertion (whether it passed or failed) depends at the evaluation of Boolean expression, e.g. (1 or 2) and 3; where 1, 2 and 3 stand for the ids of pass/fail tests configured within single test assertion.

- Session termination – as soon as the last action of characterization process is successfully finished (i.e. no error occurred or pass/fail test evaluation has passed) the session is terminated

- Product / profile configuration grouping
    1. Each characterization process (session) configuration is referred to as a profile
    2. Profiles can be grouped into products
    3. Product specifies the type of the real-world optical product the profile is related to
    4. Profile session action configuration
        a) Session actions (characterization process steps) are defined within the profile
        b) Session actions ordering is left up to the user
        c) Session action can be marked as the last one – i.e. the session is closed after it is finished successfully

- Communication with external devices
    1. Software needs to be able to communicate with external devices over TCP/IP as well as serial link oriented protocols and it needs to be possible to add further communication protocols in future versions.
    2. Each external device which is accessed by the application shall provide continually the information of its status to the user.

- Re-flashing of lens's EEPROM – it must be possible to re-upload characterization data based on historical measurement data stored within the database.

- Manual EEPROM data edition – the user needs to have the possibility of editing EEPROM data manually, i.e. it shall be possible to edit the bytes or data fields which are afterwards stored in the lens EEPROM.

- Pass/Fail result – throughout the characterization procedure operator needs to be able to display the result of pass/fail test evaluation above single characterization step (action).
- Operator shall have the possibility to specify which set of devices will be available to the software and the software shall accordingly modify allowed characterization steps.
- Access restriction – each operator needs to login prior to using the application.
  1. Profile roles:
     a) CREATE_PROFILE – allows to create new profiles
     b) VIEW_PROFILE – allows to view existing profiles to the user who has been granted access to in Profile access configuration window
     c) EDIT_PROFILE – allows the user to edit existing profiles, though only the ones who have been granted access to
     d) SAVING_WORKSPACE – allows to save workspaces
  2. Administration roles:
     a) ADD_USER – allows to create new users in application
     b) DELETE_USER – allows to remove users from application
     c) CHANGE_PASSWORD – allows to change password to any user
     d) ASSIGN_ROLE – allows to assign roles to users
  3. Application roles:
     a) VIEW_HISTORICAL_DATA – allows to view collected data via Database viewer
     b) EXPORT_HISTORICAL_DATA – allows to export data from database to external files
     c) GRANT_PROFILE_ACCESS – allows granting profile access to a user/group.
     d) OPEN_PROFILE – allows to load profile and execute characterization via Profile window

These were only the most important requirements as the rest of the requirements will be clarified while discussing particular wireframes in the next chapter. Non-functional requirements which shall be met by the software are listed in the following chapter.

Chapter 5

Non-functional requirements

Non-functional requirements are grouped into multiple parts. The context where the software shall be utilized such as operation system which the software shall be capable to be executed on and preferred technologies as the outcome of the environment requirements is discussed in the first part.

5.1 Software context

5.1.1 Software environment

Optotune utilizes throughout its structures mainly PCs with operating system Windows 7 and newer. Therefore the software needs to be able to collaborate with such the environment. Optotune does not plan to utilize Linux system to run the software and therefore the technologies which are heavily bound to the Windows platform can be employed.

5.1.2 Programming language

Optotune utilizes the .NET family programming language, mainly C#, on the previous projects and due to its simple integration with Windows platform they require the software to be written in C#/.NET technology.

5.1.3 Technologies

Optotune demands sticking with open-source technologies wherever possible. Therefore the employed database and frameworks shall be open-source based without the need of paying license fees.

5.1.4 Database

To be capable of centralizing the data at a single storage the software needs to integrate with an external database. The design of the database is included within the scope of software implementation. To be capable of reflecting easily various data types which need to be handled by the application there is the NoSql solution MongoDb employed. It is an open source database which is capable of processing large volumes of various data and has many built-in features for filtering, processing and accessing. The database exposes many drivers which make it extremely easy to integrate it into an application. Of course there is also a driver for C# language available, distributed under an open-source license. Optotune disposes with Windows server machines where the database server shall be run.

5.1.5 Frameworks

5.1.5.1 .NET 4.5

As the final software will be utilized only with Windows 7 or a newer .NET 4.5 shall be utilized.

5.1.5.2 Unity IoC

As dependency injection framework Unity IoC was chosen. It is licensed under MS-PL and is freely distributed. There is a large community behind Unity IoC provided by Microsoft forum which is free to use for anyone so there shall be enough support when dealing with some technical issues.

5.1.5.3 Managed extensibility framework

In the software there is a need to create a new module per product which is demanded to be integrated into the software. As one of the non-functional requirements is to avoid recompiling the whole system when adding a new product the software is designed in a way it is capable of loading modules at its start-up. For this purpose Managed extensibility framework which is the part of Microsoft's portfolio was picked and is distributed under the open source license.

5.1.5.4 .NET authorization and authentication framework

As each user needs to log-in prior to using the system and the system needs to be integratable with LDAP it's essential to employ suitable technology which will allow developers to implement such a feature easily. Therefore .NET authorization and authentication framework comes as a rational choice as it has been developed by Microsoft.

5.2 Communication interfaces

As all of the products which will be characterized or quality assured by the application need to integrate with different services and third party devices, it is required to create a framework capable of providing an easy way to integrate devices which communicate through either TCP/IP (namely socket connections or via SOA), serial link oriented devices or eventually by some other means, e.g. provided SDK.

5.3 Software architecture

In this section the non-functional requirements demanded from the software design and architecture are described.

5.3.1 Modules

Application consists of modules which are being used by the core. Modules contain the implementation of communication protocols and offer custom actions which can be triggered by the platform. Modules implement interfaces which are exposed by the platform. This ensures that the platform is able to handle them. Afterwards these custom implementations are stored in unity context so each of the services can demand them as a dependency. As the application needs to be easily extensible for new products without being recompiled the main framework needs to load all deployed modules from a certain repository which can be either local folder or another network location. Each module represents a single product and the characterization and quality assurance logic is implemented within a module. Modules need to be versioned and the application must be capable of running multiple versions of the same module at a time. The modules are uniquely identified by their id and version whereas id reflects the product type it is dedicated to.

5.3.2 Module architecture

The modules are collecting data about the product. Each product has its custom properties which need to be collected and evaluated by software – these are called dictionary fields. A dictionary field is nothing else but a human readable representation of a watched parameter and a number identifier assigned to it. When persisting values, the software just uses the number identifiers and when the application needs to translate an identifier into human readable form it can search for it.

The module is carrying its dictionary fields. Each version of module has to be persisted in database with its dictionary fields and transition rules among the previous and the current version of modules.

Transition rules are required from scalability point of view – in between 2 versions of module, dictionary fields could be eventually shifted or changed and when the configuration which was bound to the old version needs to be transferred onto the new one all of the references pointing toward dictionary fields need to be updated. To state an example of such a situation it could occur in module of version 1.0 there is a property “Focal power” mapped onto the index of 18. For some reason this had to be changed in module version 2– e.g. measurement device specification changed – and in the new version the same value is stored under the dictionary field id of 20. A user could eventually have legacy configuration which they would like to port over to utilize with the new version of the module. In such a legacy configuration there could be an evaluation criteria bound to “Focal power” property which it assumes is located at the dictionary field id of 18. In order to sustain the correct functionality such a reference needs to be re-mapped to 20.

5.4 Pass/fail tests – product quality assurance

Each module can eventually have its own pass/fail tests implementations packed within it. Platform has to store the configuration information about pass/fail test (requested pass/fail test parameters or vectors, its fully qualified name, version). Pass/fail tests are versioned. Hereby it allows to avoid increasing module version with each modification made within single pass/fail criteria. Whenever there is a change in pass/fail test the version of it has to be increased and the platform needs to store new configuration for the given module.

5.5 Actions

Action is a basic unit executing the characterization logic. The characterization process consists of executing a certain set of actions. The module is responsible for executing various actions. Within the action the logic is implemented (driving/measuring/characterizing the lens). There are 2 kinds of actions:

- Session actions - are directly bound to the measurement session,
- Device actions – devices need to be capable of executing actions as a part of its lifecycle – for example measurement devices require calibration to be carried out prior to measuring.

The result of both actions mentioned above can be validated via pass/fail test concept and persisted in database.

5.6 Characterization process

Characterization process consists of executing the sequence of session actions. The sequence of actions which need to be executed per product is configured and stored within the profile. After executing an action which is marked as the last one in the sequence, the session is considered to be finished and is stored within the database with a flag marking this state. It can have either a passed flag or a failed flag attached. There is a requirement to allow the user to cancel the characterization process. When the cancel is requested, all of the module actions have to be terminated properly.

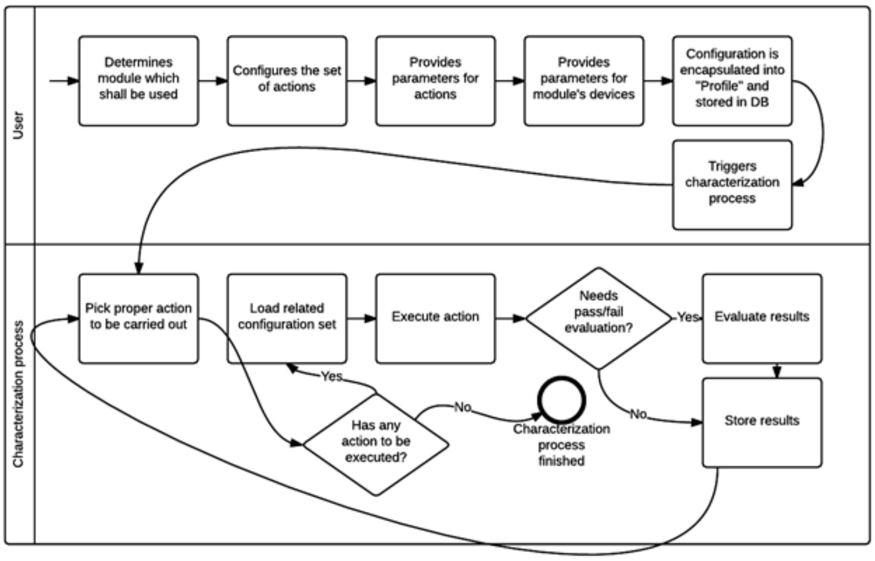

User
Determines module which shall be used
Configures the set of actions
Provides parameters for actions
Provides parameters for module's devices
Configuration is encapsulated into "Profile" and stored in DB
Triggers characterization process
Characterization process
Pick proper action to be carried out
Load related configuration set
Execute action
Needs pass/fail evaluation?
Yes
Evaluate results
No
Store results
Has any action to be executed?
Yes
No
Characterization process finished

Figure 5.1: Lifecycle of a characterization process

5.6.1 Session, distributed session

Session is the mean of grouping the set of actions carried out for a product. Each characterization process is encapsulated within the scope of session. Each product can have multiple sessions assigned whereas each of them stands for particular characterization process.

Session can be utilized across multiple profiles (different configurations). The session is only bound to the particular product. It is possible to reopen a closed session (after finishing the last action).

Sessions need to be distributed; meaning user could split-up the session execution over multiple instances of the software. To provide an example of distributed session use case there's a sapmle user story taking place next. *User creates multiple profiles with different configurations selected for the same product. Starts the measurement at the OT instance, OT1, with profile p1.*

Having all measurement actions done and passed for chose profile and configuration executed, the session is not terminated/closed yet, as he/she might have chosen the configuration without last measurement action configured (if there was a configuration with last action and the action was executed then the session is closed). Afterwards operator at the different position of the assembly line wants to do the final characterization of the product, therefore he connects the lens, which has been previously partially characterized and wants to continue in the started session. First of all, profile related to the given product must be chosen (distributed session can be used only for the product of same type) and afterwards existing session is reloaded with chose configuration. If the selected profile contains last action which was executed, then the session is terminated, otherwise it remains opened. Sessions which had been already closed are re-openable.

Below there's a bpm diagram describing section lifecycle attached.

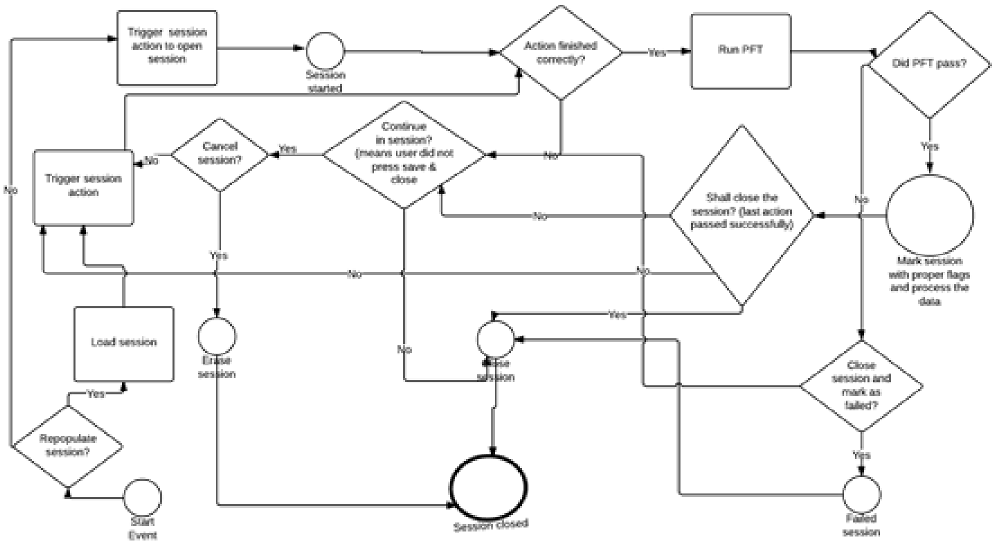

Trigger session action to open session
Session started
Action finished correctly?
Yes
Run PFT
Did PFT pass?
Yes
Mark session with proper flags and process the data
No
Shall close the session? (last action passed successfully)
No
No
Yes
Continue in session? (means user did not press save & close
Yes
Cancel session?
No
Trigger session action
No
Yes
No
Load session
Yes
Repopulate session?
Start Event
Erase session
No
Close session
Session closed
Close session and mark as failed?
Yes
Failed session

Figure 5.2: Lifecycle of a session

5.6.2 Repopulating session

After the application is re-populated with the closed session (either passed or failed one, see the diagram above), and the user re-triggers any of the session actions (this re-opens a session), the action overwrites the result of its predecessor in DB and tags the action/session to describe the situation.

5.6.3 Session inception

To be able to start the session each of the external devices needs to be in a “Ready” state. As soon as the session is started, the user is not able to trigger any device action anymore (they are disabled). If the user realizes that they need to take some other device action after the session has been triggered they are allowed to press Save and close session button within Profile window (see Profile window definition in the chapter Use cases), which will close the session and re-enable device actions. Afterwards they can repopulate with the session. Such a use case, when the user is triggering a session only for a particular subset of the actions shall be rare. If required to split-up the characterization process among multiple steps/characterization stations, then multiple profiles shall be used.

5.7 External devices

Each external device which OptoTester (the name of the software) integrates with shall be treated as a state machine. There are few general states each of the external devices may get into (because of integration with general framework) and need to be reflected within every custom device state machine. Such general states are mainly error states in case of communication problem occurs and “Ready” state signalizing special state when the device is ready to use and an operator may start the characterization process. The general states are:

- A communication error – when either serial link connection, TCP/IP connection or another protocol connection could not have been established or has failed.
- An error while processing – this error state signalizes the fact that the framework is able to communicate with a device though unpredictable exception during the processing of the response occurred.
- Timeout – Each of the devices has certain timeout period configured after which one device’s state machine is transferred into this state pinpointing the situation when something is wrong with the external device and it’s not responding to the commands.
- Ready state – this is the special state which signalizes to the framework that the device is ready to be used.

There can be also custom states, for example the state realizing the calibration of the device itself.

Chapter 6

Wireframes and use cases

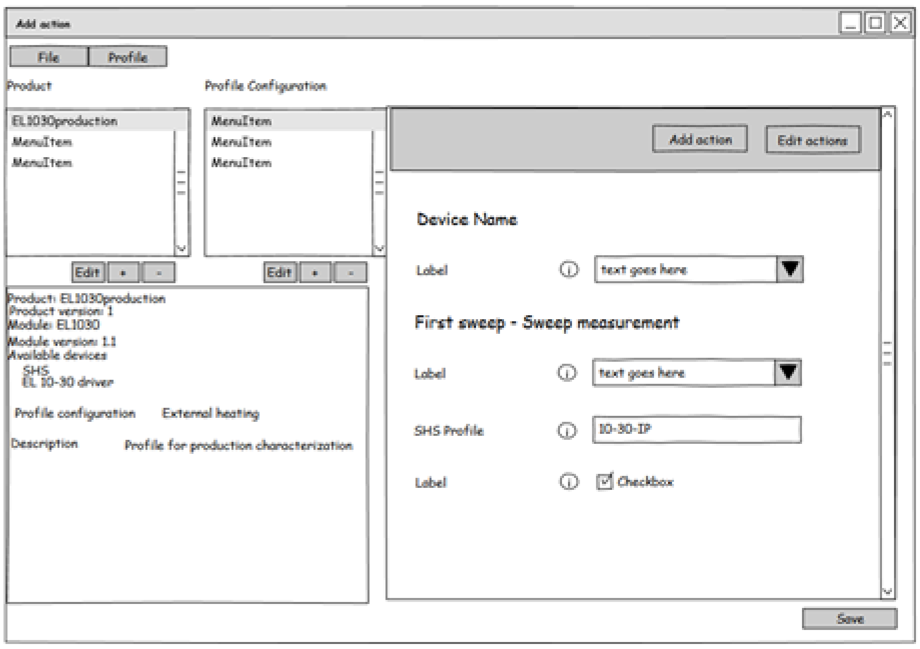
In this chapter multiple use cases capturing the most important functional and non-functional requirements are depicted. The use cases are grouped according to their relation to the particular wireframe[1]. The mapping in between use cases within the wireframe and the functional or the non-functional requirements shall be obvious and therefore this mapping is omitted.

The first wireframe deals with a profile and a product configuration which is afterwards used for characterization process. The second wireframe “Profile window” deals with the characterization procedure itself. The profile window is the most important wireframe as it is the main gateway for the operators to characterize the product. Afterwards there is a set of wireframes used for configuring pass/fail evaluation criteria. The last set of wireframe discusses parallelized lens characterization. There are many use cases which have been specified for the software but for the sake of simplicity and thesis-length restriction only the important ones are stated so the reader can get an overall basic understanding of the most fundamental task of the software. Each of the wireframes stated below has been designed as the team work of my supervisor and the project leader Mr. Christoph Romer and me.

6.1 Profile editor

Profile editor is the window where an operator is able to configure the parameters for characterization procedure.

[1]Representation of skeletal framework of a single screen depicting arranged elements devoted to accomplish particular purpose.



Product: EL1030production
Product version: 1
Module: EL1030
Module version: 1.1
Available devices
SHS
EL 10-30 driver

Profile configuration External heating

Description Profile for production characterization

Device Name

Label text goes here

First sweep - Sweep measurement

Label text goes here

SHS Profile 10-30-IP

Label ✓ Checkbox

Figure 6.1: Profile editor wireframe

6.1.1 Add product

The user starts by adding a product via utilizing "Add product" window where he picks a particular module which shall be utilized and configures the serial number feed used to validate entered product's serial number and batch properties. A description may also be provided but it is not compulsory and is only used for informative purposes. After configuring the product successfully it is displayed in the main Profile editor window products list.

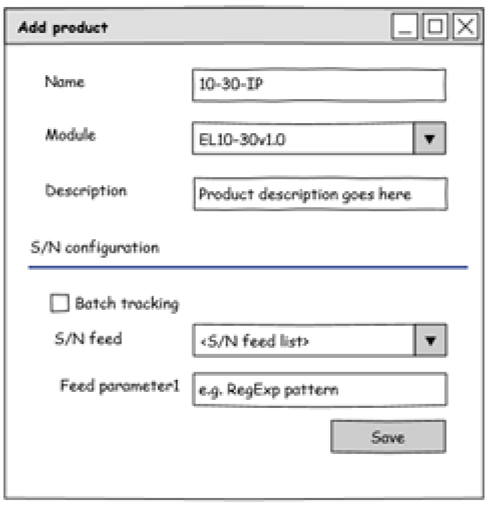


Figure 6.2: Add product - wireframe

6.1.2 Add profile

After having the product configured and picked user can create profile. In this window the name and configuration has to be entered. The user using a different configuration can change the set of devices which are required to be connected to the PC. So not always necessarily every device needs to be connected, e.g. for certain characterization steps it is sufficient to have only the lens driver connected (such as checking the sensors) whereas the measurement device is omitted.

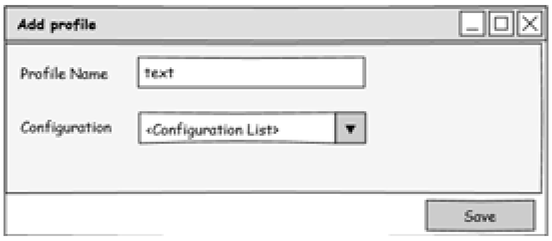
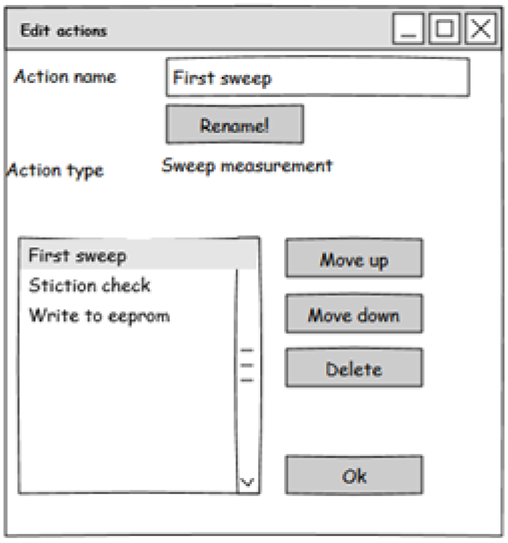


Figure 6.3: Add profile - wireframe

6.1.3 Edit actions

As soon as the user configures a profile they are able to press “Edit action button” in the main window and configure the set of characterization actions which need to be executed for this profile. They can do so by pressing “Add button” in Edit actions form which pops-up “Add action” window where they can choose the action type and enter a descriptive name which needs to be unique.

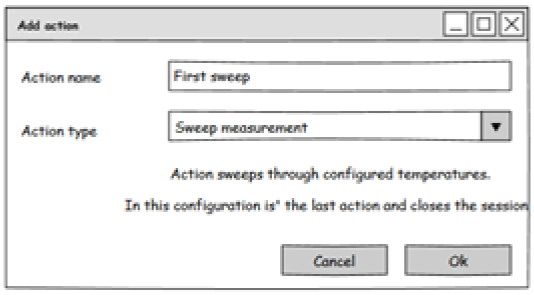


Action name: First sweep

Action type: Sweep measurement

First sweep
Stiction check
Write to eeprom

Figure 6.4: Edit actions - wireframe



Action name: First sweep

Action type: Sweep measurement

Action sweeps through configured temperatures.

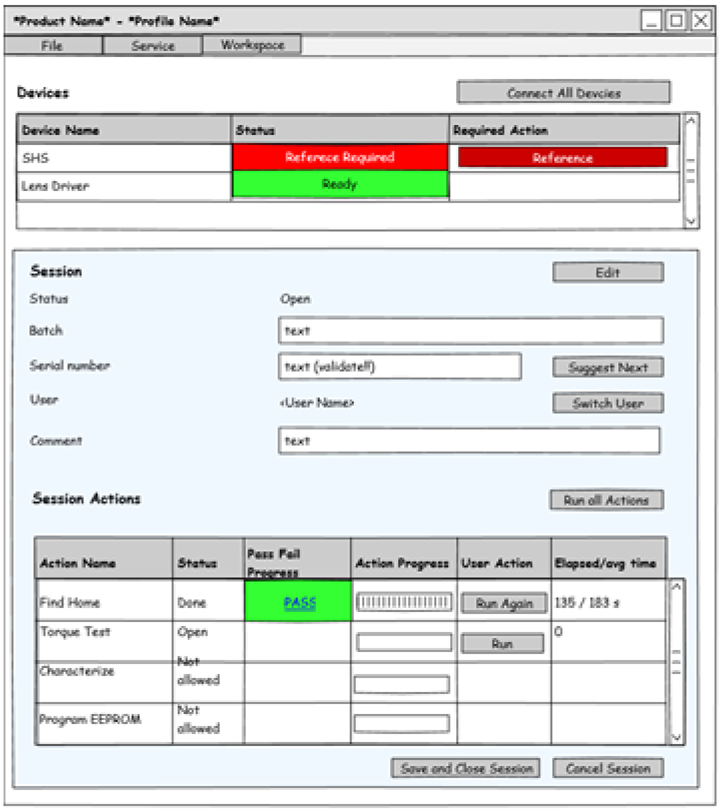
In this configuration is' the last action and closes the session

Figure 6.5: Add action - wireframe

After having a profile and the set of actions defined in the main window parameter fields for each of the configured actions show-up the user can enter their desired configuration. The devices can also request parameters, such as IP address of a device (if it is device accessed via TCP/IP) or for example com port name for serial link devices. The Pass/Fail test criteria is also treated as a profile parameter and Pass/Fail configuration window can be opened from profile editor by clicking “Configure pass/fail test” at the very bottom of displayed profile parameters.

6.2 Profile form

Profile form is the most important wireframe of the whole application. Prior to opening this window the user needs to choose defined product and related profile configuration where the characterization process – the sequence of actions itself – is defined. In the top section of window the user needs to set-up a connection to each of the devices required by configured characterization actions. As soon as all of the devices are in "Ready" state and a particular serial number and a batch number are entered the user is eligible to start the characterization process by either triggering "Run all actions" or manually running action by action. Each of the configured actions can have pass/fail test evaluation criteria assigned (configured from a profile editor) which are triggered just after the action has been finished. As soon as the action which is considered to be the last one finishes, the characterization process is considered to be finished – it can either end-up as a valid product (matched configured evaluation criteria) or a failed one. An operator is advised of this state and can act accordingly.



Device Name	Status	Required Action
SHS	Reference Required	Reference
Lens Driver	Ready	

Status	Open
Batch	text
Serial number	text (validatell)
User	‹User Name›
Comment	text

Action Name	Status	Pass Fail Progress	Action Progress	User Action	Elapsed/avg time
Find Home	Done	PASS		Run Again	135 / 183 s
Torque Test	Open			Run	0
Characterize	Not allowed				
Program EEPROM	Not allowed				

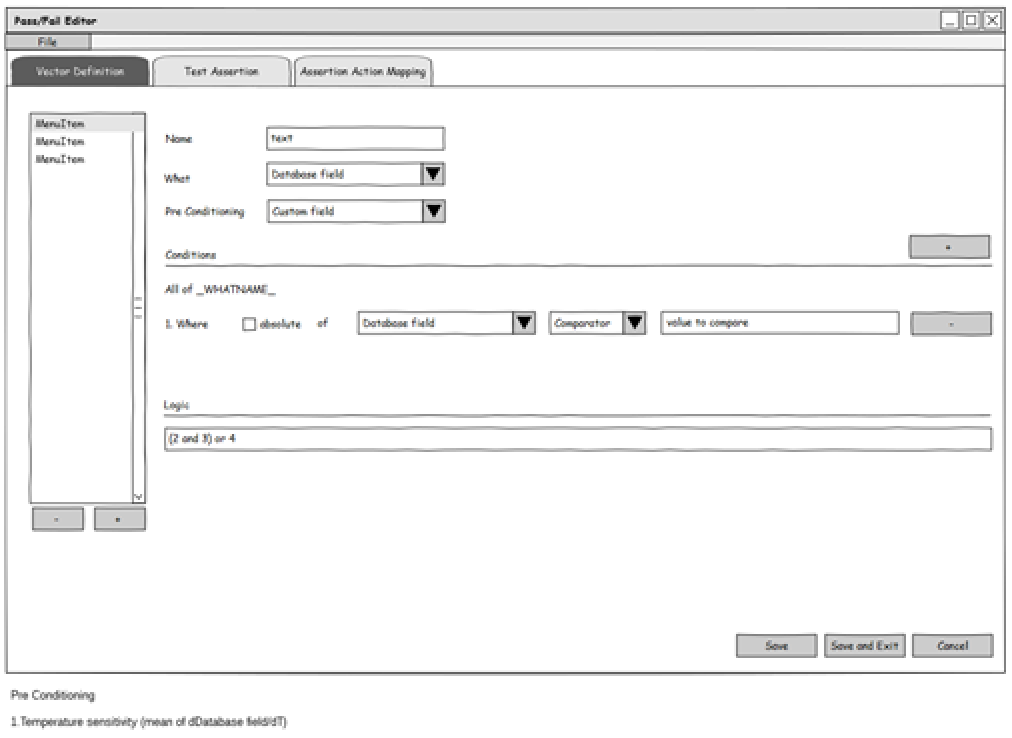
Figure 6.6: Profile form - wireframe

6.3 Pass fail test configuration

In this wireframe evaluation criteria for each of the products can be configured. The user can get into this window from the profile editor. Configuring a pass/fail test consists of 3 basic steps.

6.3.1 Configuring vector

The vector represents data of the single dictionary field. In the very first step the user needs to configure the vector of data over which they intend to do assertions. If the vector consists only of an integer or double data they can employ simple where condition based restriction of dataset via stating explicit criteria which the measurement values shall meet within the given vector. E.g. the user can define the vector of “Focal power” values which were measured at desired lens configuration.



Name: text

What: Database field

Pre Conditioning: Custom field

Conditions

All of _WHATNAME_

1. Where ☐ absolute of Database field | Comparator | value to compare

Logic

(2 and 3) or 4

Pre Conditioning

1. Temperature sensitivity (mean of dDatabase field/dT)

Figure 6.7: Wireframe depicting the vector configuration

6.3.2 Configuring the test assertion

Each of the modules (representing a single product) may contain custom data evaluation tests – pass/fail tests - which are used to decide whether the product is meeting certain quality criteria. Each of the pass/fail test accepts the vector of data as an input parameter – the vector over which the assertion itself shall be executed.

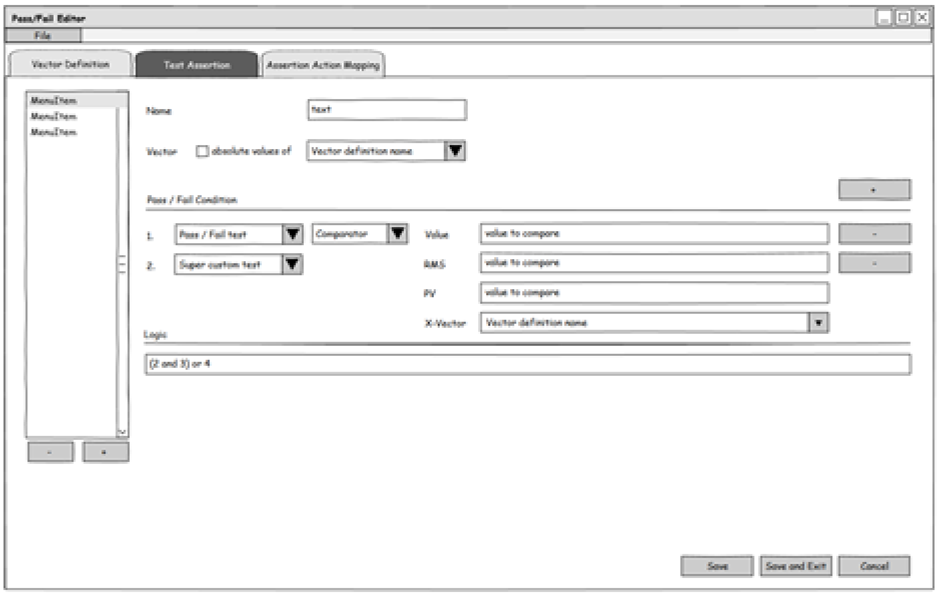

Pass/Fail Editor
File
Vector Definition | Test Assertion | Assertion Action Mapping
MenuItem
MenuItem
MenuItem
Name: text
Vector ☐ absolute values of: Vector definition name
Pass / Fail Condition
1. Pass / Fail test — Comparator — Value: value to compare
2. Super custom test — RMS: value to compare
PV: value to compare
X-Vector: Vector definition name
Logic
(2 and 3) or 4

Figure 6.8: Wireframe depicting the configuration of test assertion

6.3.3 Mapping of configured test assertions onto actions

Having the test assertions configured from within the previous step the user needs to assign them to certain actions which were configured in the profile editor. Each action after being finished (see profile use case) is capable of triggering configured set of pass/fail test assertions evaluating its results.

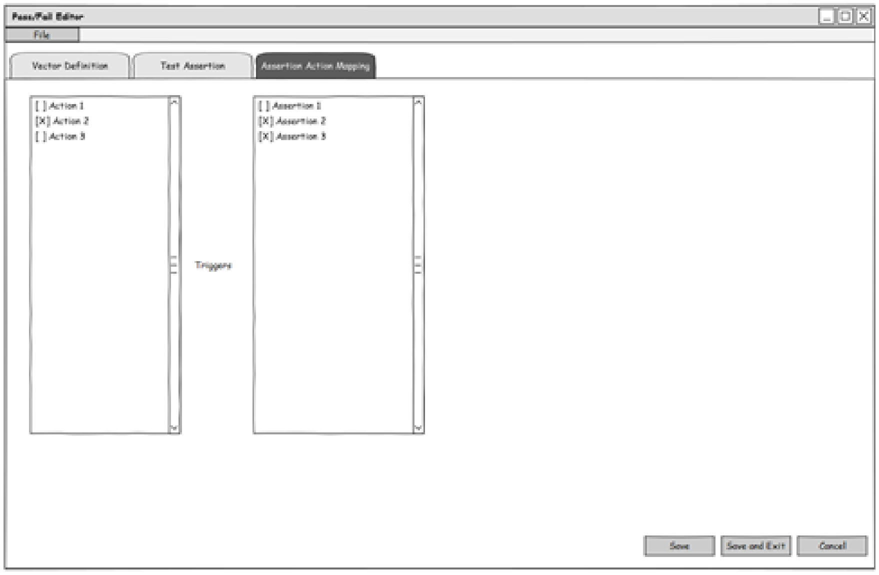


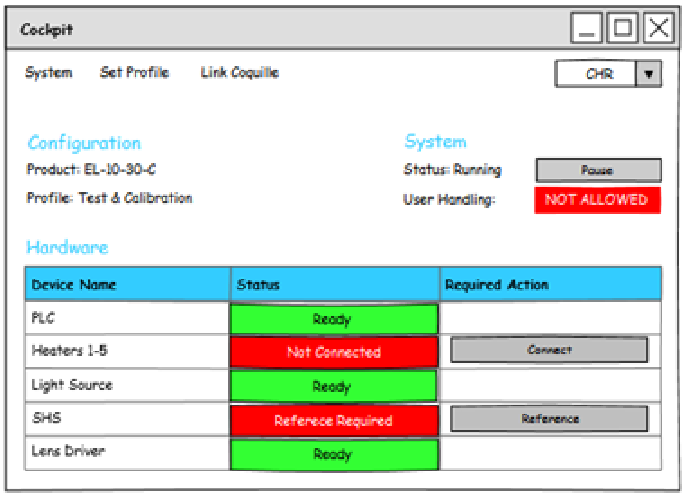
Figure 6.9: Wireframe depicting the mapping of configured test assertions onto available actions

6.4 Automation mode

In automation mode the user is eligible to characterize multiple EL family lenses at once bringing down the cost significantly – for further cost saving analyses see Economic aspects of software section. This use case swaps the way of actuating the framework and exposes possibility to control it with the minimized assistance of operator. As the highest time consumer during the characterization process is lens preparation (which occurs prior to measuring a lens, each lens needs to be characterized at different configurations therefore preparation procedures occurs multiple times per single characterization process), this process is parallelized and lenses are being pre-set to targeted configuration prior they get characterized. The automation line is moving the sled filled with lenses back and forth in a measurement device. After any of the lenses reaches its targeted configuration a PLC[2] controller adjusts it into the measurement position – mantles it within the measurement device – and the characterization can start. The software needs to be capable of handling multiple actions at a time and assigning correct data onto them within the database. Further description concerning how the automatized characterization line works is not provided as that is not the topic of this thesis. Only

[2]Programmable logic controller, digital computer used for industrial electromechanical processes (e.g. controlling of assembly lines)

requirements which shall be met by the software to allow such integration from conceptual point of view are considered.



Configuration

Product: EL-10-30-C

Profile: Test & Calibration

System

Status: Running

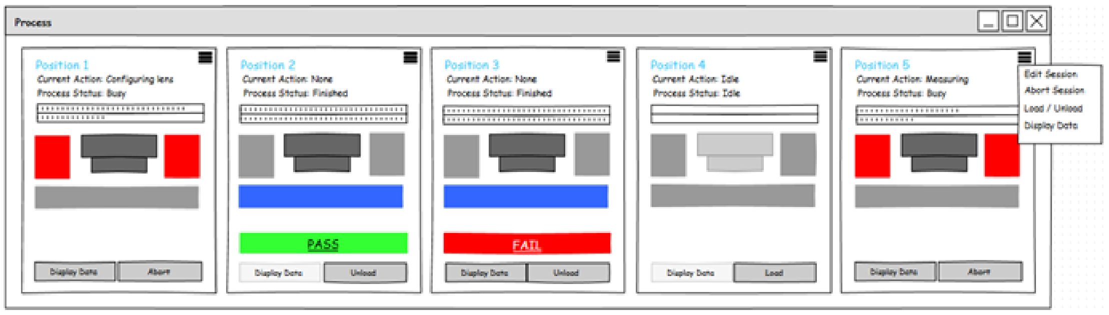
User Handling: NOT ALLOWED

Hardware

Device Name	Status	Required Action
PLC	Ready	
Heaters 1-5	Not Connected	Connect
Light Source	Ready	
SHS	Referece Required	Reference
Lens Driver	Ready	

Figure 6.10: The main gateway of the automation mode

The Cockpit UI wireframe (6.10) is the gateway for the automatized characterization where the operator needs to establish the connection with each of the devices and afterwards can start the system. As soon as the system is started and there are some positions into which the lens has been placed, the characterization process which would normally be carried out sequentially automatically starts.



Position 1
Current Action: Configuring lens
Process Status: Busy

Position 2
Current Action: None
Process Status: Finished

Position 3
Current Action: None
Process Status: Finished

Position 4
Current Action: Idle
Process Status: Idle

Position 5
Current Action: Measuring
Process Status: Busy

Figure 6.11: Depicts every characterization position

In the Process wireframe (6.11) there is the state of each of the positions depicted. Each position needs to go through various states throughout which the lens is set to targeted configurations and measured afterwards. Of course also the evaluation of measured data is the part of a single position's lifecycle. The description of position's lifecycle is provided in the diagram below.

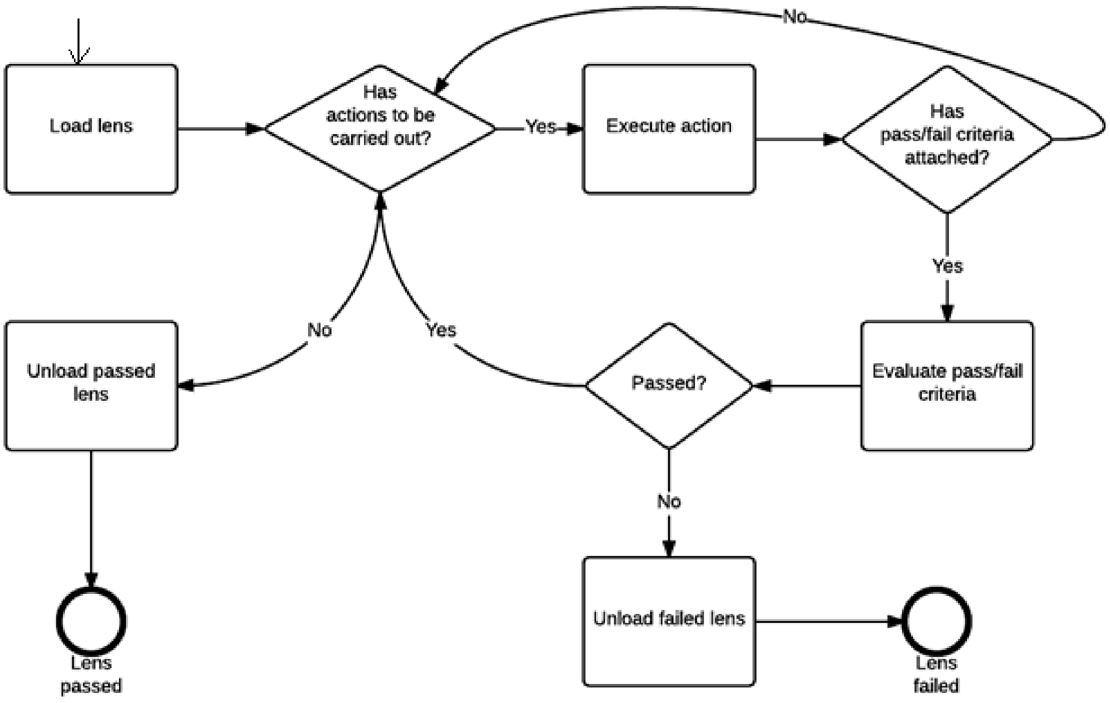


Figure 6.12: Depicts the lifecycle of a single characterization position

The application consists of many other use cases and wireframes which were omitted due to the limited scope of this paper. Therefore only the most important ones explaining the core functionality were included. The most of the omitted use cases are utilized for data evaluation and further configuration of the system.

Chapter 7

Economic aspects of software

For economic analysis the following aspects of software are considered:

- General framework decreases the time required for the implementation of test and characterization algorithms of a new product as well as the time required for incorporating change requests
- The centralization of data and the simplification of data evaluation
- The downswing of human impact and the elimination of possible process errors
- The automation of the characterization processes

In the scope of this project 3 products shall be incorporated in the final version and therefore only those are considered for the economic analysis.

7.1 General framework decreases the time required for the implementation of test and characterization algorithms of a new product

Before the inception of OptoTester software the company used to create a standalone application per product. This was producing multiple applications reusing the same code which differed only in characterization procedures and had an impact of having hard to maintain code. Such applications were also difficult to adapt to new modifications (as Optotune is still developing new products, change requests to adapt procedures are happening multiple times per month) and with ever increasing amount of the change requests the cost of maintaining is rising up. The creating of a standalone application per product lasts in an average 15 man days but such an application has a hidden cost of high maintainability and low scalability index when a software developer needs to spend approximately another 5 man days for debugging and working-in the change requests from the engineering team. Also such application only exports data to a plain text file. The cost of this fact is discussed in the next sections. It is possible to incorporate a new product to the OptoTester framework in the form of module in an average of 6 man days with all the necessary debugging included. The time required to work change requests into the existing module is taken down just to 1 man day.

Task \Application Type	Standalone application	Module in OptoTester	% of orginial time required with OptoTester
Implementation [man days]	15	6	40%
Debugging [man days]	5	1	20%
Time comparison [man days]	20	7	35%

Table 7.1: The savings issued by OptoTester in comparison with the standalone application per product

7.2 The centralization of data and the simplification of data evaluation

Data are important possession companies have – they allow them to evaluate the performance of their product, identify any weak spots and analyze the market to help improve their competitiveness. With the legacy approach of having single application per product data were exported in a single file per characterized product. Therefore it was also very laborious to get even the simple statistics out of that. The company was employing different excel sheets where the operators were manually copying exported data over and sometimes also adjusting their format. To verify each product there were only semi-automatized excel macros plotting the measured data of each product and also an interaction of an engineer was required. It meant that the following operations had to be realized per each product:

- Run the characterization – done by the software
- Export data into a file – done by the software
- Copy data into unified excel sheet – done by the operator, average time of 2 minutes per product

- Evaluate the data – done by an operator or an engineer, average time of 3 minutes per product

The step 3 and 4 will be fully automatized thanks to the information system and the time of the evaluation will be a few milliseconds long and therefore it is neglectable. The company asked for the quantification assuming production of different amount lenses per year, i.e. 3000, 5000, 10000, 20000, 50000. The real sales predictions are the subject to the NDA and will not be revealed in this thesis.

Amount of lenses [thousands]	3	5	10	20	50
saved man days	29	49	98	196	490

Table 7.2: Table depicting issued savings

One man day consists of 8.5 hours (that is an internal regulation of Optotune).

7.3 The downswing of human impact and the elimination of possible process errors

As the software also in its sequential version brings us the centralization of characterization and test processes under the single hood the human impact is minimized. In the legacy state it could have happened that an employee evaluating the data had simply mistaken and verified broken lens as well. With the software this will not be the case ever again. The system never proclaims the broken product to be the working one (unless there is misconfiguration present) and therefore the only possible way the company could ship the broken product is a mistake of operators. As lenses are being shipped within batches and the company does not track the amount of broken lenses (as it was fairly a small number) there are not statistics available to consider the savings the software would bring in this domain.

7.4 The automation of characterization processes

The software offers 2 ways of characterizing products:

- Sequential
- Automatic

In the sequential characterization as stated above the operator can only characterize single lens at a time. In the automation mode we get the possibility of operating multiple lenses at a time.

7.4.1 Automation speed up

Below possible speed up prediction with utilizing the automatized version of the system can be found. This graphical representation was prepared by Optotune's process engineering team and it's been used in this thesis with their kindly allowance. From the picture below it is apparent that the highest speed-up is achieved when dealing with 5 lenses at a time. This brings down the characterization time down to 3.4 minutes in an average per lens (instead of original 12 minutes per lens). It is the major cause of the characterization

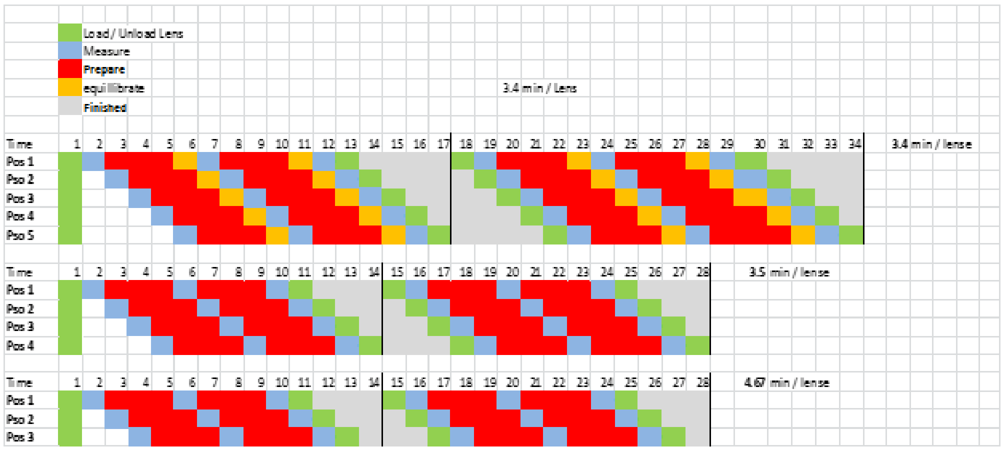


Figure 7.1: Automation speed graphical representation

and test procedure cost decrease. Particular time savings in comparison with the original time required with the ordinary sequential characterization are stated in the table below.

Amount of lenses	3000	5000	10000	20000	50000
Sequential characterization time [man days]	71	118	235	471	1176
Automation mode [man days]	20	33	67	133	333
[%] of sequential time	28%	28%	28%	28%	28%

Table 7.3: Automation mode savings

7.5 Economic assessment summary

A sum-up of benefits the software brings to the company and their quantification can be found in this section. For the quantification purposes an average salary per hour is considered and since the company is located in Switzerland a Swiss average hour salary for 2014 which is 44 CHF[3] is used.

7.5.1 One time constant cost savings

This section considers the one time savings brought by the software – it is mainly decreased workload for the creation of new modules per product compared with the creation of the standalone application. This presumes application is using 3 products as the part of delivery of the first version of the software, of course it is easily extensible for new ones later on.

Constant savings	39 man days * 8.5 hrs/man day * 44	14 586 CHF

Table 7.4: Table depicts one time constant savings

7.5.2 Periodical savings

In this section the periodical savings issued by the utilization of OptoTester on a yearly basis are stated.

Task \Amount of lenses	3000	5000	10000	20000	50000
Simplification of data evaluation	29	49	98	196	490
Automated characterization	51	84	169	337	843
Saved man days per year	80	133	267	533	1333
Saved CHF	29 920	49 8666	99 733	199 466	498 667

Table 7.5: Shows periodical savings issued each year by optotester depending on the amount of produced lenses

7.5.3 Savings issued by the software from different time horizons point of view

Possible savings which could be achieved by utilizing the software are stated in the sections above. For the overall return on investment evaluation which

[3]The value is due to Wikipedia.org and is only approximate

takes place in the very last chapter of the thesis it is necessary to evaluate the savings from different time horizons. Horizons of 1, 3, 5, and 7 years are considered. The achieved cost deductions per given period are described in the following table. The values below are the sum of constant and periodical savings.

Year \Amount of lenses	3000	5000	10000	20000	50000
1	44 506	64 453	114 320	214 053	513 252
3	104 346	164 186	313 786	612 986	1 510 586
5	164 186	263 920	513 253	1 011 920	2 507 920
7	224 026	363 653	712 720	1 410 853	3 505 253

Table 7.6: Table shows quantified savings in CHF

The economic aspects of developing the software for Optotune's purposes which are used afterwards for returning of investment and others economic indicators computations were described in this chapter.

Chapter 8

Software architecture and design

After considering the requirements and the overall asset of the system there is a design and the architecture described upon which basis the time estimation can be constructed.

8.1 Application architecture

8.1.1 Architecture overview

The final architecture consists of 3 parts:

- Database – MongoDb
- Platform
- Modules per each of the products

A sketch of the architecture and the interaction in between particular components is outlined in the picture below.

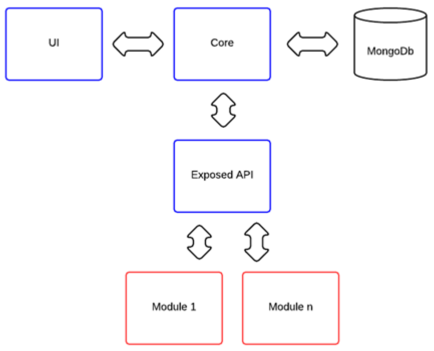


Figure 8.1: The sketch of the system architecture

The components marked with the blue border are considered to be the platform ones. These components altogether create the targeted framework offering the general functionality such as interaction with database, functionality used for data collection and general APIs for interacting with external devices over the certain set of protocols (such as support for Tcp/Ip protocol, as well as various serial link communication protocols such as Modbus and company's custom one). The framework exposes various APIs to modules allowing them to interact with the platform and utilize exposed services.

8.1.2 Database

As the requirement is to be able to store data at a centralized data warehouse the software needs to integrate with a database system which will allow the company to execute data analyses and further process their data for R&D[4] team research. The decision to utilize NoSql technology – MongoDb was made. The main reason for such a decision was the fact that throughout the optical product characterization process heterogeneous data are being measured. It is also possible that the company may need to utilize further data processing algorithms (or patterns) to evaluate their data such as map reduce. Whereas MongoDb meets the criteria perfectly to achieve the same with an ordinary RDMS would come at a higher development and maintenance cost – the main cause of this is difficult handling of varying data as it's not possible to handle them within single column and multiple different tables would have to be created or some sophisticated algorithm for data serialization employed which would bind the database onto particular platform and potentially cause troubles fetching the data with the 3rd party software. The drawback of utilizing relational solution is as well the fact each module could eventually export whatever type of data, this need means the need of adaption of relational database whenever a new module would export new type of data (which has not been included in the framework previously). Having the application running

[4]Reasearch and Development

with MongoDb, the technology allows to store heterogeneous data just within a single collection and there is no need for creating new collections over and over for each of the datatypes.

8.1.2.1 MongoDb overview

MongoDb is a document based database. Each record is treated as a separate document (analogy in RDMS would be a single record in a table). The documents are organized into collections (in RDMS analogy would be a table).

A MongoDB deployment hosts a number of databases. A database holds a set of collections. A collection holds a set of documents and a document is a set of key-value pairs. The documents have dynamic schema. The dynamic schema means that the documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.[7]

8.1.3 Modular architecture

As each of the products shall be easily integrated into the existing application without the need of recompiling the utilization of modular architecture is employed. Application is exposing certain set of interfaces which could be accessed by modules through shared library (API) whereas the implementation of them remains located in within each of modules. Custom implementation afterwards specifies the exact behavior of platform (e.g. executing different actions per module). There is a central module repository which is checked at the application's startup and all of the found modules are loaded by the software. Each configuration of the module needs to be stored within the database (for backtracking the exact characterization / evaluation algorithms used for each product).

8.2 Software design

Various frameworks empowering desired concept are utilized to achieve the targeted architecture structure from the implementation point of view. For modular architecture the Managed extensibility framework from Microsoft is used. Also Unity IoC as an inversion of control container shall be employed to support the implementation of given architecture and mainly to keep it scalable and hereby easy to maintain.

8.2.1 Platform design

The platform design underlies standardized n-tier architecture concept whereas the first tier is UI responsible for processing user impacts, the next tier is the service tier encapsulating the whole data collection and evaluation logic. The service tier interacts with the data access layer which treats the database operations. There is also one more tier – API – the set of interfaces which are implemented within the core part of the application and exposed through inversion of control toward modules.

8.2.2 Core architecture

The core is the most important part of the application where the most of the logic building up the framework is located. It is responsible for maintaining the communication with database as well as for executing the characterization process itself. The core consists of many packages but I will state a few of the most important ones:

- Service – contains the logic of application
- Repository – contains classes for accessing the database (data access layer)
- Facade – contains the implementation of facades located within API exposing core features to modules

The core part is capable of interacting with modules through the API as well as the modules are capable of interacting with the core of application through API in a vice-verse direction.

8.2.3 API architecture

The API consists mostly of interfaces which shall be accessible from modules and their implementations are located within the core part of the application. The core part is capable of interacting with API, as well as modules are capable of such an interaction though API is not "aware" of either of those. API is considered to be the middle-finger providing the mean the core and modules can communicate together.

8.2.4 Modules

Modules are implementing the interfaces exposed from API and therefore the core is capable of interacting with it. In these implementations modules are carrying the whole characterization and quality assurance logic per each of the products.

8.2.5 Inversion of Control

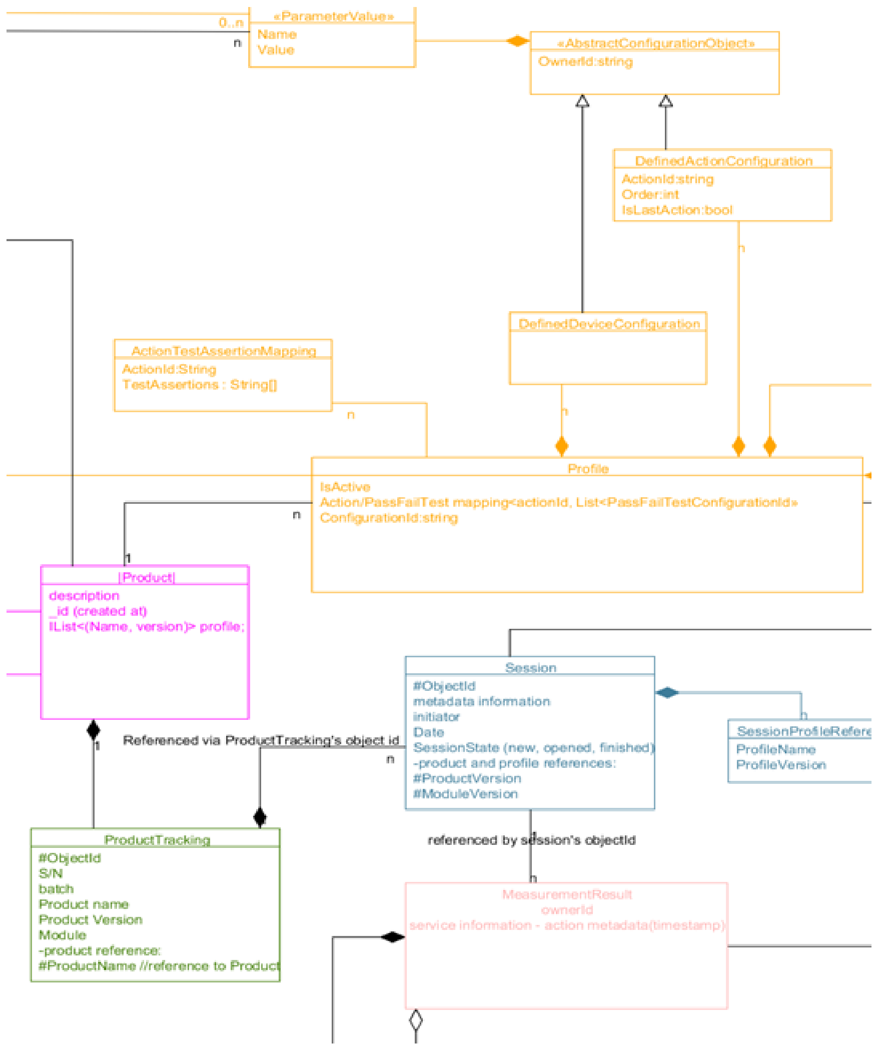
Inversion of control is a must-have of almost every application. It decreases the maintenance cost as well as ramps-up the scalability of the software solution. In this application I am utilizing Unity IoC framework which is suitable for

.NET family languages and disposes with direct support from Microsoft. The application at its startup sets up multiple IoC contexts for managing the lifecycle of various classes. Software sets up the main application context at the very beginning where each of the services gets registered and can be afterwards accessed easily from the other objects living in the scope of the same context. As soon as the application context is set-up the application loads modules found at pointed URL and builds up separate context for each of modules where the implementations of API interfaces are contained as well as the platform exported facade services allowing module to interact with it. This makes the architecture scalable and easy-to-extend for new requirements.

8.2.6 NoSql data model overview

In this section the most important part of fairly complex data model is described. Such a data model can be afterwards serialized directly from C# objects into MongoDb collections utilizing C# MongoDb driver. Unlike relational models, in NoSql modelling it is acceptable to live with a bit of redundant information as it does not support any joins out-of-box. Nor the transactions are available in MongoDb though it is atomic at the single operation level. Both of the previously mentioned is the tradeoff for having a simple implementation and an easy tool for data processing and its filtration. Also the asset of avoiding joins is the performance – MongoDb has been built for handling large datasets of various data and therefore persisting information for thousands of measurements with volume datasets is not the issue. Very important factor is also the horizontal scalability of MongoDb. Compared to relational databases, MongoDb is easily horizontally scalable ensuring high availability at a low cost.

8.2.7 NoSql data model



0..n
n
«ParameterValue»
Name
Value
«AbstractConfigurationObject»
OwnerId:string
DefinedActionConfiguration
ActionId:string
Order:int
IsLastAction:bool
n
DefinedDeviceConfiguration
ActionTestAssertionMapping
ActionId:String
TestAssertions : String[]
n
n
Profile
IsActive
Action/PassFailTest mapping<actionId, List<PassFailTestConfigurationId»
ConfigurationId:string
n
1
|Product|
description
_id (created at)
IList<(Name, version)> profile;
Session
#ObjectId
metadata information
initiator
Date
SessionState (new, opened, finished)
-product and profile references:
#ProductVersion
#ModuleVersion
n
SessionProfileRefere
ProfileName
ProfileVersion
1
Referenced via ProductTracking's object id
n
ProductTracking
#ObjectId
S/N
batch
Product name
Product Version
Module
-product reference:
#ProductName //reference to Product
referenced by session's objectId
n
MeasurementResult
ownerId
service information - action metadata(timestamp)

Figure 8.2: The most important part of data model

The description of the most import part of database model of the application is provided in the next lines. Persisting of the characterization procedure configuration as well as the tracking of measurements are present. The rest has been omitted due to the simplification purposes. As already stated, MongoDb is persisting data within collections (collection is analogy to a table in RDMS). Each record tracked within the collection is called a document. Instead of utilizing foreign keys to refer to others objects MongoDb employs nesting. E.g. having a user entity in RDMBS referring an address record through a foreign key, MongoDb would treat this requirement such as it nests the address object into the user document. Therefore there will always be some documents grouped within collections which are not nested and thus are the top-level documents. In data model depicted above there are Profile, Product, ProductTracking, Session and MeasurementResult entities stated. Each of the objects is stored within equally named MongoDb collection. Color coding of the picture captures the set of entities which are nested one to another, though each of equally colored objects are physically present within a single document, all of them are stored in the same collection. E.g. Profile collection (orange color in the picture) nests few other objects within it, like DefinedDeviceConfiguration and therefore also this entity is marked with orange color (it belongs to a Profile entity).

8.2.7.1 Persisting characterization process configuration

Characterization process itself depends on the selected module. The module determines the final product type for which the configuration shall be used, i.e. there is a module per product which is meant to be characterized by the software. Each of the module carries its set of actions which could eventually be configured as the part of characterization process. How does the characterization process look like is determined by the configuration entered by the user (Profile configuration). Profile entity persists basic information about characterization procedure such as session actions, device parameters and pass/fail criteria configuration. A single profile document contains multiple nested objects which are used to persist data mentioned in previous sentence. To store the session action configuration it nests the array of DefinedActionConfiguration objects which are carrying the information necessary to match the configured action with the action physically present within module. Each of the session actions can demand certain set of parameters used for customizing characterization procedure itself (e.g. maximum current used through the measuring). The application supports different datatypes for action parameters. To reflect such a need DefinedActionConfiguration inherits from AbstractConfigurationObject. Therefore each of DefinedActionConfiguration is also nesting the array of ParameterValue objects whereas each of parameters consists of Name/Value properties. MongoDb does not have fixed data types and therefore it allows to store different types of data within single property of the same object (it serializes every value into JSON). This approach unlike the relational one allows to have only a single collection storing various datatypes and avoids employing of different tables per type or utilization of sophisticated data deserialization logic (to get all the data into a single column within relational database). As each characterization process utilizes various external devices to either actuate product or collect information about it, such devices may also require configuration parameters to be provided (e.g. entering IP address of measurement device). And therefore it is also possible to provide parameters for devices in ProfileEditor window. Device parameters are stored within Profile document, nested as an array of DefinedDeviceConfiguration objects. They inherit from AbstractConfigurationObject – the same parent as DefinedActionConfiguration does and store the various types of parameters in the same way. Product object is used for grouping profiles. From the model point of view there is 1 to N relationship between Product and Profile entities. From the implementation point of view it is treated differently. Profile and Product entities are utilizing unique compounded indexes which consist of domain specific data. Profile entity identifier consists of a unique name, a version, and a product name used for identifying product which this entity belongs to as well as a unique module identifier. This approach allows to compound the information together to fetch the desired profile from the database easily. And having a product name included within Profile's key allows us to have equally named profile among different products and reflect mentioned 1 to N relationship between these 2 entities. Product entity key is also compounded key consisting of domain specific data such as its name and version and the reference to a module it belongs to. As one of the requirements is to be able to backtrack exact characterization configuration per every characterized product, profile and product entities are versioned (see the version property of their compound identifier). Whenever configuration of a profile/product changes it is stored as a new version of it whereas the old one remains archived and any new characterizations will be realized utilizing the most recent version found in the database. Also every product is uniquely identified by the batch and the serial number and thus it is required to keep the tracking of products in the database on the batch and the serial number basis. For this purpose there is a ProductTracking entity created which carries the key of a Product it belongs to as well as the batch and the serial number. This entity is identified by MongoDb 's unique id (MongoDb has a way of generating unique id for new document inserted into the database depending upon time of insertion, process id of database and a contingency value). MeasurementResult entity is the object used for storing the data collected throughout the execution of a configured session action. Its identifier is as well compounded key consisting of action id (identifier of the action within module implementation), owner id (the name entered for the action during the configuration within ProfileEditor window) and session id it is related to. Every measurement result holds a profile name of the profile which was configured by the user to be used for the execution of session action[5] (profile determines the set of parameters which shall be used for session action). Each measurement consists of multiple steps and exports various data. This piece of information is nested into the MeasurementResult document but the exact model description is omitted due to the scope limitation of the paper. Session entity is grouping measurements belonging to the characterization process (as the characterization process consists of the sequence of multiple session actions). It holds a reference in a form of having a field holding the value of

[5]Session action outputs the data into MeasurementResult collection

ProductTracking id. This allows to have multiple different sessions executed per single product. Also MeasurementResult entity holds a reference to a Session in the same manner as Session does with ProductTracking.

8.2.8 Utilizing MongoDb over relational database system summary

First of all the software needs to handle various data. With this approach it is possible to fiddle in whatever data required in a single collection. Unlike the RDMS where one would have to either deserialize different data types into the same structure or come up with a column or even table per datatype. On the other hand MongoDb does not allow joins. This can be sometimes painful but having data model design in a way that it nests all the related information within one collection there is no need for them (or the amount of needed joins is brought down to the minimum). Rather than joining (fetching object1 from application, having a look at a key of the other object stored within object one, fetching object 2) it is better to have de-normalized database structure. MongoDb has a native support for MapReduce pattern which allows very effective data evaluation and the post processing and therefore also further evaluation criteria may be applied on already characterized lenses without the need of 3rd party application carrying out the logic (as MapReduce can be directly implemented within MongoDb).

Chapter 9

Work breakdown structure and initial workload estimation

For planning the work break down structure standardized Gantt charts were utilized. The particular Gantt charts were left-out from the paper as they are subject to an existing NDA[6]. The main implementation of the features has been split-up into 4 milestones whereas each milestone comes with a different functionality. The extension of the software to comply with the automated characterization line is treated separately out of scope of these 4 milestones. Below there is a brief description of the functionality contained within each of milestones provided as well as initial workload estimation used for evaluating the overall costs. At the end of each milestone there is a release of application which is debugged and fully capable of production run. Overall milestone workload consists of estimated time for development, debugging and project management. Time for debugging has been taken as 10% of the time required on development and adjusted for relative difficulty level of each of the milestones whereas project management time has been estimated depending on a difficulty level of each of the milestones. Project management time shall include time spent on discussion with the product owners to clarify all potential questions and ambiguities which could arouse throughout the development.

9.1 Milestone 1

In the first milestone the attention shall be concentrated on the most important use cases offering the initial capability of software conduct characterization and quality assurance of products. Modular architecture shall be implemented within this one as well.

[6]Non-disclosure agreement

Activity	Time estimation[hours]
Development	684
Debugging	68
Project management	221
Overall	973

Table 9.1: Table depicts the workload of milestone 1

9.2 Milestone 2

The second milestone is short indeed. It only contains capability of reloading a session and therefore it allows end user to start a measurement at the one computer station and finalize it at the other one.

Activity	Time estimation[hours]
Development	48
Debugging	4.8
Project management	4.8
Overall	57.6

Table 9.2: Table depicts the workload of milestone 2

9.3 Milestone 3

In the milestone 3 features such as data filtration, reporting, and data viewing capabilities shall be implemented.

Activity	Time estimation[hours]
Development	489
Debugging	50.49
Project management	65.96
Overall	605.45

Table 9.3: Table depicts the workload of milestone 3

9.4 Milestone 4

This one is the very last milestone in which features such as user management and access restriction shall be implemented. Various improvements of the end-user experience are included in here as well, e.g. the possibility to adjust the workspace such as it could be bound with an exact user and after logging in at another software station it is going to display the same set of windows as the user configured at a different PC.

Activity	Time estimation[hours]
Development	352
Debugging	35
Project management	53
Overall	440

Table 9.4: Table depicts the workload of milestone 4

9.5 Automation extension

Automation mode has been estimated out of general milestones planning due to the fact that this extension of the system is demanded by the project which covers automated characterization line itself. Extension of the system for capability of handling multiple products at a time whereas acting as the master of whole characterization process has been estimated as follows in the next table.

Activity	Time estimation[hours]
Development	401
Debugging	30
Project management	20
Overall	451

Table 9.5: Table depicts the workload of automation extension

Chapter 10

Return on investment evaluation

To evaluate the overall return on investment it is necessary to estimate the cost of it upon estimations included in the previous chapter. After summing up the hours of each of the milestones and including automation extension the time needed for the implementation and establishment of the system – 2526 man hours is computed. This is the most likely estimate which is used as an input for more precise Three Point Estimation method.

10.1 Three point estimation

The three-point estimation technique is used in management and information systems applications for the construction of an approximate probability distribution representing the outcome of future events, based on very limited pieces of information. While the distribution used for the approximation might be a normal distribution this is not always so and for example a triangular distribution might be used, depending on the application.[8]

The output of such estimation is the weighted value E and standard deviation, SD, which are computed as follows:

E = (A + 4M + B) / 6

SD = (B A) / 6

A – The best case estimate

E = (A + 4M + B) / 6

M - The most likely estimate

B – The worst estimate

The output of the Gantt estimation stated in the previous section represents M – the most likely estimate of 2526 man hours. Considering the 20% margin for the best case estimate and the worst case estimate get to 2020 man hours for the best case estimate whereas the worst case estimate is 3031 man hours. This gives us results as shown below:

E = 2525.8

SD = 168.5

To get even more precise estimation the confidence level of 95% is calculated. For the simplification purposes only the approximate confidence level is computed which is close enough to the exact computation and therefor it's fine for the estimation purposes.

Confidence level = E +- 2* SD = 2526 +- 337

Therefore there is 95% chance the estimation will fall into the interval of <2189, 2863>man hours. When considering the same average wage used for quantifying the savings, the approximate cost is computed as <96316, 125972>CHF and the expected cost is as high as of 111 144 CHF.

10.2 Return on investment

A generic term to define a number of analytical tools for measuring the financial benefits of an investment, including cash-on-cash, internal rate of return, equity dividend and financial management rate of return.[9] When simplified it is nothing besides a factor depicting the benefit to the investor from a certain investment which is relative so it's easy to cross compare various investments. For the purpose of the thesis following formula is used to compute the return on investment:

ROI = (gain from investment – cost of investment) / cost of investment

The higher ROI gets the higher the efficiency of investment is. For evaluation purpose the ROI on a 7 year yield return is computed and the cost of maintenance per year is neglected.

ROI formula used for computations could eventually get negative which would mean developing the software would not bring sufficient savings therefore the overall project would produce lost. As the outcome of the table above there are multiple variants how the return on investment could evolve depending at the amount of characterized lenses per year.

Cost of lenses per year	3000	5000	10000	20000	50000
96 316 CHF	1.33	2.78	6.4	13.65	35.4
111 144 CHF	1.02	2.27	5.41	11.7	30.54
125 972 CHF	0.78	1.89	4.66	10.2	26.83

Table 10.1: Table shows various return on investment depending on the cost of software and the amount of produced lenses per year.

10.3 Internal rate of return

The internal rate of return (IRR) or economic rate of return (ERR) is a rate of return used in capital budgeting to measure and compare the profitability of investments. It is also called the discounted cash flow rate of return.[10]

This factor depicts the desirability of a project. The higher the value gets the more desirable the project proves to be. It can be also used to cross compare projects requiring the same initial investment. Considering the NPV discounted by the cost of capital we get to a real incomes which can be expected from an investment. IRR states the highest possible cost of capital which is affordable for the project to get at least to a 0 revenue (so the company doesn't lose money). So as soon as there's an investment opportunity which cost of capital is less than IRR, it is worth to take it. Therefore the investment shall produce revenue.

$$\mathrm{NPV} = \sum_{n=0}^{N} \frac{C_n}{(1+r)^n} = 0$$

Figure 10.1: Generic formula for IRR computation, one needs to express r out of it

To get the IRR it is needed to express r from the formula stated above. Further mathematical description is omitted as it's not the main topic of the paper. Below there's the table containing computed IRR values for different presumption of produced lenses.

Cost of lenses per year	3000	5000	10000	20000	50000
96 316 CHF	29%	55%	111%	217%	530%
111 144 CHF	23.1%	45.8%	95.3%	187.9%	459.5%
125 972 CHF	18%	39%	83%	165%	26.83

Table 10.2: Table depicts various IRR coefficients per different cost and produced lenses amounts.

10.4 NPV

For the computation of NPV the discount of 8% has been chosen as it is considered to be quite a productive area and the possibility of producing income in optics is high. In the table below there are net present values after 7 year horizon stated per different cash flows (depends on the amount of produced lenses per year).

Cost of lenses per year[CHF]	3000	5000	10000	20000	50000
96 316	67 559	163 716	404 109	884 895	2 327 252
111 144	53 829	149 986	390 379	871 165	2 313 522
125 972	40 100	136 257	376 650	857 435	2 299 793

Table 10.3: Table depicts various net present values in CHF.

After considering various economic aspects it is apparent the software will produce sufficient returns to pay back original investment. The returns are not including eventual competition advantage as well as possible interconnection with further data analysis system used for improving the process and the quality of company's products. In computations above the cost of maintenance has been neglected.

Chapter 11

Software summary

The thesis provides the overview of Optotune's characterization and quality assurance processes. Upon the basis of such a description the weak spots of characterization and quality assurance process are pinpointed and possible improvements by the software suggested. The software supporting mainly following activities has been designed:

- Integration of the most important processes in one place
- Ensuring the reliability of each characterization process
- Downswing of human impact at each product validation
- Automation
- Backward data evaluation

The designed software shall consist of multiple parts. To allow the inclusion of various products without the need of recompiling the project, modular architecture is employed. Modules will interact with the platform part, where the logic for data collection and evaluation shall be located, through exposed APIs. APIs are utilized by the core part as well as modules to exchange the information concerning characterization process, data access, data evaluation, etc... This architecture provides scalable concept easily extensible in the future for new products. As the application is collecting various types of data NoSql solution MongoDb is employed as the data storage. This approach allows an easy-to-maintain and horizontally scalable mean of data storage furthermore it has built-in features for data post-processing. The platform offers generic way of the characterization algorithm configuration (as each of the characterization and quality assurance algorithms may require parameters). User can configure various characterization action collecting the data from multiple external devices and afterwards assign pass/fail criteria onto each of them determining whether the certain quality criteria are met (these characterization actions and quality assurance criteria vary from product to product).

"A solid software foundation for the product characterization and quality assurance processes has been created. This provides a scalable way for maintaining various processes under a single hood allowing the centralization of complex logic. Additionally, this framework allowed us to minimize the system idle times and therefore to maximize the system throughput which in turn decreases overall production costs." Christoph Romer, CIO.

Conclusion

The goals of the thesis were:

- Analyze characterization and quality assurance processes of optical products
- Determine possible improvement areas via employing supporting software
- Design the software reflecting these enhancements employing NoSql solution MongoDb
- Evaluate benefits of such improvement considering the aspects of integration, automation, high reliability, and minimal human impact on the validation of each product
- Evaluate benefits and the cost of system via employing various economic markers, namely ROI, IRR, NPV

In the thesis the following has been done:

- Processes for characterization and quality assurance has been described from an abstracted manner which is sufficient for designing the targeted software
- Possible areas for improvements achieved via employing software solution were pinpointed
- Functional and non-functional requirements has been collected and analyzed. These requirements are mapped into the wireframes which can be used during the implementation.
- The software for empowering Optotune's characterization and quality assurance processes has been designed

- The architecture, design and the model part of the application is presented within the scope of this thesis whereas the emphasis has been put on the modularity and the scalability of the final solution

As the company is evolving the software needs to be capable of integrating further products into it with the minor intervention to the general framework. Designed framework offers capabilities for data collection and evaluation as well as covers the general communication with various external devices.

Upon the created analysis the software development time has been estimated and planned into 4 milestones and one special release for automation version (as the automation version is the part of another project). Each of the milestones brings different sort of functionality and at the end of each milestone the new release is created and debugged.

Upon the initial process analyses possible savings which could be achieved by the software are estimated. The savings estimation has been evaluated in various horizons, namely, 1, 3, 5 and 7 years. The overall savings brought by the software also depends on the amount of lenses characterized per year. As the exact prediction is the subject to NDA it's assessed for multiple assumed amounts (3000, 5000, 10000, 20000, 50000). The worst case savings issued by the software after 7 years has been estimated to 224 026 CHF whereas the best case savings after the same time period were estimated at 3 505 253 CHF.

At the very end of the thesis there's the economic asset of the software evaluated. For the worst case the cost has been estimated at 125 972 CHF, the best case estimation is 96 316 CHF. The return on investment, the internal rate of return as well as the net present value coefficients has been computed for various cash-flows depending on a different presumptions of sold lenses per year. The best-case ROI (the highest savings issued, the lowest possible cost achieved) shall be up to 35.39. The worst-case has been estimated at 0.77. It is apparent under any presumed circumstances the project will bring an additive value to the company as the ROI is always positive (negative ROI means project lose money). Currently the automation extension is under the development. The first 2 milestones are already finished and deployed for the production use. In the initial estimation only 3 products has been presumed to be characterized by the software. As the company is evolving fast, it has already increased into 5 products and in the near future it will be even more. As the integration of new products was seamless (without the need for the platform adaption), the initial architecture has proven to be scalable and robust enough.

Bibliography

[1] Wikipedia, opened encyclopedia. A simple zoom lens system. 02 2008. Available from: http://en.wikipedia.org/wiki/Zoom_lens#/media/File:Zoomlens1.svg

[2] Optotune AG. Speckle reduction measured with a CCD camera. Available from: http://www.optotune.com/images/stories/Laser-speckle-reduction-with-intensitiy-profiles.gif

[3] Optotune AG. Working principle of lens. Available from: http://optotune.com/technology/focus-tunable-lenses

[4] Optotune AG. EL-6-18. Available from: http://www.optotune.com/images/stories/EL-6-18.jpg

[5] Optotune AG. EL-10-30. Available from: http://www.optotune.com/images/products/EL-10-30-C.jpg

[6] Dainty, J. C. *Laser Speckle and Related Phenomena.* 1984, ISBN 0-387-13169-8.

[7] MongoDB, INC. MongoDb description. Available from: http://www.mongodb.org/about/introduction/

[8] Ministry of Defence. Three point estimates and quantitative risk analysis. 2007.

[9] Denise Evans, O. W. E. *The Complete Real Estate Encyclopedia: From AAA Tenant to Zoning Variancess and Everything in Between.* 2007, ISBN 0071476385.

[10] M.A.Main. *Project Economics and Decision Analysis, Volume I: Deterministic Models.*

Appendix A

Acronyms

EEPROM electrically erasable programmable read-only memory, is a type of non-volatile memory which is used to store small amount of data

RS232 standardized serial link communication interface

Session in the scope of this thesis session is considered to be the single characterization process instance. I.e. characterizing a single product and triggering various actions throughout the characterization process, each of these actions will be in the same session.

MongoDb open-source NoSql database solution

NoSql mechanism of storage and data retrieval which is other than ordinary relational database management systems

SDK software development kit. Encompasses the set of tools that allow the creation of software or simplify the integration of the 3rd party libraries.

MsPl Microsoft's open-source license

OptoTester the name of the designed software.

Man day A person's working time for a day. Optotune's standard working day consists of 8.5 hours.

Appendix B

Contents of enclosed CD

```
readme.txt ........................ the file with CD contents description
src ......................................... the directory of source codes
└── thesis .............. the directory of LaTeX source codes of the thesis
text ............................................. the thesis text directory
├── BP_Kredatus_Simeon_2015.pdf ........ the thesis text in PDF format
└── BP_Kredatus_Simeon_2015.ps ........... the thesis text in PS format
model .............. the folder containing model image of the application
└── model.png ....................... the image of the application model
```