UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Kodjovi Hippolyte-Fayol Toulassi

Software Tool for Validation of Chromatographic Analytical Method

Master's Thesis (30 ECTS)

Supervisor(s):

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

Tartu 2019

Software Tool for Validation of Chromatographic Analytical Method

Abstract:

Many industries rely on analytical procedures to analyze various substances. In the medical field they are used to perform laboratory analyzes. In the pharmaceutical industry they are used to determine and quantify the active component of a drug product as well as impurities. In the food industry they are used to identify the properties of foods and their ingredients. An analytical procedure can be assimilable to the algorithm of a chemical analysis. Due to their widespread use, analytical procedures must be validated. The validation process will prove that the chemical analysis described by the analytical procedure is judicious and fit for its intended use case. That is, the chemical analysis can accurately measure the compound it is supposed to measure. Sadly, that validation process, currently, is performed manually by analytical chemists. The completion of analytical procedure validation manually is tedious and potentially error-prone. Therefore, accessible systems that can assist analytical chemists during analytical procedure validation should be made available to them. These systems will not only ensure the consistency of the result but also alleviate the workload of analytical chemists. The Department of Chemistry of the University of Tartu has acknowledged the need of such systems and launched the implementation of one named ValChrom. This thesis highlights the implementation details of ValChrom – a web-based application for analytical procedure validation, after evaluating the strengths and shortcomings of existing similar software solutions.

Keywords: analytical procedure validation, analytical method validation, chromatography, analytical procedure validation software, analytical method validation tool

CERCS: P300, P170

Tarkvaral ine tööriist kromatograafilise meetodi valideerimiseks

Tänapäeval toetuvad paljud valdkonnad erinevate ainete analüüsimiseks analüütilistele protseduuridele. Meditsiinivaldkonnas kasutatakse neid laborianalüüside tegemiseks. Farmaatsias kasutatakse neid ravimite aktiivsete komponentide ja nende koguste määramiseks ning defektide tuvastamiseks. Toitainetööstuses määratakse nende abil toitude ja nende koostisosade omadused. Analüütilist protseduuri võib vaadelda keemilise analüüsi algoritmina. Nende protseduuride suure populaarsuse tõttu on vajalik saada neid valideerida. Valideerimine tõestab, et analüütilise protseduuri poolt kirjeldatav keemiline analüüs on antud otstarbe jaoks mõistlik: et sellega saab vajalikku ühendit piisava täpsusega mõõta. Kahjuks teostatakse protseduuride valideerimine tänapäeval käsitsi analüütiliste keemikute poolt. Käsitsi valideerides võtab aga see palju aega ja on kerge teha vigu. Seega on vajalik analüütiliste keemikute töö hõlbustamiseks luua süsteeme, mis kindlustaks tulemuse korrektsust ja teeks kogu protsessi kergemaks. Tartu Ülikooli keemia instituut on tunnistanud selliste süsteemide vajalikust ja alustas ühe sellise süsteemi - ValChrom'i - arendust. See lõputöö hindab olemasolevate lahenduste tugevaid ja nõrki külgi ning räägib analüütiliste protseduuride valideerimiseks mõeldud veebirakenduse ValChrom implementatsioonist.

Märksõnad: analüütiliste protseduuride valideerimine, analüütilise meetodi valideerimine, kromatograafia, analüütiliste protseduuride valideerimise tarkvara, analüütilise meetodi valideerimise tööriist

CERCS: P300, P170

Contents

List of Figures

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 chromatographic 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 is 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. Subsequently, data integrity cannot be guaranteed.

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[1]. 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. This thesis presents the implementation of ValChrom.

The implementation of ValChrom discussed in this thesis consists mainly of a frontend application in VueJS and a backend application is Python Django. The backend system and the front-end system communicate through a Representational State Transfer (REST) Application Programming Interface (API). The backend abstracts the analytical method validation process and provides means to represent domain concepts such as an analytical method, a validation guideline and an analytical method validation report. It takes care of the creation and mutation of those domain concepts and exposes them as resources. The frontend is a thick client application that transforms

[1] https://valchrom.ut.ee

the data from the backend and renders it in a user-friendly manner. The frontend is in charge of creating interfaces that mirror the analytical method validation processes and provides the desired user experience.

This thesis is divided into five chapters. This introductory chapter is followed by chapter 2, which provides some context and background about the problem domain. Its sections give an introduction about chromatography – an analytical method and about analytical method validation. A review of existing solutions for analytical method validation on the market is offered. The project's requirements and the technologies used are also discussed. Chapter 3 details the system design. It describes the system's architecture, its domain model and elaborates on the system's resources. Chapter 4 discusses the way the system has been packaged and shipped. An overview of testing activities conducted during the project is given. We conclude this thesis in Chapter 5 and hint at some possible improvements that can be made to the delivered system.

This software project was a joint work with Grace Achenyo Okolo, 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. Grace Achenyo Okolo and I were respectively in charge of the frontend application and the backend application. This thesis focuses on the implementation of the backend application whilst Grace Achenyo Okolo's focuses on the implementation of the frontend application.

Only Section 1 of this thesis was written jointly with Grace Achenyo Okolo. All the other sections are individual work.

2. Background

This chapter starts with an introduction to the problem domain. It gives an overview of chromatography as an analytical technique and of validation of an analytical method. It continues with a review of existing software tools that are used to validate analytical methods. The chapter ends with a specification of the system's requirements and a discussion of the technologies used.

2.1. Chromatography

Chromatography is an analytical technique that makes it possible to separate compounds present in a mixture from one another. Compounds can exist in liquid or gas phases and have different affinities for other chemical particles. The technique leverages differences in partitioning behavior of the compounds between two immiscible phases. The two phases are referred to as the mobile phase and the stationary phase [1]. The mobile phase is liquid in the case of liquid chromatography, and gas in the case of gas chromatography Compounds additionally have distinct physical proprieties such as their solubility, boiling point, capability to absorb ultraviolet light that are also involved in the chromatographic analysis process.

In practice, in a High-Performance Liquid Chromatography (HPLC) system, for example, a high-pressure pump is used to generate a continuous flow of the mobile phase which is also known as the eluent. A small amount of the mixture to be analyzed is then injected into the mobile phase flow. The mobile phase flow subsequently carries the mixture into the chromatography column where the separation of mixture components takes place. The chromatography column is a tube which holds the stationary phase. The dimensions of the column namely the internal diameter and the length can significantly impact the efficiency, sensitivity, and speed of the analysis. Often, the choice of column dimensions is based on the chromatographic application and on the number of compounds present in the mixture [2]. Commonly in liquid chromatography 5-25 cm long columns with internal diameters in the range from 1-4.5 mm are used. The stationary phase consists of porous spherical micro-particles of 2-5 μm in diameter [2]. Inside the chromatography column, the respective affinities of the compounds under inspections for the particles present in the stationary phase cause the respective compounds to elute at different points in time, ensuring their separation. A detector spots the arrival of each compound against a background of the mobile phase as they exit the column. The detector is connected to a computer system which interprets the arrival of a new compound as an electrical signal. The intensity of the signal helps in determining the concentration of the compound. A graph of the signal as a function of time is sketched by the computer system. That graph is referred to as a chromatogram [3]. A chromatogram depicts peak heights or areas plotting the concentration of the compound or analyte present in the mixture as a function of time. The detector takes advantage of the analytes' properties to ascertain their arrival. Since those properties vary from one compound to the other often a combination of different types of detectors is used [3]. The possibility of combining detectors lead to the appearance of the so-called Liquid Chromatography Mass-Spectrometer LC/MS which couples a mass spectrometer to a high-performance liquid chromatography system resulting in more extensive information about a compound from one single injection. Figure 2-1 shows the components of a High-Performance Liquid Chromatography system.

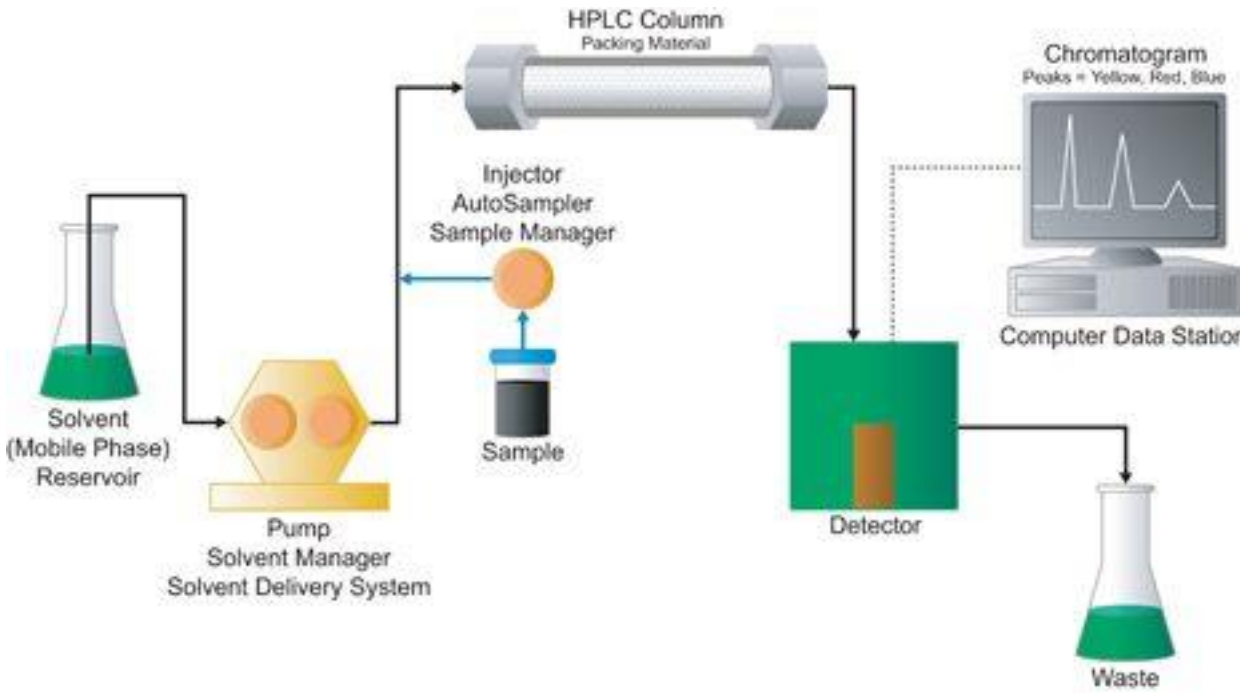

Figure 2-1 High-Performance Liquid Chromatography (HPLC) System

Waters (2018). High-Performance Liquid Chromatography [HPLC] System. [image] Available at: http://www.waters.com/waters/en_MT/How-Does-High-Performance-Liquid-Comatography-Work%3F/nav.htm?cid=10049055&locale=en_MT.

2.2. Validation of Analytical Procedure

The term "analytical procedure" refers to the step-by-step description of the necessary activities that need to be carried out while performing an analysis [4]. It is assimilable to an algorithm of the analysis.

The validation of an analytical procedure refers to the process used to demonstrate that the analytical procedure is robust, appropriate and suitable for its intended purpose [5], [6]. The term "analytical procedure" can be used interchangeably with "analytical method" [7].

Four most common types of analytical procedures can be distinguished [6]:

- Identification tests;
- Quantitative tests for impurities' content;
- Limit tests for the control of impurities;
- Quantitative tests for the active moiety in samples of a drug substance or a drug product or for other selected component(s) in the drug product.

The purpose of identification tests is to identify an analyte in a sample by examining a characteristic of the sample against that of a recommended standard. Tests for impurities are intended to exhibit the purity characteristics of a sample. Assay procedures' goal is to quantify the analyte present in a given sample [6].

During the validation of a given analytical procedure, a certain number of characteristics called validation characteristics or validation parameters must be considered. These characteristics are specificity, linearity, range, quantitation limit, detection limit, accuracy, precision, and robustness.

2.2.1. Specificity

An analytical method is said to be specific for a compound when the method can unequivocally determine the compound in a sample without the interference of other components. Specificity is confirmed when samples containing the analyte yield positive results and samples not containing the analyte yield negative results [5]. Specificity is tightly linked to the primary goal of chromatography which is the adequate separation of a given mixture [8]. Specificity is best demonstrated by the resolution R_S of two adjacent components appearing on a chromatogram. The resolution is a measure of the quality of separations, it expresses to what extent the peak of an analyte can be separated from that of an adjacent analyte.

2.2.2. Accuracy

The accuracy also termed trueness of an analytical method serves as the measure of the proximity between the test results observed using the method and the true value. The true value is a generally agreed upon authoritative value [6].

2.2.3. Precision

The precision of an analytical procedure indicates the degree of scatter between a set of measurements collected from repeated use of the analytical procedure against multiple samplings of the same sample under the recommended settings. Precision is often denoted as the standard deviation or coefficient of variation of a statistically significant series of measurements [6]. Based on the experimental conditions and environment we can distinguish:

- Repeatability – results obtained from the method while performing during a short time interval under identical settings
- Intermediate precision – results from within-laboratories variations such as different days, analysts, equipment
- Reproducibility – the result of collaborative studies between laboratories

2.2.4. Linearity

The linearity of an analytical method is the capability of the method to extract, within a given range, test results that are in some way proportional to the concentration of analyte in a sample. The linearity of an analytical method indicates to what extent a calibration curve representing the signal as a function of the concentration of analyte satisfies a linear equation [1], [6].

2.2.5. Range

According to the ICH [6] guideline “the range of an analytical method is the closed interval between the upper and lower concentration of analyte in a sample for which, it has been proven that the analytical method has a suitable level of precision, accuracy, and linearity”.

2.2.6. Detection limit

The detection limit of an analytical procedure represents the smallest concentration of analyte in a sample that can be reliably detected by the analytical procedure but not necessarily quantitated as

an exact value. It is defined as a peak whose signal is at least three times the noise registered by the system [1], [6].

2.2.7. Quantitation limit

The quantitation limit of an analytical method represents the minimum concentration of compound in a sample that can be gauged as an exact value by the analytical procedure with acceptable precision and accuracy. Analogously to the detection limit, the quantitation limit is defined as a peak whose signal is at least ten times the noise registered by the system [1], [6]

2.2.8. Robustness

The robustness of an analytical procedure indicates the degree of reliability of the procedure. It is the ability of the analytical procedure to not be altered by small intended variations in method parameters. Robustness can be used to establish system suitability parameters [6].

2.3. Existing Solutions

Analytical method validation is both a regulatory requirement and a scientific necessity. Therefore, it must be completed with the utter-most rigor. In an effort to get rid of possible human errors that can happen during the method validation process, a handful of software solutions that automate the entire process have been implemented. Unfortunately, they usually do not come as standalone solutions but rather as additional modules on top of a fully-fledged enterprise-wide software system. It follows a steep cost of acquisition rising to EUR 100000 at time.

Empower 3 Method Validation Manager (MVM) is distributed as an option for Empower 3 Chromatography Data Software developed by Waters Corporation which also specializes in laboratory equipment such as Ultra High-Performance Liquid Chromatography UHPLC procurement. Empower MVM integrates nicely with the chromatography systems procured by Waters Corporation and a few other vendors to directly fetch experiment data from the data station thus eliminating one of the most error-prone steps in a typical manual validation process which is data transfer to additional software applications. Empower MVM allows to automatically manage the validation workflow, check the status of ongoing validation studies, perform statistical calculations and generate validation reports. As a result, organizations implementing Empower MVM can envision a reduction of up to 80% in time and cost pertaining to the validation process. Moreover, the solution implements the latest regulations regarding data security [9].

Fusion Analytical Method Validation is part of the Fusion QbD Software Platform developed by a company named S-Matrix. It is often used in combination with Fusion LC Method Development which is a tool that is designed to assist analytical chemists in analytical method development. While the association of these two software tools can be of tremendous help for an experienced analytical chemist, it often is a source of confusion for much less experienced analytical chemists and steepens their learning curve. Fusion Analytical Method Validation is referred to as “the software that does it all” by its creators. It can be used to validate liquid chromatography methods as well as gas chromatography methods. Fusion Analytical Method Validation can be integrated with chromatographic data stations from multiple vendors including Agilent, Thermo, and Waters. It provides ways to automate method validation experiments. It is statistically rigorous, supports multiple analytes, creating complete reports for each. In addition, Fusion Analytical Method Validation is fully compliant with validation guidelines from the United States’ Food and Drug Administration (FDA) and the International Conference on Harmonization of Technical

Requirements for Registration of Pharmaceuticals for Human Use (ICH). The solution's main shortcoming is probably the fact that it is not available as a SaaS [10].

ValGenesis Validation Lifecycle Management System VLMS implemented by ValGenesis, Inc. is a web-based solution that companies can leverage to effectively manage all types of validations activities including analytical methods validation. The system is designed to get rid of the inefficiencies found in manual validation lifecycle management. Analytical method lifecycle management with ValGenesis VLMS is entirely automated. Like Fusion QbD Software Platform, ValGenesis integrates analytical method development and validation in one system; providing the ability to access data and documentation related to method development and validation in one single repository [11].

Enoval from PharmaLex is a software that provides a means to validate physico-medical analytical methods and generate matching validation reports. Enoval is fully compliant with validation guidelines from the FDA, ICH, European Medicines Agency EMA and United States Pharmacopeia USP. Enoval is a software as a service, therefore, users always get access to the latest version with no extra update fees. Enoval is the existing software closest to the solution implemented in the present thesis.

2.4. Scope and requirements

In a software development project, requirements analysis is the phase where stakeholders' needs are identified [12]. This is a critical step to ensure success in the development of the project since it establishes clearly and unambiguously what is required, and what is expected as a deliverable at the end. A series of five interviews conducted with analytical chemists from the University of Tartu's Chemistry Department namely Koit Herodes, head of the analytical chemistry testing center and Asko Laaniste, chemist in chair of analytical chemistry allowed us to elicit the envisioned system's functional requirements and get a glimpse of its desired quality attributes.

The first interview focused on getting a general idea of analytical chemistry and method validation, the way analytical method validation has been performed and the pain points that the system should solve. The project's deadline and success criteria were also discussed. It transpired from this first interview that analytical method validation can be regarded as a three-step process namely the planning step, the experiment step, and the reporting step. Each one of these steps have been discussed in a separate interview. The last interview focused on user management in the system.

The planning step encompasses all activities that help in preparing the lab experiments. During this step, ValChrom should provide means for the chemist to clearly document the analytical method under validation. The compounds also known as analytes of interest in the analytical method should be specified along with their targeted concentration and the units that will be used for the measurements made during the experiments. The chemist also carefully selects the guideline that the validation process will be based on. Several validation guidelines have been defined and agreed upon by the scientific community to make analytical method validation a systematic process. Each guideline recommends a set of methods to test the different validation parameters. These methods are referred to as assessment methods and they come along with a set of criteria that could be used to assert the validity of a given validation parameters. A validation parameter is deemed valid when the experiment results satisfy the criteria. Guidelines usually define default values for the criteria. Depending on the use case, the chemist can select a subset of the recommended assessment methods that are more suitable for the analytical method and define

custom values for the criteria. The outcome of the planning step is an experimental plan that can be downloaded as a PDF file. An experimental plan is a document that compiles the list of lab experiments that need to be completed as part of the ongoing method validation process along with instructions to be followed by the chemist while conducting the experiments. Figure 2-2 shows the Business Process Model Notation (BPMN) diagram of the planning step.

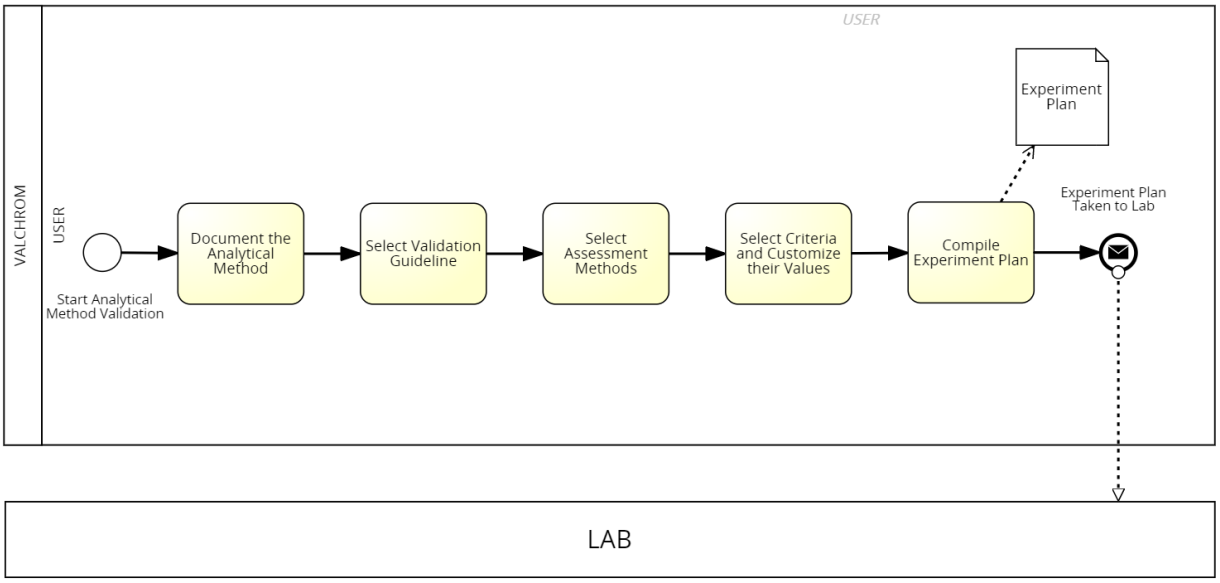

Figure 2-2 BPMN Diagram - Planning Step

The experiment step includes performing lab experiments and evaluating the various validation parameters. After conducting the experiments in a lab, the chemist is responsible for converting the output of the lab equipment into a data format supported by ValChrom and upload it into the system. ValChrom should ensure the validity of the uploaded data. Once the data is successfully uploaded, ValChrom should proceed to evaluate the different validation characteristics. ValChrom should provide analytical chemists the ability to perform method validation against three international guidelines namely ICH[6], Eurachem[13] and EMA-BA[14]. As such the system must implement all the assessment methods defined by each of these guidelines along with their criteria assessment logic. Figure 2-3 displays the BPMN diagram of the experiment step.

In the reporting step, the output of the various lab experiments conducted during the method validation process as well as the outcomes of the evaluation of the validation characteristics are compiled in a report that can be downloaded as a PDF file. According to the audience targeted by the report, it can have different levels of granularity. This behavior should be achieved by means of report templates. ValChrom should provide by default two report templates and possibly allow users to define their own. Figure 2-4 depicts the BPMN diagram of the reporting step.

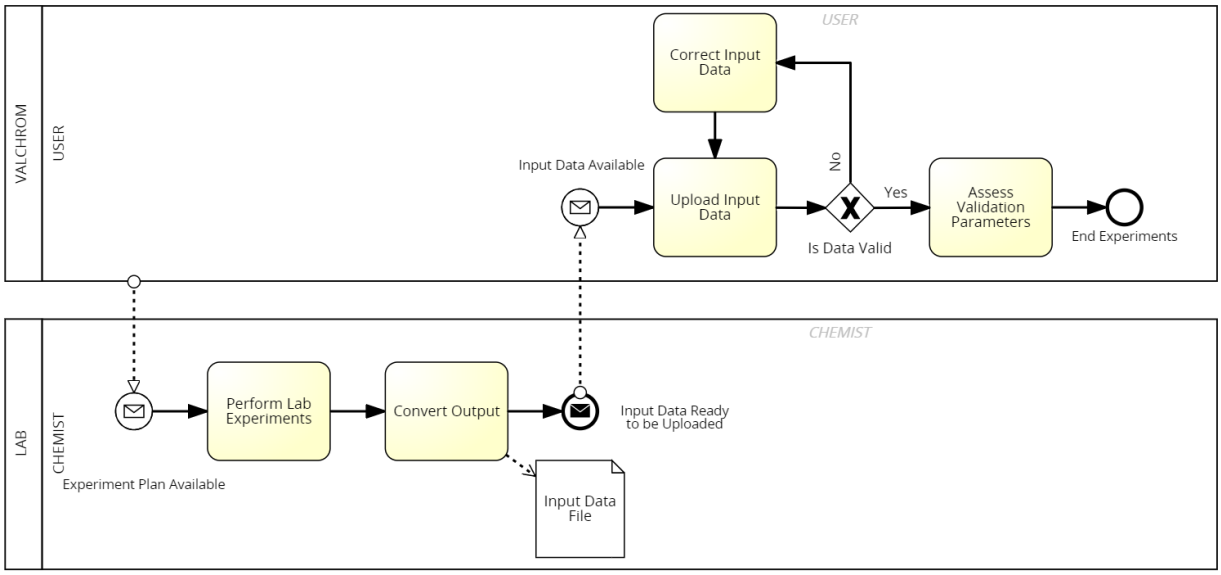

Figure 2-3 BPMN Diagram - Experiment Step

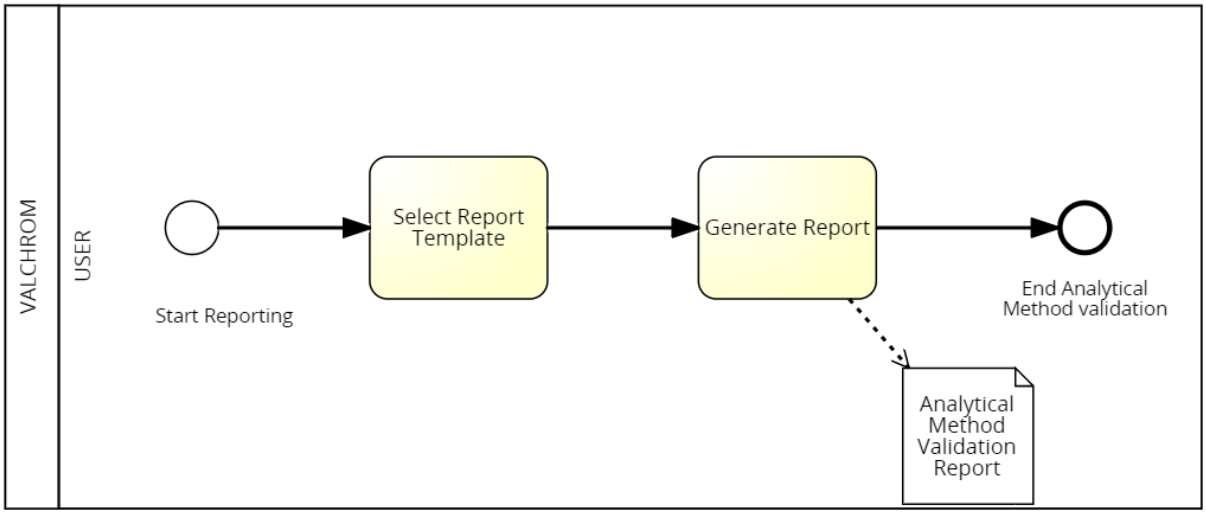

Figure 2-4 BPMN Diagram - Reporting Step

Moreover, analytical methods are intellectual properties; therefore, they should be protected. Ideally, ValChrom users should only have access to analytical methods that they own and possibly those owned by their organization. For regulatory compliance purposes all data pertaining to method validation procedures must be stored by ValChrom and no modification should be allowed to them once the validation report has been generated.

In addition to its functional requirements, ValChrom is expected to provide its users a pleasant experience. To achieve that goal, the system should be intuitive to use and allow analytical chemists without prior experience with similar tools to seamlessly ramp up after two weeks of

utilization. The system should also be able, to provide an output for computations related to a typical analytical procedure in less than two seconds with up to 200 concurrent users. The system expected to have a 95% uptime.

2.5. Technologies Used

This section presents the technical details considered during the selection of the technology used to implement the software tool and motivates the choice made.

A fundamental requirement of the system envisioned by the chemistry department is for it to be web-based. It follows that the main technology to be used in the implementation of the software tool will be a web development framework.

Our initial choice was Phoenix, a web development framework written in Elixir. Elixir is a general-purpose functional programming language that runs on the Erlang virtual machine. This choice was mainly motivated by the fact that all members of the team that will implement the software tool were already familiar with the framework. Moreover, Phoenix offers some scaffolding capabilities that will allow us to quickly bootstrap the project along with the RESTful API that will expose the system's resources to the outside world.

This initial choice has quickly been challenged by the fact that the system to be developed needs to perform extensive mathematical and statistical computations and there was no mature mathematical and statistical computation library available in Elixir. Provided that some chemists from the chemistry department had already been using scripts written in R as part of their analytical method validation processes, we considered implementing all the heavy mathematical and statistical computations in R and use them within the Elixir based application through means of interoperability. R is a programming language and environment specifically designed for statistical computation and graphing. Interoperability refers to the ability of distinct programming languages to natively pass messages and data with one another as part of the same system [15].

While the Erlang virtual machine on which runs the Elixir programming language offers several means to achieve interoperability with external programs, this adds an extra layer of complexity to the project. Also maintaining and context-switching between multiple programming languages can affect the team's productivity and hinder the project's success. All these reasons made us consider another web development framework.

Django is a web development framework written in Python. Python is an interpreted, general-purpose programming language. It is a versatile programming language due to its multi-paradigm characteristic. Python supports procedural, imperative, reflexive, object-oriented and functional programming. Python's strength lies in its comprehensive standard library and its extensive list of community-contributed modules.

Python is one of the most widely used programming language for data science [16] giving us the confidence that it can handle all our computational needs. Python's community-contributed modules NumPy and SciPy constitute the ideal combination to perform scientific computation. NumPy augments the Python programming language with powerful data structures that allow efficient computation of multi-dimensional arrays and matrices whilst SciPy supplies a large library of high-level mathematical functions to operate on these matrices and arrays.

Django was created in 2003 by web programmers at the Lawrence Journal-World newspaper[2] and publicly released in 2005. The framework which is currently in version 2.2.2 fits perfectly into the Python ecosystem since it comes like any other Python package and does not require any specific setup apart from a Python virtual environment. Django facilitates the creation of web applications by allowing components reusability and pluggability. Developers can easily reuse components from previous Django projects in their new project. They can implement common web development functionalities such as user authentication and authorization simply by plugging some existing libraries into their project and tailor the necessary parameters to their needs. The framework is also backed up by an impressive community making it easy to find help regarding technical issues. The framework implements the Model View Controller architecture and embeds an Object-Relational Mapper (ORM) that maps Python objects referred to as models into relational database tables.

Python's computational capabilities and Django's web development features satisfy all our programming needs.

[2] https://en.wikipedia.org/wiki/Lawrence_Journal-World

3. Solution

This chapter outlines the main lines of the software solution. It starts with a description of the architecture of the system along with its different components. A presentation of the domain model and the resource API follows. The chapter finishes with an overview of the system's controllers.

3.1. Architecture of the overall system

According to ISO/IEC/IEEE 42010:2011(en) Systems and software engineering — Architecture description[17], a system's architecture defines its "fundamental concepts or properties in its environment embodied in its elements, relationships, and in the principles of its design and evolution". The software solution implemented in this thesis is related to the field of analytical chemistry. This is a field about which none of the developers had prior knowledge. Misunderstanding the system requirements or making wrong assumptions about them was a major risk. To mitigate this risk, the team decided to follow an agile software development methodology with short sprints of one-week duration. This gives us the opportunity to weekly sync up with chemists and get their feedback on the current state of the system. Our overall system architecture benefited tremendously from that rapid feedback cycle as it gets amended according to the outcome of those weekly meetings.

The core of the system followed Django's MVC pattern. Requests for resources go through Django's router that directs them to the relevant controller. The controller processes the requests and returns the appropriate response. Request processing at the controller level generally involves performing some business logic. In case of complex business logic like executing the mathematical and statistical computations or generating the analytical method validation report PDF file, the controller delegates to a service or a helper module. The controller also delegates the serialization of results to a serializer module and eventually returns the serialized data. Serialization is the process of converting data into a format that can be transmitted over a network. The model holds a representation of a domain concept. It handles all interactions with the database. It also implements some domain-specific business logic such as the rounding of computation results to a given number of significant digits.

Besides catering for the system's functional requirements, our architecture must also make sure that the system produces an output for a typical analytical procedure within two seconds as specified in the system's requirements in Section 2.4.

During analytical method validation, multiple mathematical computations are executed. Upon integration of the first computation functions, a certain latency has been noticed in the system's response time when running computation for given analytical method validation. The observed latency continued growing as new computation functions were being implemented. Investigations revealed that many database operations were being performed to gather the data required. To alleviate the load from the database, the queries have been optimized and a caching service that would hold the most up to date version of the data added.

While the caching service considerably improved the performance, its effect could only be noticed at the second request for the resource. It turned out that the initial design of the system led to an execution of all computation related to a given analytical method validation at once. Moreover, the execution was synchronous. A modification has been introduced in the design to allow individual execution of computation functions and the possibility to run them in background. A queuing system has also been added to account for high traffic periods.

Method validation deals with sensitive data that can either be proprietary or essential to regulatory compliance. Most of the existing analytical method validation solution only offer locally installed instances giving users total control over their data. Since our system is web-based with a centralized database, users have to entrust us with their data. Therefore, there is a need to ensure that their data is available to them at all time, especially in the case of a technical outage on our side. A scheduler service has been added to perform daily database backups. The backups are saved locally, and a copy pushed to remote storage.

In addition to addressing current system requirements and technical challenges that they spawn; the system's architecture should also prepare the system to handle future increase in the number of users. In that regard, a reverse proxy service has been set up. The reverse proxy works as an intermediary between the system and the outside world. When the need would arise, many instances of the system could be spawned, and the reverse proxy could be used as a load balancer to distribute incoming traffic across the various instances of the system to provide high availability. The reverse proxy could also serve a security purpose. It would help the system being less vulnerable to attacks from the internet by dropping suspicious requests. Another advantage of the reverse proxy is the fact that it would ensure that maintenance activities completed on the system are transparent to end users.

Figure 3-1 shows the overall architecture of the system.

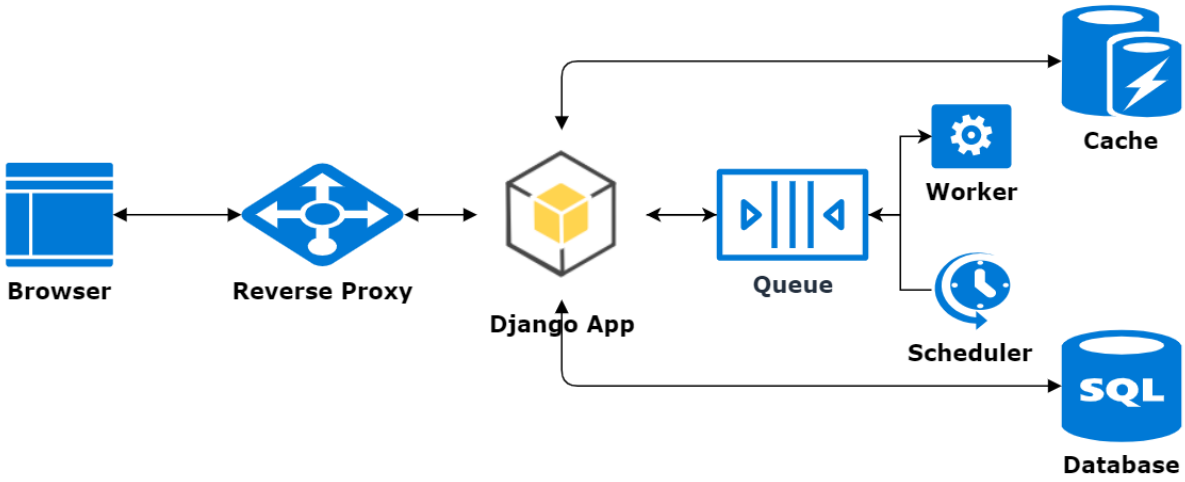

Figure 3-1 System Overall Architecture

During the review of existing software used for analytical procedure validation, one of the shortcomings that have been uncovered is their complexity. This complexity due to their association with other tools as mentioned in Section 2.3 is an important usability issue. ValChrom sets to solve this issue by designing a system that has a straightforward workflow while still embodies the highest scientific rigor. In ValChrom an analytical method validation process starts with the creation of an analytical method and a validation plan template. The analytical method and a validation plan template will in the next step be combined into an experiment plan that the chemist can follow in the laboratory. After completing the experiments in the laboratory, the raw output of the experiments will be saved in a new dataset. This data will serve as input for the computational module. The output of the computations will also be saved in the same dataset. The

dataset can be compiled into a report based on a report template after it is evaluated. Figure 3-2 presents the application's general workflow.

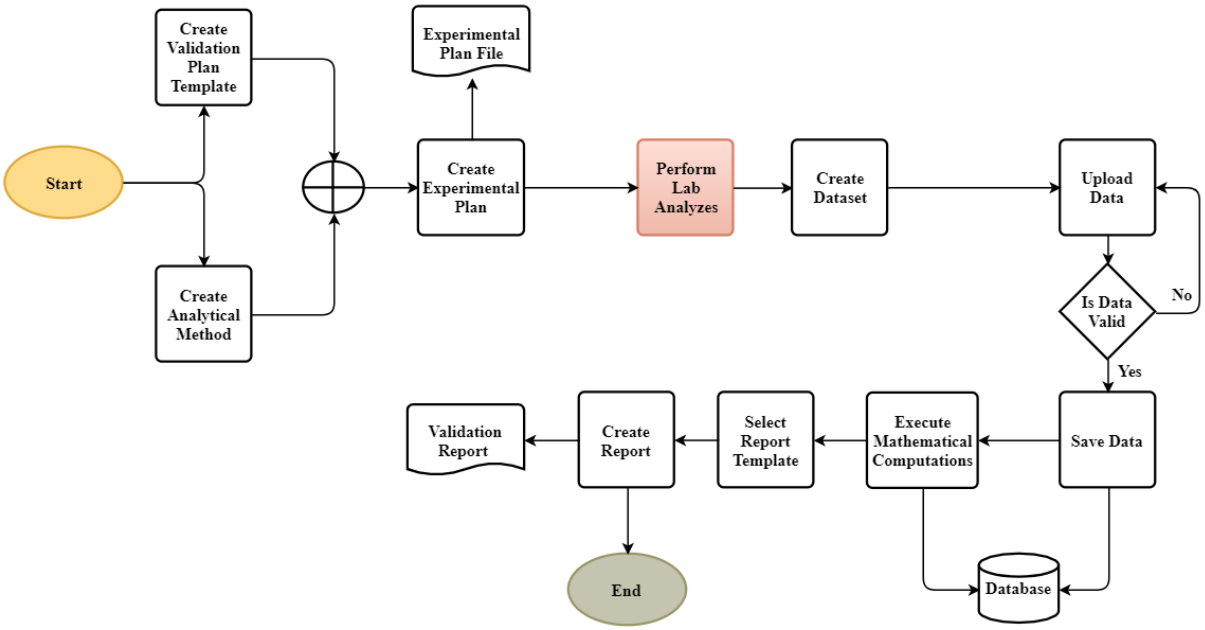

Figure 3-2 Flowchart of the validation process using ValChrom

3.2. Data formats and domain model

The first challenge encountered at the beginning of the project was to come up with a domain model that accurately captures the problem domain. The series of interviews conducted with analytical chemists and the review of existing tools satisfying similar requirements have been a good starting point. The main difficulty was in the definition of a ubiquitous language that could be easily understood by both the developers and the chemists. Thus, the following terms and meanings associated with them have been agreed upon.

– Analytical Method: the analytical procedure that is subject to validation.
– Analyte: a compound of interest in the analytical method
– Guideline: an official analytical procedure validation guideline [6], [13]
– Validation plan template: a materialized form of a guideline. Guidelines define a range of assessment methods that chemists can choose from to test validation parameters. A validation plan template specifies the very assessment methods and criteria that will be used during an analytical method validation process.
– Experiment plan: a compilation of laboratory experiments to be performed as part of an analytical procedure validation based on a given validation plan template.
– Experiment data: properties of analytes measured during laboratory experiments or computed by the system's computational module (target concentration, residual …)
– Experiment data value: the value of an experiment data
– Dataset: the collection of experiment data and experiment data value for a given analytical procedure validation process.

- Report template: a pattern that the analytical procedure validation report will follow.
- Report: compilation of the evaluations of the various validation parameters along with the dataset used to perform them. The report can also contain remarks or interpretations of the chemist on the evaluations.
- Series: a set of laboratory measurements done at once without interruptions.
- Levels: samples with intentionally different concentration level within a series
- Parallels: samples with intentionally same concentration level within a series
- Replicates: different measurements of one sample within a series

We equally convene on the naming style for experiment data within the system and the format to use for uploading laboratory analysis output into the system. ValChrom would expect the laboratory measurements to be provided as Comma Separated Values (CSV) with experiment data as headers. A row in the CSV would correspond to a measurement of one sample for one analyte. It follows that information regarding the corresponding analyte, series, level, and parallel must be added to the given row. The CSV format has been chosen because it can be used with any text editor or spreadsheet software and does require any technical knowledge. Figure 3-3 shows an excerpt of a laboratory experiment output formatted as expected by ValChrom.

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 3-3 Example CSV File

One of the keys to a successful product development endeavor is the establishment of a business model that clearly identifies the product's users and customers, states how the product will bring value to them and finally pinpoints the way to target them and convey information about the product to them. Often, users are customers. Users are people or systems that interact with the product whilst customers are those paying for the product. In ValChrom's case, users are analytical chemists and customers are the laboratories the analytical chemists work for. ValChrom's marketing strategy is to design a product targeted at analytical chemists, offering them the best possible user experience and rely on them to recommend the product to their laboratories which will get a subscription. This translated into the prioritization of the features to implement during this thesis. The current version of the system could be considered as a Minimum Viable Product MVP. Figure 3-4 shows the domain model of the MVP. The concept of organization or laboratory is not included in this model. It will be part of a future update to the system.

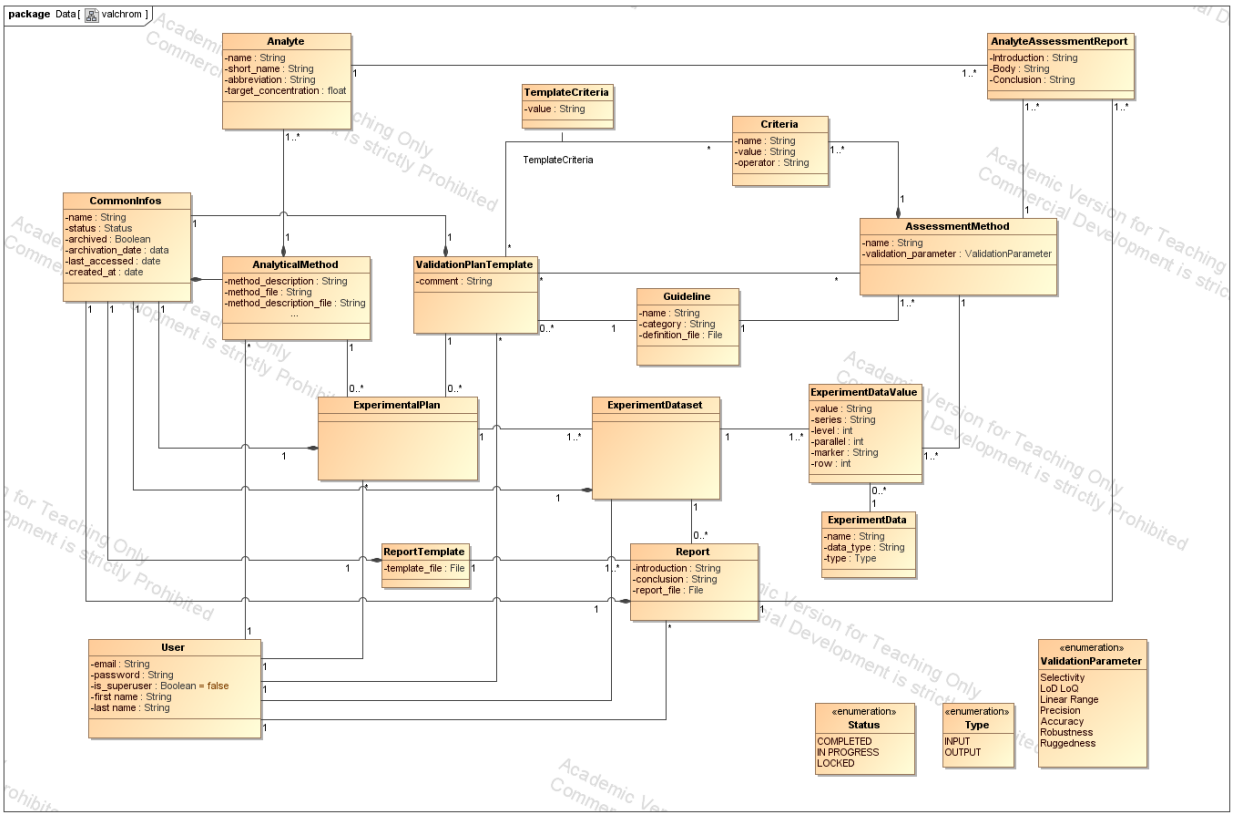

Figure 3-4 Domain Model

3.3. REST API

This section documents the Application Programming Interface (API) designed to expose the system's resources to the outside world. The frontend application communicates with the backend through this API. The section is organized in tables that summarize the list operations supported by each resource.

An analysis of the relationship between the different concepts present in our domain helped in aggregating them and subsequently derive the REST API from those aggregates. As an example, an analyte only exists in the context of an analytical method, therefore, it makes sense to aggregate these two concepts. The analytical method class will be the aggregate root since an analytical method object is required in order to access an analyte object. A similar conclusion can be inferred about guidelines, assessment methods and criteria. In fact, criteria only make sense in the context of an assessment method and an assessment method in turn only exists as part of a guideline. The other roots identified are validation plan templates, experimental plans, experiment datasets, experiment data, reports and users.

The "`/datasets/:id/assessments/:aid/output`" endpoint listed in Section 3.3.6 is responsible for triggering the calculations for the assessment method identified by "`aid`" within the dataset identified by "`id`".

3.3.1. Analytical Methods

Method	URI Template	Relation	Current State	New State	Comments
POST	/methods	create			Creates a new analytical method
GET	/methods	get_all			Retrieves all analytical methods created by the authenticated user
GET	/methods/:id	get			Retrieves a specific analytical method created by the authenticated user
PUT|PATCH	/methods/:id	update			Updates a specific analytical method created by the authenticated user
PUT|PATCH	/methods/:id/archive	archive	active	archived	Archives an analytical method
POST	/methods/:id/duplicate	duplicate			Creates a new analytical method from the current
Analyte					
GET	/methods/:id/analytes	analytes			Retrieves the list of analytes linked to a specific analytical method

3.3.2. Assessment Methods

Method	URI template	Relation	Current state	New state	Comments
POST	/assessments	create			Creates a new assessment method
GET	/assessments	get_all			Retrieves all assessment methods
GET	/assessments/:id	get			Retrieves a specific assessment method
PUT	/assessments/:id	update			Updates a specific assessment method
Criteria					
GET	/assessments/:id/criteria	criteria			Retrieves the list of criteria associated with a specific assessment method

3.3.3. Guideline

Method	URI Template	Relation	Current State	New State	Comments
POST	/guidelines	create			Creates a new guideline
GET	/guidelines	get_all			Retrieves all guidelines
GET	/guidelines/:id	get			Retrieves a specific guideline
PUT|PATCH	/guidelines/:id	update			Updates a specific guideline
GET	/guidelines/:id/assessments	assessments			Retrieves the list of criteria associated with a specific assessment method

3.3.4. Validation plan templates

Method	URI template	Relation	Current state	New state	Comments
POST	/templates	create			Creates a new validation template
GET	/templates	get_all			Retrieves all validation templates created by the authenticated user
GET	/templates/:id	get			Retrieves a specific validation template created by the authenticated user
PUT|PATCH	/templates/:id	update			Updates a specific validation template created by the authenticated user
PUT|PATCH	/templates/:id/archive	archive	active	archived	Archives an validation plan template
GET	/templates/:id/guideline	guideline			Retrieves the guideline that a specific validation

					plan template is based upon
GET	/templates/:id/assessments	assessments			Retrieves the list of assessment methods associated with a specific validation plan template
GET	/templates/:id/criteria	criteria			Retrieves the list of criteria associated with a specific validation plan template

3.3.5. Experiment plan

Method	URI template	Relation	Current state	New state	Comments
POST	/plans	create			Creates a new experiment plan
GET	/plans	get_all			Retrieves all experiment plans created by the authenticated user
GET	/plans/:id	get			Retrieves a specific experiment plan created by the authenticated user
PUT|PATCH	/plans/:id	update			Updates a specific experiment plan created by the authenticated user
PUT|PATCH	/plans/:id/archive	archive	active	archived	Archives an experiment plan
GET	/plans/:id/template	template			Retrieves the validation plan template associated with a specific experiment plan
GET	/plans/:id/method	method			Retrieves the analytical method associated to a specific experiment plan

3.3.6. Experimental datasets

Method	URI Template	Relation	Current State	New State	Comments
POST	/datasets	create			Creates a new dataset

GET	/datasets	get_all			Retrieves all datasets
GET	/datasets/:id	get			Retrieves a specific dataset
PUT|PATCH	/datasets/:id	archive	active	archived	Archives a specific dataset
POST	/datasets/:id/duplicate	duplicate			
GET	/datasets/:id/template	template			Retrieves the validation plan template associated with the dataset
GET	/datasets/:id/method	method			Retrieves the analytical method associated with the dataset
GET	/datasets/:id/analytes	analytes			Retrieves the analytes associated with the dataset
GET	/datasets/:id/result	result			Retrieves the output of the mathematical computation
POST	/datasets/:id/assessments/:aid/input	post			Uploads input data for a specific assessment method within a dataset
GET	/datasets/:id/assessments/:aid/input	get			Retrieves input data for a specific assessment method within a specific dataset
POST	/datasets/:id/assessments/:id/output	post			Starts mathematical computation for

					a specific assessment method within a dataset

3.3.7. Experiment Data

Method	URI template	Relation	Current state	New state	Comments
POST	/data	create			Creates a new experiment data
GET	/data	get_all			Retrieves all experiment data
GET	/data/:id	get			Retrieves an experiment data
PUT|PATCH	/data/:id	update			Updates an experiment data
DELETE	/data/:id	delete			Deletes an experiment data

3.3.8. Experiment Data Value

Method	URI template	Relation	Current state	New state	Comments
POST	/data-values	create			Creates a new data value
PUT|PATCH	/data-values/:id	update			Updates a data value

3.3.9. Report templates

Method	URI template	Relation	Current state	New state	Comments
POST	/report-templates	create			Creates a new report template
GET	/report-templates	get_all			Retrieves all report templates
GET	/report-templates/:id	get			Retrieves a report template
PUT|PATCH	/report-templates/:id	update			Updates a report template

3.3.10. Reports

Method	URI template	Relation	Current state	New state	Comments

POST	/reports	create			Creates a new report
GET	/reports	get_all			Retrieves all reports
GET	/reports/:id	get			Retrieves a report
PUT|PATCH	/reports/:id	update			Updates a report
PUT|PATCH	/reports/:id/archive	archive	active	archive	Archives a report
POST	/reports/:id/duplicate				

3.3.11. Users

Method	URI template	Relation	Current state	New state	Comments
POST	/users	create			Creates a new user
GET	/users	get_all			Retrieves all users
GET	/users/:id	get			Retrieves a user
PUT|PATCH	/users/:id	update			Updates a user
DELETE	/users/:id	delete			Deletes a user

3.3.12. Accounts

Method	URI template	Relation	Current state	New state	Comments
POST	/accounts/signup	signup			Creates a new user
POST	/accounts/login	login			Logs a user in
POST	/accounts/logout	logout			Logs a user out
POST	/account/password/change	change_password			Updates password
POST	/accounts/password/reset	reset_password			Resets password
POST	/accounts/password/reset/confirmation	confirm_reset			Confirms a password reset action

GET	/accounts/user	current_user			Retrieves the authenticated user
PATCH	/accounts/user	update_user			Updates details of the authenticated user

3.3.13. Token

Method	URI template	Relation	Current state	New state	Comments
POST	/token/refresh	refresh			Refreshes the token of the authenticated user
POST	/token/verify	verify			Verifies the authenticity of a given token

3.4. Controllers and Resource Objects

Several classes have been derived from the domain model. Each one of those classes has a controller associated with it. Most of the controllers perform Create Read Update and Delete (CRUD) operations. Some of them handle more advanced business logic. The controllers rely on serializers to translate the python objects into resources.

We can distinguish two types of resources in the system. On one hand, we have system owned resources that cannot be modified by users and user-owned resources on the other. System owned resources include guidelines, assessment methods, criteria, and experiment data. User-owned resources are analytical methods, validation plan templates, experiment plans, datasets, report templates, and reports.

In the context of the problem domain, user-owned resources are referred to as elements. An important requirement of the system demands that users only have access to elements that they created. A custom filter has been implemented to remove all objects not related to the currently authenticated user from the result of a database query before passing it down for serialization and returning it to the user. All controllers based on elements implement this filter.

Elements are generally connected to some others downstream and their lifecycle is dependent on that of the downstream one. As an example, a report is based on a report template, therefore, a report constitutes a downstream element for a report template. Elements can be in four different states. A newly created element is in the “In Progress” state. From that state, it can transition to “Archived” state or “Completed” state. Due to traceability and auditability needs, elements deletion is not allowed in the system. The “Archived” state is assimilable to a deletion in the problem domain. The “Completed” state means that the element can be used during the creation of a downstream element. On creation of a downstream element, the current element moves to the “Locked” state. A “Locked” state conveys the message that the element has other elements that depend on it, therefore they can no longer be edited. The element comes back to the “Completed”

state when all connected downstream elements are archived. Figure 3-5 displays elements' state diagram.

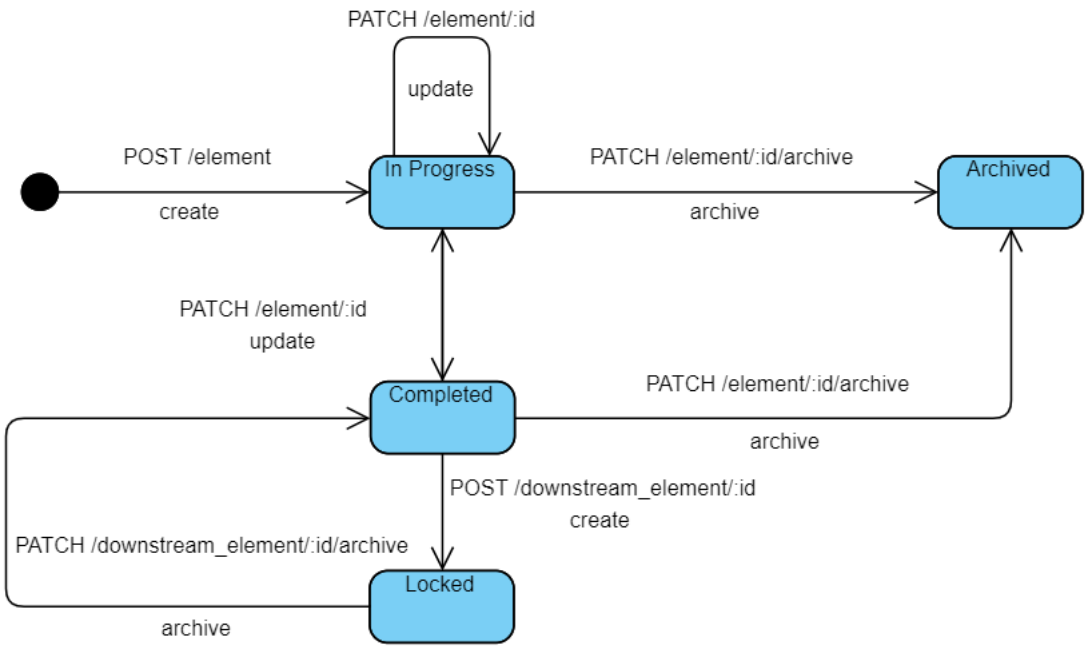

Figure 3-5 Element State Transition

The report element has a specific lifecycle. A report is never "Locked" since there is no other element that depends on it. Figure 3-6 displays a report's state diagram.

Controllers based on elements handle state transition. Each of those controllers implements an *archive* action that takes the element from its current state to the "Archived" state or returns an error when it's not possible. They also implement an *update* action which can be used to transition between "In Progress" and "Completed" and a *duplicate* action that creates a new resource using the current one as a template.

Changes are not allowed on system owned resources. The principal constituents of a guideline are assessment methods which in turn are associated with multiple criteria. Criteria cannot exist on their own, therefore a change to a criterion would imply a change to the related assessment method, which would also imply a change to the guideline. The cascading effect will de facto affect the validation plan template and ultimately the entire analytical method validation process. Only a system administrator can make changes to system owned resources. Controllers associated with system owned resources are set up in a way that any changes on them would result in the creation of a new version of the guideline they are linked to. This ensures a deterministic outcome of the analytical method validation process provided that the output of laboratory experiments has not changed.

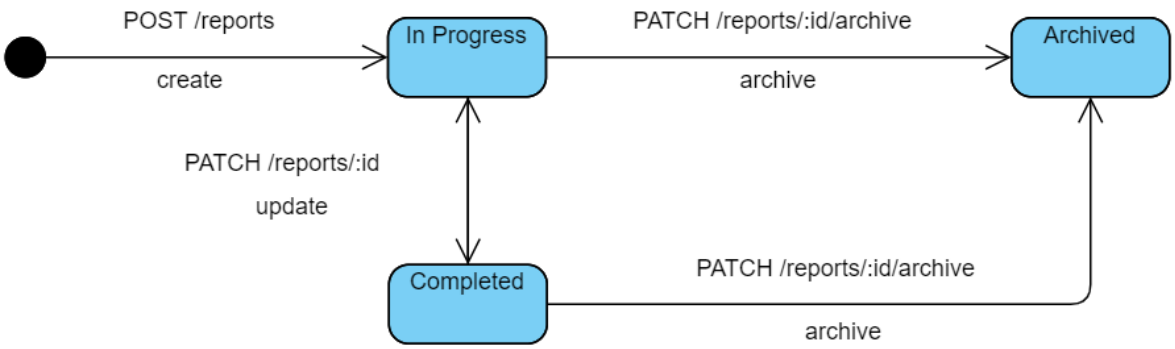

Figure 3-6 Validation Report State Transition

A dedicated controller handles the upload of laboratory experiments output into the system. The data is uploaded by assessment method. The data is expected to be in a CSV format. For each assessment method, a set of constraints is defined for the expected data. The uploaded data must meet each one of the constraints so that the computational function associated with the assessment method executes properly. The constraints are generally about the number of measurements, that is the number of series, levels, and parallels present in the data set. The controller ensures that all the constraints are met before extracting each data point in the CSV file into an experiment data value object. All the data values extracted from the file are saved in bulk in the database and the cache for a successful upload. When some data has been previously uploaded for a given assessment method, a new upload overwrites the previous data. In some limited cases, graphs – chromatograms, in a PNG format or a JPG format would be uploaded. The controller ensures that the assessment method that the data is being uploaded for expects a graph and only graphs with the expected file extension are accepted. Multiple chromatograms can be uploaded simultaneously.

The controller in charge of uploading data into the system provides an additional function to retrieve data linked to a given assessment method. The data is arranged and serialized in a way that it could be easily rendered in a tabular manner in the frontend. It is assumed that most users would use spreadsheet software to compile the laboratory experiment output in the format expected by ValChrom. Rendering the data in a tabular way in the frontend would give them a similar look and feel as in the spreadsheet software.

To finish a result controller has been designed to handle the execution of computational functions. This controller should ideally process requests asynchronously in the background and saves the results of the computations both in the database and in the cache. In the system's intended normal workflow, a request should be sent to this controller after every successful data upload. This ensures that no waiting time is experienced when users want to check the overall result of the analytical method validation. It also makes it easy to have access to intermediate results after each upload.

4. Packaging and Testing

This chapter gives an account of the technologies used and activities performed to continuously verify and ship the system under development. The chapter starts with an introduction to the containerization technology followed by a detail of the steps taken to leverage it to package the application. The chapter continues with a presentation of the continuous integration pipeline that supported the software development process. The chapter ends with a description of the testing activities completed during the project.

4.1. Containerization

In 2008 some engineers from Google submitted a patch to the Linux kernel. The patch named *cgroups*[18] – control groups introduced into the Linux kernel features like the aggregation and isolation of a group of processes, the measurement of their resource utilization and the limitation of their resource usage. These features will later be leveraged to develop containerization.

Containerization is a software packaging system that bundles the software source code together with all its dependencies required for it to run. The dependencies include libraries, executables and configuration files. The resulting bundle referred to as a container can run consistently on any infrastructure. Containers are now used as an alternative to virtual machines. Unlike virtual machines, containers do not embed a copy of the operating system. A container runtime engine ensures that all containers running on a host machine share the same operating system.

Containerization is achieved in a three-step process. It starts with a manifest that describes the state of the container. The manifest is used to generate a snapshot also known as an image of the container. Finally, the container runtime engine produces the actual container from the image.

The adoption of the containerization technology has been accelerated by the advent of the open source container runtime engine Docker. Docker rapidly became the industry's standard for containers thanks to its ubiquitous concept of packaging and its simple developer tools like Docker Compose. Docker Compose or simply Compose is a tool offered by Docker to define and run Docker applications with multiple containers. It helps to manage the complexity inherent in orchestrating a multi-container environment. Using Compose, one can spin up an entire environment with a single command.

In the Docker terminology, the manifest is called *Dockerfile* and the container image a Docker image. A *Dockerfile* has been put together for each one of the services and components in the overall architecture.

Figure 4-1 shows the *Dockerfile* of the core system – the Django application. The system is built for Python 3. The first line of the *Dockerfile* declares a Python 3.7.1 official docker image as the base image that will be used to build our system's image. The Python 3.7.1 official docker image provides a Python 3 environment setup on a Linux Ubuntu operating system, relieving us from installing Python. The *Dockerfile* continues with the creation of an *app* directory to which the requirements file of the system is copied. This directory is the working directory where our system's source code will reside. In a Python project, the requirements file contains all the packages used. The next instruction installs the database client needed by the Linux operating system to connect to an external database. All the dependencies of the system are then installed, and the source code of the project copied to the working. The last instruction in *Dockerfile* ensures that the *wait-for-it.sh* file is executable. *wait-for-it.sh* is a script that halts the execution of a

program until a dependent service is ready to receive connection. The Django application depends on a database service, a cache service, and a queuing service. The *wait-for-it.sh* script will ensure that all those services are available and ready before our system is launched.

```
FROM python:3.7.1
RUN mkdir /app
WORKDIR /app
COPY valchrom/requirements.txt /app
RUN apt-get update  && apt-get install -y postgresql-client
RUN pip install -r requirements.txt
COPY . /app
RUN chmod +x /app/wait-for-it.sh
```

Figure 4-1 Django Application Dockerfile

Figure 4-2 displays the *Dockerfile* of the database. The setup here is straightforward. The *Dockerfile* is based on the official docker image of the latest version of PostgreSQL. The PostgreSQL official docker image provides a PostgreSQL database installed on a Linux Ubuntu operating system. A database initialization script is copied to the appropriate directory and configured to be an executable file.

```
FROM postgres:latest

COPY db_init/db_init.sh /docker-entrypoint-initdb.d/db_init.sh

RUN chmod +x /docker-entrypoint-initdb.d/db_init.sh
```

Figure 4-2 Database Dockerfile

The database initialization starts by creating a new database user and a new database belonging to that user. It then populates the newly created database with the latest database backup available. The database initialization script can be seen in Figure 4-3.

```
#!/bin/sh
psql -U postgres -c "CREATE USER valchrom PASSWORD valchrom"
psql -U postgres -c "CREATE DATABASE valchrom OWNER valchrom"
psql -U postgres --set ON_ERROR_STOP=on $DB_NAME < /backup/$DB_NAME-latest.sql
```

Figure 4-3 Database Initialization Script

Figure 4-4 shows the *Dockerfile* of the cache service. It builds an image based on a Redis official docker image. The *redis:4-alpine* provides a Redis 4 cache server installed on a Linux Alpine operating system. A Redis configuration file *redis.conf* is copied to the relevant directory and the image is instructed to start the *redis-server* using the configuration file provided on startup of a new container. The configuration file allocates a 3 Gigabytes memory space to the cache and

defines an LRU replacement policy for the cache. LRU stands for Least Recently Used. It's a cache replacement policy that discards the least recently used item of the cache when it hits its memory limit. Figure 4-5 presents the cache configuration file

```
FROM redis:4-alpine
COPY redis.conf /usr/local/etc/redis/redis.conf
CMD ["redis-server", "/usr/local/etc/redis/redis.conf"]
```

Figure 4-4 Cache Dockerfile

```
maxmemory 3gb
maxmemory-policy allkeys-lru
```

Figure 4-5 Cache Configuration File

Like the cache service's *Dockerfile*, the reverse proxy's *Dockerfile* builds an image based on an Nginx web server official docker image and supplies the configuration file *nginx.conf* to be used by the server. *nginx:1.15.6-alpine* provide the version 1.15.6 of Nginx web server installed on a Linux Alpine operating system. The configuration file overwrites the default Nginx configuration.

```
FROM nginx:1.15.6-alpine
ADD  nginx.conf /etc/nginx/nginx.conf
```

Figure 4-6 Reverse Proxy Dockerfile

The Nginx server is configured to intercept client requests and forward them to the right system server. First, the intercepted requests are enriched with request forwarding HTTP headers namely the X-Forwarded-For, X-Forwarded-Host, and X-Forwarded-Proto that allow to respectively keep details about the client that submitted the request, the host initially targeted by the request and the protocol used between the client and the reverse proxy. Any request that is received on the reverse proxy's port 80 is systematically redirected to the secured port 443. The reverse proxy uses a regular expression (Regex) based Unified Resource Locator (URL) dispatcher to dispatch requests to the suitable host. It's worth noting that the client-facing application of our system will also be hidden behind the reverse proxy. Figure 4-7 displays the configuration of the reverse proxy

The *docker-compose* file produced in Appendix B orchestrates our system's entire infrastructure. It encompasses eight services. Three more services are present in addition to those mentioned above. The *broker* service sets up a RabbitMQ instance. RabbitMQ is a message-queueing system referred to as a message broker that holds a queue onto which a producer transfers a message that will afterward be taken off by a consumer. The messages are generally tasks to be executed.

```
worker_processes 4;
pid /run/nginx.pid;
 events {
    worker_connections 1024;
    accept_mutex on;
}
http {
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 90;
    types_hash_max_size 2048;
    client_max_body_size 0;
    proxy_connect_timeout 300s;
    proxy_read_timeout 300s;
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;
    gzip on;
    gzip_disable "msie6";
    proxy_set_header   Host $host;
    proxy_set_header   X-Real-IP $remote_addr;
    proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header   X-Forwarded-Host $server_name;
    proxy_set_header   X-Forwarded-Proto $scheme;
    server {
        listen 80;
        server_name valchrom.ut.ee;
        server_tokens off;
        location / {
            return 301 https://$host$request_uri;
        }
    }
    server {
        listen 443 ssl;
        server_name valchrom.ut.ee;
        server_tokens off;
        ssl_certificate /ssl_certs/valchrom_ut_bundle.crt;
        ssl_certificate_key /ssl_certs/www_valchrom_ut_ee.key;
        location /favicon.ico {
            access_log off;
            log_not_found off;
        }
        location /static/ {
            alias /app/valchrom/static/;
        }
        location /media/ {
            alias /app/valchrom/media/;
        }
        location /admin/ {
            proxy_pass         http://backend:4000;
            proxy_redirect     off;
            proxy_buffering    off;
        }
        location /api-auth/ {
            proxy_pass         http://backend:4000;
            proxy_redirect     off;
            proxy_buffering    off;
        }
        location /password-reset/ {
            proxy_pass         http://backend:4000;
            proxy_redirect     off;
            proxy_buffering    off;
        }
        location /api/v1/ {
            proxy_pass         http://backend:4000;
            proxy_redirect     off;
            proxy_buffering    off;
        }
        location / {
            proxy_pass         http://frontend:8080;
            proxy_redirect     off;
        }
    }
}
```

Figure 4-7 Reverse Proxy Configuration

The Django application plays the role of the producer in the case of requests destined to the result controller and the *worker* service that of the consumer. In the case of daily database backups, the *beat* service acts as the producer which periodically kicks off the database backup task that will be eventually picked up and processed by the consumer, the *worker* service.

4.2. Continuous Integration

At the beginning of the project, the team made the choice to apply a Continuous Integration (CI) development approach. In a CI development approach, all features are merged regularly into a mainline and a feature is considered done when the mainline remains healthy after the feature has been merged. The healthiness of the mainline is verified through the execution of a comprehensive list of deterministic tests. These tests are executed in an environment that is as close as possible to the production environment in which the system will ultimately be deployed. Thus, CI generally requires us to package, build and deploy the whole system in a testing environment. The team agreed upon a workflow that prevents any developer to directly push code to the master branch, the master branch acting as our mainline. At any point in time, the master branch contains the version of the application ready to be deployed. As part of the agreed-upon workflow, all new feature branch stems from the master branch. Once the implementation of the feature is completed, the developer creates a pull request targeting the master branch. The developer can then ask for a code review from team members once the CI pipeline executed successfully. The feature is merged into the master branch when at least one positive review has been received.

In this project, the CI setup is powered by Bitbucket Pipelines, Bitbucket's built-in Continuous Integration/Continuous Delivery (CI/CD) service. Bitbucket pipelines are defined in YAML. Figure 4-8 displays our CI setup. The setup starts with the definition of the steps and services that will be needed. Each step has a YAML anchor defined on it allowing us to reference it later in multiple pipelines. The *build-test* step takes care of setting up the python environment where the system will be executed, starts the application and finally executes the API tests designed with Katalon Studio. The *package* step performs the necessary tasks in relation to packaging the application into a Docker image and making it available on the container registry. The system requires a database and a cache service to run properly. The service named *db* sets up a PostgreSQL database and the one named *redis* spawns a Redis cache server. The CI setup also encompasses two distinct pipelines. One that runs whenever a new pull request is created. That pipeline only executes the build-test step to ensure that the feature branch can be safely merged into the master branch. The second pipeline runs whenever the master branch is modified. This pipeline is triggered after the feature branch is merged into the master branch. Both *build-test* and *package* steps are performed. When this pipeline runs successfully a new Docker image containing the latest version of the system is pushed to docker hub. The two pipelines execute in a Docker container created from the Katalon Studio official docker image. The Katalon Studio official docker image embeds a Katalon Studio installation within a Linux Ubuntu environment.

```yaml
definitions:
  steps:
    - step: &build-test
        script:
          - apt -qq update
          - apt -qq install -y software-properties-common
          - add-apt-repository ppa:deadsnakes/ppa
          - apt -qq update
          - apt -qq install -y python3.7
          - apt -qq install -y python3-pip
          - python3.7 --version
          - python3.7 -m pip install -r ./valchrom/requirements.txt
          - python3.7 ./valchrom/manage.py test
          - python3.7 ./valchrom/manage.py migrate
          - python3.7 ./valchrom/manage.py runserver &
          - cp -r ./katalon /katalon/katalon/source
          - katalon-execute.sh -consoleLog -retry=0 -testSuitePath="Test Suites/API Testing" -executionProfile="default" -browserType="Web Service"
        artifacts:
          - report/**
        services:
          - db
          - redis
    - step: &package
        script:
          - docker login -u $DOCKER_HUB_USER -p $DOCKER_HUB_PASSWORD
          - docker build -t $DOCKER_HUB_USER/valchrom_backend:latest .
          - docker push $DOCKER_HUB_USER/valchrom_backend:latest
        caches:
          - docker

  services:
    db:
      image: postgres
      variables:
        POSTGRES_DB: 'valchrom'
        POSTGRES_USER: 'postgres'
        POSTGRES_PASSWORD: 'postgres'
    redis:
      image: redis

image: katalonstudio/katalon
pipelines:
  pull-requests:
    '**':
      - step: *build-test
  branches:
    master:
      - step: *build-test
      - step: *package

options:
  docker: true
```

Figure 4-8 bitbucket-pipelines.yml

4.3. Testing

Testing is the process of assessing a system or its components' conformance to the specified requirements. It is the process to determine whether the right system is being built and it's being built in the right way.

Testing is usually delayed until all development is done. This approach turned out to be high-risk for software development projects. Defects found at that stage are expensive to fix and make projects run behind schedule. Thus, a new approach to testing has been introduced. In many organizations, today testing activities start as early as possible in the development process.

In this project, testing activities were conducted side by side with development activities. Testing efforts focused on ensuring that the REST API that will expose system resources to the outside world works as expected, that is, it returns the data that is expected along with the right status code and it performs actions that it is intended to perform. Katalon Studio[3] was used to perform the API Testing. Built on top of the open-source automation frameworks Selenium and Appium, Katalon Studio is an automation testing solution for API, Web, and Mobile testing. It is mostly used to

[3] https://www.katalon.com/

record interaction with web applications in the browser and replay them to perform automated User Interface (UI) testing.

Katalon Studio not only provides means to send HTTP requests to the system but also to retrieve the responses of the requests, extract data from them using *json-path* and run assertions against the data to ensure that the API works as expected. For instance, to test the endpoint "`/api/v1/experiment/methods/:id`" that returns the details of a specific analytical method, it can be checked that the response status code is indeed 200 – the HTTP response code for a successful *GET* request. The response to this request is expected to contain a list of analytes. The presence of that list can be verified, and assertions can be made on its size as well. Each endpoint can be tested atomically or for CRUD like endpoints, they can be chained. When chained, one can still run verification and assertion on each individual HTTP request. Information from previous requests can also be retrieved and passed down to the following ones in the chain. This is a good way to test how a given resource is affected by a sequence of API calls. The endpoint tests can be grouped into test cases or test suites based on the behaviors under test. Since most of the endpoints require authentication, the capabilities of Katalon Studio API testing have been leveraged to organize the tests in a way that initially creates a new user, then authenticates that user and retrieves and saves its authentication token into a global variable and finally use that variable to enrich subsequent requests. All this can conveniently be achieved from the comfort of the Graphical User Interface (GUI). Katalon studio converts the actions performed on the GUI into Groovy code behind the scenes. For fine-grained control over the tests, one can even directly modify the Groovy[4] code.

Katalon Studio also offers a nice feature that helps generate a command that can be used from a terminal to launch a given test case or test suite. This makes it easy to execute Katalon Studio tests as part of an automated system like a continuous integration pipeline.

Tests performed using Katalon Studio were limited to the functional aspect of the system. As such, CRUD operations on all the resources were tested. Access restrictions on analytical methods were also tested. That is, checking that a new user doesn't have access to existing analytical methods, but only has access to the new one that he/she creates. Status changes related to elements' lifecycle were extensively tested especially in cases where they are linked to a downstream element. A typical scenario was to verify that the status of an analytical method changes from "Completed" to "Locked" when it is used during the creation of a new experimental plan and that the status is reverted to "Completed" when the experimental plan is deleted. Tests were also performed to ensure that data upload errors were correctly reported. Upload of graphs and CSV files were equally tested.

Apart from the API level tests conducted with Katalon Studio, a lot end to end manual testing was performed. The goal of these manual tests was to assert the output generated by the system for individual validation parameters. When the implementation of a given method validation guideline was completed end to end manual tests were also conducted to assert the output of the system when the validation parameters are checked altogether.

No performance nor load testing has been conducted, therefore, the requirement stating that the system should be able to provide an output for computations related to a typical analytical procedure in less than two seconds with up to 200 concurrent users has not been validated.

[4] https://groovy-lang.org/

Conducting load tests and configuring the platform to meet the requirement mentioned above is a direction for future work. The reverse proxy integrated into ValChrom's architecture already provides a starting point to scale up the system in order to meet this requirement, but refinements to the system might still be required. Also, some additional infrastructure management functionality is likely to be required in order to execute ValChrom on a cluster of servers in order to cater for 200 concurrent users.

5. Conclusion and Future Work

At the end of this thesis, we have successfully delivered a working system. The delivered system can take its users through the entire analytical method validation process starting from the validation planning phase to the validation report phase. Even though in its current state, the system offers only one report template, it allows its users to perform method validation against three official guidelines namely ICH, Eurachem and EMA BA and a fourth one which is a combination of the previous three. The system has been presented at the Eurachem conference held in Tartu from 20th – 24th of May 2019 and received positive feedback from the participants. Eurachem is an organization that develops guidance documents for various aspects of analytical chemistry, including analytical method validation. Currently the system is being beta tested by fifteen expert users which were able to use it just after a twenty-minute tutorial.

The version of ValChrom resulting from this thesis represents a huge improvement to the initial manual method validation process. It nevertheless introduces some potentially erroneous activities into the process namely the collection and formatting of experiments results from lab equipment. Our investigation at the beginning of the project revealed that each equipment provider has its own output data format. A possible improvement to the current version of ValChrom would be the implementation of a module that will be able to directly ingest output generated by the most popular lab equipment. The ability to automatically convert the output of lab equipment into the format expected by ValChrom will make the validation process even more accurate.

Report templates currently are written as Django templates and require a comprehensive understanding of the API. Django templates are essentially HTML code augmented with Python code. The typical analytical chemist will experience difficulties in creating a customized report template. A less technical way to create report templates could be investigated.

Another potential point of improvement is the automation of the deployment of the system. In the current setting, when a new feature is implemented on the system, the CI pipeline simply bundles the new application into a Docker image and pushes it to the container registry. At that point, human intervention is required to stop all the services on the Virtual Private Server (VPS) and restart them by applying the *docker-compose* file. A possible solution would be to connect to the VPS through a Secure Shell (SSH) tunnel from within the CI pipeline and apply the *docker-compose* file.

Furthermore, it will be necessary to conduct load testing and to refine the system in order to meet the requirement of being able to provide a response in less than two seconds even with 200 concurrent users.

References

[1] L. R. Snyder, J. J. Kirkland, and J. L. Glajch, *Practical HPLC Method Development*. John Wiley & Sons, 2012.

[2] "HPLC Column Dimensions." [Online]. Available: https://www.chromacademy.com/chromatography-HPLC-Column-Dimensions.html. [Accessed: 16-Dec-2018].

[3] "How Does High Performance Liquid Chromatography Work? : Waters." [Online]. Available: http://www.waters.com/waters/en_MT/How-Does-High-Performance-Liquid-Chromatography-Work%3F/nav.htm?cid=10049055&locale=en_MT. [Accessed: 16-Dec-2018].

[4] R. Kadis, "Analytical procedure in terms of measurement (quality) assurance," in *Measurement Uncertainty in Chemical Analysis*, P. De Bièvre and H. Günzler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 1–7.

[5] C. C. Chan, Y. C. Lee, H. Lam, and X.-M. Zhang, *Analytical Method Validation and Instrument Performance Verification*. John Wiley & Sons, 2004.

[6] J. Abraham, "International Conference On Harmonisation Of Technical Requirements For Registration Of Pharmaceuticals For Human Use," in *Handbook of Transnational Economic Governance Regimes*, A. Brouder and C. Tietje, Eds. Brill, 2009, pp. 1041–1054.

[7] D. FDA/CDER/"Beers, "Analytical Procedures and Methods Validation for Drugs and Biologics," p. 18, 2015.

[8] L. R. Snyder and J. J. Kirkland, *Introduction to modern liquid chromatography*, 2d ed. New York: Wiley, 1979.

[9] Waters, "Streamline the Chromatographic Method Validation Process Using Empower 2 Method Validation Manager." 2007.

[10] "Fusion QbD Analytical Method Validation - Powered by PageTurnPro.com." [Online]. Available: http://www.pageturnpro.com/S-Matrix-Corp/58884-Fusion-QbD-Analytical-Method-Validation/default.html#page/1. [Accessed: 21-Dec-2018].

[11] "Technology Overview | ValGenesis." .

[12] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[13] B. Magnusson and U. Örnemark, *Eurachem Guide: The Fitness for Purpose of Analytical Methods – A Laboratory Guide to Method Validation and Related Topics*, 2nd ed. 2014.

[14] European Medicines Agency, "Guideline on bioanalytical method validation.", EMEA/CHMP/EWP/192217/2009, 21-Jul-2011.

[15] T. W. Malone, "Interoperability in Programming Languages," 2014.

[16] J. King and R. Magoulas, "2015 Data Science Salary Survey," p. 49.

[17] "ISO/IEC/IEEE 42010:2011(en), Systems and software engineering — Architecture description." [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:42010:ed-1:v1:en. [Accessed: 27-Jun-2019].

[18] "Index of /doc/Documentation/cgroup-v1/." [Online]. Available: https://www.kernel.org/doc/Documentation/cgroup-v1/. [Accessed: 05-Jul-2019].

Appendix A

System Requirements for Server

ValChrom can be deployed on any operating system with a working installation of the following:

- Docker
- Docker-compose

Installation / Deployment instructions

From a work station

1. Checkout *valchrom_final* repository

   `git clone` git@bitbucket.org:valchrom/valchrom_final.git

2. Start the application

   ```
   cd valchrom_final
   docker-compose up -d
   ```

3. Stop the application

   ```
   docker-compose down
   ```

The live version is available at https://valchrom.ut.ee/

The source code available at https://bitbucket.org/valchrom/

Appendix B

```
version: '3.2'
services:
  redis:
    build:
      context: ./cache/
      dockerfile: Dockerfile
    expose:
      - ${CACHE_PORT}
  broker:
    image: rabbitmq:3.7-alpine
    expose:
      - ${BROKER_PORT}
  database:
    restart: on-failure
    build:
      context: ./database/
      dockerfile: Dockerfile
    volumes:
      - data:/var/lib/postgresql/data
      - ./database/backup:/backup
    environment:
      - DB_NAME=${DB_NAME}
      - DB_USER=${DB_USER}
      - DB_PASS=${DB_PASS}
    expose:
      - ${DB_PORT}
  frontend:
    image: ${DOCKER_HUB_USER}/valchrom_frontend:latest
    expose:
      - ${FRONTEND_PORT}
  backend:
    image: &backend ${DOCKER_HUB_USER}/valchrom_backend:latest
    volumes:
     - backend-static:/app/valchrom/static
     - backend-media:/app/valchrom/media
    environment:
      - DB_NAME=${DB_NAME}
      - DB_USER=${DB_USER}
      - DB_PASS=${DB_PASS}
      - DB_SERVICE=${DB_SERVICE}
      - DB_PORT=${DB_PORT}
      - CACHE_PORT=${CACHE_PORT}
      - CACHE_SERVICE=${CACHE_SERVICE}
```

```
    command: ./wait-for-it.sh ${DB_SERVICE}:${DB_PORT} -s -t 0 -- bash -c "cd valchrom &&
python manage.py migrate && python manage.py collectstatic --noinput --clear && gunicorn
valchrom.wsgi:application -k gevent --worker-connections 1024 -t 90 -b :${BACKEND_PORT}"
    expose:
      - ${BACKEND_PORT}
    depends_on:
      - ${DB_SERVICE}
      - ${CACHE_SERVICE}
      - ${BROKER_SERVICE}
      - ${CELERY_BEAT_SERVICE}
      - ${CELERY_WORKER_SERVICE}
  worker:
    image: *backend
    volumes:
      - backend-media:/app/valchrom/media
    command: ./wait-for-it.sh ${BROKER_SERVICE}:${BROKER_PORT} -s -t 0 -- bash -c "cd valchrom
&& celery -A valchrom worker -l info"
    ports: []
    environment:
      - BROKER_PORT=${BROKER_PORT}
      - BROKER_SERVICE=${BROKER_SERVICE}
      - CACHE_PORT=${CACHE_PORT}
      - CACHE_SERVICE=${CACHE_SERVICE}
    depends_on:
      - ${BROKER_SERVICE}
      - ${DB_SERVICE}
  beat:
    image: *backend
    command: ./wait-for-it.sh ${BROKER_SERVICE}:${BROKER_PORT} -s -t 0 -- bash -c "cd valChrom
&& celery -A valChrom beat -l info"
    ports: []
    environment:
      - BROKER_PORT=${BROKER_PORT}
      - BROKER_SERVICE=${BROKER_SERVICE}
      - CACHE_PORT=${CACHE_PORT}
      - CACHE_SERVICE=${CACHE_SERVICE}
    depends_on:
      - ${BROKER_SERVICE}
      - ${DB_SERVICE}
  reverse_proxy:
    restart: unless-stopped
    build:
      context: ./reverse_proxy/
      dockerfile: Dockerfile
    volumes:
```

```
      - backend-static:/app/valchrom/static
      - backend-media:/app/valchrom/media
      - ./reverse_proxy/certbot/conf:/etc/letsencrypt
      - ./reverse_proxy/certbot/www:/var/www/certbot
    ports:
      - "80:80"
      - "443:443"
    command: "/bin/sh -c 'while :; do sleep 6h & wait $${!}; nginx -s reload; done & nginx -g
\"daemon off;\"'"
    depends_on:
      - ${FRONTEND_SERVICE}
      - ${BACKEND_SERVICE}
    stdin_open: true
volumes:
  data:
  backend-static:
  backend-media:
```

License

Non-exclusive licence to reproduce thesis and make thesis public

I,

Kodjovi Hippolyte-Fayol Toulassi

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,

Software Tool for Validation of Chromatographic Analytical Procedures,

supervised by Prof. Marlon Dumas, 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.

Kodjovi Hippolyte-Fayol Toulassi

14/08/2019