university of groningen

PASTAQ

Requirements Document

Software Engineering Project Feb 2022

Students: Mohammed Nacer Lazrak, Dominic Therattil, Tudor Dragan, Kaitlin Vos, Björn Schönrock, Teresa Ferreira & Cristian Iacob

First Supervisor: Lars Andringa

Second Supervisor: Dr A. Capiluppi

Contents

1 Stakeholders

1.1 Core Team

- Members: Tudor Dragan, Björn Schönrock, Mohammed Nacer Lazrak, Dominic Therattil, Kaitlin Vos, Teresa Ferreira, Cristian Iacob
- Supervisors: Lars Andringa, Dr A. Capiluppi

1.2 User

- Researchers
- Data scientists
- Chemists
- Biologists

1.3 External Consultant

- Alejandro Sánchez Brotons
- Prof. Dr. Peter L. Horvatovich

2 Introduction

The project presented in this document is a Graphical User Interface (GUI) which aims to ease the use of the PASTAQ (Pipelines and Systems for Threshold-Avoiding Quantification) pipeline for users that do not wish to interact with it from a terminal and need an out-of-the-box application to process their data. To be able to understand the context in which our GUI exists we need to take a closer look at liquid chromatography coupled with tandem mass spectrometry (LC-MS/MS).

LC-MS/MS is an advanced analytical chemistry technique that aims to identify and quantify proteins and their peptides in a sample. The instruments used in this technique output large amounts of data that need to first be converted through a piece of software called 'msConvert' into two formats: .mgf and .mzXML. The former is then run through another suite of software meant to identify the peptides in the sample and then converted into an .mzID file. This together with the .mzXML files are fed into PASTAQ.

PASTAQ's intended use is in metabolomics and proteomics as it employs certain specialized algorithms. The pipeline outputs data as .csv files that identify proteins and features of the sample. These results are then meant to be interpreted through Python or R. It also outputs quality control plots as .png files. The application is available as a library that can be imported and used in C++ or Python. For users that are not familiar with programming, the GUI offers a simple alternative that also includes quality-of-life functionalities.

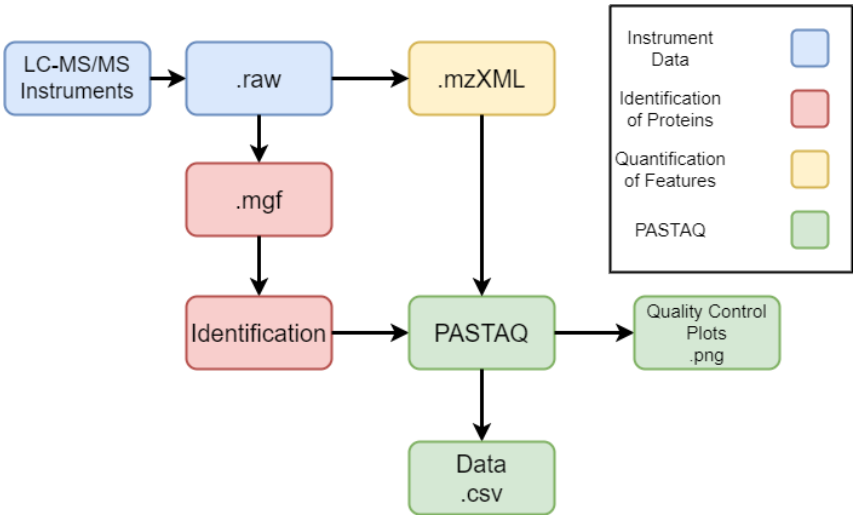

Figure 1: Overview of the ecosystem where PASTAQ is used

3 Scope

The system as a whole will cover the functionalities needed to transform .raw data to .csv files and .png files for quality control.

First, through the usage of external software, we convert .raw data to .mgf and .mzXML files. Said .mgf and .mzXML data will be used by selecting a certain execution path which, through certain search engines and external software, will produce an .mzID file.

Afterwards, our PASTAQ GUI will take the .mzID and .mzXML files and convert them to .csv files and quality control .png files.

The scope of our project is to make the functionality of the system better for the average user by giving the user said functionalities without the use of a terminal, but rather through our PASTAQ-GUI, the user should be

able to just input an .mgf file and select an execution path which utilises certain software and search engines in order to reach the end result.

Respectively our goal is also to refactor the code such that the GUI is stable and current crashes are to be removed, refactor the code such that there are no code smells to be observed, add tool-tips to explain its functionalities to the user, and maintain the software on Windows, Linux and MacOS.

4 Requirements list

We have decided to separate the requirements list based on whether said requirements are functional or non-functional, said list is ordered based on the priority of its items with optional and critical items specified.

4.1 Functional Requirements

1. The GUI must accept different types of input files, such as .mgf and .mzID. When the input file is an .mgf, it must automatically convert it to the proper format which is .mzID. When the input file is an .mzID, it will proceed normally since a conversion is not necessary.
2. The GUI must have tool-tips for the parameters to explain its functionalities to the user.
3. The GUI must be stable and current crashes must be resolved through the use of thread handling between Python and C++.
4. The software must be supported on the following platforms: Windows, Linux and MacOS.
5. The SearchGUI must be integrated into the GUI as much as possible, so that by selecting a path in the GUI will make the SearchGUI process the files accordingly.

4.2 Non-Functional Requirements

1. The GUI must be simplistic and user-friendly.
2. The GUI must be redesigned to be more aesthetically pleasing.
3. The code quality must be improved by fixing bugs and refactoring code smells.

5 Use Cases

All requirements for this single system can be related to the same Graphic User Interface (GUI). This led us to choose use cases over user stories because we think they give a better structure to our requirements. This section is divided into six different user cases, which we will consider individually, state the use case specification and present the use case diagram.

5.1 (.mgf) File Acceptance

The GUI must be able to accept an .mgf file and automatically convert it to an .mzID file.

Context: The user wishes to avoid the tedious process of manually converting the .mgf files using a terminal and instead upload an .mgf file which the GUI will automatically process into an .mzID file.

Main Success Scenario:

- The user is able to upload the file
- The .mgf files get converted to .mzID
- The newly converted .mzID file gets inputted into the GUI

Alternative Scenarios:

- The .mgf file fails to get converted to .mzID
- An error message is displayed
- Prompted to try again or just manually convert the file to an .mzID file

Pre Conditions: .mgf file is uploaded
Post Conditions: File is converted to an .mzID and inputted into the GUI

5.2 Search Parameters

The GUI should accept different search parameters that can be applied to the uploaded file.

Context: The user uploads their file to PASTAQ GUI and selects their specific search parameters from MS-Fragger or SearchGUI that can be applied to the uploaded file.

Main Success Scenario:

- The user is able to upload the file
- Selects appropriate search parameters
- The file gets processed as JSON

Alternative Scenarios:

- The user cannot upload the file
- An appropriate error message is displayed
- The set search parameters are invalid
- Error is shown, informs users which search parameters are invalid and prompts to try again.

Pre Conditions: The search parameters are successfully set
Post Conditions: The GUI processes the file and outputs a JSON

5.3 Tool-Tips

The GUI should have tool-tips to explain its functionalities to the user.

Context: The user wants to access the PASTAQ GUI, however, the user does not know how to fully work with the GUI options.

Main Success Scenario:

- The user is able to follow the tool tips

Alternative Scenarios:

- The tool-tips are not simple enough, the user gets confused
- The tool-tips are too simple, not enough information to achieve the users task

Pre Conditions:

- *Tool-tips written and explained to informed users*
- *User has enough knowledge to work with the GUI*

Post Conditions: The tool-tips help the user

5.4 Stable GUI

The GUI should be stable and current crashes should be resolved.

Context: The user wants to terminate the program by shutting down the GUI.

Main Success Scenario:

- The program is terminated with the shutting down of GUI

Alternative Scenarios:

- The program crashes upon shutting down the GUI
- The user is provided with the appropriate crash details

Pre Conditions: Program is running
Post Conditions: Program terminates

5.5 Code Smells

The program code is documented and refactored such that it has no code smells.

Context: Since the PASTAQ GUI is an open source project, there can be users with an understanding of programming who wish to update the code for their own specific needs.

Main Success Scenario:

- The user is able to easily understand the structure of the program and implement their features into the GUI

Alternative Scenarios:

- The user is unable to understand the code and can raise an issue on the repository

Pre Conditions: The user is able to understand the code and add functionality for their needs
Post Conditions: The user is able to successfully run the GUI with their new functionality

5.6 Operating System

The software is able to run on Windows, Linux and MacOS.

Context: The user wants to use the PASTAQ GUI in any Windows, Linux or MacOS- based operating system.

Main Success Scenario:

- The user is able to run the GUI on Windows, Linux and MacOS

Alternative Scenarios:

- The user is informed with an error message as to why the program is not running

Pre Conditions: Program runs on the users operating system
Post Conditions: Program Terminates successfully

6 Languages and Technologies

- R, for statistics on the outputted .csv file
- Python, the language in which our PASTAQ-GUI is written in

Before we select a path

- msConvert, converts .raw data to .mgf and .mzXML

Path 1

- SearchGUI, selects .mgf files and the search engines before outputting identification files as a .zip
- PeptideShaker, outputs .mzID files from the identification files

Path 2

- Philosopher, to perform command line analyses with MSFragger
- MSFragger, generates pep.xml files from .mzXML files
- idconvert, converts pep.xml files to .mzID so they can be parsed

After we selected a path

- PASTAQ, takes the .mzID and .mzXML files and outputs .csv files and .png files for quality control

7 Terminology

Glossary

.mgf Mascot generic format files are the standard format to store protein data. 1, 3–5, 7

liquid chromatography Separate a certain sample into its individual components. 3

mass spectrometry Separate ions by their mass-to-charge ratio and to detect them qualitatively and quantitatively by their respective mass/charge and abundance. 3

metabolomics The analysis of metabolites in a biological specimen. 3

MSFragger Database search tool for peptide identification in mass spectrometry. 5, 7

peptide Peptides are short chains of amino acid. 3

PeptideShaker A search engine independent platform for protein identification. 7

proteomics The study of proteoms (a set of proteins produced in an organism). 3

SearchGUI Open-source common interface for configuring and running protein data search. 4, 5, 7

8 Client Meeting Log

Meeting Date	Content
2022-02-25	• General introduction • Discussed available projects • Short overview of client system
2022-03-02	• Discussed project choice • Client provided overview of the whole system • Discussion about specific requirements (fix bugs, small improvements etc.) • Brainstorm possible requirements (integration of visuals and statistics)

Table 1: Meeting log

9 Change log

Date	Time	Team Member	Changes	Version
12.2.2022	13:58	Kaitlin Vos	Inserted change log table	1.0
15.2.2022	12:16	Dominic Therattil	Cover page draft	1.1
16.2.2022	16:45	Tudor Dragan	Client Meeting Log	1.1.1
19.2.2022	13:39	Dominic Therattil	Updated Cover Page	1.1.2
01.03.2022	13:39	Dominic Therattil	Updated banner for pastaq	1.1.2
02.03.2022	12:55	Björn Schönrock	Updated meeting log	1.1.3
04.03.2022	12:30	Kaitlin Vos	Meeting log changed to table format and second meeting notes entered	1.1.4
04.03.2022	13:00	Mohammed Nacer Lazrak	Writing Requirement first draft	1.1.5
05.03.2022	08:00	Cristian Iacob	Added stakeholders, system's scope and frameworks	1.1.6
05.03.2022	13:00	Kaitlin Vos	More detailed requirements	1.1.6
05.03.2022	22:00	Tudor Dragan	Introduction	1.1.8
06.03.2022	19:00	Dominic Therattil	Added use cases	1.1.9
06.03.2022	19:00	Maria Teresa	Added use cases	1.1.10
06.03.2022	19:00	Dominic Therattil	Added terminology	1.1.11
06.03.2022	23:41	Kaitlin Vos	Fixed grammar and fine-tuned introduction	1.1.12
07.03.2022	12:00	Cristian Iacob	Moved system scope to scope and added to it	1.1.13
07.03.2022	12:21	Kaitlin Vos	Terminology glossary	1.1.14

Table 2: Change log