UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Grace Achenyo Okolo

ValChrom – Software Tool for Validation of Chromatographic Analysis Method

Master Thesis (30 ECTS)

Supervisors: Professor Marlon Dumas
Associate Professor Koit Herodes
Asko Laaniste, PhD

Tartu 2019

Acknowledgements

Firstly, I would like to thank God almighty for giving me strength and His amazing grace. I would like to express my gratitude to the University of Tartu for the opportunity to study in Estonia and for giving me the chance to follow my dreams and expand my knowledge. I would like to thank Professor Marlon Dumas for believing in me and providing support and encouragement throughout my studies and my thesis. I am also grateful to my supervisors Associate professor Koit Herodes and Asko Laaniste PhD, for giving me the opportunity to be a part of developing ValChrom. Working on my thesis has helped me grow professionally and personally. Finally, I would like to thank my family and friends for their constant support. May God bless you all.

ValChrom – Software Tool for Validation of Chromatographic Analysis Method

Abstract

Quality assurance is an important aspect in all industries. Most products and services are legally obligated to undergo some quality checks before they are sold to consumers. This concept also applies to the field of analytical chemistry. The results of chemical analysis can only be accepted if the analysis follows a method that has been proven to produce correct results. Analytical methods are the blueprints that determine how the analysis should be performed, therefore they need to be carefully validated before use. Analytical method validation has received attention from regulatory bodies and expert groups who have developed guidelines by which methods should be validated. Now, analytical chemists can validate their methods using these guidelines to prove that they are fit for their intended purpose.

Analytical method validation is no easy process and it involves a lot of tasks that are currently being done manually. This creates room for error and can make the process slower and so, more expensive. Consequently, the Institute of Chemistry at the University of Tartu initiated the development of a software-as-a-service (SaaS) solution called ValChrom, that will solve the prominent problems faced in analytical method validation. This thesis presents the development of the client-side application of ValChrom. The author discusses the rationale for the project, the development lifecycle of the application and how the requirements where elicited and specified. The author presents the system design and implementation and discusses future works and possible improvements to the delivered product.

Keywords: analytical method validation software, chemical analysis, chromatography, method validation, client-side application development

CERCS: P300, P170

ValChrom – kromatograafiliste analüüsimeetodite valideerimise tarkvara

Lühikokkuvõte

Kvaliteedikontroll on kõikides tööstusharudes oluline osa tööprotsessist. Enamik tooteid või teenuseid peavad enne üleandmist kliendile läbima kvaliteedikontrolli. Kvaliteedikontroll on oluline ka analüütilises keemias. Keemilise analüüsi tulemuse usaldusväärsus on tagatud ainult siis kui analüütiline meetod on valideeritud ehk, on tõestatud, et saadavad tulemused vastavad analüüsi eesmärkidele.

Ametlikud järelvalvet teostavad asutused kontrollivad pidevalt, et keemilised analüüsid oleks usaldusväärselt kontrollitud ja teostatud. Selleks on mitmed ekspertgrupid (ICH, EMA, ISO, Eurachem jt) arendanud juhendmaterjalid, mis sätestavad analüütilise meetodi valideerimise juhised ja kriteeriumid. Valideerimise etapis on analüütilistel keemikutel tihtipeale kohustuslik järgida konkreetset juhendmaterjali, et tõendada meetodi sobilikkust konkreetses valdkonnas (nt bioanalüüside valdkonnas).

Valideerimine on küllaltki keerukas ja spetsiifilist kompetentsi nõudev töö, mille lihtsustamiseks võiks kasutada IT-vahendeid. Siiani viiakse valideerimisel vajalikud arvutused tihti läbi tabelarvutustarkvaras ja tulemused vormistatakse tekstiredaktori abil. Sel juhul on inimlike vigade arv suur ning kogu valideerimise protsess aeganõudvam. Seetõttu algatati TÜ Analüütilise keemia õppetoolis poolt teenusena pakutava (SaaS) tarkvara ValChrom arendamist. Eesmärgiks on pakkuda kasutajatele tööriista, mis abistaks kasutajat kogu valideerimise protsessi vältel.

Antud töös kirjeldatakse ValChromi kliendipoolse osa arendamist. Arutletakse projekti põhimõtete ja tarkvara elutsükli üle ning protsessi kasutaja vajaduste väljaselgitamisest kuni tarkvaralise lahenduseni. Samuti pakutakse võimalikke arendussuundi edasiseks.

Märksõnad: keemiline analüüs, kromatograafia, metoodika valideerimine

CERCS: P300, P170

1. Introduction

Before companies sell their products to people, the products must undergo tests and analyses to ensure compliance with regulations. The same principle applies to chemically engineered products. Chemical analyses are performed on them to know their quality and understand their chemical composition. Often those analyses use analytical methods to obtain the composition of the product. It may be an analysis to determine the content of active pharmaceutical ingredient of a tablet, pesticide residue in tomatoes, or an analysis to detect doping substances in athletes' bodily fluids. It is important for these analyses to produce accurate and consistent results all the time. For example, contents of active ingredient and impurities should always be accurately determined. In order to correctly perform chemical analysis, there needs to be a blueprint that specifies the steps to follow and tools to utilize. This blueprint is called an analytical method.

Often the use of analytical methods yields significant results that can affect several aspects of life. In the medical field, an inaccurate result could imply an incorrect diagnosis of a patient. In food production, an inaccurate result could imply that people will consume harmful food. That is why before an analytical method is used to perform chemical analysis, it must be validated to determine whether it produces results that comply with regulations and to ensure that it fits the intended purpose.

Even though today, sophisticated lab equipment is available to help chemists perform chemical analyses, there are still a lot of manual and time-consuming activities involved in the method validation process. These activities include reading lengthy method validation guidelines to decide which techniques to utilize to assess the different criteria and parameters of the analytical method, preparing samples for laboratory analyses, collecting and compiling the results of the analyses, performing mathematical and statistical computations on the analyses results and finally producing a document to report the analytical method's characteristics also known as validation parameters. All these activities create room for errors. In some cases, chemists may use several software tools at different stages of the validation process, which means that they need to transfer data between different tools. This can easily lead to the isolation of information in several files or databases, the loss or inconsistency of data, and the difficulty to share data and collaborate with each other.

An intuitive solution to this problem is to create a software for analytical method validation. The software will automate the process and basically eliminate the problems faced by the current analytical method validation process. This is the rationale behind ValChrom, a software project initiated by the Institute of Chemistry at the University of Tartu. ValChrom is envisioned as a software-as-a-service (SaaS) solution to help analytical chemistry laboratories plan, assess and report the validation of analytical methods in accordance with validation guidelines.

In this thesis, the author presents the development of the frontend application of ValChrom, which is built using the Vue.js Framework and how it interacts with the backend application to provide the complete ValChrom SaaS solution. The backend system and the front-end system communicate through a Representational State Transfer (REST) Application Programming Interface (API). The frontend is developed to be an interactive and dynamic single page application (SPA). It provides the user interfaces that mirror the analytical method validation process and provides the desired user experience.

This thesis is divided into eight (8) chapters. This introductory chapter is followed by chapter 2, which provides some context and background about the domain. It introduces analytical methods, specifically

chromatography and discuses about chromatographic analytical method validation. Chapter 3 discusses the problem statement and reviews existing solutions for analytical method validation on the market and then introduces ValChrom as a solution. In chapter 4, the scope of the thesis and the approach taken are discussed. The requirements are discussed and illustrated in chapter 5 while chapter six presents the system designs and chapter 7 discusses the implementation. The thesis is concluded in chapter 8 which provide some possible improvements that can be made to the delivered system.

This software project was a joint work with Kodjovi Hippolyte-Fayol Toulassi, also a student in the Software Engineering curriculum and Karl Kruuse, a software developer. Karl Kruuse handled the implementation of the computation modules needed to compute the characteristics of the analytical procedures, while Hippolyte-Fayol and the author of this thesis were respectively in charge of the backend application and the frontend application. This thesis focuses on the development of the frontend application while Hippolyte-Fayol's thesis [14] focuses on the implementation of the backend application.

Only Section 1 of this thesis was written jointly with Hippolyte-Fayol. All the other sections are the individual work of the author of this thesis.

2. Literature Review

An analytical method is a way of conducting an analysis, detailing the step-by-step procedure necessary to carry out the analysis [3]. It is used to quantitatively and qualitatively determine the chemical composition of a compound.

2.1 Chromatography

Chromatography is a type of analytical method for separating the analytes (or components) of a sample to discover what they are and their concentration level [1]. This process of separation involves two phases, the mobile phase and the stationary phase. The stationary phase is usually a porous solid that stays motionless inside a column (or tube) and acts like a resistance to the substances flowing through the column. The mobile phase is the substance that moves throughout the process, transporting the sample to be separated over the stationary phase. The individual analytes in a sample interact differently with the stationary phase due to differences in size, partitioning or adsorption etc. phenomena. These differences between the components of a sample enable them to be separated from each other [2]. Chromatography has an elaborate historical background, it began with simple column chromatography, where mobile phase moved through the stationary phase under the influence of gravity. Chromatography has since evolved and produced many modern techniques such as ultra-high-performance liquid chromatography (UHPLC), fast protein liquid chromatography (FPLC), supercritical fluid chromatography (SFC) and gas chromatography (GC).

2.1.1 Liquid Chromatography

Liquid chromatography is a type of chromatography where the mobile phase is a liquid. The separation process takes place in a column (special tube). In a liquid chromatography system, the stationary phase is held in place inside the column. The column is where the mixture of the sample and the mobile phase passes over the stationary phase and then the sample components get separated from each other. These analytes are eluted from the other end of the column in the order of their separation.

2.1.2 High-performance liquid chromatography (HPLC)

High-performance liquid chromatography (HPLC) is a type of liquid chromatography that makes use of a high-pressure pump to pass the mobile phase and sample mixture as a pressurized liquid solvent through a column containing the stationary phase [8]. The diagram in Figure 1 shows the components that make up a High-performance liquid chromatography system. This system is designed to support a standard HPLC process described below.

HPLC Process

The system has a Reservoir that contains the mobile phase (a solvent which will be transported throughout the system). The Pump is used to supply and regulate the flow of the mobile phase into the system. The sample is introduced into the system using the Injector which injects the sample into the flow of the mobile phase. The mobile phase acts as a transporter, flowing into the column where the stationary phase is attached to the hardware of the column. As the mixture of the sample and mobile phase flows through the column, the analytes in the sample get separated. After which they flow out from the column and into the detector [8]. The Detector registers the individual analytes that pass through it. Then they can either be stored for further analysis or discarded depending on the purpose of the separation. The detector is connected to a Computer data station which identifies and quantifies the concentration of the analytes that are being eluted from the column after separation. It collects and

stores the electrical signals and then generates a Chromatogram - change of detector signal over the time of chromatographic analysis. One HPLC system can have multiple detectors depending on the sample or analytes characteristics.

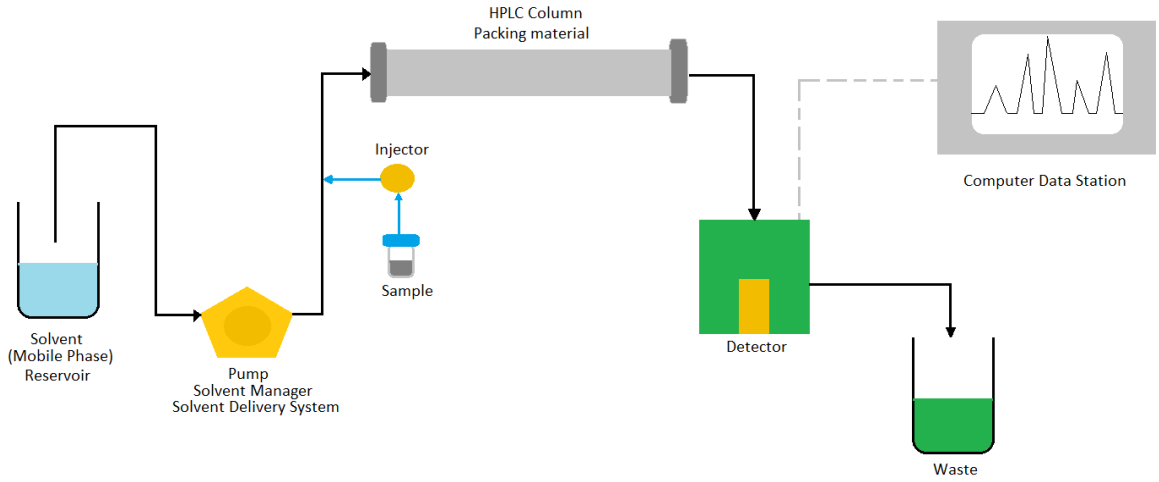


Figure 1 High-Performance Liquid Chromatography [HPLC] System

The column

Figure 2 provides a clearer illustration on how the analytes of a sample are separated in a column. The sample used in this example is a mixture of blue, yellow and red dyes which now appears as a dark substance. The first image is a snapshot of the column at time zero, when the sample is injected into the column and it appears at that point as one single dark band. The arrows indicate the direction of flow of the mobile phase.

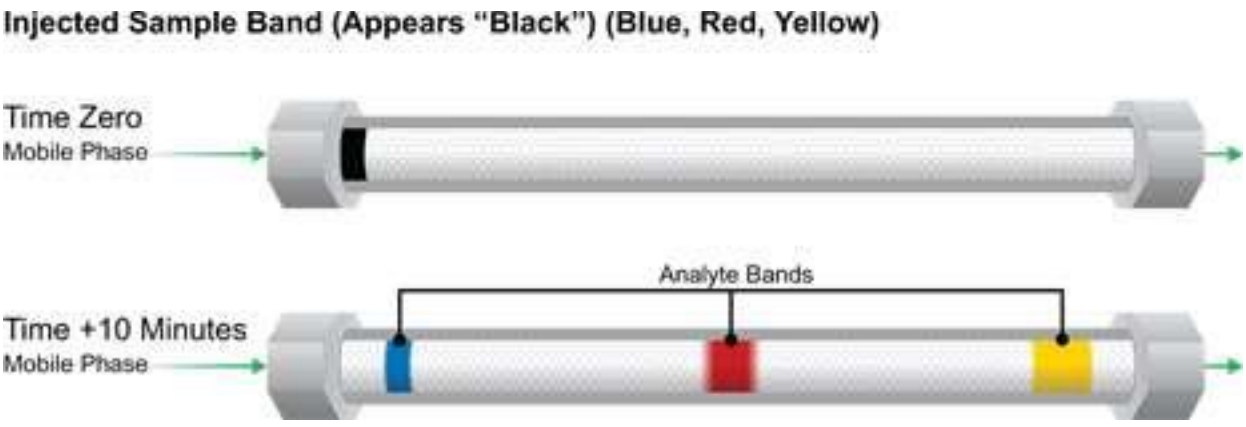


Figure 2 [8] How a Chromatographic Column Works

The second image of the column shows the situation ten minutes after the sample was injected. During this time the mobile phase continued to flow through the column over the stationary phase while moving the analytes in the sample further along the flow direction. This has caused the individual dyes to separate into three different band moving at different speed. The yellow band moved the fastest and is at the point of exiting the column. This is because the yellow analyte is more attracted to the mobile phase and so it moves faster compared to red and blue dye. The blue analyte on the other hand is more attracted to the stationary phase and so has a slower speed and it is the most retained analyte of the sample. The red band is moderately attracted to the mobile phase and so it moves at a moderate speed. This difference in speed of analytes is what enables them to be chromatographically separated.

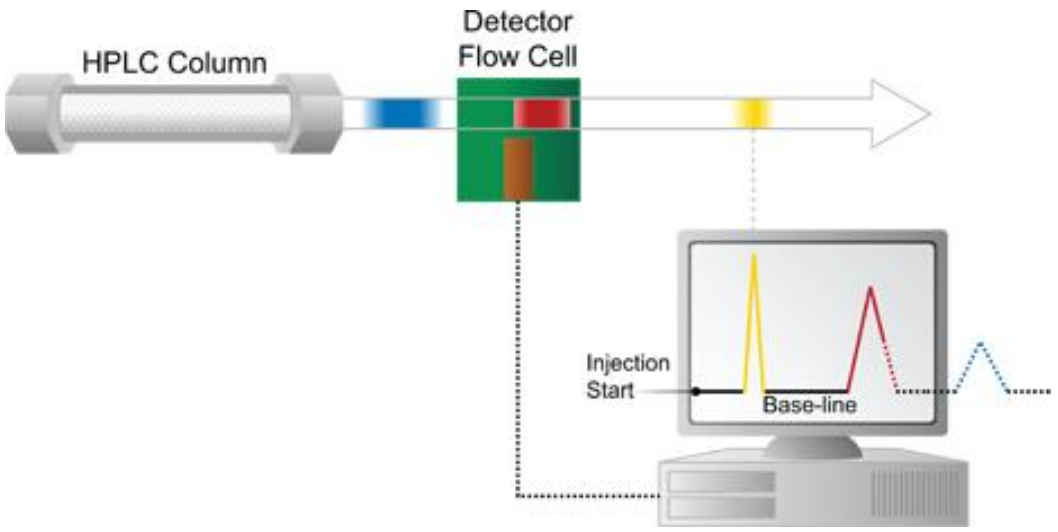


Figure 3 [8] How Peaks Are Created

The Detector

The exit point of a column is connected to a detector. Figure 3 shows how different analytes leave the column and enter the detector. The detector has a flow cell that detects every analyte that enters into it and then sends an associating electrical signal to a computer data station about the analyte it detected.

The Chromatogram

The chromatogram is a visual representation of the result of the separation. The computer data station plots the chromatogram based on the electrical signals it receives from the flow cell of the detector (see Figure 4). The computer data station starts plotting the chromatogram from time zero, when the sample is first injected [8]. That is why it starts off as a straight line, because the detector has not detected anything other than the pure mobile phase (called the baseline). As an analyte enters the detector, it sends a stronger signal to the computer data station. The stronger signal causes the plot to create an upward curve to the proportion of the concentration of the analyte, it goes high and as the signal reduces then it reduces till it returns to a straight line. These “mountains” in the chromatogram are called peaks. Peaks are registered for every analyte that the detector encounters.

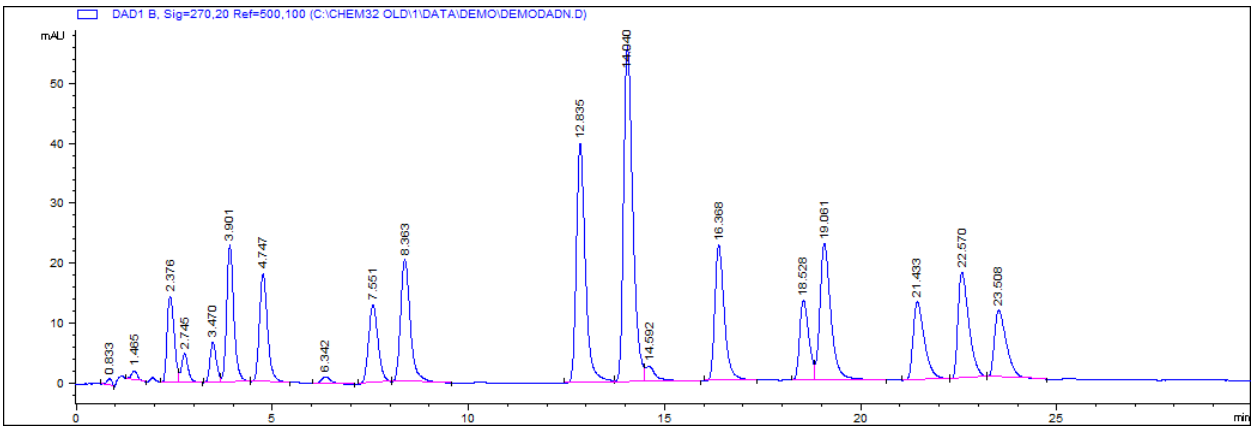


Figure 4 Chromatogram

2.2 Analytical Method Validation

The objective of analytical method validation is to determine whether an analytical method used for an analysis fits its intended purpose [1]. Analytical method validation is a vital aspect of a good analytical practice and sometimes it is obligatory by law. The results from validation are used to ascertain the reliability, quality, and consistency of methods. The validation of analytical methods must be done before they are introduced into routine use. Also, revalidation of an analytical method needs to be carried out whenever the state for which the analytical method was validated changes.

Substantial attention is given to validation from regulatory bodies and industrial committees because of its importance in the field of analytical chemistry. This has resulted in various international bodies (organization and conferences) creating guidelines about analytical method validation. Some of the most important bodies and validation guidelines are:

- Eurachem (European network of analytical chemistry) analytical method validation.
- ICH (International Conference on Harmonization) validation of analytical procedures.
- EMEA (European Medicines Agency) bioanalytical method validation

2.2.1 Validation Parameters

All guidelines specify a set of parameters that must be assessed to determine whether an analytical method is valid or not. Each parameter evaluates a different aspect of a method. Below are some of the most common parameters currently being used in validation.

Linearity

The linearity of an analytical method is demonstrated by its ability to derive results from the analysis that are directly proportional to the level of concentration of analytes in a sample within a specified range. Linearity is sometimes evaluated visually by examining a graph of signals plotted as a function of analyte concentration. This graph is called the calibration graph. It shows the concentration level of an analyte and can be used to predict the concentration level in an unknown sample. Linearity is evaluated by inspecting the calibration graph that contains signal heights (peak areas) as a function of the concentration level of the analyte.

Limit of detection

The limit of detection of an analytical method is the minimum amount of an analytes in a sample that can be detected but cannot be exactly quantitated as a precise value. It is an acceptable level to say that an analyte is present in a sample, but it does not say how much of it is present.

Limit of quantitation

The limit of quantitation of an analytical method is the minimum amount of analyte in sample that can be quantitatively determined with appropriate accuracy and precision.

Precision

The precision of an analytical method is measured by how close the individual results of analysis are to each other or how well they agree when the method is applied several times to samples of the same type. The acceptability of precision depends on the type of analysis being done. The objective of validating an analytical method with respect precision is to make sure that the analytical method will output homogeneous results. For example, a sample is taken and analysed five times and then the individual results are compared to assess how close they are to each other.

Trueness

The trueness of an analytical method checks how close the results are to the true value (concentration). If the results are close to the true value, the average is on the true value when plotted, then the analytical method is precise with high trueness. Otherwise, if the results are only close to each other but not to the true value then the precision is good, but the trueness is low.

Accuracy

The combination of precision and trueness forms the accuracy. An analytical method is accurate when it is true and precise. It is measured by how close the results generated from an analysis are to the true or acceptable value and how close the individual results of analysis are to each other. The guideline by ICH suggests that accuracy should be determined using a minimum of nine analyses. Every guideline has established a criterion – the maximum deviation of results from true value for it to be considered accurate. The trueness and precision of an analytical method is sometimes regulated by laws for example, the quality of drinking water.

Selectivity/Specificity

The specificity of an analytical method is measured when it is possible to undoubtedly establish that a signal is only due to a specific analyte, in the presence of other analytes. This test of identification must be able to map each peak with its corresponding analyte.

Range

The range of an analytical method is the interval between the highest and lowest concentration level of analytes in a sample, where the method is considered applicable.

Ruggedness/Robustness

The ruggedness or robustness of an analytical method is measured by how much (small but intentional) variations of the experimental conditions (that are likely to occur during the routine usage in the process of performing an experiment) affect the result. If the method is unaffected by these variations, then it is robust.

2.2.2 Validation Process

The validation of analytical method begins after it has been developed. Then one or multiple guidelines will be chosen by an analytical chemist(s). The selected guideline(s) will determine the parameters and experiments that should be used to validate the method. The next task for the analytical chemist(s) is to create a well-documented experimental plan. This experimental plan is created using the information in the analytical method and in the selected guideline(s). The experimental plan is taken into the laboratory where the necessary experiments will be conducted in accordance to the plan.

After the experiments have been successfully performed, the analytical chemist(s) will have the results. These results are often in the form of tabular datasets, with each column in the dataset represents some aspect of the experiment, and the rows contain the values that were recorded. Each test in an experiment produces its own result dataset. The volume of the experimental dataset produced from running experiments greatly depend on the experiment themselves. The product owners of ValChrom, who are themselves expert analytical chemist, states that each assessment method produces its own result dataset containing the results for each analyte in the analytical method. Figure 5 is an example of an experimental dataset for an assessment method called "Based on calibration graph | Slope and standard deviation of y-intercept" containing results for two (2) analytes "ACE" and "THC". The attributes in an experimental dataset vary depending on assessment methods.

name	tr	area	expected concentration	series	level	parallel	marker	analyte
1.0 mg/ml acetamiprid	18.979	38489473	1.019024673	08.05.2019	1	1	Calibation	ACE
0.8 mg/ml acetamiprid	18.95	31285035	0.813941697	08.05.2019	2	1	Calibation	ACE
0.6 mg/ml acetamiprid	18.965	23799618	0.612960681	08.05.2019	3	1	Calibation	ACE
0.4 mg/ml acetamiprid	18.923	15866617	0.406306764	08.05.2019	4	1	Calibation	ACE
0.2 mg/ml acetamiprid	18.894	8000115	0.204248167	08.05.2019	5	1	Calibation	ACE
0.1 mg/ml acetamiprid	18.82	3932841	0.100684207	08.05.2019	6	1	Calibation	ACE
0.08 mg/ml acetamiprid	18.868	3142822	0.078779525	08.05.2019	7	1	Calibation	ACE
0.06 mg/ml acetamiprid	18.872	2351487	0.060463607	08.05.2019	8	1	Calibation	ACE
1.0 mg/ml thiacloprid	13.4	3848947	1.02	08.05.2019	1	1	Calibation	THC
0.8 mg/ml thiacloprid	13.95	3128503	0.8	08.05.2019	2	1	Calibation	THC
0.6 mg/ml thiacloprid	13.965	2379968	0.61	08.05.2019	3	1	Calibation	THC
0.4 mg/ml thiacloprid	13.923	1586617	0.4	08.05.2019	4	1	Calibation	THC
0.2 mg/ml thiacloprid	13.394	800115	0.2	08.05.2019	5	1	Calibation	THC
0.1 mg/ml thiacloprid	13.32	393241	0.1	08.05.2019	6	1	Calibation	THC
0.08 mg/ml thiacloprid	13.363	314222	0.078	08.05.2019	7	1	Calibation	THC
0.06 mg/ml thiacloprid	13.372	235187	0.06	08.05.2019	8	1	Calibation	THC

Figure 5 Experimental Dataset

The datasets need to be mathematically analysed and validated against the validation criteria which are specified for assessment methods in the guideline(s). This involves the analytical chemist(s) performing statistical and mathematical evaluations on the datasets. The results of the evaluations will tell if the analytical method is valid or not based on the criteria specified by the guideline(s). Once the datasets have been analysed, the analytical chemist(s) creates a detailed report of the entire process so far. The report is a cumulative document from all the stages starting from the method description, guideline

selection, experimental plan, experimental dataset generation and validation analysis. All these phases are explained in a report.

3. Background

3.1 Problem Statement

The frequency, how often a laboratory has to validate an analytical method depends on the field and type of laboratory. For example, laboratories, which often develop analytical methods for new analytes and operate in regulated field (like pharmaceutical industry), perform different validations virtually daily. In addition to validating newly developed methods, every method needs revalidation if substantial changes are introduced: new sample type, new analyte or different instrument is used. Also, methods have to be revalidated at least once in five years.

Due to frequency and high significance of method validation, there is a need for the validation process to be fast and accurate. However, most analytical chemistry laboratories do not have the luxury of describing their process using the words “fast and accurate”. From interviews with the product owners of ValChrom, who are also domain experts, it could take a laboratory up to two weeks to conduct an analytical method validation process from start to finish if they have never done it before and up to one week if they have done it before and have a system or process to follow. A significant portion of this time is spent on mundane but essential task in the validation process e.g. information transfer, calculations, result checking, report writing etc. This is an inefficient use of personnel time and resources.

Another problem with this process of method validation is that there is a lot of tasks requiring human interaction. This leaves a lot of room for errors and mistakes. An analytical chemist is required to utilize several tools to complete a method validation, e.g. Word Processing for reporting, Statistical tools for calculations and Excel to view datasets. All these tools add extra complexity to an already complex process. There is a need for a solution that can automate and simplify the process for chemists to enable them focus on the actual analytical chemistry tasks.

3.2 Existing Solutions

Several software solutions have been created to automate the process of analytical method validation. Unlike other fields, analytical chemistry and method validation do not have a variety of software solutions to automate its processes. Below are some existing solutions for analytical method validation:

Analyse-it is a software solution that helps to automate aspects of method validation [4]. It is developed by a company called Analyse-it as an extension of Microsoft Excel and all input data and results are stored in Excel workbooks. It is compliant to a few validation guidelines. The software analyses an experimental dataset using guideline-compliant statistical computations and then generate a result about the validation conducted [4].

Fusion is an analytical method validation (hosted) software developed by a company called S-Matrix to work with chromatography data station software [5]. Fusion is compliant to two validation guidelines, ICH and Food and Drug Administration (FDA). It provides experiments needed to support method validation and automated calculations, graphing and reporting [5].

3.3 ValChrom

ValChrom is a SaaS platform that fully automates the process of analytical method validation and eliminates problem faced in traditional analytical method validation. Analytical chemists no longer need to spend hours reading long documents of guidelines to create experimental plans, users simply need to enter the information of their method and select the guideline they want to use - ValChrom will automatically generate an experimental plan. ValChrom currently supports three distinct guidelines and offers a combination of all three as a fourth option; Eurachem, ICH (International Conference on Harmonization) and EMA (European Medicines Agency).

With ValChrom, analytical chemists no longer need to perform the statistical computations on their experimental datasets by themselves. ValChrom takes the datasets and performs the necessary computations required by the selected guideline(s) and presents the results in a clear and detailed report with all the input data and graphs generated. With ValChrom, all the analytical chemist needs to do is make high-level decisions while the software takes care of the tedious but necessary tasks. ValChrom can be accessed from anywhere with a stable internet connection and these are the benefits that have been achieved by implementing ValChrom:

- users will always have the latest version of the software.
- Paperless analytical method validation processes to eliminate unnecessary costs and data inconsistencies and errors.
- Real-time analytical method validation results.

There are potential disadvantages associated with developing ValChrom as a SaaS solution. There is the issue of a customer's data not being stored on their premises. This could raise concerns about data confidentiality and some customers (e.g. large pharmaceutical companies) might be resistant to the idea that their highly confidential business data will be stored outside their premises and that it can be accessed by a SaaS provider. Another issue is that the software cannot be accessed if the SaaS service is temporarily unavailable. This could result in disruptions to the customer's business. There are also security risks involved in transferring data between the customer’s premise and the SaaS provider that could raise concern from potential clients.

Considering the advantages and disadvantages of the SaaS approach, the major reason for the product owners to choose SaaS is the fact that ValChrom is still at an early stage of development. It is very important at this stage for the solution to be tested and validated by potential users and clients. Using the SaaS approach provides the ability to ship out new versions easily and quickly to get valuable feedback from a wide range of users.

Target User characteristics

ValChrom is aimed at one type of user at this stage of development and this is described below:

- A chemist who is familiar with the field of Chromatographic Method Validation.
- Has a computer.
- Possess basic knowledge of computers and English language.
- Accustomed to using the Internet.
- Have access to high-speed internet connection.
- Have an updated version of one of the modern browsers on their computer.
- Knowledge of Microsoft Excel or similar programs, to work with .csv files.

Operating Environment

ValChrom can be accessed through any modern web browser on a computer that is connected to a high-speed internet connection. It does not require any installation.

Browser Support

ValChrom works on recent versions of Firefox, Chrome, Edge, Opera and Safari, with cookies and JavaScript enabled.

4. Scope and Approach

4.1 Scope of Thesis

This thesis focuses on the development of the Frontend application of ValChrom. The application communicates with the Back-end API of ValChrom which was developed alongside the Frontend application. The scope of work includes:

- Requirements elicitation and specification
- Design and Validation
- Implementation
- Testing

Each of these stages will be further explained in this thesis.

4.2 Approach

Team Structure

The team was comprised of two product owners, one data scientist and two developers: frontend and backend. The two product owners (also the thesis supervisors) are Associate professor Koit Herodes and Asko Laaniste, PhD. They possess a wealth of knowledge about the domain of Analytical method validation. They developed the initiative of ValChrom as a product. The product owners defined the requirements for the system. During the development process, they tested and validated the designs and deliverables against their requirements.

The mathematical modules of the application were developed by Asko Laaniste and Karl Kruuse. These modules performed analyses on the experimental datasets. The backend of ValChrom was developed by Hippolyte Fayol, a Software Engineering student and the author of this thesis developed the frontend application.

Development Process

The entire project was designed using the agile software development process, which is an iterative development process. The requirements evolved and pivoted throughout the project. Therefore, it was important to keep an active collaboration between all the stakeholders of the project. The scrum methodology was used for the development which follows the principle that a project should progress through a series of fixed-length iterations called sprints.

Estimation and Planning

The project was structured and divided into sprints of one week. Each sprint started by defining a goal and then all the features of the product that needed to be implemented during that sprint to achieve the goal. These features were then discussed and refined before adding them to a collection called the

sprint backlog. For each sprint, a backlog was defined which described the amount of work that should be completed during that sprint. At the end of the sprint, the team produced a potentially shippable product increment which was tested and validated by the product owners before it got accepted as completed. The backlog was then updated according to the amount of work completed.

5. Requirements Elicitation and Specification

The goal of this section is to present the requirement specification. This section provides a view of ValChrom from a user centric perspective, the goals and requirements. This software requirements specification will show the different use cases of ValChrom and the steps necessary to accomplish certain tasks.

5.1 Requirements Elicitation

Discussion Sessions

The first method used to elicit requirements was discussion sessions. At the first meeting, the product owners introduced ValChrom, the goals and expectations. They introduced the existing software, which was not in use as it had not met all their requirements. Discussion sessions were held to present the domain and the requirements of the software. The discussions were done with the aid of screen mock-ups to visualize the user interfaces and user experience that was expected.

Prototyping

The second method used for gathering requirements was prototyping. After the discussion sessions, the author of this thesis would follow up and create user interface prototypes of the features that had been discussed. These prototypes will then be tested and validated by the product owners. If they met their requirements then they would be implemented, if not the prototypes will be revised to implement whatever changes where specified. This was an essential method to make sure time is not spent implementing features that were not what the product owners wanted. The discussion sessions and prototype presentations happened weekly during the sprint meetings.

5.2 Functional Requirements

ValChrom Process Description

Here is a description of the process as described by the product owners during the requirements elicitation phase. The software is accessible online through a modern browser on a computer that is connected to the internet. On entering ValChrom, a visitor or user will see the home page. The home page will contain textual and visual contents that describe ValChrom to them. A visitor will be able to navigate to the About page and Contact page and will have the option to create an account on ValChrom. Existing users simply login to their account. After a visitor has created an account, they become a system user. A user will be able to access the six (6) modules of the system namely: Analytical Methods, Validation Plan Templates, Experimental Plans, Experimental Datasets, Report Templates and Reports.

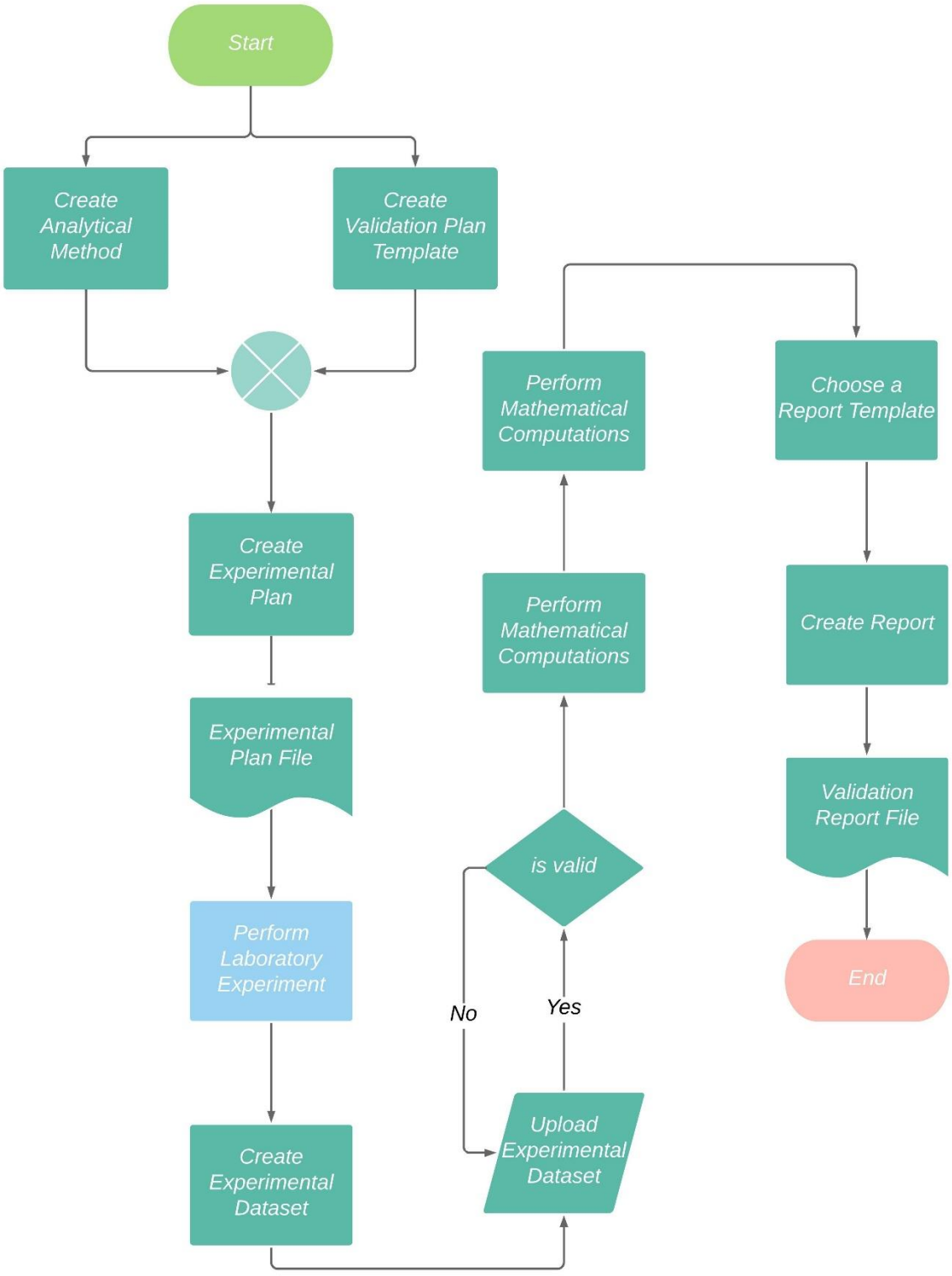


Figure 6 Flow of ValChrom

Figure 6 illustrates the process flow that a user goes through to perform an analytical method validation. They start from either the “Analytical Methods” or the “Validation Plan Templates” module of the system. The user will create an Analytical Method by providing all the required information and saving it. In the Analytical Methods section of the system a user will be able to see a list of all the methods they have created. This list will be grouped into two collections namely, *Active* and *Archived*. The analytical methods that are actively in use will appear in the *Active* list and the methods that were archived by the user will appear in the *Archived* list. In the active list the user can search, view, modify, duplicate and archive analytical methods. In the archived list, a user can only search and duplicate an analytical method. Once all the information has been added to the analytical method, the user can change the status from *IN PROGRESS* to *COMPLETED*. Status helps inform users of the progress of each method in the active list. Completed analytical methods can be accessed from other modules of the system.

The next stage is in the “Validation Plan Templates” section of the system. The user will create a validation plan template using one of the four guidelines supported by the application. The user will create a validation plan template by selecting all the options relevant to their validation and set the expected values of validity and save it. Like the “Analytical Methods” section, the user will also be able to view all their validation plan templates in two lists: *Active* list and *Archived* list. Similarly, in the active list the user can search, view, modify, duplicate and archive validation plan templates. In the archived list, a user can only search and duplicate validation plan templates. When all the information has been provided for a template, the user can mark it as complete, which will change the status from *IN PROGRESS* to *COMPLETED.* This status will allow the validation plan template to be accessed from other sections of the system.

When a user has created an analytical method and a validation plan template, the next step in the process is the experimental plan. The user will navigate to the “Experimental Plans” module of the system and create a new experimental plan using the completed analytical method and validation plan template that was created previously. After saving the experimental plan the system generates an experimental plan document from the analytical method and validation plan template. This document will be used by the user in the laboratory to perform the specified experiments to test the validity of the analytical method. The experimental Plan module also has an Active list and Archived list with the same functionalities to search, view, modify, duplicate and archive. Once all the information is satisfactory to the user, the status can be changed from *IN PROGRESS* to *COMPLETED*.

When the results from the experiments are ready, the user will proceed to the “Experimental Datasets” module of the application. Experimental plans and Experimental datasets have a one-to-one relationship. This is because one laboratory experiment cannot produce more than one result. At this stage of the process the user will create an experimental dataset using the completed experimental plan created earlier. In the experimental dataset entity, there are subsections for the various experiments performed. The user will be able to upload the laboratory result datasets to their relevant subsection. Once the user has uploaded all the result data from the experiment, the user will be able to see an overview or a detailed view of the validation results. The “Experimental Dataset” module also has an Active list and Archived list. In the active list the user can search, view and archive experimental datasets. In the archived list, a user can only search an experimental dataset. Once all the information has been added to the experimental dataset entity, the user can change the status from *IN PROGRESS* to *COMPLETED*. This status change will allow the experimental dataset to accessible from other modules of the system.

At this stage of the process, the user has completed the analytical method validation. The next step is to create a report of the process so far. The user will navigate to the “Reports” module of the system and create a new report using the completed experimental dataset entity. The report is structured according to a report template which can be viewed in the “Report Templates” module of the application. The user will provide all the necessary information for the report and save it. After creating a report, a user will be able to see a list of all the reports they have created. In the active list the user can search, view, download, duplicate and archive reports. In the archived list, a user can only search and duplicate a report. Once all the information has been added to the report, the user can change the status from *IN PROGRESS* to *COMPLETED*. This status change helps inform the user of the progress of each report in the active list.

Use cases

This section further illustrates the functional requirements of the frontend application of ValChrom as use cases. As described above, there are two actors that interact with the application, visitors and users. Below their functional requirements are illustrated in individual diagrams in Figure 7 and Figure 8 respectively. The six modules in the application have a similar structure of presenting their entities as data collections. These data collections have similar requirements that are reflected in the use case diagram in Figure 8. Appendix B contains comprehensive outline of the user cases implemented in the frontend application of ValChrom.

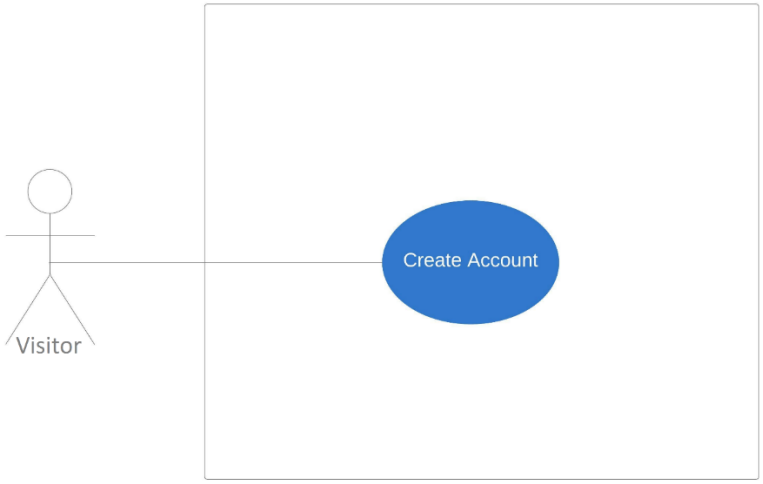


Figure 7 Visitor - Use Case

Figure 8 User Use Cases

User Stories

User stories are agile project management tools that help to define the requirements of a system and reveal the benefits or value of that requirement. These user stories present a high-level and simple description of the front-end application's requirements from the perspective of the end-users of the system. They were used during development to describe the functional requirements. Since the requirements where being set by non-technical product owners it was relevant to use a tool that would enable them communicate domain needs into implementable feature description. Figure 9 gives a glimpse of how the user stories were structures and Appendix A contains a comprehensive list of the user stories used to implement the frontend application of ValChrom.

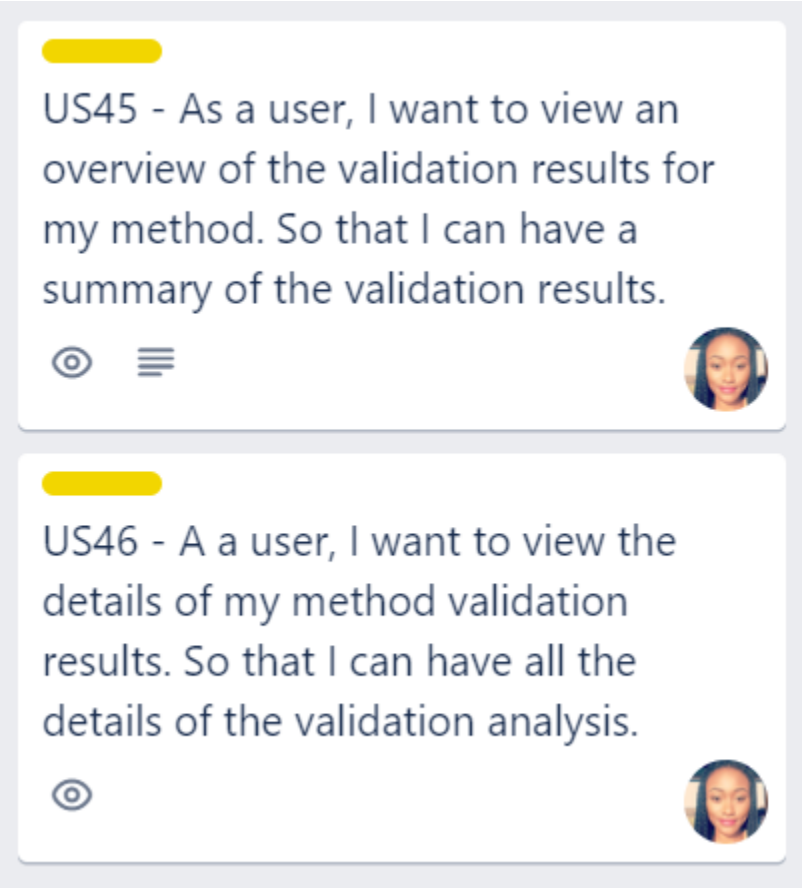


Figure 9 Sample User Stories

5.3 Non-functional Requirements

This section contains the non-functional requirements of the Frontend application of ValChrom. These are the non-behavioural requirements or quality attributes of the system derived from the requirements elicitation with the product owners. These quality attributes have been categories according to those defined in ISO/IEC 9126 [9]. They were used to establish the standards used to judge the operation of the system.

FUNCTIONALITY

Security

- In order to control access to the functionalities of the application, users account validation must be strictly enforced. A user must be authenticated from the Backend API prior to gaining access to any UI features (except the home page, about, contact, Signup and Login).
- Users must be logged out of their session after a period of inactivity in the system and their authentication token must be deleted from memory.
- The system must not save any cache or cookies containing a user's login credential (email and password) on the user's computer. The reason for this is to prevent the user's account from being compromised if the user's computer or local storage gets hacked.

USABILITY

Learnability

- 80% of novice users should be able to learn and operate the major features with assistance of a tutorial.
- The major feature of the system must be accessible from the navigation bar of the system.
- The UI elements must be intuitive to the system users.

Attractiveness

- All UI elements must have a consistent format, placement and style.
- Conceptual integrity in the UI of the system.
- Professor Koit Herodes must be satisfied with the user interface design of the system.

MAINTAINABILITY

- A software developer that is new to the project and has up to 6 months experience in coding, should be able to fix issues in the code and add new features in less than 2 person-days.
- Frontend architecture design must be done with modularity in mind so that maintainability can be done efficiently.

6. System Design

This section of the thesis specifies the architecture and software design decisions used for the implementation of the frontend application of ValChrom system. The different diagrams in this section are used to show distinct properties of the application to properly communicate all the essential requirements of the system.

6.1 Context-Level Data-Flow Diagram

Data flow diagram (DFD) is a graphical representation of the flow of information through the processes in a system. The goal of using DFD is to make system requirements clear and identify major data transformations that will be implemented in the system [10]. The Context diagram helps to establish the boundaries and the scope of the system. Figure 10 shows the context of the application (referred to as ValChrom Client). ValChrom client interacts with the users by collecting and presenting information. The client is a "Thick Client", that is it performs processing and structuring of the data retrieved from the backend database before it is presented users.

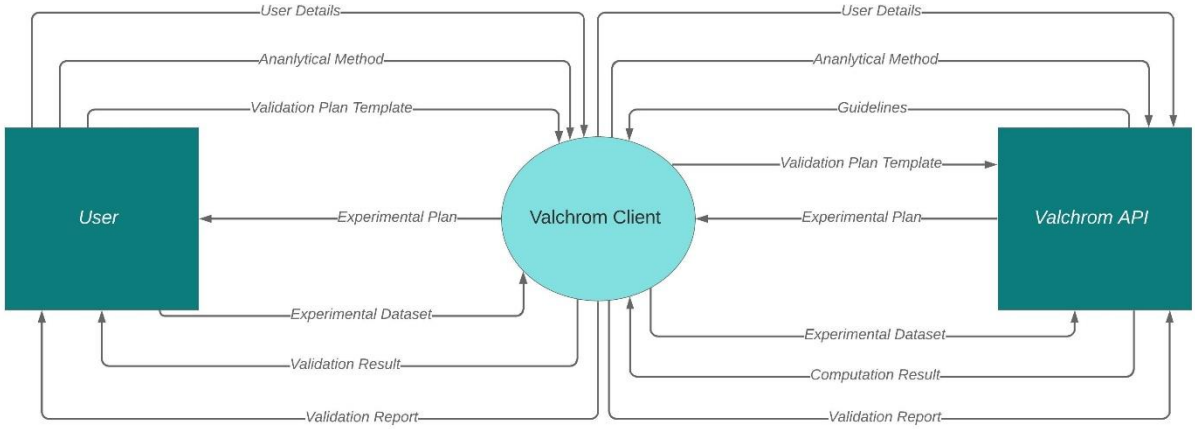


Figure 10 Context Diagram

- Figure 10 presents ValChrom Client and shows its interaction with external entities, namely the user and the backend API. The user inputs data and the ValChrom Client validates it and the sends it to be stored in the database through the API. The user saves information for their analytical method, validation plan template, experimental plan and experimental dataset. Once all that information is saved, the user will then be able to view the validation result for the analytical method. The API performs mathematical analysis on the experimental dataset based on the Eurachem (European network of analytical chemistry) analytical method validation.
- ICH (International Conference on Harmonization) validation of analytical procedures.
- EMEA (European Medicines Agency) bioanalytical method validation

Validation Parameters selected by the user in the validation plan template. The computation results derived from performing the mathematical analyses are sent to the Client when the user makes a

request to view the validation results. The computation results received from the API do not say whether the analytical method is valid or not. The validity of the analytical method (based on each validation parameter) is computed in the Client application using a service.

Due to the API and database design of the backend, the frontend needs to perform this real-time processing of the computation result to deduce the validity of an analytical method. The Client structures the data and computes the validity results before presenting it to the user. The user can simply view in their browser or decide to print out a PDF report. When a report is generated by the Client, it also sends the PDF version to be stored in the database through the API. The reason for storing the PDF versions of the report is to have an audit trail. It is a way of taking a snapshot of the process so that it could be traced when and with what data a report was generated. Also, the product owners would like to add a signature functionality to the system in later versions, so when a report is generated and signed it will be saved as a PDF file.

6.2 Sequence Diagram

Sequence diagram is a visual representation of the interactions in a system. It shows how operations occur, what messages are sent and received and when they happen in the lifeline. Everything is ordered in a sequential manner according to the time of occurrence.

The sequence diagram below will further illustrate the process of retrieving a Validation result of an analytical method. Prior to this the user has saved the information for their analytical method, validation plan template, experimental plan and experimental dataset.

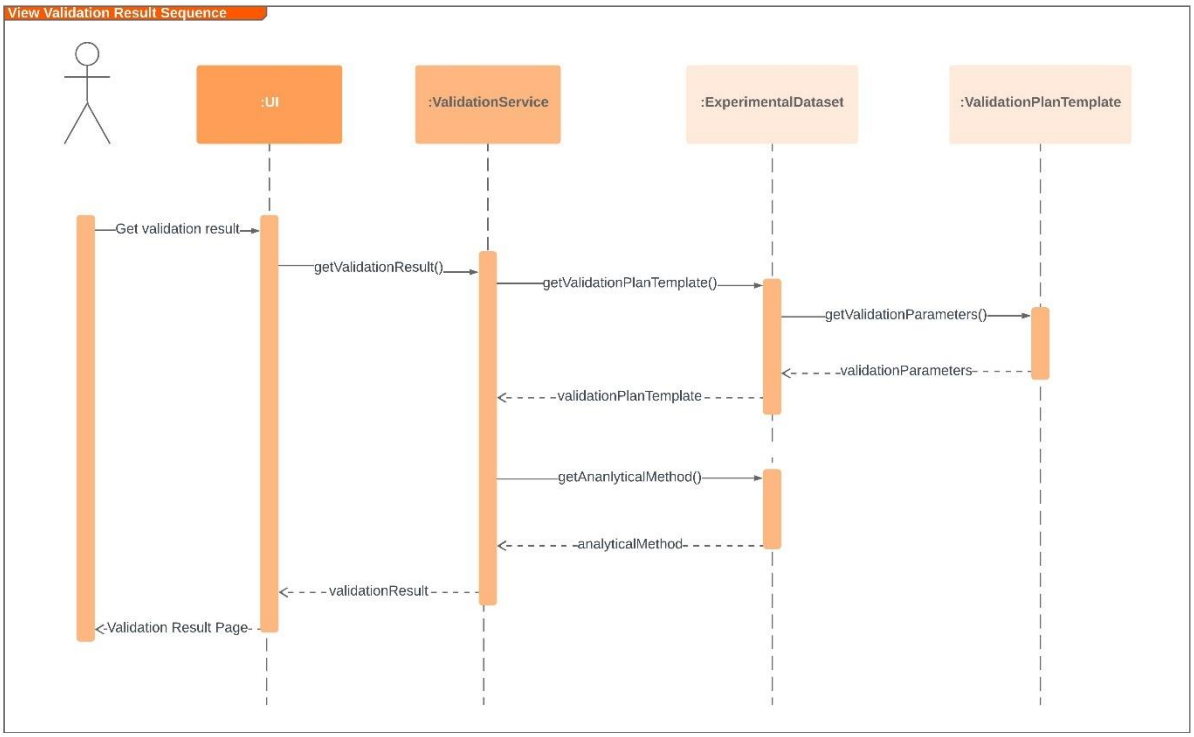


Figure 11 Sequence Diagram

Figure 11 shows the user making a request through the application UI to get the validation request. The UI forwards this request to a service. This service is a script that gathers all the necessary data and performs the analysis to transform the data received into the information the user requires. This service will be discussed in detail in the Validation Result Computation subsection of the Implementation section. Finally, the resulting information is sent back to the UI and it is display in a specified structure for the user.

6.3 Domain Model Diagram

A domain model is a visual representation of the real-world entities of the domain being modelled [11]. In the case of ValChrom, the domain modelling is used to translate the requirements of the domain into a conceptual solution. Figure 12 shows the conceptual model of ValChrom that incorporates the conceptual classes, associations between conceptual classes, and the multiplicity of their associations.

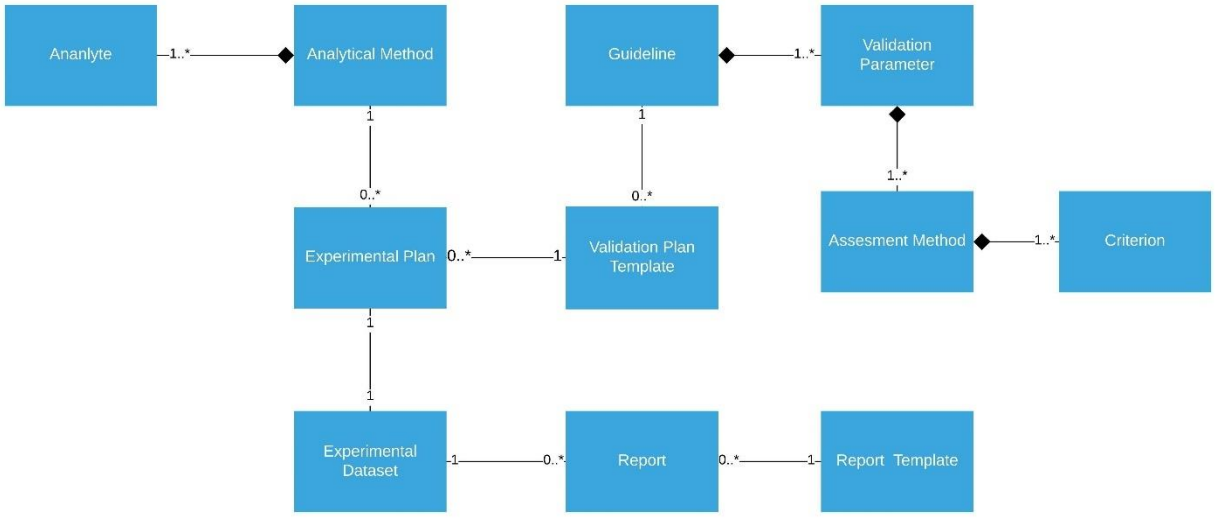


Figure 12 Domain Model

Figure 12 shows that an analytical method is composed of one or many analytes. A validation plan template is derived from one guideline. A guideline is composed of multiple validation parameters and each validation parameter is composed of multiple assessment methods. Assessment methods are the ways a validation parameter can be assessed. Each assessment method is composed of one or more criteria that needs to be satisfied to prove that a method is valid. An experimental plan is made from combining an analytical method and a validation plan template. This experimental plan can only produce (link to) one experimental dataset entity. An experimental dataset object can be used to produce multiple reports and a report is structured according to one report template.

6.4 State Chart Diagram

A state chart (or sometimes referred to as automata, state diagram or state machine) is a graphical representation of the different states of an entity. The five main entities in ValChrom, namely: Analytical Methods, Validation Plan Templates, Experimental Plans, Experimental Datasets and Reports, respond to various event during their lifecycle and change from one state to another. Figure 13 is used to model the dynamic nature of these entities.

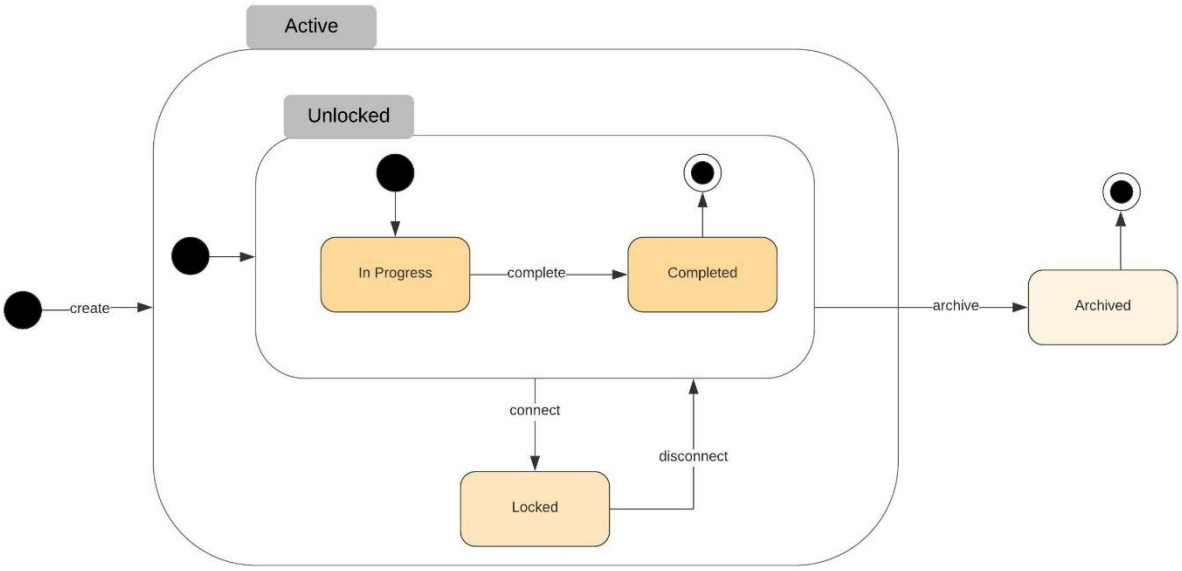


Figure 13 State Chart

When an entity is created it enters the state “Active”. In the active state, an entity can be viewed, modified, completed and archived. In the “In Progress” state an entity can be marked as complete so that it can be accessed from other modules of the system. When an active entity is used (connected) in another module, it becomes “Locked” and when it is no longer used (disconnected) it becomes “Unlocked”, and an unlocked entity can be archived. When an entity is archived, it can no longer change to another state and so its lifecycle ends.

7. Implementation

This section of the thesis explains the process involved in implementing the frontend application of ValChrom. It discusses the technology stack used to develop the application along with the reasons for choosing them. It then presents the architecture of the application and the implementation of the major features of the application.

7.1 Technology Stack

The development of the application was done using Vue.js Framework using JavaScript, HTML, and CSS. Bulma CSS framework and Buefy UI component library were used to create the User Interface (UI) designs. Visual Studio Code was used as the code editor and Bitbucket as the version control repository for the application code.

Frontend Framework

The application is a SPA, that is a web application that runs inside a browser and interacts with a user by dynamically modifying elements on the current page rather than loading a new page from the server. A SPA performs all the page rendering in the frontend and only requests the server when new data is required or needs to be stored or refreshed. Angular, React and Vue are the most popular frontend frameworks (according the number of stars on their GitHub repositories) and they are used for building user interfaces and SPAs. Each framework has its advantages and disadvantages and they were considered when deciding which framework to use for ValChrom client. The factors considered for each framework are discussed below:

Learning Curve:

The product owners had a strict deadline to present the software at the Eurachem conference in May 2019 and so they needed the development to progress at a fast pace. They required a framework that will have a low learning curve for the developer and thus translate to a quicker implementation of features. The author of this thesis had more experience working with Vue.js than Angular and React. Angular and React are considered to have a higher learning curve than Vue [16] so Vue had an advantage.

Popularity:

The product owners also required the technology stack to be popular enough in Estonia so that they could easily find and recruit developers to work on the application. Popularity was also a priority for the author of this thesis because popular frameworks have communities were developers can ask for help when they are stuck. They also have design pattern or solution for commonly faced problems which can help save time during development. React is the most popular framework according to a survey conducted by State of JS [17] followed by Vue and then Angular.

Core Features:

Angular is intended for large applications and it comes with modules that handle task like templating, state management, form validation and processing, HTTP communication and routing etc. out of the box which makes it a relatively heavyweight framework. React and Vue on the other hand are considered lightweight frameworks. At their core, they only represent the view layer of an application. Therefore, features like state management, form validation and processing, HTTP communication and routing are left to be customized. Developers can choose what features they would like to add to the framework and customize is as required by their project. React and Vue have support from their ecosystem of libraries that provide solutions for most common problems. The lightweight nature of React and Vue

was an advantage when considering the framework to use for ValChrom bearing in mind the time constraint set for the project, there was no need to add features not needed at that time.

Final Decision:

After comparing Angular, React and Vue against the constraints and requirements of ValChrom, Vue was chosen as the framework to implement the frontend application. Vue has a strong advantage with learning curve and it is popular enough to easily find new developers to continue the project. Vue also supported the features required to implement ValChrom and it offered the flexibility to customize what features we use (to maintain a lightweight project).

Vue Libraries

This subsection will introduce the libraries that were added to the core Vue.js to customise the framework to suit the requirements need.

Vue Router:

The Vue Router library is the official router for Vue.js applications. A router is a part of a SPA that synchronizes the address in the browser address bar with the currently displayed view. The router is what makes the URL change when navigation is triggered and helps to display the correct view matching the URL.

Vuex:

Vuex is the official state management pattern and library for Vue.js. Managing state in a SPA that has multiple components can be challenging. Vuex library is used to manage the state in the application.

Axios:

It is a promise-based HTTP client for JavaScript that provides a frontend application with a single interface for handling asynchronous HTTP requests to RESTful APIs.

Buefy:

It is a lightweight library based on Bulma framework that offers responsive UI components for Vue.js applications. Bulma is a CSS framework based on Flexbox that offers the elements to create web interfaces. It offers pre-built UI components that comply with web standards and best practices.

Integrated Development Environment (IDE)

Several factors were considered when selecting the tool to use for the development of the application, they are discussed below. Three popular JavaScript IDE/source code editors were compared before selecting one, they are: Visual Studio Code (VSCode), Atom and Sublime Text. VSCode, Atom and Sublime Text are ranked high in a survey conducted by the State of JS about the most used JavaScript tools [18].

Similarities:

All three tools are lightweight when compared to other IDEs like Eclipse and IntelliJ [19]. They have the advantage of being cross platform and supporting syntax auto-completion. VSCode provides syntax auto-complete by IntelliSense based on imported modules, function definitions, and variable types. Atom provides autocomplete through the autocomplete-plus package.

Cost:

VSCode has the advantage of being the only one among the three that is free of charge to use. Atom and Sublime Text come with some charges. It was important not to incur any unnecessary cost in development in areas where there are free solutions available.

Features:

When comparing the features that each tool offers out of the box, VSCode offers more features than Atom and Sublime text. VSCode supports building and debugging of applications out of the box while Atom provides installable packages and Sublime text does not support building and debugging.

Atom and VSCode both have Git integration, while Sublime offers git integration through plugins.

Final Choice:

Overall VSCode offers more functionality than Atom and Sublime Text but it is worth mentioning that VSCode features are relatively limited in comparison to other IDEs like Eclipse and IntelliJ, but these IDEs are not considered for this project because they are heavyweight and have more functionalities than was required for this project. VSCode was chosen as the IDE for implementing the frontend application of ValChrom because it was the closest fit based on the factors considered.

Version Control

Three of the most used repository management services were considered and compared for this project, they are GitHub, Bitbucket and GitLab. Here are the factors that determined which service to use:

Cost:

All three services offer free unlimited public git repositories, but we needed a private repository for ValChrom to keep the source code secured. GitHub charges for private repositories while Bitbucket offers free private Git repositories and each repository can have up to 5 users. The most cost efficient is GitLab which lets an unlimited number of users on an unlimited number of private Git projects for free.

Usability:

It was important for the product owners to be able to follow up on the implementation and see the daily progress made in development. Therefore, it became important to choose a repository management services that the product owners felt comfortable working in. Bitbucket had the advantage in this area as the product owners found it more usable than GitLab and GitHub.

Final Choice:

Overall Bitbucket was a better fit for this project than the others and it was used as the repository for the entire ValChrom project.

7.2 Architecture

As mentioned earlier, the Vue core library is focused on the view layer of an application which allows it to be a lightweight framework, but this implies that it does not enforce any form of application architecture. Application architecture can vary differently across several projects depending on their requirements and constraints.

The architecture used for the development of this application is the Component-based architecture. It is a hybrid between the layered architecture and the feature-based architecture.

Examples of the layered architecture are the Model-View-Controller (MVC) and Model–view–viewmodel (MVVM). With this architecture, the application is separated into presentation layer, processing layer and data layer. The approach satisfies the concept of separation of concerns and decoupling but in a SPA this architecture can become complex and difficult to maintain as the application grows, this is because the SPA itself is not separated in layers but in features and components that are reactive to user interactions.

In a feature-based architecture, the application is separated by features. This helps to mirror the domain because the code is grouped by the domain entities and their functionalities. A possible area of complexity forms when we have services that need to communicate with each other, or we have elements that are used across several features that could be reused.

The Component-based architecture takes the benefits of the layered architecture and the feature-based architecture and combines them in a new architectural design. In this architecture, the application is first separated into modules like the feature-based architecture and the within each feature the code is separated into the layers of the MVVM. Common features and services that are used in several parts of the application can be reused.

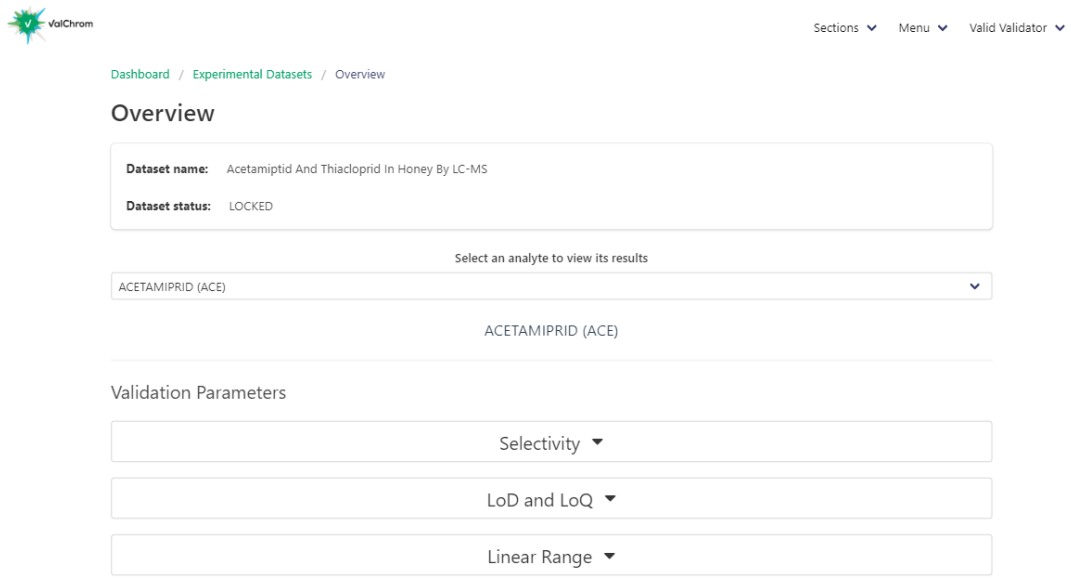


Figure 14 Validation Result Overview Page

To explain the component-based architecture further, let us examine the web page in Figure 14. To a user, this page contains horizontal bars, texts and dropdown menus but from an architectural perspective there are only components (representing features). They are placed together to make up a UI view. These components are organised in a tree structure which begins with a root node (or main component). This tree structure is the foundation of component-based architecture. Figure 15 shows the page in Figure 14 from an architectural perspective. Component-based architecture is a method for

separation individual elements of a large user interface (or components) into independent, and self-sustaining micro-systems.

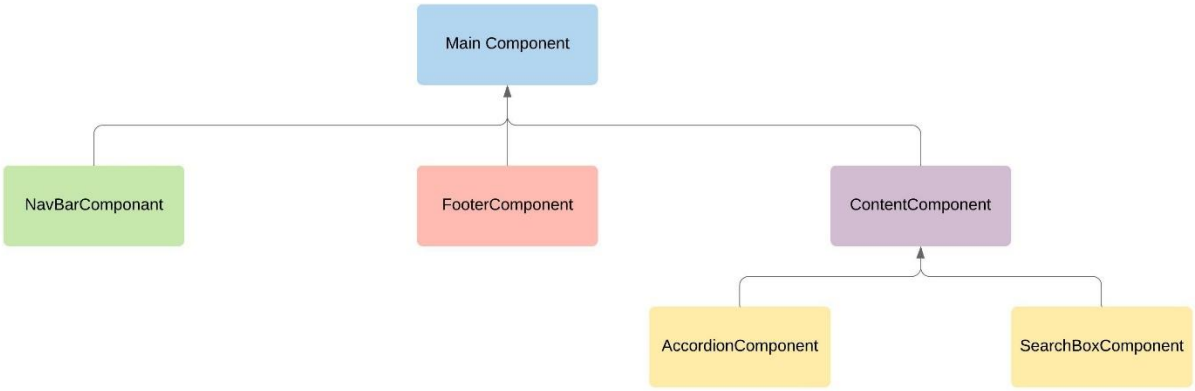


Figure 15 Architectural Perspective of Validation Result Overview Page

Project Structure

The initial setup of the project was done using the Vue command line interface (CLI). Vue CLI is developed by the Vue team to provide the ability to quickly scaffold a new project and produce a structure like the one in Figure 16.

Figure 16 Project Structure

Figure 16 shows the directory structure of the project after the initial setup. This is the standard project structure that is provided by Vue CLI on creating a new project. The purpose of each item in the directory is explained below:

- The ***node_modules*** directory is where all the libraries needed to build a Vue application are stored.
- The ***public*** directory holds any static assets that do not need to run through Webpack (JavaScript module bundler) when the project is being built e.g. the ***index.html*** file.
- The ***assets*** directory holds files like images, fonts and documents that will be imported into a component.
- The ***components*** directory holds all the components of the application that will be used as building blocks in the views of the application. They are the "Dumb" components, also referred to as presentational. They handle the look and feel and can be reused across different views. They are grouped by the domain entity e.g. method, template, plan and dataset etc.
- The ***views*** directory holds the components that are routed. They represent pages that the users will see. They are the "Smart" components also referred to as the containers. They handle how things work and state changes. They are grouped by the domain entity e.g. method, template, plan and dataset etc.
- The ***App.vue*** file is the root component where all other components are nested into.
- The ***main.js*** file is the entry point of the application that renders the App.vue component and mounts it to the Document Object Model (DOM).
- The ***router.js*** file contains the routes used in the application by Vue Router.
- The ***store.js*** file holds the Vuex elements and properties of the application.
- The ***.gitignore*** file specifies to git what files and directories to ignore.
- The ***babel.config.js*** and ***package.json*** files specify to node package manager (npm) how to handle dependences and identify the project.

7.3 Application

Entry Point

The ***main.js*** file is the entry point of a Vue project (see Figure 17). In this file, an instance of Vue is created and all the libraries needed in the project are imported. The ***main.js*** file is responsible for rendering the root component (***App.vue***) into the HTML Document Object Model (DOM) in the ***index.html*** file. The ***App.vue*** file is the root component where all other components will be nested.

```
main.js > ...
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
import Buefy from 'buefy'

Vue.use(Buefy, {
  defaultIconPack: 'fas'
})

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount('#app')
```

Figure 17 main.js file

Component Structure

The components in this project are structured as single file components. This means that for each component, the structure (HTML), behaviour (JavaScript) and styling (CSS) are encapsulated into one file. The reason for this approach is to keep related information together for better separation of concern. This is how the component-based architecture is applied, each component is a feature and then each component is an encapsulation of the layered architecture. The resulting structure of a single file component is shown in Figure 18, consisting of three parts, template (HTML), script (JavaScript) and style (CSS).

```
src > App.vue > {} "App.vue"
  1 > <template> ···
 41
 42 > <script> ···
108
109 > <style scoped> ···
187
```

Figure 18 Single File Component

Navigation

Now zooming in on one component in the application, the NavBarComponent. This component represents the feature for navigating the application. It is displayed as the navigation bar at the top of the application. The component is directly nested in the ***App.vue*** component. It displays the hyperlinks for the different pages and features in the application. The navigation feature works with the Vue

Router through the ***router.js*** file. The Vue router is configured with the routes for each view (or page) in the system. Figure 19 show how routes are configured in the application. The "path" is the URL of that view and the "component" is the view (or component) that will be rendered when that URL is accessed. The Vue Router monitors the URL of the browser for change. Once the URL changes, it searches the ***router.js*** file for the path that matches the new URL and then it displays the corresponding view.

```
{
  path: '/about',
  name: 'About',
  component: About,
  meta: {
    breadcrumb: [
      { name: 'Home', link: '/'  },
      { name: 'About' },
    ]
  }
},
```

Figure 19 Route configuration

Breadcrumbs

The breadcrumb feature works with the navigation feature to provide a dynamic navigation trail for the user, to help them to be aware of where they are in the system. To implement this feature, an extra property is added to the route configurations called "breadcrumb" (see Figure 19). The content of this property is an array containing the navigation trail of the user. Whenever a user navigates in the application, the breadcrumb feature fetches the content of the breadcrumb property and then it generates and displays the navigation trail to the user. The breadcrumb component is place in the ***App.vue*** file below the NavBarComponent. This pattern of implementation is ideal because the breadcrumb and routes are all in a central location in the ***router.js*** file and if there are any changes in future, only one file needs to be changed.

Security

Security was achieved in the application by combining the functionalities of the Vue Router, Vuex store and Axios. Visitors can access the login page, the signup page and the home page, the about page and the contact page. All other views are restricted to only authenticated users. User authentication is handled by the Login feature. When a user provides their login credentials, the login feature makes a HTTP request to the backend API to authenticate the user. If the login operation succeeds, the API sends back an authentication JSON Web Token (JWT) for that user's session [20]. Every subsequent HTTP request made to the API during a user's session must be sent with a JWT token as a means of

identification. Therefore, when the login feature receives the JWT token from the API response, it saves it in the Vuex store. That way it can be retrieved later in the user's session.

The backend has a specified length for each session, therefore if the user wants to use the application for a prolonged time, the frontend needs to request a token refresh from the backend. This is simply to ensure that the user continues interaction with the secured resources of the system. Alternatively, if the user wants to end their session or if the user has been inactive for a long time (10 minutes), the Logout feature is invoked. The logout feature sends a HTTP request to the backend API to end the user's session and then it removes the JWT token that was stored in the Vuex store.

The navigation guards provided by Vue router were used to protect the secured pages from being accessed by unauthenticated users. In the application router (the ***router.js*** file) a navigation guard is added to all the secure routers that require a user to be authenticated before they can view them. The guard used is called the "beforeEnter" guard of Vue router. It is a property added to a route configuration and its value is a method which will determine whether to redirect to the desired route or to cancel it. This method is implemented to interacts with the Vuex store and check if the current user has an active session before it allows the user to access the requested route.

REST API Calls

The API calls are the interactions between the frontend and the backend applications of ValChrom. The frontend application uses Axios to send asynchronous HTTP calls (GET, POST, PATCH/PUT and DELETE) to the API. The API calls are made in the smart components (views). When an API call is made, there is an implicit promise to return its result. Two things can happen either the result is returned, or an error is returned. Axios provides a way to handle both cases. In this application the Vuex store is used as a repository to store the data returned from API calls. If an error is return, the appropriate message is displayed to the user that requested a resource.

State Management

Vuex provides state management that offers a way to make state changes in the application. The need for state management became evident as the application grew, each component had a version of state that it utilised and there needed to be a way to communicate changes in state between components. One instance of this is the case of the Validation Result view (called Overview) which nests multiple child components and each of them needs to interact and modify the same states. Using Vuex store provided the ability to consolidate all the states into a single location that can be accessed by every component.

To utilise Vuex in this application a ***store.js*** file was created to hold the implementation of state management. Inside this file there are four (4) main properties state, getters, mutations and actions (see Figure 20). The **state** property serves as the "single source of truth". It is implemented as a single object that comprises of all the application-level state. Simply put, it is an object that contains variables of the different state in the application. The **getters** property is a collection of functions that use the state in the state property and then returns a value based on the manipulations in the function. The return value of a getter function is cached based on its dependencies, it only gets re-evaluated when a dependency has changed its value. One example of a getter function in the application is **authenticated,** this function checks to see if there is a JTW token for the user in the store and then it checks if the token is still active. The return value is of type Boolean, to indicate whether a user is authenticated or not.

Figure 20 Anatomy of Vuex Store

The mutations property is the only way states in the store can be modified. An example of its usage is when an active session of a user is about to expire and the user would like to continue using the application, as mentioned earlier, the frontend sends a request to the backend to refresh the token. The backend then sends a new token and this token needs to be saved to make future request to the backend. Since session tokens are stored in the Vuex store, the new token needs to be updated. There is a mutation function called **REFRESH_TOKEN.** This mutation is committed by calling the REFRESH_TOKEN function and passing in the new token value. The function will then modify the state for the JWT token in the store. This is how state management is handled in the application.

Validation Result Computation

This is a core functionality of ValChrom. It is responsible for providing users with the validation result of their chromatographic analytical method. The implementation of this feature involved making an API call to the backend to fetch the computation results for the experimental datasets uploaded by the user. The response comes back as a json object that contains all the input experimental datasets and their computation results. The response data fetched is in a structure that represents how the data is stored in the database. This structure is very different from how it needs to be presented to the users on the UI.

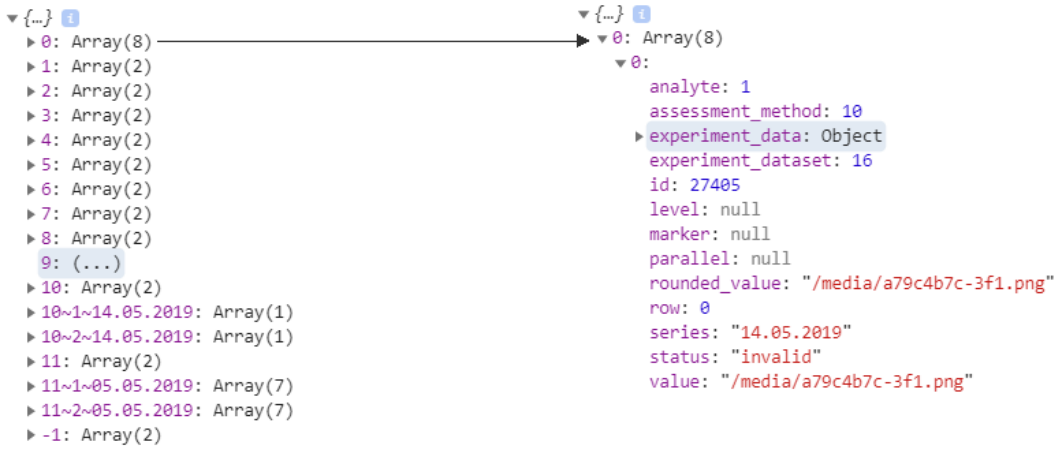


Figure 21 Anatomy of Computation Results

Figure 21 shows an example of the response data received from the backend after requesting the computation results. The data comes as a JSON object containing keys and values. The values of the experimental datasets uploaded by the user (the input data) are the items whose keys are positive integers, while the negative integers represent any additional input data that the user entered into the system through a form field and not by uploading a dataset. The computation result values are the items whose keys contain integers combined with the symbol ~. The format of the computation result item key is:

<assessment method ID>~<analyte ID>~<series>

A service was created to restructure the data into a format that can be displayed to users through the UI components. This service also separates the data based on their type (input and computation result). The restructured data is then pass to another service to compute the validation results.

Several API calls are made at the stage of computing the validation result. All the data about the current experimental dataset is needed to compute validity. The Analytical method and all the associating analytes are fetched, the validation plan template data comprising of all the validation parameters and their assessment method and the criteria are fetched. The computation service contains functions that compute the validity of an analytical method. The result of the validity computation is then displayed on the UI for the user to see as the “decision” that is valid or invalid see Figure 22.

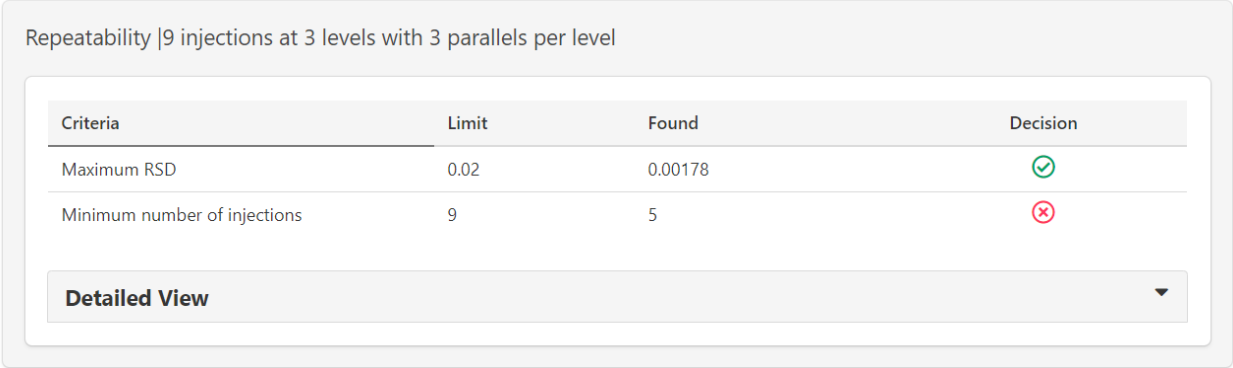


Figure 22 Analytical Method Validation Results

User experience (UX) and User Interface (UI)

The overall experience that a person has as they interact with a software is a crucial determinant of whether the person will continue using the software or not. This concept is referred to as the User Experience (UX). The UX of an application is formed by every aspect of the application like functionality, processes and the user interface. The UI of an application is the part of an application that the users interact with directly and has a high impact on the overall UX. Here are some fundamental principles that were followed to implement the UI of ValChrom in a way that would satisfy the USABILITY of the product owners and provide the desired UX (see Figure 24).

User's need:

This was a major goal behind every UI in the application and it was achieved through design validation with the product owners (who are experts in the domain and understand the needs of analytical chemists). The designs were also validated by other analytical chemists that were not part of the project to get different perspectives on how potential users will interact with the application.

Simplicity and Clarity:

The UI was designed in a way that the content of every page is presented in a simple and clear way. Whitespaces are essential in making a UI look simple and not over cluttered. Detailed information is held in dropdown components to keep the pages light. The application has a simple colour theme of green and white, and light shades of grey are used to highlight some areas in the pages. The breadcrumb UI component have a simple design and they make it possible for the user to always know where they are in the application. The navigation menus are placed in the order that makes sense to an analytical chemist and it mirrors the progression of the validation process (see Figure 24).

Consistency:

The UI design is consistent throughout the application. This is achieved by reusing the same UI component across the entire application. This provided a consistent layout, colours, font, buttons etc. The consistency across the application helps users recognise similar elements even when they are on a different page. The UI element also follows modern UI standards across the web and so users don't need to get used to strange icons of events in the application (see Figure 24).

Confirmation and Feedback:

The application communicates with the users to help them understand what is going on. This is done by providing informative pop-ups when an action succeeds or fails, asking user to confirm their action before proceeding with an action (see Figure 23).

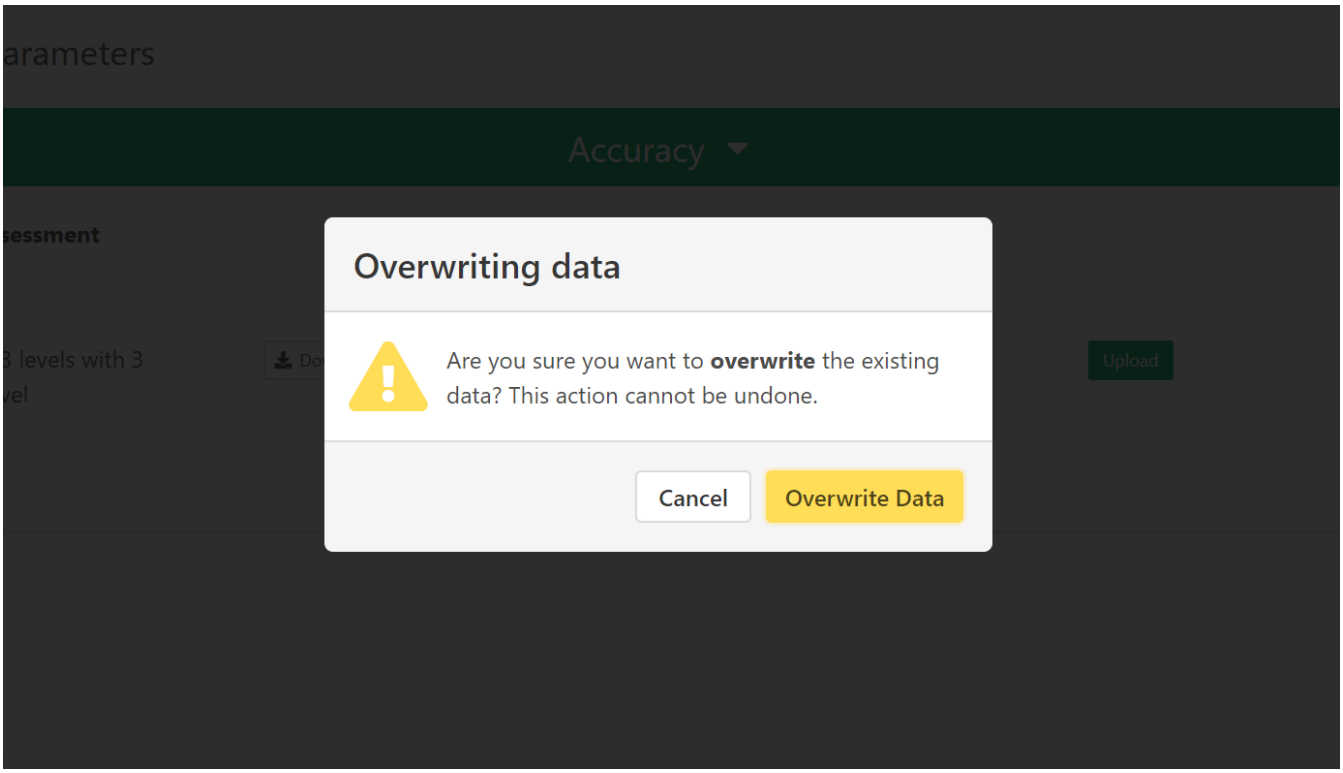
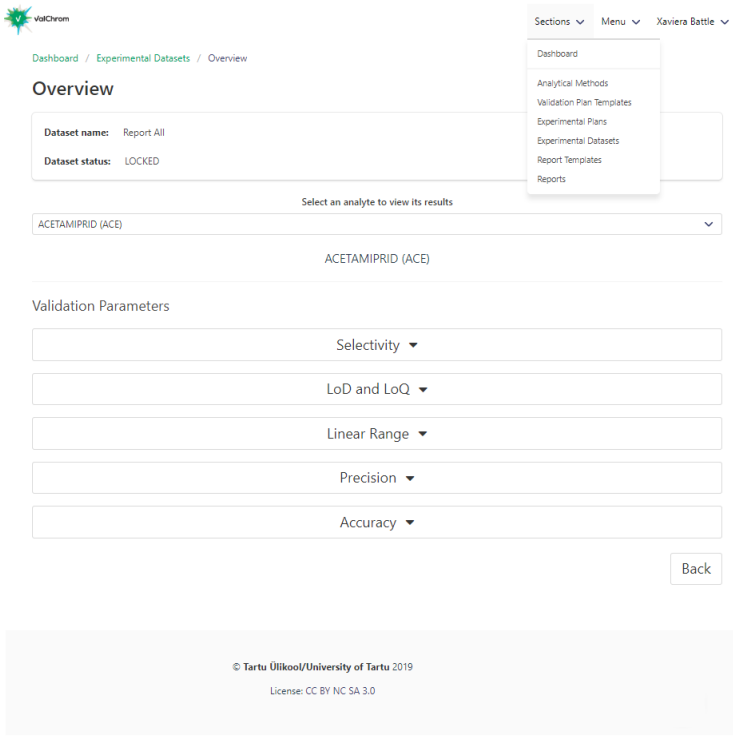


Figure 23 ValChrom UI Designs – Confirmation and Feedback



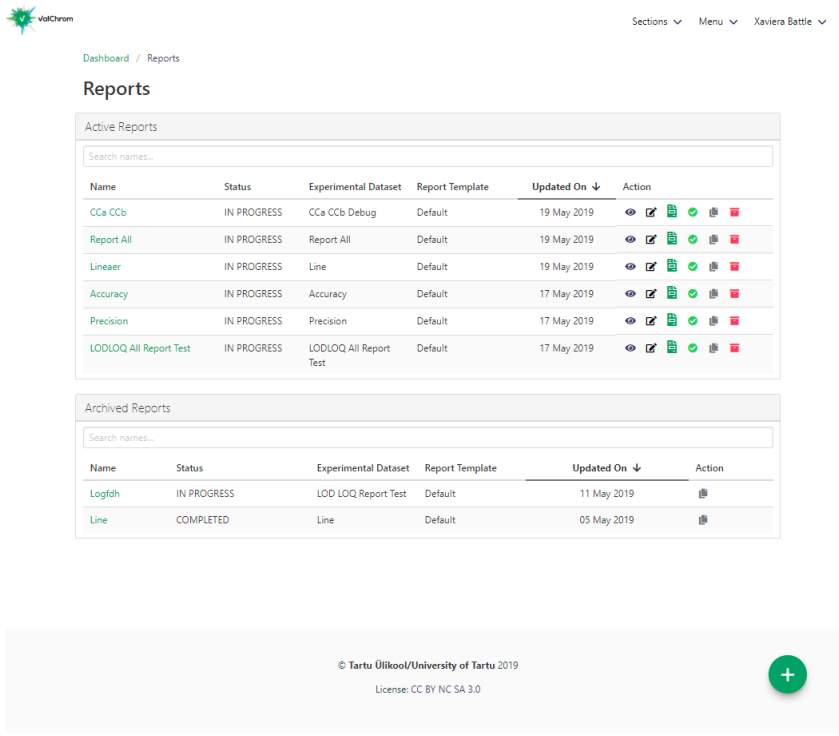


Figure 24 ValChrom UI Designs

7.4 Testing

The frontend testing was not automated, it was done using browser testing and it required the developer (the author of this thesis) and the product owners to work collectively and test UI elements and features on multiple devices and from different viewports. The testing process was conducted every week as new features were implemented. The goal was to verify that a given feature met the end user's expectations and requirements. The scope of a test was defined by the description of the feature in the user story. This helped to make sure the product owners and the developer were on the same page.

Tools for Testing

Browsers:

The browser tools used for inspecting, and testing of code were Chrome Dev Tools, and Edge Developer Tools. The browsers used for testing the UI of the application where Firefox, Chrome, Edge, Opera and Safari.

An alternative approach to manual browser testing would have been to use a tool like Selenium which is an automated testing suite for web applications. The reason this approach was not used was due to the constraint of the project. The product owners did not consider automation testing to be of high priority at the time due to the strict deadline set for the project. A higher priority was given to the implementation of new features and manual browser testing.

Bug Tracking:

For bug tracking, screenshot tools were used to capture the bugs and Trello was used to report the bug and assigning and managing the fixes.

8. Conclusion and Future Work

The goal of this thesis was to develop the client-side application for ValChrom. This goal was met by developing and delivering a client-side application that interacts with the server-side application to deliver a software-as-a-service (SaaS) solution. The software provides a solution for the entire process of chromatographic analytical method validation. ValChrom was presented at the Eurachem conference that took place in Tartu on the 20th to 24th of May 2019 and was met with positive feedback and reaction from the audience. The delivered software is still under active development by other developers. It is continually tested by a team of fifteen analytical chemists who were onboarded to the application using a twenty minutes tutorial. The client-side application of ValChrom that was delivered at the end of this thesis was a huge improvement to the existing implementation of ValChrom. The product owners were closely involved in the development process and this allowed the final product to meet their requirements. All UI elements and the functional features were validated by the product owners. Therefore, frontend application and the entire software solution provided the desired user experience the clients required.

Although the delivered application fulfils the client's requirements, those requirements were specified for a specific time frame. There are some improvements to the application that would need to be addressed moving forward. One of the main points to improve would be the scope of functionality implemented by the client-side application. As mentioned earlier, the client-side application is a thick-client because it handles some of the data processing of the system. It is not best practice to handle such computations on the client-side, but this architectural decision was made due to the capabilities of the API and the way the database was designed. A possible improvement to the system would be to move this computation logic to the server-side so that the client-side will be responsible for rendering of data and providing the desired user experience. Another possible improvement point would be to the architecture of the client-side application, the current architecture first separates components as views (routed pages) and components (presentational components) and then further groups them by modules. This architecture works for the current size of the application but as the application continues to grow and implements new features it might become complex to maintain.

A possible improvement would be to structure the architecture the other way around by first separating components by modules and then inside every module further separate the components as views and components. The goal would be to have a highly modular application architecture so that features can be plugged in easily or removed easily. This style of architecture is highly maintainable and provides a level of independence between the modules of the application.

Another possible improvement to the system would be to use Selenium or similar automated testing suites for testing the client-side application to make the testing process more structured and time efficient.

9. References

1. European Medicines Agency. (1995). ICH Topic Q 2 (R1) Validation of Analytical Procedures. *Text and Methodology*. CPMP/ICH/381/95 (1), p1-15.

2. Khanacademy.org. (2018). *Principles of chromatography.* Available: https://www.khanacademy.org/test-prep/mcat/chemical-processes/separations-purifications/a/principles-of-chromatography. Last accessed 20th December 2018.

3. González, A. Gustavo, and M. Ángeles Herrador (2007) A Practical Guide to Analytical Method Validation, Including Measurement Uncertainty and Accuracy Profiles. TrAC Trends in Analytical Chemistry 26(3): 227–238.

4. Analyse-it. (2018). *Method Validation Edition.* Available: https://analyse-it.com/products/method-validation. Last accessed 20th December 2018.

5. S-Matrix Corp. (2018). *Fusion Analytical Method Validation.* Available: http://www.smatrix.com/fusion_lc_method_validation.html. Last accessed 20th December 2018.

6. Ravisankar, P. (2013). *Analytical method validation.* Available: https://www.slideshare.net/banuman35/analytical-method-validation-by-pravisankar. Last accessed 20th December 2018.

7. *HPLC: A Practical User's Guide*. https://epdf.tips/hplc-a-practical-users-guide52ee5a326ca7ce44d9e488bb9a0f39ac78269.html, accessed December 21, 2018.

8. *How Does High Performance Liquid Chromatography Work?* : Waters N.d. http://www.waters.com/waters/en_MT/How-Does-High-Performance-Liquid-Chromatography-Work%3F/nav.htm?cid=10049055&locale=en_MT, accessed December 21, 2018.

9. ISO. (2011). *ISO/IEC 25010:2011.* Available: https://www.iso.org/standard/35733.html. Last accessed 10th Oct 2019.

10. Lucidchart. (2019). *What is a Data Flow Diagram?* Available: https://www.lucidchart.com/pages/data-flow-diagram#section_2. Last accessed 18th Oct 2019.

11. Larman, C. and Kruchten, P (2002). *Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and the Unified Process*. United States: Prentice Hall Professional. 627.

12. Martin Fowler (2002). *Patterns of Enterprise Application Architecture*. United States: Addison-Wesley Professional. 560.

13. Vue.js. (2019). *Documentation.* Available: https://vuejs.org/. Last accessed 29th Oct 2019.

14. Kodjovi Hippolyte-Fayol Toulassi. (2019). *Software Tool for Validation of Chromatographic Analytical Method.* Available: https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=67352&year=2019

15. Giorgi Bakradze. (2019). *How to Choose the Best Front-end Framework.* Available: https://www.toptal.com/javascript/choosing-best-front-end-framework. Last accessed 31st Oct 2019.

16. Nikhil Tyagi. (2019). *Top 11 JavaScript Frameworks For 2019.* Available: https://www.lambdatest.com/blog/top-javascript-frameworks-for-2019/. Last accessed 31st Oct 2019.

17. State of JS. (2018). *Front-end Frameworks.* Available: https://2018.stateofjs.com/front-end-frameworks/overview/. Last accessed 31st Oct 2019.

18. State of JS. (2018). *Other Tools.* Available: https://2018.stateofjs.com/other-tools/. Last accessed 31st Oct 2019.

19. G. Andrew Duthie. (2015). *The New Shiny: Is Visual Studio Code for You?.* Available: https://devhammer.net/2015/07/01/the-new-shiny-is-visual-studio-code-for-you/. Last accessed 31st Oct 2019.

20. Flavio Copes. (2018). *JWT authentication: When and how to use it.* Available: https://blog.logrocket.com/jwt-authentication-best-practices/. Last accessed 31st Oct 2019.

21. Guru99. (2019). *What is Selenium? Introduction to Selenium Automation Testing.* Available: https://www.guru99.com/introduction-to-selenium.html. Last accessed 31st Oct 2019.

Appendix A

USER STORIES

ID	As a	I want to	So that
US01	Visitor	Create an account on the system.	I can have access to features of the system that are for registered users.
US02	User	Login to the system with the email and password I registered in the system.	I can access all my data and have access to features of the system that are for users.
US03	User	View my account information	See all the information I have saved in the application
US04	User	Edit my account information	Modify the information I have saved in the application
US05	User	Create an analytical method and save it.	I can perform validation analysis on it
US07	User	Separate my analytical methods into two groups; active and archived	So that I have a clear view of their progress
US06	User	View all my active analytical methods.	I can have an overview of all the active methods I have created.
US08	User	View all my archived analytical methods.	I can have an overview of all the methods I have archived.
US09	User	Sort all my analytical methods by specific attributes	I can have a sorted overview of my methods based on a specific attribute.
US10	User	View the details of an active analytical method.	I can review all the information I saved in it.
US11	User	Edit the details of an active analytical method.	I can modify the information I saved in it.
US12	User	Change the status of active analytical methods from “IN PROGRESS” TO “COMPLETED”.	I can specify which methods are complete and ready for use.
US13	User	Archive an active analytical method.	I can remove analytical methods that are not needed anymore from my active view.
US14	User	Duplicate an active or archived analytical method.	I can create an active replica of it
US15	User	Create a validation plan template using a guideline and save it.	I can use it to validate analytical methods
US16	User	Separate my validation plan templates into two groups; active and archived	So that I have a clear view of their progress

US17	User	View all my active validation plan templates.	I can have an overview of all the active templates I have created.
US18	User	View all my archived validation plan templates.	I can have an overview of all the templates I have archived.
US19	User	Sort all my validation plan templates by specific attributes	I can have a sorted overview of my validation plan templates based on a specific attribute.
US20	User	View the details of an active validation plan template.	I can review all the information I saved in it.
US21	User	Edit the details of an active validation plan template.	I can modify the information I saved in it.
US22	User	Change the status of active validation plan templates from “IN PROGRESS” TO “COMPLETED”.	I can specify which templates are complete and ready for use.
US23	User	Archive an active validation plan template.	I can remove the validation plan templates that are not in use anymore from my active view.
US24	User	Duplicate an active or archived validation plan template.	I can create an active replica of it
US25	User	Create an experimental plan using an existing analytical method and validation plan template.	I can use it to perform laboratory experiments
US26	User	Separate my experimental plans into two groups; active and archived	So that I have a clear view of their progress
US27	User	View all my active experimental plans.	I can have an overview of all the active experimental plans I have created.
US28	User	View all my archived experimental plans.	I can have an overview of all the experimental plans I have archived.
US29	User	Sort all my experimental plans by specific attributes	I can have a sorted overview of my experimental plans based on a specific attribute.
US30	User	View the details of an active experimental plan.	I can review all the information I saved in it.
US31	User	Edit the details of an active experimental plan.	I can modify the information I saved in it.
US32	User	Change the status of active experimental plans from “IN PROGRESS” to “COMPLETED”.	I can specify which plans are complete and ready for use.

US33	User	Archive an active experimental plan.	I can remove the experimental plans that are not in use anymore from my active view.
US34	User	Duplicate an active or archived experimental plan	I can create an active replica of it
US35	User	Create an experimental dataset using an existing experimental plan.	I can save information about my laboratory experiments.
US36	User	Separate my experimental datasets into two groups; active and archived	So that I have a clear view of their progress
US37	User	View all my active experimental datasets.	I can have an overview of all the active experimental datasets I have created.
US38	User	View all my archived experimental datasets.	I can have an overview of all the experimental datasets I have archived.
US39	User	Sort all my experimental datasets by specific attributes	I can have a sorted overview of my experimental datasets based on a specific attribute.
US40	User	View the details of an active experimental dataset.	I can review all the information I saved in it.
US41	User	Upload data into an active experimental dataset.	I store the result data of my experiment.
US42	User	Change the status of active experimental datasets from “IN PROGRESS” TO “COMPLETED”.	I can specify which datasets are complete and ready for validation.
US43	User	Archive an active experimental dataset.	I can remove the experimental datasets that are not in use anymore from my active view.
US44	User	Duplicate an active or archived experimental dataset	I can create an active replica of it
US45	User	View an overview of the validation results for my method	I can have a summary of the validation results
US46	User	View the details of my method validation results	I can have all the details of the validation analysis
US47	User	Create a report using a report template and an experimental dataset.	I can have a complete account of the validation process I performed.
US48	User	Separate my reports into two groups; active and archived	So that I have a clear view of their progress
US49	User	View all my active reports.	I can have an overview of all the active reports I have created.
US50	User	View all my archived reports.	I can have an overview of all the reports I have archived.

US51	User	Sort all my reports by specific attributes	I can have a sorted overview of my reports.
US52	User	View the details of an active report.	I can review all the information I saved in it.
US53	User	Change the status of active reports from “IN PROGRESS” TO “COMPLETED”.	I can specify which reports are complete and ready for use.
US54	User	Archive an active report.	I can remove reports that are not needed anymore from my active view.

Appendix B

USE CASES

Membership Use cases

Use Case: Create Account

Name	Create Account
Actor	Visitor
Goal	The purpose of this feature is to create an account for a visitor to access the system.
Input	The visitor inputs an email, password and other information.
Output	The visitor is informed that the account has been created or that the operation failed.
Main Scenario	A visitor enters ValChrom site and wishes to use its features. Before the visitor can access the system, they must have a user account.
Pre-condition	Visitor is connected to the internet and using a modern internet browser.
Post-condition	The visitor receives a user account.
Exceptional Scenario(s)	In the case of an error, the user will be informed about it and told that the account was not created.

Use Case: Login

Actor	**User**
Goal	The purpose of this feature is to enable system users have access to system features.
Input	The user inputs their email, password.
Output	The user is informed that they have successfully logged into the system or that the login operation failed because of specified reasons.
Main Scenario	A user enters ValChrom site and wishes to use its features. The user is prompted to login with their email and password.
Pre-condition	User is connected to the internet and using a modern internet browser. The must have a valid user account.
Post-condition	The user receives access to system features.

Exceptional Scenario(s)	In the case of an error, the user will be informed about it and told that the login operation failed.

Use Case: Logout

Actor	**User**
Goal	The purpose of this feature is to enable authenticated users to end their session on the web application.
Input	The user clicks the logout button.
Output	The user is informed that they have successfully logged out of the system or that the logout operation failed because of specified reasons.
Main Scenario	A user has been authenticated and has an ongoing session. The user clicks on the logout button.
Pre-condition	User is connected to the internet and using a modern internet browser. The user must have an ongoing session.
Post-condition	The user is taken out of their session to access system features.
Exceptional Scenario(s)	In the case of an error, the user will be informed about it and told that the logout operation failed.

Collections Use cases

Use Case: Sort Collection

Actor	User
Goal	Enable users to sort collections of objects (e.g. analytical methods)
Main Scenario	A user enters a module of the system and views a collection of objects. The user can sort; in ascending or descending order, the collection based on different attributes (e.g. date of creation, status, last updated etc)
Pre-condition	User is logged in to the system.
Post-condition	The user views a sorted collection of objects.

Use Case: Search Collection

Actor	User
Goal	Enable user to search a collection of objects (e.g. analytical methods)
Input	The user inputs their search criteria.
Output	A collection of objects that meet the search criteria.
Main Scenario	A user enters a module of the system and views a collection of objects. The user can search for a specific object in the collection based on different attributes (e.g. name, status etc)
Pre-condition	User is logged in to the system.
Post-condition	The user views objects in the collection that match the search criteria.

Use Case: Mark an Item as Complete

Actor	User
Goal	Enable users to mark an item in a collection of objects (e.g. analytical methods) as complete
Main Scenario	A user enters a module of the system and views a collection of objects. The user can mark an item as complete when they entered all the information require and fulfilled all the task for that object.
Pre-condition	User is logged in to the system.
Post-condition	The status of the object changes from “IN PROGRESS” to “COMPLETED”.

Analytical Methods Use cases

Use Case: Create Analytical Method

Actor	User
Goal	Enable users to create an Analytical method object in the system
Input	Analytical method name, description, analytes and other required information
Output	A new object of Analytical method.
Main Scenario	The user enters the Analytical Methods module of the system. The user selects the option to create a new method. The system provides a form to enter in the data and the user submits the form.
Pre-condition	The user is logged in to the system.
Post-condition	The system creates a new object of Analytical method with all the data the user provided.

Use Case: View all Analytical Methods

Actor	User
Goal	Enable users to view all the Analytical methods they have created.
Main Scenario	The user navigates to the Analytical methods module of the system and views all the method they have created.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays the analytical methods that have been created by the user.

Use Case: View an Analytical Method

Actor	User
Goal	Enable users to view an Analytical method from the list of those they have created.

Main Scenario	The user navigates to the Analytical methods module of the system and views all the method they have created. User selects one to view in more detail.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays all the information stored about the analytical method selected by the user.

Use Case: Edit an Analytical Method

Actor	User
Goal	Enable users to modify an Analytical method object in the system
Main Scenario	The user enters the Analytical Methods module of the system. The user selects a method to modify. The system provides a form for the user to modify any information about the Analytical Method and save it.
Pre-condition	The user is logged in to the system.
Post-condition	The system saves the new information the user provided about the Analytical method.

Use Case: Archive an Analytical Method

Actor	User
Goal	Enable users to archive an Analytical method from the list of active Analytical methods they have created.
Main Scenario	The user navigates to the Analytical methods module of the system and views all the active method they have created and then selects one to archive.
Pre-condition	The user is logged in to the system.
Post-condition	The archived analytical method is moved from the collection of active methods to the list of archived methods.

Use Case: Duplicate an Analytical Method

Actor	User
Goal	Enable users to duplicate an Analytical method from the list of those they have created.
Main Scenario	The user navigates to the Analytical methods module of the system and views all the method they have created and select one to duplicate. The system prompts the user to enter a name for the new Analytical method.
Pre-condition	The user is logged in to the system.
Post-condition	The system creates a new Analytical method using the new name provided by user and the information from the duplicated method.

Validation Plan Templates Use cases

Use Case: Create Validation Plan Template

Actor	User
Goal	Enable users to create a Validation plan template object in the system
Input	Validation plan template name, comment and other required information
Output	A new object of Validation plan template.
Main Scenario	The user enters the Validation plan template module of the system. The user selects the option to create a new validation plan template. The system provides a form to enter in the data and the user submits the form.
Pre-condition	The user is logged in to the system.
Post-condition	The system creates a new object of Validation plan template with all the data the user provided.

Use Case: View all Validation Plan Templates

Actor	User
Goal	Enable users to view all the Validation plan templates they have created.
Main Scenario	The user navigates to the Validation plan templates module of the system and views all the validation plan templates they have created.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays the validation plan templates that have been created by the user.

Use Case: View a Validation Plan Templates

Actor	User
Goal	Enable users to view a Validation plan template from the list of those they have created.
Main Scenario	The user navigates to the validation plan templates module of the system and views all the templates they have created. User selects one to view in more detail.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays all the information stored about the validation plan template selected by the user.

Use Case: Edit a Validation Plan Template

Actor	User
Goal	Enable users to modify a validation plan template object in the system
Main Scenario	The user enters the validation plan templates module of the system. The user selects a template to modify. The system provides a form for the user to modify any information about the validation plan template and save it.

Pre-condition	The user is logged in to the system.
Post-condition	The system saves the new information the user provided about the validation plan template.

Use Case: Archive a Validation Plan Template

Actor	User
Goal	Enable users to archive a validation plan template from the list of active validation plan templates they have created.
Main Scenario	The user navigates to the Validation plan templates module of the system and views all the active templates they have created and then selects one to archive.
Pre-condition	The user is logged in to the system.
Post-condition	The archived validation plan template is moved from the collection of active templates to the list of archived templates.

Use Case: Duplicate a Validation Plan Template

Actor	User
Goal	Enable users to duplicate a Validation plan template from the list of those they have created.
Main Scenario	The user navigates to the validation plan templates module of the system and views all the templates they have created and select one to duplicate. The system prompts the user to enter a name for the new validation plan template.
Pre-condition	The user is logged in to the system.
Post-condition	The system creates a new validation plan template using the new name provided by user and the information from the duplicated template.

Experimental Plans Use cases

Use Case: Create Experimental Plan

Actor	User
Goal	Enable users to create an Experimental plan object in the system
Input	Experimental plan name and other required information
Output	A new object of Experimental plan.
Main Scenario	The user enters the Experimental plans module of the system. The user selects the option to create a new experimental plan. The system provides a form to enter in the data and the user submits the form.
Pre-condition	The user is logged in to the system.
Post-condition	The system creates a new object of experimental plan with all the data the user provided.

Use Case: View all Validation Plan Templates

Actor	User
Goal	Enable users to view all the experimental plans they have created.
Main Scenario	The user navigates to the Experimental plans module of the system and views all the experimental plans they have created.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays the experimental plans that have been created by the user.

Use Case: View an Experimental Plan

Actor	User
Goal	Enable users to view an experimental plan from the list of those they have created.
Main Scenario	The user navigates to the experimental plans module of the system and views all the experimental plans they have created. User selects one to view in more detail.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays all the information stored about the experimental plan selected by the user.

Use Case: Edit an Experimental Plan

Actor	User
Goal	Enable users to modify an experimental plan object in the system
Main Scenario	The user enters the experimental plans module of the system. The user selects an experimental plan to modify. The system provides a form for the user to modify any information about the experimental plan and save it.
Pre-condition	The user is logged in to the system.
Post-condition	The system saves the new information the user provided about the experimental plan.

Use Case: Archive an Experimental Plan

Actor	User
Goal	Enable users to archive an experimental plan from the list of active experimental plans they have created.
Main Scenario	The user navigates to the experimental plans module of the system and views all the active experimental plans they have created and then selects one to archive.
Pre-condition	The user is logged in to the system.

Post-condition	The archived experimental plans is moved from the collection of active experimental plans to the list of archived experimental plans.

Use Case: Duplicate an Experimental Plan

Actor	User
Goal	Enable users to duplicate an experimental plan from the list of those they have created.
Main Scenario	The user navigates to the experimental plans module of the system and views all the experimental plans they have created and select one to duplicate. The system prompts the user to enter a name for the new experimental plans.
Pre-condition	The user is logged in to the system.
Post-condition	The system creates a new experimental plan using the new name provided by user and the information from the duplicated experimental plan.

Experimental Datasets Use cases

Use Case: Create Experimental Dataset

Actor	User
Goal	Enable users to create an Experimental dataset object in the system
Input	Experimental plan name and other required information
Output	A new object of Experimental dataset.
Main Scenario	The user enters the Experimental datasets module of the system. The user selects the option to create a new experimental dataset. The system provides a form to enter in the data and the user submits the form.
Pre-condition	The user is logged in to the system.
Post-condition	The system creates a new object of experimental dataset with all the data the user provided.

Use Case: View all Experimental Dataset

Actor	User
Goal	Enable users to view all the experimental datasets they have created.
Main Scenario	The user navigates to the Experimental datasets module of the system and views all the experimental datasets they have created.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays the experimental datasets that have been created by the user.

Use Case: View an Experimental Dataset

Actor	User

Goal	Enable users to view an experimental dataset from the list of those they have created.
Main Scenario	The user navigates to the experimental datasets module of the system and views all the experimental datasets they have created. User selects one to view in more detail.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays all the information stored about the experimental dataset selected by the user.

Use Case: Upload Data

Actor	User
Goal	Enable users to upload data into an experimental dataset object in the system
Input	A CSV file containing result data
Output	A modified object of Experimental dataset containing more data.
Main Scenario	The user selects an experimental dataset to upload data. The system provides a form for the user to upload data into the experimental dataset and save it.
Pre-condition	The user is logged in to the system.
Post-condition	The system saves the data the user provided for the experimental dataset.

Use Case: View Validation Result

Actor	User
Goal	Enable users to view the validation result of an experimental dataset from the list of those they have created.
Main Scenario	The user navigates to the experimental datasets module of the system and views all the experimental datasets they have created. User selects one to view its validation result.
Pre-condition	The user is logged into the system.
Post-condition	The system computes and displays the validation result of the experimental dataset selected by the user.

Use Case: Archive an Experimental Dataset

Actor	User
Goal	Enable users to archive an experimental dataset from the list of active experimental datasets they have created.
Main Scenario	The user navigates to the experimental datasets module of the system and views all the active experimental datasets they have created and then selects one to archive.
Pre-condition	The user is logged in to the system.

Post-condition	The archived experimental datasets is moved from the collection of active experimental datasets to the list of archived experimental datasets.

Report Templates Use case

Use Case: View all Report Templates

Actor	User
Goal	Enable users to view all the report templates available in the system.
Main Scenario	The user navigates to the Report templates module of the system and views all the report templates in the system.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays the report templates in the system.

Use Case: View a Report Template

Actor	User
Goal	Enable users to view a report template from the list of report templates in the system.
Main Scenario	The user navigates to the report templates module of the system and views all the report templates in the system. User selects one to view in more detail.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays all the information stored about the report template selected by the user.

Reports Use cases

Use Case: Create Report

Actor	User
Goal	Enable users to create a report object in the system
Input	Report name, report template, experimental dataset and other required information
Output	A new object of Report.
Main Scenario	The user enters the Reports module of the system. The user selects the option to create a new report. The system provides a form to enter in the information and the user submits the form.
Pre-condition	The user is logged in to the system.
Post-condition	The system creates a new object of Report with all the data the user provided.

Use Case: View all Reports

Actor	User
Goal	Enable users to view all the Reports they have created.
Main Scenario	The user navigates to the Reports module of the system and views all the reports they have created.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays the reports that have been created by the user.

Use Case: View a Report

Actor	User
Goal	Enable users to view a Report from the list of those they have created.
Main Scenario	The user navigates to the Analytical methods module of the system and views all the reports they have created. User selects one to view in more detail.
Pre-condition	The user is logged in to the system.
Post-condition	The system displays all the information stored about the report selected by the user.

Use Case: Archive a Report

Actor	User
Goal	Enable users to archive a Report from the list of active reports they have created.
Main Scenario	The user navigates to the Reports module of the system and views all the active report they have created and then selects one to archive.
Pre-condition	The user is logged in to the system.
Post-condition	The archived report is moved from the collection of active reports to the list of archived reports.

License

Non-exclusive licence to reproduce thesis and make thesis public

I,

Grace Achenyo Okolo

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

ValChrom – Software Tool for Validation of Chromatographic Analytical Methods,

supervised by Prof. Marlon Dumas, Prof. Koit Herodes and Asko Laaniste.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Grace Achenyo Okolo
01/11/2019