Software to Support Laboratory-Scale Process Test
Abstract

The purpose of this study was to describe a construct of an application software designed to support laboratory test equipment for pyrolysis/coking process test equipment for the use at the Process Metallurgy Group (PMG) in the University of Oulu. The prior research in the fields of laboratory automation, usability in laboratory context, data gathering, operational safety and linking to larger laboratory IS, and a brief summary of design science research and it's methodology were presented. The study described the context of the software development, including the Process Metallurgy Group in the University of Oulu; and pyrolysis and coking processes. The design and the development processes of the PYROLYSIS software were described, as was the evaluation of the software. A model of hardware virtualization, application-device communication and the UI design were presented. Finally, a tentative model for remote alert system via SMS was presented.

Keywords

laboratory automation, software development, design science research

Supervisor

University lecturer, Ph.D. Raija Halonen
Foreword

This study was founded to my experience of working at the laboratory of the Process Metallurgy Group (PMG) at the University of Oulu. My personal motivation was to improve the design of the software used to support the experimental research in PMG. During the preparation of this thesis, a large number of people have provided invaluable contributions to this work.

I would like thank the whole staff of the Process Metallurgy Group of the University of Oulu, and especially D.Sc.(Tech.) Hannu Suopajärvi, M.Sc.(Tech.) Riku Mattila, M.Sc. (Chem.) Satu Huttunen and D.Sc.(Tech.) Eetu-Pekka Heikkinen. Their assistance in providing information of metallurgical processes and during the user interface testing was invaluable. Also, I would like to thank M.Sc.(Tech.) Juho Haapakangas for providing help during the final proof-reading of this thesis.

I would like to thank my supervisor, Ph.D. Raija Halonen, for guidance, support and encouragement during the research and writing processes. Also, I would like to thank M.Sc. (Tech.) Pertti Seppänen about his comments and suggestions during the final stages of the thesis.

Finally, I would like to thank my father Uolevi Kokkonen for his encouragement and support of my studies, and my beloved wife Satu for her constant support and loving care during this process.

This thesis is dedicated to the memory of

my sister Aila and my mother Aili.

Tommi Kokkonen

Oulu, April 17th, 2015
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COTS</td>
<td>Commercial Off The Shelf</td>
</tr>
<tr>
<td>IS</td>
<td>Information System</td>
</tr>
<tr>
<td>LIMS</td>
<td>Laboratory Information Management System</td>
</tr>
<tr>
<td>MSUI</td>
<td>Main System User Interface</td>
</tr>
<tr>
<td>PCP</td>
<td>Process Control Program</td>
</tr>
<tr>
<td>PMG</td>
<td>Process Metallurgy Group</td>
</tr>
<tr>
<td>SAB</td>
<td>System Advice Box</td>
</tr>
<tr>
<td>SMB</td>
<td>System Message Box</td>
</tr>
<tr>
<td>UCD</td>
<td>User Centered Design</td>
</tr>
</tbody>
</table>
## Contents

Abstract .......................................................................................................................... 2
Foreword .......................................................................................................................... 3
Abbreviations .................................................................................................................. 4
Contents .......................................................................................................................... 5
1. Introduction .................................................................................................................. 6
2. Prior research .............................................................................................................. 8
   2.1 From components to an automated system .......................................................... 8
   2.2 Usability of laboratory software ........................................................................ 10
   2.3 Requirements for measured data ........................................................................ 13
   2.4 Safety-critical software and preparing for emergency situations ...................... 14
   2.5 Part of a larger information system ..................................................................... 14
   2.6 Summary ............................................................................................................. 16
3. Research methods ...................................................................................................... 17
   3.1 Design science as research paradigm .................................................................. 17
   3.2 Seven guidelines for design science research .................................................... 20
   3.3 Three cycle view of design science research ...................................................... 22
   3.4 Summary ............................................................................................................. 23
4. Context ....................................................................................................................... 24
   4.1 Process Metallurgy Group as environment .......................................................... 24
      4.1.1 Laboratory and users .................................................................................. 24
      4.1.2 Previous and planned laboratory software .................................................. 25
   4.2 User requirements ............................................................................................... 26
      4.2.1 Pyrolysis and coking as industrial processes ............................................. 26
      4.2.2 Process test requirements for system ......................................................... 28
      4.2.3 Hardware requirements ............................................................................ 29
      4.2.4 Software requirements ............................................................................. 29
5. Construct .................................................................................................................... 31
   5.1 Design .................................................................................................................. 31
      5.1.1 Hardware .................................................................................................... 31
      5.1.2 Main software components ....................................................................... 32
      5.1.3 User interface components ....................................................................... 34
      5.1.4 Virtual devices for instruments .................................................................. 37
      5.1.5 Data logging and storage components ...................................................... 41
      5.1.6 Remote alert components ....................................................................... 42
   5.2 Implementation and testing ................................................................................... 42
   5.3 Evaluation ............................................................................................................. 43
      5.3.1 Architecture and component reuse ............................................................. 43
      5.3.2 Usability ..................................................................................................... 43
      5.3.3 Fault survivability and user safety ............................................................... 44
   5.4 Summary ............................................................................................................. 45
6. Discussion and implications ...................................................................................... 46
7. Conclusions ................................................................................................................. 50
References .................................................................................................................... 51
Appendix A. Software requirements ............................................................................. 57
Appendix B. Hardware................................................................................................. 58
Appendix C. Device communication protocols ............................................................. 60
Appendix D. MSUI ......................................................................................................... 62
Appendix E. SMB and SAB examples .......................................................................... 63
1. Introduction

The purpose of this study was to describe the construct of the application software (PYROLYSIS) designed to support a laboratory test equipment for pyrolysis/coking equipment utilized at the Process Metallurgy Group (PMG) at the University of Oulu.

This study was motivated by the need to improve laboratory automation software development at PMG. A well designed and developed software can assist the scientific experiment by automating the process control and by gathering the measurement data from different laboratory devices effectively. As the experiments in PMG can run overnight without local supervision, safety features designed into a system improve operation safety of the experiments. Improvements in usability have impact to efficiency by reducing training and diminishing possible user-borne mistakes.

The prior research describes the process of developing combined system from separate devices (Pollard, 2001) and provides examples of software architecture solutions (Thakur, Chen & Leister, 1999; Cuadrado, Luque de Castro & Gómez-Nieto, 2006). The importance of designing a good user interface is discussed and methods to achieve it are presented (Barrington, 2007), as are methods of testing usability (Holzinger, 2004). The importance of storing essential experimental data is well documented (Eurolab, 2006; OECD, 1995), as is the importance of data quality (Potthof, Walk & Rieger, 2013; Potthof, Lütjohann & Jung, 2014). Ideally test equipment is a part of a larger system designed for safe storage and effective utilization of measurement data (Frey, 2004; Rubacha, Rattan & Hosselet, 2011; Prasad & Bodhje, 2012). Safety critical software development is briefly discussed and methods for remote observation of experiments (Echols, Smith & Nirschl, 2004; Callaghan, Harkin, McGinnity & Maguire, 2008) and possibility of SMS alert during an emergency situation (Aziz, Muhamad, Wahab, Alias, Hashim & Mustafa, 2010; Jubadi & Sahak, 2009) are presented.

The focus of this study was an application software used to support experimental work at the PMG research laboratory. The research question was formulated as follows:

RQ: What kind of a laboratory software will support the pyrolysis and coking process test equipment of PMG?

Research topics included the development of laboratory software; usability issues; data quality; operational safety; and integration with a larger laboratory IS.

The study was a constructive research as nature and followed the methods of design science research as described by Hevner, Ram, March and Park (2004). A software artifact was designed, implemented and evaluated. Key paradigms were the three cycle view to design science (Hevner, 2007) and seven guidelines for design science research (Hevner et al., 2004). Software architectural design and component reusability were evaluated descriptively, comparing the construct against principles described in prior research and a prior version of software components developed at PMG. In order to evaluate the usability of PYROLYSIS software, end user testing was utilized as it was
considered to provide the most reliable assessment. Finally, operational safety of PYROLYSIS was evaluated by testing input field ruggedness, PCP safety and hardware malfunction tolerance.

In this study, a solution for laboratory automation was presented. A simple model for laboratory hardware virtualization was developed, as also a model for data transfer between application and hardware devices. In order to improve user response to exceptions during the process experiments, a notice system was developed to provide information of situation arranged in order of seriousness. Finally, a tentative model for remote alert via SMS was presented.

This study contains an overview of the prior research on laboratory automation software development. Key topics are the development of laboratory automation software; implementation of usability during the development process; quality of the gathered data; operational safety; and integration to a larger laboratory IS. This overview of the prior knowledge is followed by a presentation of research methods used in this study, describing the principles and evolution of design science research in IS science. The context of this study is described, giving a brief description of the Process Metallurgy Group (PMG) laboratory as a target for laboratory automation software development. Principles of pyrolysis and coking processes are described and key process requirements for both hardware and software presented. Construct chapter describes the design and the development of the PYROLYSIS software, and the evaluation of the construct against usability, software architecture and operational safety. The findings are presented and discussed with a reference to prior knowledge in the field of laboratory automation. Finally, conclusions are drawn, limitations of this study are discussed and topics for further research are presented.
2. Prior research

In the following chapters, prior research concerning laboratory automation from the point of view of software development is discussed. Topics cover general development of automated laboratory test equipment; effective user interface design; accumulation and storage of measurement data; handling of emergency situations; and linking the automated test equipment as part of a larger laboratory information system.

2.1 From components to an automated system

Pollard (2001) describes a development process for a basic laboratory automation project. According to Pollard, a typical laboratory equipment contains far more potential for automation than is generally realized. Addition to basic laboratory instruments, another requirement is a PC and an application software. This application software can either be obtained from a specialist vendor, or one can use do-it-yourself (DIY) -strategy. (Pollard, 2001.) Should the DIY strategy be selected, McDowall (2004) provides an extensive view to risk management in laboratory automation development projects, providing a listing of potential risk factors and a thorough treatment of each risk factor; description of various failure types; and a number of general rules to achieve satisfactory results from an automation project. Liscouski (2006) emphasizes an importance of the use of specialists in laboratory automation projects and promotes the term laboratory automation engineer.

From a hardware point of view, a typical limitation on laboratory automation is a lack of serial (RS232, RS485) ports. One solution for this is the use of USB-to-serial adapters. When selecting these adapters, it is recommended to evaluate them in terms of stability and robustness. (Bernlind & Urbaniczky, 2009.)

Thakur, Chen and Leister (1999) provide a basis for architectural design of laboratory automation software. They emphasize the use of component based software engineering. The key design principle is modularity: according to Thakur et al., a single "monolithic" system is more vulnerable than a component-based, causing the whole system to crash even in case of a single device failure. Chen, Thakur and Leister (1999) extend the discussion of Thakur et al. concerning the architecture of laboratory automation software. Chen et al. describe, among other things, configuration management; event, alert and communication management; information and log management; scheduling; and device control; and operation and monitor management.

Cuadrado, Luque de Castro and Gómez-Nieto (2006) provide a comprehensive description of the use of object-oriented software development (OOD) paradigm in laboratory automation. They describe both device and data modelling, using OOD methods as inheritance, polymorphism and encapsulation. Cuandraro et al. created a model for hierarchical representation of analytical laboratory automation according to definitions of IUPAC (International Union of Pure and Applied Chemistry), dividing instruments to logical categories and sub-categories. Figure 1 presents an example this model, where only two branches are fully modelled; and attributes and methods are
omitted. Cuandraro et al. have also modelled analytical data presentation so that all the relevant information for traceability is maintained.

Figure 1. Example of hierarchical modelling of laboratory hardware (Cuadrado et al., 2006).

Various development tools and programming languages for laboratory automation and application development have been reported in the literature. Use of Microsoft’s Visual Studio and its supported programming languages, Visual Basic, Visual C++ and Visual C# has been reported, among others, by Harkness, Crook and Povey (2007); Echols, Smith and Nirschl (2004); and Delaney, Echenique and Marx (2013).

National Instrument’s LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) is a system design software, providing tools ”... to build any measurement or control application in dramatically less time...” (NI, 2015). It is a graphical programming language, implementing a data-flow paradigm ie. instead of writing the source code the application is "drawn" by using graphical software components (Elliot, Vijaykumar, Zink & Hansen, 2007). LabVIEW components can also be combined with components developed with more "traditional" software development tools, for example with Visual Basic or C (Elliot et al., 2007). Some examples of the use of LabVIEW in laboratory application development are a micro-beam X-ray fluorescence spectrometer (Wrobel et al., 2012); an automated laboratory robotics system (Elliott et al., 2007); a sequential injection analysis - capillary electrophoresis (SIA-CE) system with UV detection, and an analytical setup for studying the inhibition of enzymatic reactions using a SIA system with FTIR detection (Wagner, Armenta & Lendl, 2010); and an automated hyperspectral imaging / multispectral imaging application (Wang, Li, Tollner & Rains, 2012).
According to the survey conducted in 2006 by the Association for Laboratory Automation (ALA) (Hamilton, 2007), Visual Basic and Excel were the dominant tools with 73% total share (see Figure 2) when developing applications for laboratory automation, in contrast to VB's and especially Excel's small usage in general software development when compared to more common C++ and Java. In March 2015, in general software development the most used programming language was C, Visual Basic .NET being 9th, Visual Basic 10th, and LabVIEW 50th. Hamilton (2007) considers the preference for LabVIEW and Excel at the expense of more "traditional" programming languages stemming from the fact that the automation community has a preference to the "simpler" tools.

Figure 2. Usage of different development tools for laboratory automation development in 2006 (Hamilton, 2007).

Finally, when using COTS (Commercial-Off-The-Shelf) software components for laboratory automation development, various evaluation processes have been presented for COTS selection, for example by Lin, et al. (2007); Comella-Dorda, Dean, Morris and Oberndorf (2002); Menon, McDermid and Hubbard (2009); and Jha and Bali (2012).

2.2 Usability of laboratory software

ISO 9241-11 defines usability (as cited in van Kuijk, 2012) as “The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.”

Barrington (2007) asserts that usability is not deemed an important subject when developing laboratory automation software, but is more of an afterthought: The technical solution is designed and implemented first and the UI is then fitted on top. Poor usability on laboratory software can, however, cause additional training costs; increased time on reading user manuals; expensive user errors; time wasted using an ineffective UI; complex features left unused; or an instrument that does not integrate

well with existing systems and working practices. Barrington emphasizes the design of usability early in a software development project, using principles of user centered design (UCD). The UCD process includes three phases: establishing the context of use; designing for usability; and evaluating usability. To define the context of use, intended users, tasks they are to perform and the environment in which the product will be used must be identified. According to Barrington, designing for usability includes 10 key points: (Barrington, 2007.)

- **Visibility of system status:** User should be kept informed what is going on and he/she should never be left in any doubt as to the state of the system or the results of his/her actions.

- **Match between system and the real world:** The interface should use terminology and concepts that are familiar to the user and relevant to the task of the system.

- **User control and freedom:** User should be able to navigate through system and interface without penalty – like loss of entered values – and should be informed clearly when exiting a form. User should be able to undo/redo any actions.

- **Consistency:** Language and structures should be consistent both internally within the interface and externally with any standards, industry conventions or practices in target laboratory.

- **Error prevention:** When possible, the UI should prevent user from making errors, especially if those errors can lead to serious consequences.

- **Minimize user memory load:** User should be presented with available options and required data formats and should not need to remember information of one part of the UI to another.

- **Flexibility and efficiency:** The UI should support both novice and experienced user. For example guides – like set up wizards – should be included for the novice user.

- **Aesthetics and minimalist design:** The UI should present important information clearly and should not swamp the user with irrelevant data.

- **Help user recognize, diagnose, and recover from errors:** Error messages should be plain, straightforward and devoid of any system codes or computer jargon. The user should be provided with clear description of problem and instructions how to either recover from an error or to obtain help.

- **Help and documentation:** Most laboratory systems will have sufficient features to warrant user documentation and a help system – for example a tool-tip system.

When designing the UI, the developer can use user interface design patterns (UIDP). UIDPs are well-documented user interface and user-system interaction solutions for known and frequently occurring user interface problems. These solutions have been shown to benefit end users; they have already been implemented, evaluated, and approved. Having been recognized successful, these patterns are generally collected and organized as collections. (Janeiro, et al., 2009.)
When designing safety-critical systems, number of specific UIDPs exists. Mahemoff, Hussey and Johnson (2001) have listed a number of design solutions safety-critical systems. Given patterns are divided to four groups: task management, task execution, information, and machine control. The following list provides an example of each group (Mahemoff et al., 2001):

- Stepladder (Task management): Complex tasks are split into a chain of simpler tasks.
- Behaviour constraint (Task execution): The system should prevent the user from requesting hazardous actions.
- Trend (Information): As humans are not good at monitoring, the system should compare and contrast the current state with previous states.
- Shutdown (Machine control): If shutting down the system is simple, inexpensive and leads to a safe state, low risk state, the system should shut down in event that a hazard occurs.

Connelly, Burmeister, MacDonald and Hussey (2001) expand and modify pattern presented by Mahemoff et al. (2001). An example of implementing UIDPs to radio therapy equipment is presented with justifications to each pattern (Conelly et al., 2001).

According to Barrington (2007), the last stage in UCD process is the evaluation phase. Holzinger (2004) provides a more thorough description of different usability testing methods. Holzinger divides these methods into two categories: usability inspection methods, used without end users; and test methods, used with end users.

Usability inspection methods include heuristic evaluation (HE), cognitive walk-through (CW) and action analysis (AA), all of these having multiple versions. In heuristic evaluation, usability experts go through each dialogue or interactive UI component and inspect whether the established usability principles are satisfied. Non-experts can also be used, but the results tend be not as good. In a CW test, a step-by-step simulation of user behaviour to complete a given task is made. CW test can be made by using only usability specialists or by using mixed team of software developers, usability specialists and end users. In AA methods, emphasis is made on what the testers do, by using either detailed or more cursory observation of actions taken to complete a given task. (Holzinger, 2004.)

Test methods include thinking aloud (THA), field observation, data logging and querying the users. As with usability inspection methods, test methods have multiple versions. In THA the test user, while using the system, thinks out loud. Thus verbalizing their thoughts, the users provide information how they view the system. THA can provide important information why users do something. Field observation is the simplest of all methods. The key is to act as unobtrusive as possible, lest to cause false results due interference of the test users. Data logging provides a detailed information of timing; or statistical analysis of usage or error. Finally querying user by interviewing or by using questionnaires provides overall impression of the usability of UI. (Holzinger, 2004.)
2.3 Requirements for measured data

According to Potthof, Lütjohann and Jung (2014), to ensure the quality of scientific data, the integrity and authenticity of the data has to be guaranteed, and the data must be traceable to the original measurement data. They emphasise digital signing of data to verify its authenticity. Similarly, Potthof, Walk and Rieger (2013) state that a data management in scientific and research process needs to comply to rules of good scientific practice (GSP). According to Potthof et al. (2013), a long term storage of data should be enabled, both data and meta-data should be stored, and a care should be taken when selecting a format of data.

A number of principles, standards and regulations exists which address the recording and storing measurement data during scientific experiments. Among these are GLP, ISO/IEC 17025:2005 and FDA Title 21 CFR Part 11.

The phrase Good Laboratory Practice (GLP) refers to a number of methods to be used to maintain good quality on laboratory research. One description of GLP is formulated as “Good laboratory practice (GLP) is a standard by which laboratory studies are designed, implemented, and reported to assure the public that the results are correct and the experiment can be reproduced exactly, at any time in the future. In less technical terms, GLP is the cornerstone of all laboratory-based activities in any organization that prides itself on the quality of the work it performs” (Biopharma, 2003). Different institutions have provided formal standards for the implementations of GLP principles. OECD provide multi-part directive for implementation of GLP (OECD, 1998) and included to this, a directive of the application of GLP to computerised systems (OECD, 1995). Hassler et al. (2006) provide an excellent guide to implement research data recording according OECD GLP regulations.


In pursuance of GLP and its requirements for electronic data, Food and Drug Administration (FDA) of the United States of America has established a set of regulations on electronic records and electronic signals (FDA, 1997). This set of regulations provides a basis of electronic data gathering and defines a criteria under which the electronic data can be considered as valid as traditional paper records. Although the set of standards is mainly targeted to pharmaceutical research, it has a peripheral effect to any measurement data recording on automated laboratory systems. As the original Title 21 CFR 11 documentation (FDA, 1997) was considered as unclear and difficult to apply to practice, FDA published a guidance documentation of the implementation of regulation (FDA, 2003).
2.4 Safety-critical software and preparing for emergency situations

According to Knight (2002) plenty of definitions exist for the term safety-critical system. He argues that an intuitive definition for safety-critical system works fairly well. Whether system is safety-critical derives from the consequences of the failure: if the failure of system can lead to unacceptable consequences, then the system is deemed safety-critical. Knight divides safety-critical systems into four categories. Traditional systems are those which have traditionally been considered as safety-critical. These include commercial aircraft, nuclear power plants or medical care systems. Failure in these systems can cause life-threatening danger. Non-traditional systems are those which are not normally deemed as safety-critical, yet in case of failure serious damages and even loss of life can incur. Knight (2002) gives a telephone system as an example: a long lasting failure of emergency call system can cause loss of life. By system design and manufacturing Knight refers to an effect of tools used to create safety-critical systems. Finally, information system safety is critical, as attacks against both public and private networks can have devastating effects. Knight sees that in future the number and type of safety-critical systems will increase, especially as embedded computer systems become more common.

Software robustness can be defined as “the extent to which software can continue to operate correctly despite the introduction of invalid inputs” (IEEE, 1990). At a minimum, the robust software must handle inputs, that are out of range; are of wrong type; or are in wrong format, without degradation of those functions not dependant of aforementioned inputs (Pullum, 2001, pp 27). In the case of wrong inputs, the following strategies can be adapted: request a new input; use the last acceptable value for input variable; or use pre-defined value for the input (Pullum, 2001, pp 27). Design of fault-tolerant laboratory automation is discussed by Thakur, Chen and Leister (1999). Pullum (2001) provides various programming techniques to achieve faulty-resistant software.

In case of remote observation of experiment status, Echols, Smith and Nirschl (2004) provide an example of web-based system which can monitor the status of more than hundred analytical instruments. Although the instruments are controlled by proprietary software provided by the instrument vendor, a glue-code is applied to provide an interface between the instrument applications and web-components (Echols, Smith & Nirschl, 2004). Another example of remote control of laboratory experiment is provided by Callaghan, Harkin, McGinnity and Maguire (2008). The use of SMS to alert of critical situations have been reported by Aziz, Muhamad, Wahab, Alias, Hashim and Mustafa (2010); and Jubadi and Sahak (2009).

2.5 Part of a larger information system

Primary data gathered during research has great influence for all experiment-driven interpretation (Frey, 2004). To store and utilize this data, the use of Electronic Laboratory Notebooks (Rubacha, Rattan & Hosselet, 2011), grid-like information structures (Frey, 2004), LIMS (Prasad & Bodhje, 2012), and semantic web (Frey & Bird, 2013) provide some solutions to the problem.

Electronic laboratory notebooks (ELN) are becoming more common and are replacing paperbound laboratory notebooks in research. Although earlier frowned upon by governmental organizations, due to the pressure from industry ELNs are accepted as
valid media for storing research records. In 2011, at least 35 different commercial ELNs existed in market. Primary market audience groups for ELN are research & development (R&D); biology; chemistry; quality assurance and quality control; and multi-discipline researchers. (Rubacha, Rattan & Hosselet, 2011.)

Frey (2004) describes a “dark laboratory” or a fully automated laboratory where laboratory instruments “talk” with each other with no human intervention. According to Frey, the challenge is not how to enable automatic experimentation and data gathering but how integrate software, people and experiments to form a more effective environment for research. In perfect world all the data would be available to a researcher, but in reality there exist limitations to a full access. Firewalls, access control, communication between different systems, and effective storage of data and meta-data are some problems to be solved. Frey (2004) sees a grid-like IS with an effective middleware as a solution to this problem. The middleware layer allows more uniform style of access to the resources with quite different underlying systems. The fully developed grid infrastructure enables the researcher to “sit in the centre of a virtual world with simple, rapid access to a wide range of physical, computational, and informatics resources”, as Frey (2004) concludes.

A laboratory information management system (LIMS) is a software package that can manage laboratory samples, users, instruments, and data (Piggee, 2008). The roots of LIMS were in laboratory automation and first steps towards LIMS systems were taken in 1973 (Gibbon, 1996). According to Prasad and Bodhje (2012), before late 1970's a laboratory sample data management and an analysis results reporting were manually handled which is an error-prone process. First commercial, first generation (1G) LIMS became available in 1982, consisting a single minicomputer with automated reporting tools. Since then, LIMS has progressed through second (2G), third (3G) and fourth generation (4G) information systems, and isolated systems have evolved into a client-server based architecture supported by data-base systems. Both closed and open-source software solutions are available, and a choice of platform includes both Microsoft and Linux/UNIX operating systems. The latest generation of LIMS are no longer limited to test and sample information management, but provide an all-encompassing laboratory data management. Prasad and Bodhje (2012) conclude that a current trend in LIMS development is in user-friendliness and integrated data management solutions.

Semantic web is a term presented by Berners-Lee, Hendler and Lassila (2001) and its idea is to include the information in web with semantic content, turning the web into a “web of data”. Considered as a revolution (Berners-Lee et al., 2001), using published articles as a yardstick it has matured over time into a well-established IS discipline (Stuckenschmidt, Schuhmacher, Knopp, Meilicke & Scherp, 2013). Frey and Bird (2013) see Semantic Web as a “middle-ground” between a bunch of files stored on personal workstations by researcher, and a DBMS and/or LIMS solution. According to Frey and Bird, Semantic Web as a solution is more controlled than storing plain files but less rigid than DBMS or LIMS. Frey (2009) describes application of Semantic Web to the three normal research life-cycle phases: the planning phase, the execution phase, and the dissemination phase. During the planning phase, the Semantic Web can speed up the planning process by providing the researchers means to find others interested of the same problems; to find people with skills needed for the research problem; or to provide funding for the research project. During the execution phase, Semantic Web techniques, especially meta-data embedding, enable to gather experimental data in such a way that it can be used globally and is easily searchable. Finally, Semantic Web
techniques can make a final report of the research more searchable, especially when gathering information for further research project. (Frey, 2009.) Frey and Bird (2013) list multiple future key development areas, including data management and integration; discovery and access control; metadata; vocabularies, data provenance; and tools for searching data.

2.6 Summary

When developing application software for an automated process test equipment, one should take into account various different requirements, including hardware and software requirements; usability; requirements for data and its storage; possible emergency situations; and broader usability of measurement data. Figure 3 summarizes some factors described by prior literature referred in Chapters 2.1, 2.2, 2.3, 2.4 and 2.5:

![Figure 3. Some factors affecting to laboratory automation software development.](image)

The figure is not an all-encompassing presentation of all the factors concerning the laboratory application development, but it gives a quick glance of the multiformity of the problem.
3. Research methods

In the following chapters, IS Design Science as research paradigm is described, including different “schools” of IS design science. Different main-stream theories are briefly described. More in-depth description for design science research (Hevner, Ram, March & Park, 2004; Hevner, 2007) is provided.

3.1 Design science as research paradigm

Iivari (2007) provides an extensive analysis of IS as design science. Ontology, epistemology, methodology and ethics of design science are discussed. From methodological point of view, Iivari states that as much of research in computer sciences consists of constructing artefacts, term constructive research should denote specific research methods required for constructing artefacts. To differentiate between practice of building IT artefacts and IS design science, Iivari encourages to try to specify reasonably rigorous research methods.

According to Carlsson (2007), IS design science has two main schools: Information Systems Design Theory (IDST) and Design Science Research (DSR). IDST is described by Walls, Widmayer and El Sawy (1992) and falls outside boundaries of this study. Different views to DSR theory and principles have been described by Nunamaker, Chen and Purdin (1991), March and Smith (1995), Hevner, Ram, March and Park (2004) and Cao et al. (2006). Key similarity between these four DSR theories is their multimethodological approach to IS research.

In their article, Nunamaker et al. (1991) present a multimethodological model for IS research (Figure 4). The model includes four research strategies – theory building, systems development, observation and experimentation – and interactions existing between these strategies:

- **Theory building**: This includes, for example, activities of developing new ideas, concepts, methods or models. Theories can be used to suggest research hypothesis, guide the design of experiments and conduct systematic observations.

- **Experimentation**: Category includes laboratory tests, field experiments and different simulations. Experimentation can provide results to be used to refine theories and improve systems.

- **Observation**: Strategy of using unobstrusive methods like case studies, field studies and sample surveys. Observation can help the researcher to get a general feel of the research problem.
• **Systems development**: This consists of five stages – concept design, constructing the architecture of the system, prototyping, product development and technology transfer. In this model, systems development is a hub, interacting between all other strategies.

Of the four research strategies presented in this model, none can be considered to stand alone. Instead, all strategies are complimentary, providing feedback to other strategies.

![DSR model](image)  
**Figure 4.** DSR model (Nunamaker et al., 1991).

March and Smith (1995) describe a two-dimensional framework for IT research. They provide and define four basic outputs (artefacts) of design research: construct, model, method and instantiation. These four form the first dimension of their model. Second dimension is formed by combining two basic natural science activities – theorize and justify – with two basic activities of design research – build and evaluate. Forming two axes from these dimensions creates their DSR model presented in Figure 5. According to the model, any of design research artefact can be assessed against the four basic science activities:

- **Build**: Artefact is built to demonstrate its feasibility and provide an object for evaluation. The basic question is "Does it work?".

- **Evaluate**: Artefact is evaluated to determine whether any progress has been made.

- **Theorize**: Based on evaluation of artefact, theory or theories can be developed explaining its characteristics and interaction with its environment.

- **Justify**: Any theories developed must tested by gathering additional evidence to support or reject them.
Model devised by March and Smith has greatly affected to later models presented by Hevner at. al. (2004) and Cao et al. (2006).

<table>
<thead>
<tr>
<th>Research activities</th>
<th>Build</th>
<th>Evaluate</th>
<th>Theorize</th>
<th>Justify</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constructs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Instantiation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5.** DSR model (March & Smith, 1995).

Hevner at. al. (2004) extended the model of March and Smith by adding two domains - environment and knowledge base (Figure 6). Environment consists of people, organizations and technologies relevant to design research process. Environment provides a problem – business needs – which design science process solves by developing an artefact. Knowledge base contains prior knowledge applicable to this particular design problem, to be used during development and evaluation of an artefact. A successful artefact feed back an application to environment and additions to knowledge base, Hevner et al. conclude.

Hevner et al. (2004) provide also seven guidelines for DSR and this model is refined by Hevner (2007). These are to be described more thoroughly in Chapters 3.2 and 3.3.

**Figure 6.** DSR model (Hevner et al., 2004).

Cao et al. (2006) derive their model from models described by Nunamaker et al. (1991), March and Smith (1995), and Hevner et al. (2004). The model combines two research paradigms – design science and behavioral science (Figure 7).
It endeavours to show and to describe three different kinds of interaction between these two domains:

- **Build-theorize**: Building an artefact can act as a proof of a claim made by behaviour science theory. Or vice versa, theories found on behaviour science domain can aid on designing new artefacts.

- **Result-result**: Results from system evaluation methods can interact with interpretation of results from theory testing, and vice versa.

- **Result-design**: results or findings from a research method used in system evaluation interacts with research design of an another method used in theory testing, and vice versa.

The model created by Cao et al. (2006) strives to point out that technology and behavior are inseparable. IS research should be conducted in an integrated way, consisting of multiple research activities and methods. Combining the two research paradigms provides more comprehensive understanding to the problem at hand.

### 3.2 Seven guidelines for design science research

Hevner et al. (2004) describe seven guidelines for DSR as a basis which researchers and other stakeholders can use either to plan research or to evaluate it. The guidelines are presented next.

*Guideline 1: Design as an artefact*. Design science research must produce an artefact: a construct, a model, a method or an instantiation.

*Guideline 2: Problem relevance*. There must be a real-life, important business problem, to which DSR produces a solution. This is provided by environment domain (see Figure 5).

*Guideline 3: Design evaluation*. The designed artefact must be rigorously evaluated, so that the artefact’s quality can be assessed. Evaluation can be made by applying strict
mathematical comparisons if applicable metrics are present. For example, quality of a new algorithm can be compared to the older models by making test runs in comparable environment. If direct numerical evaluation is not suitable, other relevant attributes can be used, for example functionality, usability, customer satisfaction or reliability. Different evaluation strategies are summarized in Table 1.

Table 1. Different evaluation methods (Hevner et al., 2004, p. 86).

<table>
<thead>
<tr>
<th>Observational</th>
<th>Case study</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Field study</td>
</tr>
<tr>
<td>Analytical</td>
<td>Static analysis</td>
</tr>
<tr>
<td></td>
<td>Architecture analysis</td>
</tr>
<tr>
<td></td>
<td>Optimization</td>
</tr>
<tr>
<td></td>
<td>Dynamic analysis</td>
</tr>
<tr>
<td>Experimental</td>
<td>Controlled experiment</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
</tr>
<tr>
<td>Testing</td>
<td>Functional (black box) testing</td>
</tr>
<tr>
<td></td>
<td>Structural (white box) testing</td>
</tr>
<tr>
<td>Descriptive</td>
<td>Informed argument</td>
</tr>
<tr>
<td></td>
<td>Scenarios</td>
</tr>
</tbody>
</table>

Guideline 4: Research contributions. Effective DSR must provide clear contributions in the areas of the design artefact, design construction knowledge and/or evaluation knowledge. Three types of contributions can be found from DSR, of which one or more must be present in results:

- **The design artefact**: Most commonly the contribution of DSR is artefact itself. It must provide a novel solution to a prior unsolved problem or it must prove to be improvement to earlier solutions.

- **Foundations**: Creative and well evaluated constructs, models, methods or instantiations that extend and improve the existing foundations in the design-science knowledge base are important contributions.

- **Methodologies**: Results can provide new methods which to use during evaluation phase of future DSR. Especially measures and evaluation metrics are valuable addition to knowledge base.

Guideline 5: Research rigor. Both construction and evaluation of the artefact must be conducted applying rigorous methods. Rigor in this context means the effective use of knowledge base, techniques, evaluation methods and metrics to develop and assess the artefact.

Guideline 6: Design as search process. Taking into account the problem environment, an effective solution to the problem must be searched. The search process commonly
follows generate-test-cycle: a version of artefact is created, followed by evaluation and new ideas to be included to next version to be evaluated.

**Guideline 7: Communication of research.** Results of the research must be effectively presented to the research community. Target groups include both technology-oriented and management-oriented audiences. To technology-oriented audience, the report must provide detailed instruction, so that applicability of solution can be estimated and if required, artefact implemented. Management-oriented audience requires information of artefact's feasibility to the organization: should organization commit itself to construct or purchase the artefact.

The previous seven guidelines provide a backbone against which a DRS research project is to be planned, conducted and evaluated. All of the items must be present on the research project and therefore the list forms a kind of check-list for researcher to undergo when evaluating the research.

### 3.3 Three cycle view of design science research

Hevner (2007) refined further DSR theory. As in Hevner et al. (2004), the three domains are described: environment, design science research and knowledge base. Environment consist of people, organizational systems and technical systems. The real-life, important problem required by design science is found from this domain. Knowledge base contains relevant scientific theories, expertise and experience, and meta-artifacts, or to put it shorter: what is already known from previous research. Third domain, design science research is the development-evaluation process, where researcher develops artifacts and evaluates results. Figure 8 presents a graphical version of the model.

The three cycles in this model are relevance cycle, design cycle and rigor cycle. Relevance cycle describes interaction between business environment and design science research. The environment provides research problem which is to be solved by designing the artefact and if needed, a platform for field test or case study. The artefact then “returns” to business environment, providing new techniques and means for business and can also provide a new research problem to be studied.

**Figure 8.** Three cycle view of design science research (Hevner, 2007).

Rigor cycle is the interaction between knowledge base and design science research process. From knowledge base a researcher obtains the prior research knowledge concerning the research problem at hand. When new information is found during design
science research process, it is reported to the research community as described in Guideline 7 on chapter 3.2, enlarging knowledge base regarding area of research.

Design cycle is the process, where a researcher will develop and evaluate artefacts. Evaluation will provide new ideas which are incorporated to design, creating an improved artefact and this will be evaluated in turn, forming cycle-like development process.

3.4 Summary

Multiple models of DSR exists, later models deriving influences from previous ones. All previously presented models have a common aspect, a multi-methodological approach to IS research (Cao et al., 2006).

![Figure 9. Three cycle view of design science research adapted to reflect laboratory software development. Adapted from Hevner (2007).](image)

A three cycle view of design science research (Hevner, 2007) provides an elegantly condensed model, describing domains, key processes and key interactions of DSR. Figure 9 shows this model modified to reflect laboratory automation software development.
4. Context

In the following chapters, the Process Metallurgy Group as a target environment is described and requirements for the application software are presented as they were at the time of this study.

4.1 Process Metallurgy Group as environment

The Laboratory of Process Metallurgy in the University of Oulu was established 1991 as a subunit of the Faculty of Technology. In 2014 the unit was renamed as the Process Metallurgy Group, referred hereafter as PMG. In 2015 its mission is summarised as follows: “The mission of the Laboratory of Process Metallurgy[sic] is to carry out research on metallurgical and other high-temperature processes and to produce for the metallurgical industry diploma engineers and doctors, who understand and master the phenomena of high-temperature processes in the metallurgical industry. The research activities of the laboratory based on the understanding of reactions, reaction kinetics, thermodynamics and mass and heat transfer phenomena in high-temperature processes.” (PMG, 2015).

4.1.1 Laboratory and users

Process research methods in PMG includes numerical modelling, water models, high temperature experimentation and supporting analytical methods i.e. optical and electro-optical microscopy, thermoanalytical (DSC-TG) methods, mass spectrometry, Raman spectrometry, XRF and XRD. Numerical modelling includes fluid flow calculations and thermodynamic calculations conducted by using either commercial or in-house developed software. Research in PMG is largely done in conjunction with metallurgical industry which provides research problems to be solved. According to the previous professor Dr. Jouko Härkki, research in PMG “most resembles the work in engineering office”. The PMG laboratory has following special characteristics:

- **Use of high temperatures:** Temperatures during high-temperature experiments generally vary between 600°C and 1700°C, with the maximum attainable temperature of ca. 2000°C.

- **Long lasting experiments:** Process experiments often last more than 8 hours and it's not uncommon that they take more than 20h. During long experiments it is not possible to maintain local supervision during the whole test.

- **Use of hazardous gases:** Typical reduction metallurgy experiments include carbon monoxide and hydrogen as reactive gases.

- **Heavy reliance to custom-built test equipment:** Typically a research project require development of a custom test equipment with included control software.
Permanent technical staff in laboratory includes two persons, whose responsibilities are maintenance of equipment; design, implementation and purchase of new research equipment; guidance of students and researcher in use of laboratory equipment; and if needed, conducting the experiments for researchers. Most laboratory experiments are conducted by undergraduate students in their diploma thesis studies or post-graduate students during their doctoral studies. As their technical skills vary, heavy emphasis is placed on effective orientation and usability of laboratory devices.

4.1.2 Previous and planned laboratory software

Software used in PMG laboratory can be broadly divided into two categories: vendor supplied and custom-build. Vendor-supplied software consists of applications which are delivered by equipment vendor as a part of standard commercial laboratory analysis equipment. No source code is available and any upgrade or modification must be handled through vendor. Custom-built software is tailor-made to specific task and only to be used in PMG laboratory. It is built either by in-house development or by outsourcing software development. In either case, the software package includes source code and can be later modified by PMG. As pyrolysis/coking process test equipment development project is a topic of this research, it will be discussed separately. The following listing of PMG laboratory process test equipment with built-in software is not inclusive as some disused equipment and software are omitted.

**Blast Furnace Simulator (BFS)** is a device designed to model physical and chemical reactions of a blast furnace used in iron making process, but it is also used for other high-temperature experiments. Users are mainly post-graduate students. Main hardware components are high-temperature furnaces, an analytical grade scale, gas flow control units and a PC. The BFS provides automated dynamic temperature control between 25°C and 1200°C with simultaneous dynamic atmosphere control for nitrogen, carbon dioxide, carbon monoxide, hydrogen, water, potassium and sulphur, and sample mass measuring. A typical sample size is 100 g or 100 cm³. Experiments are conducted using user-defined process programs. The BFS was developed by outsourcing software development, using Turbo Pascal as development environment with some COTS software components included. The BFS control software was designed to run under Microsoft Windows 2000 and can be presently considered as a legacy system. Tentative plans have been made for software and hardware modernization project, but due to the fact that the system works satisfactorily and is constantly in heavy use, upgrade is not deemed as a high priority. For more detailed description of BFS system and example of use, see Iljana et al. (2012).

**TGA 2.0** is a custom-built thermogravimetric analyser, used both for research and teaching. It is a lighter version of the BFS, consisting of high-temperature furnace, gas flow control units, an analytical grade scale and a PC. Sample size is typically 5g or 5 cm³. Like the BFS, TGA 2.0 provides automatic user defined dynamic control of temperature between 25°C and 1500°C, gas atmosphere with maximum of 6 separate gases with additional gaseous water, and sample mass measuring. TGA 2.0 control software was developed as in-house project using Visual Studio 2010. Source code is in C# and TGA 2.0 was designed to run under Windows 7. Based on user feedback, a next version under project name TGA 3.0 is under development. Key target for development
is hardware and software usability, as TGA 2.0 is used by students with short orientation.

**Layer Furnace System (LFS)** is a process test equipment designed to simulate progress of reaction front in sample bed of metallurgical raw material. The LFS consists of four high-temperature furnaces stacked vertically and controlled individually, gas flow control units, infra-red gas analyser unit for measurement of carbon dioxide, carbon monoxide and hydrogen content, and PC. Control software provides a dynamic temperature and atmosphere control with automatic timed gas sampling and analysis from six sampling locations. The LFS control software was developed as in-house project using Visual Studio 2010. Source code is in C# and LFS was designed to run under Windows 7.

**Optical dilatometer** is a test device designed to measure dimensional changes of sample material and/or wetting angle between molten sample and solid sample disc on variable static atmosphere. Main components of the system are high-temperature furnace, a CCD camera and a PC. System includes in-house developed automatic dimension/wetting angle measurement software with project name DAKOTA. DAKOTA was developed using Mathlab and designed to run under Windows 2000. The dilatometer system is near obsolete with immediate requirements for either custom development of a new version, or purchase of off-the-shelf high-temperature dilatometer with included software.

**EffArcSystem** is a coming process test equipment development project and is designed for high-temperature research of electric conductivity within sample material. EffArcSystem consists of high-temperature furnace, gas flow control units, electric current and voltage measurement device, and a PC. As of February 2015, hardware has been designed and manufactured, a tentative concept for GUI has been developed and software development is planned to start at May 2015. System will be developed by using Visual Studio 2010 and C#. Software development project aims to re-use software components, especially virtual device controllers, developed for coking/pyrolysis software.

When comparing the different process test equipment used by PMG, it is possible to recognize some common software components. Typically any process test equipment includes furnace temperature and gas flow control devices and these are promising targets for software reuse. Other possible reusable components include process program parsing, process variable control, data gathering and storage, and data visualization.

### 4.2 User requirements

In the following chapters, pyrolysis/coking process tests equipment user requirements are described. Pyrolysis and coking processes are briefly described and user requirements are given from viewpoints of processes, hardware and software.

#### 4.2.1 Pyrolysis and coking as industrial processes

Pyrolysis can be described as thermal decomposition of material in absence of oxygen. Depending from starting materials, pyrolysis yields charcoal, aliphatic and aromatic volatile organic compounds, carbon monoxide, carbon dioxide and hydrogen. Pyrolysis can be divided into fast, intermediate and slow pyrolysis based on heating rate used in
the process. Heating rate amongst other things effects total yield and ratio of different chemical components. (Suopajärvi et al., 2013)

Interest in charcoal (Figure 10) production has increased since charcoal is a renewable, highly reactive and truly clean carbon source. Compared to fossil energy sources, charcoal contains virtually no sulphur or mercury. (see Antal & Grønli, 2003)

![Charcoal produced by pyrolysis](image)

Figure 10. Charcoal produced by pyrolysis (H. Suopajärvi, used with permission.)

The blast furnace (BF) process is the most used technology to produce hot metal for steelmaking purposes. The core process in BF involves reduction of iron oxides into metallic iron by using carbon and hydrogen containing reducing agents. Typical materials for this are coke, heavy oil, pulverized coal, natural gas and hot reducing gas (Suopajärvi et al., 2013). Metallurgical coke (Figure 11) is the main carbon source and has five important functions in BF process (see Yanga et al., 2014):

- coke carbon is the main source of energy
- carbon and carbon monoxide reduce iron oxides
- carbon carburizes the hot metal
- coke creates a permeable support for the burden
- coke acts as a gas distributor

The coke making process is defined as the heating of natural, organic, mostly solid materials in an oxygen deficient atmosphere in order to concentrate the carbon. In the coking process, coal and possible additional raw materials are heated gradually up to 1100°C in coke batteries. Due to the geometry of the coke battery, a reaction front advances through material, inducing physical and chemical reactions. Typical coking process lasts ca. 20h. Addition to coke, other products include coal tar, methane and hydrogen (see Babich & Senk, 2013)
New environmental regulations and sanctions have created pressure to reduce carbon dioxide emission by lessening the use of fossilized raw materials, use of recycled plastic during coking process is researched. Typically, about 1% of coal can be substituted with plastic without any adverse effects to coke. (see Kato et al., 2006)

4.2.2 Process test requirements for system

Based on information on pyrolysis and coking processes, process requirements for the test equipment are as follows:

- **Process control**: During an experiment, the system must follow a process program, which defines process variable settings in relation to process time.

- **Atmosphere control**: During both pyrolysis and coking processes, at least minimum an inert atmosphere must be maintained. If test are to be conducted with variable atmosphere, multiple gas flow control units are required. An automated gas flow rate and composition control under the process program control is required.

- **Heating and temperature control**: Maximum process temperature of ca. 1250°C should be attainable, and heating/cooling rates must be controllable. An automated temperature control under the process program control is required.

- **Recovery of distillate**: Pyrolysis and coking processes provide complex mixtures of liquid compounds as valuable by-product. Analysis of these by-product gives valuable information concerning the process. A condenser/distillation unit with coolant circulation is required to recover by-products from outlet gas.

- **Lengthy experiments**: Maximum process time can be in excess of 20h and no local supervision can be provided during the whole of test run. The system must be fully-automatic and robust with some built-in fault tolerance and if feasible, a remote alert system should be included.

These requirements are to be considered stable, with no additional requirements expected.
4.2.3 Hardware requirements

Derived from process test requirements and user requirements, following hardware requirements are presented:

- **High-temperature furnace**: Furnace with maximum temperature greater than 1250°C with remote control capability. If feasible, the furnace should also include possibility of creating a thermal gradient, simulating a reaction front advance in real processes.

- **High-temperature reaction vessel**: A hermetically sealed reaction vessel for pyrolysis and coking. The vessel must be manufactured from suitable refractory material with temperature resistance up to 1300°C with a gas-tight lid with inlets and outlets for reaction gases.

- **Gas flow controllers with remote control**.

- **Temperature data logger**: Multi-channel temperature logger with suitable measurement probes and remote control.

- **Condenser/distillation unit**: Dual-stage condenser and distillation unit for recovery of by-products.

- **A PC with required communication ports and wiring**: A PC is to be used to provide user a control of hardware devices; to provide automatic control of process test; and to gather and save the measurement data.

Hardware requirements were implemented by using existing laboratory devices whenever possible. Custom-built metal components were mainly provided by engineering works at the Faculty of Technology and glassware for condenser/distillation unit were purchased.

4.2.4 Software requirements

By combining the information obtained through interviews of users; from prior research of laboratory automation; and from process and hardware requirements, a following description of software requirements is created. Following listing is in informal format. For more detailed presentation of user requirements for the software, see Appendix 1.

- Software architecture is to be based on modular design and component reuse should be implemented whenever it is possible. Feasible principles described in Chapter 2.1 should be followed in design and implementation.

- User interface must comply with principles described in Chapter 2.2 and must provide functions defined by the user.

- Application controls pyrolysis/coking process variables by using user-defined process program divided into segments. Tool for creating and editing process programs must be included to the main application.
- Visualization of process variables, both set and actual values, and process test data during experiments should be provided to the user.

- Measurement data must comply with both requirements of research and laboratory standards presented in Chapter 2.3. Data must be saved both locally to hard-drive and to a remote repository on PMG RAID server. Data import tool with data formatting is to be provided.

- Application should be fault-tolerant and in case of failure, user must be provided with accurate information about the source of failure and its seriousness. When feasible, principles and methods presented in Chapter 2.5 must be implemented into software.

The software was developed as an in-house project and without using any COTS software components. Development was made in close co-operation with the user and by following principles described in Chapters 3.2 and 3.3.
5. **Construct**

In the following chapters, design, implementation and evaluation of pyrolysis/coking process test equipment are described.

5.1 **Design**

The design chapter describes the design process of the application software for pyrolysis/coking process test equipment. The software development project was named as PYROLYSIS and the same name was used of the application. As knowledge of the hardware is essential for the design of the application software, a description of the hardware is presented in Chapter 5.1.1, followed by description of application and its components in Chapters 5.1.2 - 5.1.6.

5.1.1 **Hardware**

Hardware was based as widely as possible to existing devices in the laboratory. When necessary, items were ordered from laboratory suppliers or manufactured in the workshop of the Faculty of Technology. Following listing describes main hardware items:

- **Furnace**: Dual-chamber high-temperature furnace manufactured by Entech with MoSi$_2$-thermal elements was selected. Its maximum effective temperature is approximately 1600°C and the furnace is controlled with Eurotherm 2408/2208e paired temperature controllers enabling either master-slave -mode or individual furnace chamber control. The controllers use a Modbus communication protocol over RS485 line. For description of Modbus, see Appendix 3.

- **Reaction vessel**: Custom-made crucible was ordered from Haldenwanger GmbH, manufactured of Halsic-RX silicon carbide. Maximum temperature resistance was 1500-1600°C, depending of atmosphere and the process material load. A lid for the vessel was manufactured by the workshop of the Faculty of Technology. The lid included an inlet and an outlet for process gases and nine ducts for K type thermocouple probes. The ducts acted also as suspension accessories for an internal thermal insulation shield. The inlet, the outlet and the ducts were fitted with standard Swagelok fittings.

- **Gas flow control**: Three Brooks 5850s mass flow controllers were used. The controllers use HART communication protocol over RS232C or RS485 line and in this construct, RS485 multi-drop was used. For description of HART, see Appendix 3.

- **Process temperature logging**: Hewlett-Packard, currently Agilent, 34970A data logger using 34901A 20-channel multiplexer card was used. The data logger enabled temperature data retrieval from maximum of 20 separate channels by using K type thermocouples provided by Sarlin. Communication to the logger
was handled by using SCPI protocol over RS232C line. For description of SCPI, see Appendix 3.

- **Condenser/distillation unit:** Glassware for condenser/distillation unit was provided by Lenz Laborglas GmbH. The unit was assembled from standard laboratory items with ground joints and PTFE sleeves as to facilitate modifications of the unit during research projects.

- **Controlling PC:** Commercial laptop unit was used. As the PC had a single serial ports, one RS485 line was set up by using Nokeval DCS770 USB-to-RS485 adapters and one RS232C by using Sunix 1009B USB-to-RS232 adapter. Both adapters were routinely used in PMG.

- **Remote storage:** QNAP TS-469 Pro network-attached storage (NAS) was used. NAS was equipped with four SATA hard drives configured to RAID-5 array.

Process test equipment was assembled to the laboratory premises of PMG. For the technical drawings and the photo of the process test equipment, see Appendix 2.

### 5.1.2 Main software components

Software was designed to consist of two packages: the main package and the virtual device package. Reasoning for this was software reuse. The main package contained mostly non-reusable components, which were to be typically tailored to suit each process control application. The virtual device package however contained reusable software components which were to be utilized with minor modifications or with no modification at all in following software development projects in PMG. Figure 12 presents the overall structure of the PYROLYSIS software and its key components.

![Figure 12. Structure of the PYROLYSIS software.](image)

Main package was an entry point of the application and therefore it was compiled to executable file (.EXE). Virtual device package was compiled as a library file (Dynamically Linked Library, .DLL). Table 2 summarises packages and their functionalities:
Table 2. Software packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pyrolysis.exe</td>
<td>• user interface components, including PCP editor and SMB/SAB components</td>
</tr>
<tr>
<td></td>
<td>• process control program parser (PCPP)</td>
</tr>
<tr>
<td></td>
<td>• system watchdog components</td>
</tr>
<tr>
<td></td>
<td>• data storage and system log components</td>
</tr>
<tr>
<td>DeviceControl.dll</td>
<td>• serial communication tools, including GUI components to set communication settings.</td>
</tr>
<tr>
<td></td>
<td>• virtual devices (furnace and gas flow)</td>
</tr>
<tr>
<td></td>
<td>• XML communication information parsing tools</td>
</tr>
</tbody>
</table>

In addition to .EXE and .DLL files, PYROLYSIS included two settings files. These files were in XML format, as it was considered to be the most flexible format. Table 3 summarises these files and the information they contained.

Table 3. The settings files.

<table>
<thead>
<tr>
<th>Setup file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settings.xml</td>
<td>• various local and remote data path definitions</td>
</tr>
<tr>
<td></td>
<td>• hardware and communication settings</td>
</tr>
<tr>
<td></td>
<td>• safety limits for devices</td>
</tr>
<tr>
<td></td>
<td>• installed gases</td>
</tr>
<tr>
<td>BrooksGasCoefficients.xml</td>
<td>• gas flow correction coefficients used for internal calculations when using Brooks mass flow controllers</td>
</tr>
</tbody>
</table>

Process Control Program Parser (PCPP) was designed to convert Process Control Program (PCP) into instructions to hardware. PCPP takes an user-defined XML file as input. These files were user defined and in XML format. Process Control Program (PCP) is segmented and each segment contains the following information:

- **Segment duration in minutes**
- **Dynamic temperature profile.** Profile is defined as a polynornical function \( T = c_0 + c_1 t + c_2 t^2 + \ldots + c_n t^n \), where \( T \) is target temperature in Centigrades and \( t \) is elapsed time from the start of current segment in minutes.
- **Static gas atmosphere.** Total flow rate and gas composition is defined. Maximum of three gases can be defined.

Segments were to run in sequence, until all the segments have been completed and the program terminates.

Finally, *ProcessWatchdog* component was designed to provide safety during the use of PYROLYSIS in automated mode. It has following triggers:

- **Gas flow control failure:** If one or more gas flow controllers are in “FAIL” state, an emergency is triggered.
- **Temperature control failure**: If furnace control is in “FAIL” state, an emergency is triggered.
- **Data logger failure**: If the data logger component is in “FAIL” state, an emergency is triggered.
- **Furnace overheat**: If either furnace chamber overheats, an emergency is triggered.
- **Sample overheat**: If three or more sample thermocouples indicate overheating, an emergency is triggered.

For description of “FAIL” state, see Chapter 5.1.4. When an emergency is triggered, *ProcessWatchdog* informs the user and if possible, initiates independently an emergency system shut-down.

### 5.1.3 User interface components

Design of the user interface was started as a first task of design/implementation and the designing of the UI was made in co-operation with end users. The process followed the design-cycle described in Chapter 3.3. Basic principles to UI design and the role of UI in emergency situations were taken into account.

The GUI design was made by using Visual Studio 2010 as both design and demonstration tool. To hasten the design process, no paper prototyping was used. Following the design cycle paradigm, rapid design-evaluate -cycles were used (see Figure 13). During evaluation, test persons were used. These test persons were potential future users for PYROLYSIS software. General layout, UI components and use of colours were tested and the feedback from the users provided basis for additional development.

![Figure 13. UI design cycle.](image)

Main System User Interface (MSUI) was designed to fit approximately 2/3 of 1920×1080 display and it was situated to the left edge of the available display area. Remaining 1/3 of the display is reserved to the System Message Boxes (SMB) and Situation Advice Boxes (SABs). Figure 14 shows the general layout of the user interface. The principle was that the SMBs or SAB should not mask the MSUI or vice versa. MSUI is described in Appendix D.
System Message Box system was designed to inform user concerning different status changes. SMB messages were divided into three different categories according to descending order of situation gravity:

- **Normal message**: These types of messages informed the user about non-critical situations. Typical message is a notice about a normal termination of process program after all the program sequences have been completed.

- **Caution message**: The caution message refers to a situation when something unexpected had occurred but either the system was able to handle the situation or the problem is not significant to warrant the termination of the experiment. Typical example is a malfunction of a single thermocouple used to measure temperature of the sample material during the experiment.

- **Warning message**: The warning messages refer to a situations where the user must provide immediate action due to the serious and even potentially dangerous situations. Typical example was loss of communication between PYROLYSIS software and the high temperature furnace.

SMBs were designed to follow consistent colour theme and were implements to maintain topmost position in display in relation to other active applications. Table 4 summarises the design of three SMBs. For graphical presentation of SMBs, see Appendix E.

<table>
<thead>
<tr>
<th>SMB type</th>
<th>Description</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Provides routine information of experiments. Does not require a user intervention.</td>
<td>Green + white</td>
</tr>
<tr>
<td>Caution</td>
<td>Provides information of less-than-serious situations. User intervention is optional.</td>
<td>Yellow + black</td>
</tr>
<tr>
<td>Warning</td>
<td>Provides information of serious and possible hazardous situations. User intervention is required.</td>
<td>Red + black</td>
</tr>
</tbody>
</table>

**Figure 14.** General GUI layout of PYROLYSIS
The colour selection was designed after a survey to the test users. Also, as PMG employed at least one staff member with deuteranopia, the colour selections were tested to be discernible by using conversion to grey-scale image.

By pressing “HELP” button in SMB, the Situation Advice Box (SAB) for the current exception opened. The SAB provided the user with step-by-step instructions how to cope with the exception. For example of SAB related to a warning SMB, see Appendix E.

When a SMB is created, MSUI adds it to a container. If more SMBs are created, MSUI handles the visibility order (layering) of SMBs according to following rules:

- More serious message goes over less serious one(s). Therefore warning is always over caution and caution always over normal message.
- In case of SMBs with same level, older goes over newer MSB. Thus, the user is not distracted with new MSB when solving the current problem unless the newer message requires more urgent attention.

In case of emergency, the user focus on topmost message and reacts to it. After the topmost exception is resolved, then the user shifts one's attention to the next SMB and proceeds until all the SMBs are acknowledged. MSUI has various sub-components which were designed to assist the user in normal tasks. These sub-components include:

- **PCP editor**: Tool is used create or edit PCP files.
- **Settings editing tool**: Tool provides user access to various application settings, including data paths and hardware communication settings.
- **Auxiliary measurement point definition tool**: This tool is used to mark optional temperature measurement points sited in the test equipment.

To achieve robustness, key input fields in MSUI and its subcomponents were provisioned with acceptable values. For example when setting temperature for furnace chamber, if user attempts to enter value which is too high or low the system warns the user and the value is not accepted.

Finally, an emergency shut-down button was included to the UI. For the graphical design, a very common metaphor of mechanical emergency shut-down switch was used. In Figure 15, examples in “real-life” equipment and the emergency shut-down button of PYROLYSIS are shown.

![Figure 15](image)

**Figure 15.** Real-life emergency switches (left and center) and a graphical representation used in PYROLYSIS.
When pressed three times within 5 seconds, an emergency shut-down was initiated. During the emergency shut-down, heating elements of the high-temperature furnace were to be switched off and gas flow is switched to inert gas, either nitrogen or argon depending on the installed gases.

5.1.4 Virtual devices for instruments

Virtual devices were the most valuable re-usable components in development project, as most of the experimentation in PMG utilizes some form of gas flow, either as an inert protective atmosphere or as a reactive gas during the experiment process, and temperature controller to control furnace temperature. Previous software solutions were already made in PMG and these components formed a basis for design of new components. A design-evaluation development cycle was used until a workable solution was attained. A class model for laboratory instrumentation was developed and is shown in Figure 16. Public attributes marked in the following UML diagrams were C# public properties.

Main abstract class was Device. Device implemented only a single attribute, status. This enumeration reflected to one of possible five real world situations: off, initializing, online, shutting-down or failed. Table 5 shows these states for Device and their descriptions.
Table 5. Basic states of Device and their description.

<table>
<thead>
<tr>
<th>Instrument status</th>
<th>Description of status</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>Instrument is turned off</td>
</tr>
<tr>
<td>INITIALIZING</td>
<td>Instrument is initializing and although working correctly, is not presently usable. Indicates that the application/user must wait before the device is ready.</td>
</tr>
<tr>
<td>ON</td>
<td>Instrument is ready to use.</td>
</tr>
<tr>
<td>SHUTDOWN</td>
<td>Instrument is shutting-down. Indicates that application/user should wait until shutdown process is completed.</td>
</tr>
<tr>
<td>FAIL</td>
<td>Instrument has failed for any number of reasons. This includes also a possible communication failure to hardware device.</td>
</tr>
</tbody>
</table>

Device status can be used to keep the user and the application aware of current operational situation of the hardware: whether the device is available to use immediately or after an initialization, or the device has failed.

For message transfer between application and laboratory hardware, a DeviceMessage class was designed. This class contains both outbound message from application to hardware device and inbound reply from hardware to application. Addition to these, class also contains an identifier which DeviceCommunication object uses to forward the reply to the sender Device object. Both message and reply are in a form of byte array.

Device class also defines two abstract functions UpdateStatus() and MessageIn(). Both implementations were left to any particular subclass, as tasks varied from device to device. UpdateStatus() was designed to get status information from hardware devices during those command cycles when no other, more important communication is required. UpdateStatus() has DeviceMessage object as return value. MessageIn() method takes in DeviceMessage object containing a reply byte array from the hardware device and parse it to obtain information about the device states.

Most common communication technologies in use of PMG were RS232 and RS485. Some devices used USB, but these were rare: during writing of this study only two devices out of about 25 different laboratory devices used a genuine USB-communication, the rest relying to a serial communication by either RS232 or RS485. To enable computer-device communication, a set of software components were implemented.
Figure 17. A class diagram of communication components.

Figure 17 presents a class diagram of communication components designed for the PYROLYSIS system. An abstract \texttt{DeviceCommunication} superclass forms a basis for different subclasses implementing serial, USB or network communication to laboratory hardware. The core of the class is a command queue working in first-in-first-out principle. If the queue is empty, \texttt{DeviceCommunication} object cycled through its container of Device objects calling \texttt{UpdateStatus()} method of each device on its turn. When messages are added to the queue, cycling through \texttt{UpdateStatus()} methods is paused and the messages in queue are passed from application to device(s) in first-in-first-out principle. After the command queue is empty, the \texttt{UpdateStatus()} -cycle is continued. Rationale for the use of \texttt{UpdateStatus()} was to utilize communication as effectively as possible. In typical process control situation, active adjustment of device encompass approximately 10% of data transfer between application and hardware, the rest being typically queries about the state of the device: current furnace temperature or gas flow rate.

\texttt{SerialDeviceCommunication} class inherits \texttt{DeviceCommunication} and provides tools for serial (RS232 or RS 485) communication. In Figure 18, a typical communication example between \texttt{SerialDeviceCommunication}, \texttt{Eurotherm2408} and hardware device is presented as a sequence diagram.

Figure 18. Example sequence diagram of communication between software and hardware components.
In Figure 17, the application calls Eurotherm2408 object to execute a command to the hardware device. Eurotherm2408 object passes a DeviceMessage object containing the command bytes to SerialDeviceCommunication object to be passed to the hardware via serial port. The reply bytes from the hardware are then added to the DeviceMessage object and DeviceMessage object is then passed back Eurotherm2408 object, which parses the reply bytes and alters its internal states to correspond the information contained in the reply.

In PGM, instrumentation used for gas flow control varied from a simple, straight-forward 1-in-1-out -type controllers to more complex, multi-gas generators. The MFC controller in Figure 19 describes the most simple form of a gas flow controller. Both input and output of the controller are the same gas and virtual device just makes necessary adjustments to the flow rate. The potassium gas generator in Figure 18 is more complex. The generator provides a test process with gaseous potassium, which is generated in high-temperature reaction between potassium carbonate and carbon according to the high-temperature chemical reaction (1):

$$K_2CO_3 + C \rightarrow K_2 (g) + 3 CO (g) \quad (1)$$

Nitrogen acts as a carrier gas and does not react during the reaction between potassium carbonate and carbon. The case is then 1-in-3-out -system.

In Figure 18 a sulphur gas generator is presented, which uses high-temperature chemical reaction (2) to provide elemental, gaseous sulphur to the process.

$$SO_2 + 2 C \rightarrow S (g) + 2 CO (g) \quad (2)$$

Nitrogen is used as carrier gas to expedite SO₂ flow and the system is 2-in-3-out. Therefore gas controllers used by PMG are n-in-n-out models. Older version of gas controller virtualization used in PMG did not take this into account but was designed only to support single gas output. To more accurately reflect the real-life situation, a GasFlow class was designed to provide information of output gas composition and flow rate as this information was considered vital when calculating gas mixtures used in process. GasFlow includes total flow rate of gas mixture and its composition. In addition, GasFlow class was designed to include overloaded subtraction, addition, division and multiplication operators to facilitate more complex gas composition and flow calculations.
Figure 19. Gas flow controller and gas generators used in PMG.

High temperature furnaces were virtualized as *Furnace*-class. *Furnace* inherits *Device* and adds two abstract methods, *SetTemperature()* and *GetTemperature()* as these functions are common to all temperature control devices. *SetTemperature()* is used to set target temperature for the furnace and *GetTemperature()* to retrieve the furnace actual temperature. As high-temperature furnaces can have multiple, independently controllable chambers, both methods were designed to include index to indicate, which chamber is affected. *Furnace* class was then inherited by subclasses representing actual temperature controllers. These subclasses were designed to implement more numerous, equipment-specific controls of hardware, for example:

- Power level
- PID control settings
- Hardware limits
- Sensor-break fall-back procedures

The *DataLogger* class was the hardest to virtualize, as the configuration of device varies greatly from task to task. The solution used was to design a class with only very rudimentary internal components and heavily customize it case-by-case by using inheritance. In case of PYROLYSIS, a customized *PyrolysisLogger* class was designed to suit this application. The *PyrolysisLogger* was not designed for reuse and was therefore implemented as a part of PYROLYSIS.EXE-package.

### 5.1.5 Data logging and storage components

*DataWriter* class was designed to provide measurement data logging and *LogWriter* to provide event logging during the experiments. During run time, a *DataWriter* object is instantiated, enabling logging of measurement data to both local and remote storage. Two *LogWriter* objects handle user comments and systems exceptions to both local and, in case of user comments, to remote data storage.

*DataWriter* was designed to store required measurement data to provide measurement traceability. Measurement data was designed to include date and time stamp and the measurement data is preceded with operator initials and a short description of the
experiment. User comments were designed to complement the measurement data by providing additional information of the progress of the experiment. Finally, the system exception log was designed to provide additional information in case of failed experiment.

As PMG did not employ LIMS nor any other general laboratory information storage system, RAID server was used as remote storage media. This simplified the implementation of both DataWriter and LogWriter but lacked the possibilities of more elegant storage solutions.

5.1.6 Remote alert components

Remote alert was designed to apply only for the warning messages ie. those messages which indicated critical failures in the system. Designed remote alert system consisted of SMS message component, which sends a SMS to listed technical personnel via internet based SMS provider. The SMS alert was also designed to integrate with ProcessWatchdog component (see Chapter 5.1.2). The alert SMS was designed to provide the same information as SMB, but without SAB information. Key principle of remote alert system is presented in Figure 20.

![Figure 20. Planned SMS alert system.](image)

Due to tight time constraints during the software development project, implementing SMS alert was planned to be included later on to PYROLYSIS 2, the next version of PYROLYSIS. Also, remote alert software components designed to PYROLYSIS were intended to be utilized on other software development projects of PMG.

5.2 Implementation and testing

Actual coding of PYROLYSIS was made during March 2015. Visual Studio 2010 was used as development tool and the application and libraries were implemented by using C# and .NET framework components. The selection of programming language was based on previous experiences of the author and was also the most commonly used programming language in PMG.

Component testing was conducted continuously during implementation phase. First tested components were UI as their testing required additional personnel. Other software components were tested during development phase either by integrating them into already developed and tested components or by using special test tools.
Full system test was not possible, as due to problems with delivery of high-temperature furnace control components. Simulated testing was however conducted by using Eurotherm 3508 controller. Although different model than the actual temperature controller, the communication protocol is the same and differences arise only from addresses used for device. During this simulation testing, other hardware components were the same defined in Chapter 5.1.1. The serial communication to hardware was tested by using specific serial port logger tools, which enabled byte-by-byte checking of both outbound and inbound data.

5.3 Evaluation

In the following chapters, evaluation of PYROLYSIS software is described. Main targets for evaluation were architecture and component re-use; usability; and fault survivability and user safety.

5.3.1 Architecture and component reuse

Architecture and component reuse were estimated against the future use in PMG. Generally it was considered to be an improvement when comparing to older version in use during the time of writing. Following improvements were detected:

- Integration of virtual devices into a single integrated structure.
- Implementation of general communication control class capable of handling different devices in a same RS485 line.
- Improvement of gas flow controller virtualization to encompass more complex gas generators used in PMG.
- The new design was estimated to be more reusable when compared to the previous version. The .DLL -library provides a basis for addition of new device controlling software components.

Hence the designed architecture was considered to be a workable model for a use in PMG test equipment development.

5.3.2 Usability

Usability of PYROLYSIS was tested by using PMG technical staff and researchers as test personnel. Three test users were used. Test was conducted according following phases:

- *Introduction phase*: Test user was familiarized with PYROLYSIS user interface. Key functions were described.
- *Test phase*: Users were instructed to test the user interface and to simulate normal operations during set up of experiment, during normal operation and during emergency situations. The test users were observed during this phase.
- *Feedback phase*: Users provided feedback about the user interface. Feedback phase was conducted by interview.
Following general estimations were obtained from the test users.

- PYROLYSIS UI was evaluated to be clear and minimalistic. No unnecessary components were observed.

- UI was estimated to a low learning curve. Tasks were easy to complete and generally users had no problems select correct functions. Notable was that no test person proposed addition of tool-tips to UI components or addition of integrated user manual.

- SMB/SAB system was found to be useful addition, especially for more inexperienced users.

Following critics and proposals for improvement were obtained during feedback interview.

- **Lack of real-time dynamic graphs.** These were estimated to be helpful to observe trends during the experiment and to maintain situation-awareness of the experiment process.

- **Emergency shut-down switch operation:** The emergency shut-down button function received both positive and negative comments. Also, during the test phase, the emergency button was only UI component requiring additional instructions of its use. Critics were mainly aimed to three-click-triggering and to replace it, key-click combination or single-click activation were suggested.

- **Integrated check-list for non-automated components:** Pyrolysis test equipment includes also non-automated hardware components, which user has to initialize manually. User feedback suggested integration of check-list window to MSUI to certify that these actions have been done. This check-list is shown before the start of PCP and the user must complete the list by checking all items before PCP is started.

Based on the comments obtained during the user evaluation, PYROLYSIS UI was estimated to be suitable for use in the pyrolysis and coking experiments conducted in PMG laboratory.

### 5.3.3 Fault survivability and user safety

Fault-survivability and user safety was evaluated by using following testing methods:

- **User input evaluation:** Input fields were tested against improper inputs. The system should not accept any input values which could potentially endanger either the system or users. Special care was taken to test PCP editor tool against the possibility of preparing potentially dangerous process programs. No such defects were noted.

- **SMB/SAB components:** The SMB/SAB system was tested by launching multiple simultaneous simulated warning, caution and normal SMB messages. The system should arrange the SMB according to principle presented in Chapter 5.1.3 and the SMB components should be over any other application window.
Based to the results of the tests, SMB/SAB system was evaluated to works according to the design principles.

- **Process Control Program (PCP) safety:** PCP parsing was tested by providing manually made PCP which included out-of-range values and badly formed XML markup. The system should detect such values and prevent the use of defective PCP. Based to the results of these tests, the system was evaluated to be safe.

- **Hardware failures:** Testing was conducted by cutting the main power of hardware devices or simulating such a situation. The system should respond as planned, providing appropriate SMB for the user and terminating the automated experiment if applicable to the situation. According to these tests, PYROLYSIS was evaluated to be safe against hardware malfunctions.

Based on above tests, PYROLYSIS was estimated to be safe for use in PMG laboratory environment. Remote alerting system should, however, be included to the next version.

## 5.4 Summary

The design and the development project of the PYROLYSIS software was made by using a develop-evaluate cycle described in Chapter 3.3. The development project was started from UI design and proceeded with development of application-hardware communication with required software components. During UI development, user feedback was utilized for evaluation process. Special emphasis was targeted towards the design of virtual devices reflecting the hardware devices used in laboratory experiments. After development of successive version a suitable solution was found. To achieve a safe system, both input field checking and a process watchdog were designed and implemented.

Evaluation process concentrated into three main areas: usability; architecture and reuse; and fault survivability and user safety. Usability was tested by using potential end users to evaluate quality of user interface. The UI was generally considered to be of good design and the test group provided number of improvements to be included into a next version. Both component architecture and component re-usability were considered to have been improved when comparing to previously used version. Fault survivability and user safety were estimated to adequate for use in PMG during long term experiments, although remote alert system should be included to the next version.
6. Discussion and implications

The focus of this study was to describe a laboratory software, which was to support the pyrolysis/coking process test equipment of PMG. Research question was formalized as “What kind of a laboratory software will support the pyrolysis/coking process test equipment of PMG?”. The study followed the principles of design science research presented by Hevner, Ram, March and Park (2004) and Hevner (2007).

The PYROLYSIS software was developed by do-it-yourself (DIY) method, as the software had to be custom-built and the outsourcing was estimated to have no advantages. McDowall (2004) discuss the merits of buying a commercial solution versus developing the system in-house and in case of in-house development, provide an excellent risk-assessment tool. Based on assessment rules by McDowall, the PYROLYSIS software development project was estimated to be that of a low risk. The factors which lowered the risk were well defined project scope and deliverables, no need of multiple interfacing computer systems, straight-forward system requirements and low demand of resources. The PYROLYSIS project had an additional advantage having the software development personnel continuously in close contact with the end users, enabling quick response to any information request. Pollard (2001) provide an example of automating a process test equipment by starting from a very basic settings. In this study, the starting point was previously developed laboratory automation software systems, enabling a rapid start. The use of Visual Studio 2010 and C# as development tools was made from personal preference, as LabVIEW was also available for development and it has provided good result in laboratory automation software development (Elliott, Vijayakumar, Zink & Hansen, 2007; Wagner, Armenta & Lendl, 2010; Wang, Li, Tollner & Rains, 2012).

Barrington (2007) emphasises the importance of early planning of the UI. Therefore, during the PYROLYSIS development process, the UI design was the starting point of the development project. The UI development followed the cyclic process defined by Hevner (2007) by design followed by evaluation. By integrating end users early to UI design process, the UI was tailored to suite the need of PMG technical and research staff. Comments and suggestions obtained from the test users gave invaluable contributions and provided novel ideas. Compared to earlier used development procedure of the software developer designing UI with limited or non-existent feedback from the users, early integration of users to the UI development process improved the usability of the end product. Real-life metaphors were used when applicable. Thus, the emergency button was designed to relate a real life versions of emergency switches used for example in power tools common in the Faculty of Engineering. This follows Barrington's (2007) instruction to match UI components with real-life counterparts.

Mahemoff, Hussey and Johnson (2001) provide number of design patterns for safety-critical system UI design and the PYROLYSIS system applied number of these patterns. The “Shutdown” pattern was used to provide the user or the system watchdog means to quickly run down the experiment in case of potentially hazardous situation. “Behaviour Constraints” were placed to user input fields to restrict the user using out-of-limit values for the system. Based on the user feedback, a dynamic graph system employing
the “Trend” pattern was decided to be included to the next version of PYROLYSIS. Addition to these UI safety features described by Mahemoff et al. (2001), System Message Box (SMB) stack was designed to provide the user relevant information of test process, especially exceptions during the experiment process. In case of emergency, the SMB stack was designed to provide the user required information in order of importance. Thus, the user was to be able to concentrate one's full attention to the most serious problem in the top of SMB stack and ignore less serious SMBs. The user can then proceed through the exceptions without the need to use time for evaluation in which order one should proceed. In case of less experienced user, Situation Advice Box (SAB) was designed to provide the user step-by-step instructions for a problem solving. The SAB system was considered important as the technical expertise of the user can vary greatly and it follows Barrington's (2007) rule of providing the user the means to solve the problem.

PYROLYSIS components were divided into two categories, reusable and non-reusable. The UI components, excluding the SMB/SAB system, were considered yield only a few reusable items and solutions to be passed on to a new software projects in PMG. Therefore the UI components were packaged as an executable file instead of as a library. Similarly, ProcessWatchdog, DataWriter and LogWriter components were considered to be tightly bound to the specific requirements of the pyrolysis and coking processes and were packaged into the executable. As an afterthought, as the SMB/SAB system was estimated to be a reusable component, the development of a UI component library containing the SMB/SAB components for the use in PMG would pay dividends. The main reusable components were the device virtualization components. As the hardware devices used for the pyrolysis/coking process test equipment were to be used in other test equipment development projects, existence of the completed software component library expedite the system development process.

The software components developed to virtualize hardware devices were organized to a tree-like, three-tier mode. An abstract superclass Device was derived to sub-classes virtualizing different basic laboratory device types having common traits, for example (furnace) temperature or gas flow controllers. These were further derived to manufacturer and model specific classes, implementing a broader set of device functions, including the communication protocols. Cuadrado et al. (2006) provide a similar solution to model laboratory hardware. Their model is a more data-centric and is designed to follow commendations of IUPAC, especially to maintain traceability of calibration and analysis results (Cuadrado et al., 2006). Compared to the model of Cuadrado et al. (2006), the model designed in this study for the use in the PMG laboratory was more simplified. Three tiers were considered a satisfactory solution without need to further divide the laboratory equipment into sub-categories. Both the model of Cuadrado et al. (2006) and the model designed for the PYROLYSIS follow the basic tenets of object-oriented software design and development.

One key design and development area was the communication between the PYROLYSIS application and the hardware. The communication was arranged so that if no other, more significant commands were to be transmitted to hardware, the available bandwidth was to be used for status updates of the devices. By designing a general communication control class to handle the data transfer, linking of multiple separate devices to the same physical line was made possible. It was also noticeable that although most devices implemented numerous commands or comparable instructions, generally it was necessary only to implement a fraction of these. In PYROLYSIS
project, only two commands were implemented to both furnace temperature controller and gas flow controllers. This greatly simplified the development process of virtual controllers, cutting time required for design, development and testing. As modern computers have a limited number of serial communication ports, Bernlind and Urbaniczky (2009) recommend a thorough selection process for USB-to-serial adapters. The RS232 and RS485 adapters used in conjunction with PYROLYSIS were models used previously in the PMG laboratory and their robustness was well tested.

The GLP (OECD, 1998) emphasises the storage of raw data emanating the experiment so that the measurement data is traceable back to the experiment and the operator. To enable these requirements, the PYROLYSIS system was designed to include date and time stamping of raw data. Also, operator identification and short description of the experiment were enabled to be inserted to the data file. The tools designed to log any abnormal behaviour of the PYROLYSIS system during the experiment improve the overall measurement data quality. The measurement data gathered and logged during the experiment was considered to be sufficient by requirements provided by PMG. Potthof, Lütjohann and Jung (2014) describe a digital signing process to guarantee the integrity of the measurement data. No need for digitally sign the measurement data in PMG was deemed necessary, although this requirement could change should PMG pursue an accreditation at a later stage.

PYROLYSIS was not integrated into a larger IS, as at the time of the writing PMG did not have any LIMS nor any other comparable laboratory IS solution in use. According to Frey (2004), a laboratory with integrated measurement data storage and management systems enables more efficient use of data. The utilization of a RAID server in the PYROLYSIS system provided some functions of distributing data but only in a limited scale.

Aziz, Muhamad, Wahab, Alias, Hashim & Mustafa (2010); and Jubadi and Sahak (2009) provide a description of SMS alert system which can inform a user about any exceptions during an experiment. The similar system was tentatively designed for the PYROLYSIS software, but it's implementation was determined to be included into the next version of PYROLYSIS, as also to upcoming process control software systems. A use of commercial, internet-based SMS gateway was estimated to be a viable solution. Echols, Smith and Nirschl (2004) present a web-based system to observe the experiment process in real-time. Although not integrated into the PYROLYSIS system, a web-based observation of the laboratory equipment could provide additional safety to the working practices of PMG.

Hevner et al. (2004) provide seven guidelines to evaluate the research in the context of design science research. In this study, a software artefact was presented. The basis for the design and the development process was an existing need to support the experimental research of pyrolysis and coking processes by using custom-built software for automating the experiments. The existing knowledge base was researched by using literature survey, enabling the use of existing solutions as a baseline for new designs. The design and the development project employed design-evaluate cycle to both in UI development as when developing other software components. The product of this development process was the PYROLYSIS software system to be deployed to PMG, and the system was evaluated against usability, operational safety and architectural and reusability solutions. Finally, the description of the developed artefact and the findings were reported.
This study provided a description of software to support pyrolysis/coking test equipment used in the PMG laboratory. The software architecture and main components were described, as also the solutions used in the UI design process. Finally, the PYROLYSIS software evaluation process was reported.
7. Conclusions

In this study, a construct of a software (PYROLYSIS) developed to support the research of pyrolysis and coking processes was presented. The software architecture and key components were described. A model for hardware device virtualization and serial communication was designed. The UI design of the PYROLYSIS system was designed to include a system message component capable of providing the information of possible hazardous situation in order of seriousness, enabling the user to concentrate in a single exception at a time. Finally, a tentative design for SMS alerting system was presented.

The PYROLYSIS system was developed based on the needs of the PMG research activity. Although the architectural solutions were kept as general as possible, the applicability of the presented solution to other environments in other research fields may be a limited one. Due to the unavailability of the full hardware suite, the evaluation phase was conducted simulating the system responses. Real-life long term testing of PYROLYSIS would have resulted more reliable information of faulty tolerant features and usability issues of the system.

Remote observation of the experimental process and automated alerting system in the context of PMG could provide a topic for further research. The development of the systems capable of observing experimental processes and making necessary decisions of shutdown and alerting of technical staff would provide valuable information on the field of automated decision making and safety-critical system development. As the PMG laboratory lies behind the firewall of the University of Oulu, the research could also include aspects of information security.

Another topic for the future research would be comparing the use of LabVIEW against the more traditional development tools when designing laboratory automation systems. The research could yield information on the rapidity and flexibility of the development. The research could also include the use COTS software components during the development process.
References


# Appendix A. Software requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ-1</td>
<td>PYROLYSIS must control process variables during pyrolysis/coking process test. Controllable variables include furnace chamber temperature and gas flow. Control is enabled using user-defined process control program (PCP).</td>
</tr>
</tbody>
</table>
| REQ-2       | Process Control Program has following attributes:  
- it is divided into segments  
- within each segment, process temperature values are defined by using polynomical function in form $T = c_0 + c_1t + c_2t^2 + c_3t^3 \ldots + c_n t^n$, where $T$ is target temperature for controller and $t$ is elapsed time from the start of the segment  
- total gas flow and its composition |
| REQ-3       | PYROLYSIS must log variables during pyrolysis/coking process test. Variables include:  
- date and time  
- temperature values of both furnace chambers (set and actual)  
- temperature values of up to 9 measurement points from reaction vessel  
- temperature values of up to 7 auxiliary measurement points  
- gas flow and composition (set and actual) |
| REQ-4       | Tools must be provided for creating and editing Process Control Programs. Tool should prevent user from making a potentially hazardous PCP directly or indirectly. |
| REQ-5       | PYROLYSIS must be controllable both manually providing directly temperature and gas flow from UI; and automatically by using Process Control Program. |
| REQ-6       | Robustness must be provided: user must be prevented from providing harmful or out-of-range input values. Should user attempt this, he must be informed of correct values and format. |
| REQ-7       | User must be kept informed if set values and actual values in process do not coincide and the disparity is significant by changing text color, using bold font or by other suitable means. |
| REQ-8       | The user is to be provided with emergency shutdown option. Emergency shutdown should instantly terminate any PCP in progress, set temperature values in both furnace chamber to zero and switch to inert gas flow if present. |
| REQ-9       | PYROLYSIS should provide user with clearly observable signal an information concerning any exceptions during the experimental process. These signals should be ordered according the severity of exception. The user should be provided with step-by-step instructions how to resolve the exception. |
| REQ-10      | PYROLYSIS should maintain two event logs:  
- User event log, where user can write comments during the experiment  
- System event log, where any exceptions will be noted |
Appendix B. Hardware

Figure 21. Hardware schematics of the pyrolysis-coking test equipment.
**Figure 22.** Photo of the pyrolysis/coking test equipment hardware.
Appendix C. Device communication protocols

Modbus

Modbus is a serial communication standard developed by Modicon in 1979 and has since became a *de facto* standard. It is an open standard supported by Modbus organization. Modbus enables of reading and writing either bits or words to device or from device. Example data structure for command *READ N WORDS* is presented in Figure 23:

![Figure 23. Structure of a Modbus command.](image)

The Cyclic Redundancy Check (CRC) used is CRC16. In PMG, the Modbus protocol is used in the recent versions of temperature controllers manufactured by Eurotherm.

Sources:
Wikipedia (http://en.wikipedia.org/wiki/Modbus)
Eurotherm Series 2000 Communication handbook (HA02630)
Modbus organization (http://www.modbus.org/)

HART

HART (*Highway Addressable Remote Transducer protocol*) is a communication standard developed by Rosemount. Example of command structure from master (PC) to slave (Brooks 5850s) is presented in Figure 24:

![Figure 24. Structure of a HART command.](image)

The one-byte checksum is calculated by successively employing XOR operation to all bytes between the start byte and the last data byte. In PMG, the HART protocol is used in Mass Flow Controllers (MFC) used to control the gas flow and manufactured by Brooks.

Sources:
HART Communication Foundation (http://en.hartcomm.org/)
SPCI

SPCI (Standard Commands for Programmable Instruments) is an ASCII character-based serial communication standard used in various measuring devices. If multiple commands are sent in the same transmit, commands are separated with semicolon (;). Example command (3)

```
CONF:TEMP TC,K, (@102);DISP:ON
```

sets the channel number 102 in HP 34970A to measure temperature by using K-type thermocouple and switches the device front display on. Commands are terminated either to `<line feed>` character (hex 0A) or to a `<carriage return>` followed by `<line feed>` (hex 0D + hex 0A). As the commands and replies consists of standard human-readable characters, SPCI capable devices can be used with general terminal devices. In PMG, the SPCI protocol is used in various measuring devices, including general data logger / switch units manufactured by Hewlett Packard.

Source:
Figure 25. Main System User Interface (MSUI) component.
Appendix E. SMB and SAB examples

Figure 26. Three different level messages by using System Message Box (MSB) component.

Figure 27. System Advice Box (SAB) related to the warning MSB in Figure 26.