VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Software Engineering | Project Technical specification

Solar panel project

Area 1

Done by:

Arnas Vidžiūnas
Mindaugas Neseckas
Aleksandras Kuznecovas
Valentinas Straigis
Tomas Jefimovas

Supervisors:

Lekt. Virgilijus Krinickij
Lekt. Gediminas Rimša

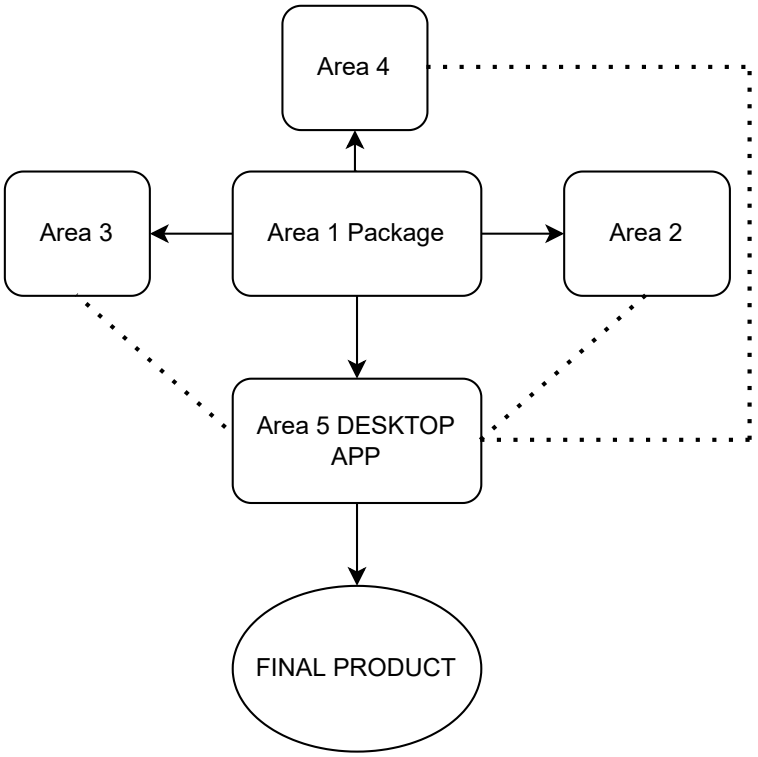
Vilnius
2022

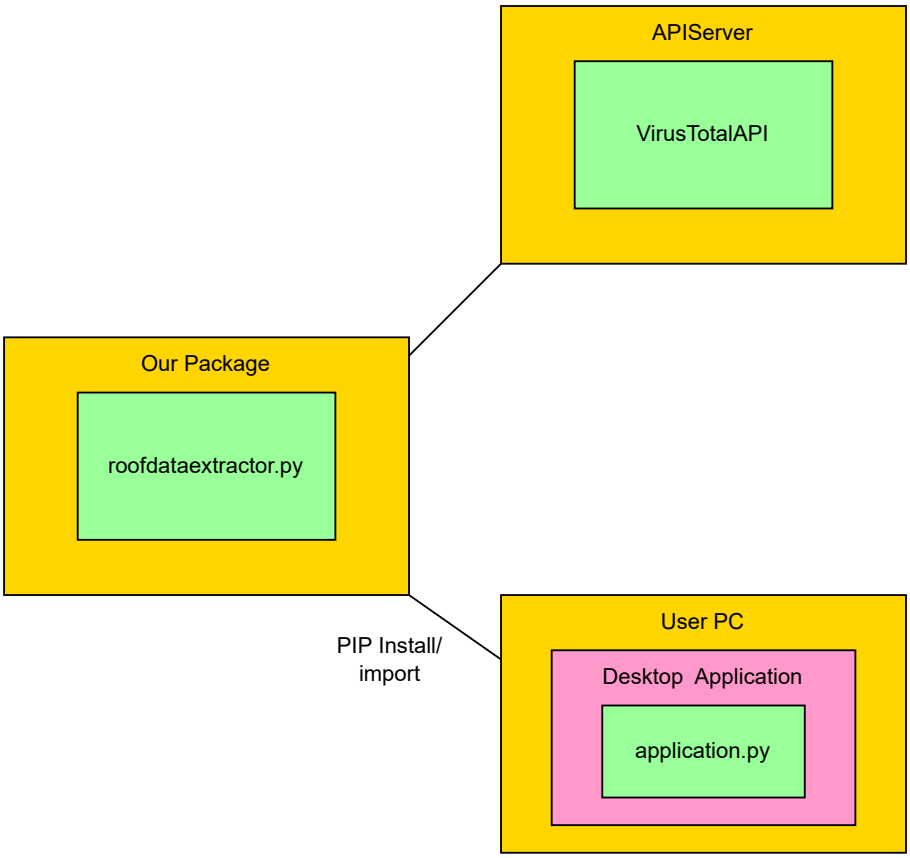
Contents

Overview

This is a technical specification for a package which will read any text file which has an XML format inside. The file will be provided by the user in an XML formatted file that would be processed and output a JSON file with filtered data. The provided file will contain roof's or collection of multiple roof panels measurements, coordinates, and other details about the roof. The package will include file and data validation in order to protect the final program from malicious scripts and other dangerous content. Our area's goal is to extract required data which is specified by other teams.

Our area's package highly contributes to other areas' work. We extract and filter certain data from the given XML formatted file and output it to the JSON file in order to make other areas' work less complicated. Our package is used in every other area of the project.





Technologies and Tools

Programming language - Python (Recommended version: 3.10, Minimum version: 3.6)
Unit testing - PyTest
IDE - VS Code
Version Control tool - Git and GitHub
Project management and issue tracking - Jira
Documentation writing tool - Overleaf LaTeX

PIP (package installer for Python) libraries used in the project:

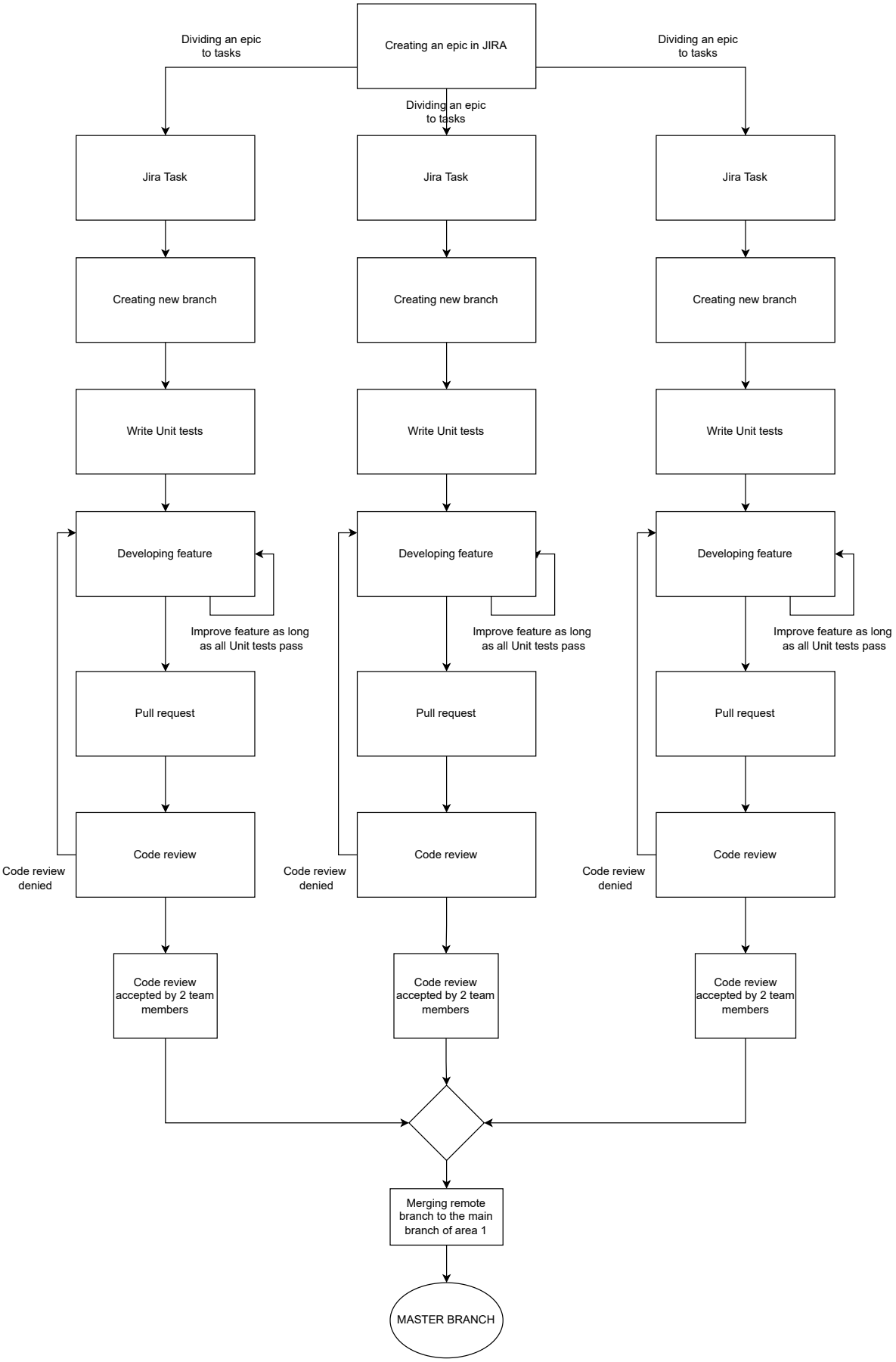
- Bpypolyskel
- XMLJSON
- DefusedXML
- PyTest

Workflow

- Creating an epic in Jira. The epic is divided into smaller tasks, which are separated between each team member.

- Creating a new branch for that Jira task.
- If needed, writing the unit test, which firstly fails. We agreed to use test driven deployment.
- Develop an improvement for the package (develop a feature). Every change has to pass the unit tests.
- Opening a pull request for the development branch (main of area 1) if everything is done and the code passes all the tests.
- Code review of the pull request. The code review has to be done by at least 2 people.
- If the feature passes two code reviews, merging this branch to the main branch of area 1.

comment: the graph provided below is not full, since we wanted to have it nicely fit in the page. There should be 2 more identical columns on the sides.


Creating an epic in JIRA
Dividing an epic to tasks
Dividing an epic to tasks
Dividing an epic to tasks
Jira Task
Jira Task
Jira Task
Creating new branch
Creating new branch
Creating new branch
Write Unit tests
Write Unit tests
Write Unit tests
Developing feature
Developing feature
Developing feature
Improve feature as long as all Unit tests pass
Improve feature as long as all Unit tests pass
Improve feature as long as all Unit tests pass
Pull request
Pull request
Pull request
Code review
Code review
Code review
Code review denied
Code review denied
Code review denied
Code review accepted by 2 team members
Code review accepted by 2 team members
Code review accepted by 2 team members
Merging remote branch to the main branch of area 1
MASTER BRANCH

Testing

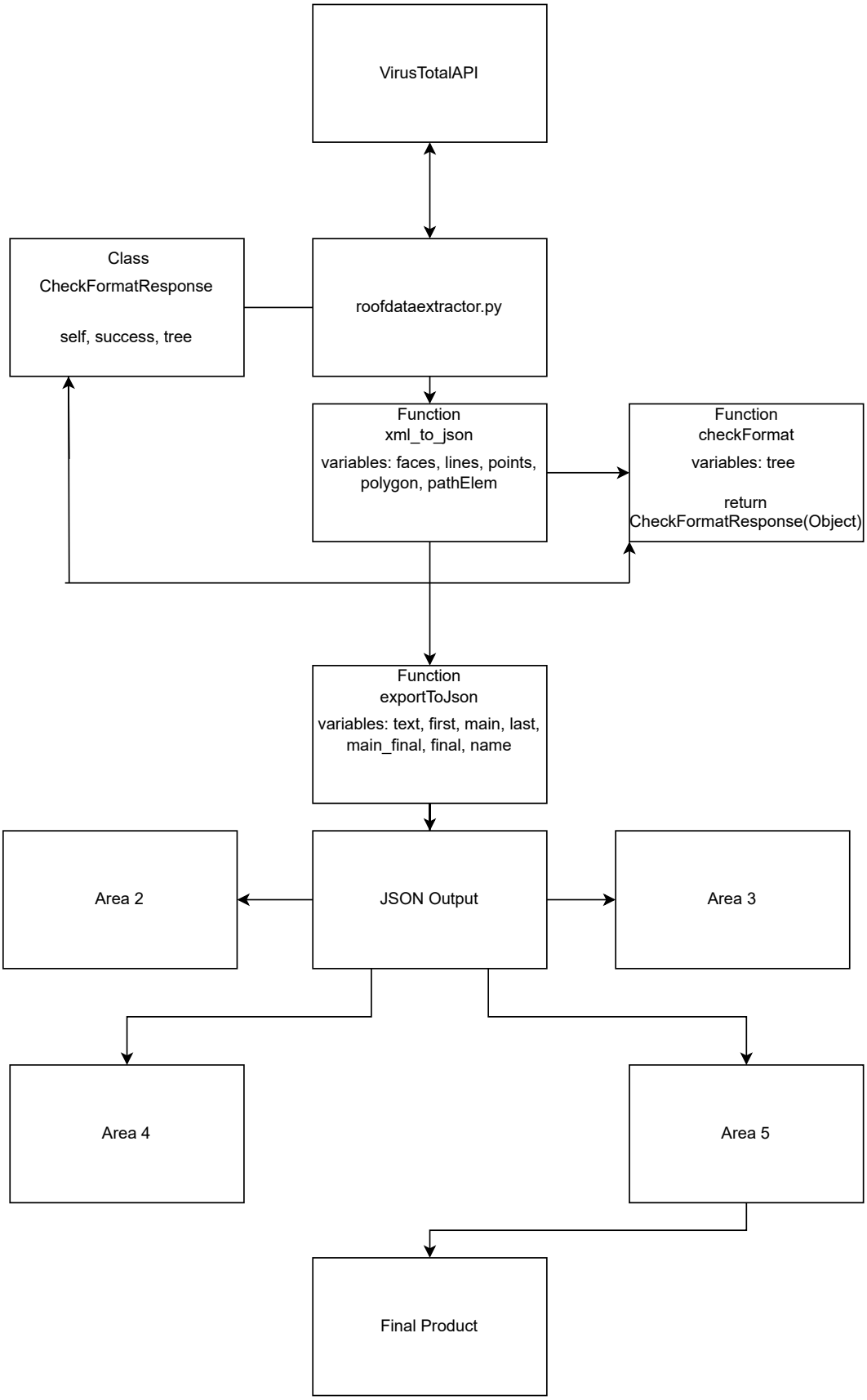
Test driven development will be used to check if our package is going to work. We are using PyTest for unit tests. The main usage of our unit tests are on reading XML formatted file, extracting certain data, printing on JSON file and comparing output file with the input file.

The main help we get from using unit tests is that we have them done before we write actual code for the feature. This helps us to reduce clutter in our final code.

Written Unit Tests by code version:

- **Version 0.1** - Tests are throwing a lot of different file extensions to see if the package reads only XML files.
- **Version 0.1.1** - Tests are throwing a lot of different file extensions with XML format inside to see if the package reads only XML formatted files.
- **Version 0.2** - Tests checks if the provided XML file has roof measurements inside.
- **Version 0.3** - Tests help us to compare XML file content and JSON file output.

Structural Aspects



Dynamic Aspects

- The package receives one file as an input.
- Firstly, the package calls VirusTotalAPI to check if the file is not malicious.
- If the given file is safe, the package calls XmlToJson function, which instantly calls checkFormat function.
- If the given file is not formatted as an XML file, the package prints: “The file is not well formatted ”.
- If the given file is formatted as an XML file, the package continues reading that file.
- If the given file is formatted as an XML file and after the reading it founds, that the XML file is written with no roof measurements it prompts an error to change the input file.
- If the file is well formatted, with no malicious scripts, the package continues its reading and calling exportToJson function().
- The package outputs a JSON file as a result.

Error Handling

- In case of incorrect data, the package will prompt an error. The user will be asked to fix it, rewrite or simply change the file.
- The package will cycle through the content of the provided file to check if it is written in an XML format. For example, the provided file could have any extension like *.txt, *.pptx, *.docx, but the content is written in an XML format, so the provided file will be accepted.
- The package will terminate itself and prompt errors if the malicious scripts are detected in the provided file.
- If the provided file is empty, the package will prompt an error and ask the user to fix the file.
- If the package can not handle an unexpected operation it will prompt an unexpected error.
- ...