National Institute of Applied Sciences and Technology

CARTHAGE UNIVERSITY

Graduation Project

Specialty : **Software Engineering**

A flexible data analysis library for hybrid pixel detectors

Presented by

Bechir Braham

INSAT Supervisor : **Dr. Guesmi Ghada**
Company Supervisor : **Dr. Froejdh Erik**

Presented on : **30/09/2024**

JURY

M. President FLEN (President)
Ms. Reviewer FLENA (Reviewer)

Academic Year : 2023/2024

Acknowledgements

Thank you all!

Table of Contents

List of Figures

List of Tables

Abstract

This is the english abstract of your project. It must be longer and presented in more details than the abstract you write on the back of your report.

General Introduction

La Table de matière est la première chose qu'un rapporteur va lire. Il faut qu'elle soit :

- Assez détaillée [1]. En général, 3 niveaux de numéros suffisent;
- Votre rapport doit être réparti en chapitres équilibrés, à part l'introduction et la conclusion, naturellement plus courts que les autres;
- Vos titres doivent être suffisamment personnalisés pour donner une idée sur votre travail. Éviter le : Conception , mais privilégier : Conception de l'application de gestion des ... Même s'ils vous paraissent longs, c'est mieux que d'avoir un sommaire impersonnel.

Une introduction doit être rédigée sous forme de paragraphes bien ficelés. Elle est normalement constituée de 4 grandes parties :

1. Le contexte de votre application : le domaine en général, par exemple le domaine du web, de BI, des logiciels de gestion ?
2. La problématique : quels sont les besoins qui, dans ce contexte là, nécessitent la réalisation de votre projet?
3. La contribution : expliquer assez brièvement en quoi consiste votre application, sans entrer dans les détails de réalisation. Ne pas oublier qu'une introduction est censée introduire le travail, pas le résumer;
4. La composition du rapport : les différents chapitres et leur composition. Il n'est pas nécessaire de numéroter ces parties, mais les mettre plutôt sous forme de paragraphes successifs bien liés.

[1] Sans l'être trop

Part I

Chapter 1

Chapter I

Project Context and Scope

Summary

Introduction

In this first chapter we will present the Paul Scherrer Institute, the host company of the project. We will also introduce

1 Presentation of The Host Company

The Paul Scherrer Institute (PSI) is the largest research institute for natural and engineering sciences within Switzerland. Created in 1998, the institute is located in Canton Aargau and employs around 2300 people. PSI is composed of 8 main research centers:

- Center for Life Sciences
- Center for Neutron and Muon Sciences
- Center for Nuclear Engineering and Sciences
- Center for Energy and Environmental Sciences
- Center for Photon Science
- Center for Scientific Computing
- Theory and Data
- Center for Accelerator Science and Engineering.

In addition the Paul Scherrer Institute is famous for its synchotron: the Swiss Light Source (SLS). The SLS is a third-generation synchrotron light source, which provides high-brilliance photon beams with high spectral resolution and tunable energy. It is used for a wide range of experiments in materials science, biology and chemistry. [Bög02; Web; Lv+22]

2 Detector's Group Presentation and Work

The Detector's Group is one of the research groups at the Paul Scherrer Institute. It is part of the Laboratory for X-ray Nanoscience and Technologies (LXN) which itself is part of the Center for Photon Science.

The group is responsible for the development of new detectors, the maintenance of the existing ones, and the development of software for the data acquisition and analysis. It is responsible for the development of new technologies for the detectors, such as new sensors, readout electronics, and data acquisition systems. The group is also involved in the development of new techniques for the data analysis, such as image processing, pattern recognition, and machine learning. These Detector's are an integral part of the beamline's setup, and are used to detect the x-ray photons that are emitted by the synchrotron. Many experiments in different fields can use these detectors such as for studying the properties of materials [But+24], protein structures [Pom+09], crystallography [Leo+23], biology [Lem+23; Dul+24], and many other fields of research.

On the software side, the group is responsible for the development of the software that is used to control the detectors, acquire the data, and analyze it. The software is developed in C++ and is used by the scientists to perform their experiments. The main package maintained by

the detector's group is the "SLS Detector Package" available publicly on https://github.com/slsdetectorgroup/slsDetectorPackage. The package provides several binaries such as:

- **slsReceiver** The receiver server acquires incoming data from detectors using UDP and listens for configurations from host machine using TCP.
- **slsDetectorGet** Used by the host machine to request the configuration on the Receiver or the Detector.
- **slsDetectorPut** Used to configure parameters on both the Receiver and Detector.
- **slsDetectorGui** A graphical user interface (GUI) that receives data from the Receiver and displays it.

The list of binaries is not exhaustive, and the package contains many other binaries for simulating detectors, analyzing data, and many other utilities.

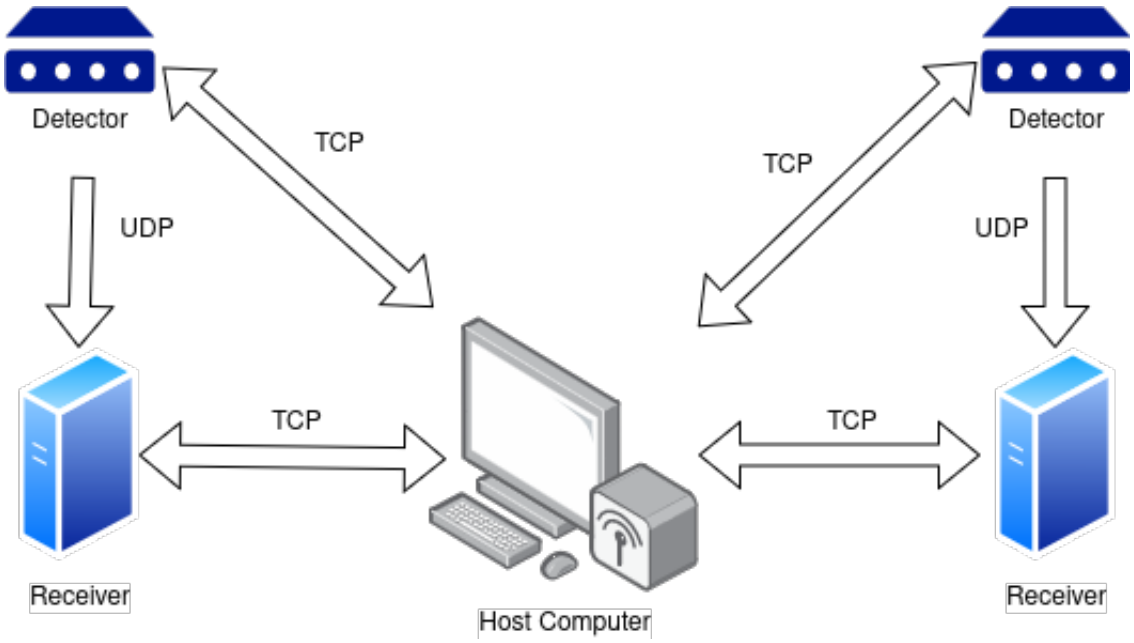

Figure I.1 – slsDetectorPackage setup for two detectors. Configuration uses TCP while data streaming from detector to slsReceiver uses UDP

3 Problem Statement

3.1 Existing Solution

The detector's group includes scientists, software engineers, firmware engineers, chip designers and many more roles. The group is diverse and the libraries' usage differs from one user to another. In general the usage of the libraries includes: acquiring data from network , configuring receivers and detectors, storing incoming data, processing data on the fly or after storing it. For the standard functionalities users rely on the slsDetectorPackage binaries. But the slsDetectorPackage is a generic software developed for public use and has very broad functionalities. Hence, for specific use cases scientists might need to write their own scripts or change the slsDetectorPackage source code and build again.

3.2 Limits of The Existing Solution

The slsDetectorPackage is a very powerful software package, but it has some major limitations.

3.2.1 Code Complexity

First, the code is very complex and has a steep learning curve. The code is written in C++ and uses many advanced features of the language. The code is also very large, with many thousands of lines of code. This makes it difficult for new users to understand how the code works, and to modify it to suit their needs. In addition, scientists are not software engineers, and in case of a bug, new feature or a specific use case they will be exposed to complex code that they are not familiar with. This might include the need to understand C++ code, multi-threading, network programming, and many other advanced topics.

3.2.2 Code Rigidity

Second, the code is very rigid and inflexible. The code is designed to work in a specific way, and it is difficult to modify it to work in a different way. This means that scientists are limited in what they can do with the code, and they are forced to work within the constraints of the existing code. This can be very frustrating for scientists, who may have specific requirements that are not met by the existing code. Furthermore, some of the speicific use case code is very brittle and can break easily if the code is modified. It lacks proper testing, error handling and logging. This makes it difficult to maintain and extend the code.

3.2.3 Code Duplication

Third, the code is duplicated in many places. Scientist often rely on their own scripts to perform specific tasks. This leads to each scientist having their own version of the code, which is difficult to maintain and update. and also results in the use of sub-optimal code, which is not efficient or reliable.

3.2.4 Data Storage Limitations

As the detectors become more and more advanced, the amount of data that they produce is increasing. This has made processing the incoming data in real-time a must. The slsDetector-Package provides very limited functionalities for processing data on the fly. This means that scientists are forced to store the data on disk and process it later. This is not ideal and is becoming less practical as the amount of data can be very expensive to store and process. The new library should be designed to process around 10GB/s of data in real-time.

4 Project Goals

The goal of this project is to develop a new library that will address the limitations of the existing software. The new software package will be designed to be simple, flexible, and efficient. It will be easy to use, and will be designed to meet the needs of scientists and engineers.

The library should include functionalities for acquiring data from receiver servers, stream data to receivers, read and write raw data files and numpy files, includes commonly used algorithms for data processing and it should expose a C++ and a Python API.

In addition, code should be well tested, documented, and should include logging and error handling. The library should be designed to be extensible, so that new features can be added easily in the future. and also flexible so that it accomodates different and upcoming use cases.

The library should be designed to be efficient, so that it can process data in real-time, and should be able to handle large amounts of data. It should use parallelism to distribute the load on multiple cores, and should be able to take advantage of the GPU for processing data. On the other hand it should abstract the low level details of the hardware and network communication to make it easy to use for the scientists.

5 Work Methodology

We used the **Kanban** methodology to manage the project. Kanban is a subsystem of the Toyota Production System (TPS), which was created in 1940s to control inventory levels, the production and supply of components, and in some cases, raw material. [JG10]

The board is divided into several columns:

- **Backlog** Contains all the tasks that need to be done.
- **To Do** Contains the tasks that are ready to be worked on.
- **In Progress** Contains the tasks that are currently being worked on.
- **Done** Contains the tasks that are completed.

5.1 Tools

We used Github Projects to manage the Kanban board. Github Projects is a tool that allows you to create a Kanban board and manage your tasks. It is integrated with Github, so you can link your tasks to your code, and track your progress easily. We also used Github Issues to create tasks, and Github Pull Requests to review and merge the code.

The Kanban boards were available for all the group members (involved in project or not), so that they can see the progress of the project, and contribute to it if needed. The boards were updated regularly, and the progress was tracked using the boards. The boards were also used to plan the work, and to assign tasks to the group members.

5.2 Benefits of Kanban

The Kanban methodology has several benefits:

- **Visibility** The Kanban board provides a visual representation of the work that needs to be done, and the progress that has been made.
- **Flexibility** The Kanban board is flexible, and can be easily adapted to the needs of the project.
- **Efficiency** The Kanban board helps to prioritize the work, and to focus on the most important tasks.
- **Collaboration** The Kanban board is a collaborative tool, and can be used by all the group members to track the progress of the project.

5.3 Meetings

We had regular meetings with the group members to discuss the progress of the project, and to plan the work. On each Tuesday, The whole group meets for about an hour to discuss the progress of the multiple projects that are being worked on. This meeting helps to keep everyone informed about the progress of the projects, and to identify any issues that need to be addressed.

In addition, on each Friday, a one-on-one meeting is held with the project supervisor to discuss the progress made during the week, and to plan the work for the next week. This is a more detailed meeting where we discuss the tasks that need to be done, and the kep track of the progress of the project.

Furthermore, the group has an open door policy, where anyone can ask for help, or discuss any issues that they are facing. Knowledge sharing is encouraged, and regular short discussions help overcoming the roadblocks that one might encounter.

6 High Level Planning

Even though the work methodology is highly flexible, we have a high level planning that we follow. The project is divided into several phases:

- **Phase 1: Research and Project Setup** In this phase, we researched the existing solutions, identified the limitations of the existing software, setup the project, and developed a high level architecture for the new library.
- **Phase 2: Implementation of the File Module** In this phase, we implemented the file IO module, which is responsible for reading and writing raw data files and numpy files.
- **Phase 3: Implementation of the Network Module** In this phase, we implemented the network module, which is responsible for acquiring data from receiver servers, and streaming data.
- **Phase 4: Implementation of the Processing Module** In this phase, we implemented the processing module, which includes commonly used algorithms for data processing.
- **Phase 5: Implementation of the Python API** In this phase, we implemented the Python API, which allows users to use the library from Python.
- **Phase 6: Documentation, Testing and Evaluation** In this phase, we tested the library more thoroughly, evaluated the performance, documented the code and added tutorials.

It is important to note that the phases are not fixed, and can be adapted to the needs of the project. The phases are used to plan the work, and to track the progress of the project. For example the python API was implemented in parallel with the other modules, as it was very useful to test the library and to get feedback from the users.

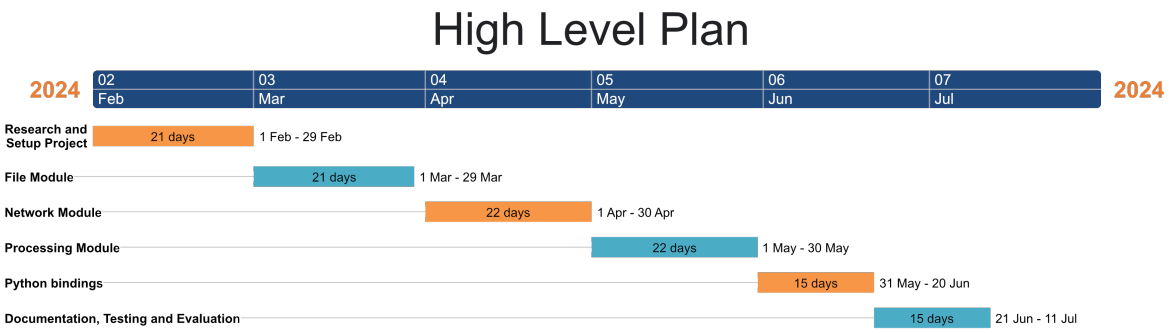

Figure I.2 – Gantt Diagram of the 6 phases of development. A margin was left at the end of the project to account for holidays, vacation days and development delays

Conclusion

La conclusion est en général sans numérotation, et n'apparaît pas dans la table des matières.

Part II

Chapter 2

Chapter II

Requirement Specification and Overall Architecture

Summary

Introduction

The requirement specification is the first step in the software development process. It is the basis for the design and implementation of the software system. It is the process of defining, documenting and maintaining the requirements of the system. The requirements are the description of the system services and constraints that are to be implemented.

In this chapter we will present the requirement specification of the project, we will define our actors, the functional and non-functional requirements of the system. We will also present the overall architecture of the project and the design principles that we will follow.

1 Requirement Specification

1.1 Actors Identification

Our library is developed for one main actor: the scientist. The scientist is the person who is going to use the library to develop new data processing algorithms, acquire data from different IO sources, analyze the data and visualize the results. It is assumed that the scientist is very competent but is not necessarily an expert in software development. The library is designed to be easy to use and to provide a high level of abstraction to the scientist.

1.2 Functional Specification

1.3 Non-Functional Specification

2 Overall Architecture and Guidelines

2.1 Onion Architecture

2.2 SOLID Principles

2.3 C++ Specific Design

2.3.1 Templates Metaprogramming

2.3.2 C++ Idioms

2.3.3 C++ Limitations

2.4 Project Architectural Design

2.4.1 Project Modules

2.4.2 Project Architecture

2.4.3 Class Diagram

Conclusion

Faire ici une petite récapitulation du chapitre, ainsi qu'une introduction du chapitre suivant.

Part III

Chapter 3

Chapter III

Implementation of a Flexible Data Analysis Library for Hybrid X-ray Particle Detectors

Summary

Introduction

1 Project Setup

1.1 Project Structure

1.2 Build System

1.3 Testing and Continuous Integration

2 Core Module Implementation

3 File IO Module Implementation

4 Network IO Module Implementation

5 Processing Module Implementation

6 Python Bindings Implementation

Conclusion

Part IV

Chapter 4

Chapter IV

Library Applications and Evaluation

Summary

Introduction

1 Examples of Library Applications

2 Parallelization and Multi-threading

3 Performance Evaluation

Conclusion

Conclusion and Perspectives

Bibliography

[Bög02] M Böge. "First operation of the swiss light source". In: *Proc. EPAC*. Vol. 2. 2002.

[But+24] Tim A Butcher et al. "Ptychographic nanoscale imaging of the magnetoelectric coupling in freestanding BiFeO3". In: *Advanced Materials* (2024), p. 2311157.

[Dul+24] Christian Dullin et al. "In vivo low-dose phase-contrast CT for quantification of functional and anatomical alterations in lungs of an experimental allergic airway disease mouse model". In: *Frontiers in medicine* 11 (2024), p. 1338846.

[JG10] Muris Lage Junior and Moacir Godinho Filho. "Variations of the kanban system: Literature review and classification". In: *International journal of production economics* 125.1 (2010), pp. 13–21.

[Lem+23] Tali Lemcoff et al. "Brilliant whiteness in shrimp from ultra-thin layers of birefringent nanospheres". In: *Nature Photonics* 17.6 (2023), pp. 485–493.

[Leo+23] Filip Leonarski et al. "Kilohertz serial crystallography with the JUNGFRAU detector at a fourth-generation synchrotron source". In: *IUCrJ* 10.6 (2023).

[Lv+22] Q. Z. Lv et al. "High-Brilliance Ultranarrow-Band X Rays via Electron Radiation in Colliding Laser Pulses". In: *Phys. Rev. Lett.* 128 (2 Jan. 2022), p. 024801. DOI: 10.1103/PhysRevLett.128.024801. URL: https://link.aps.org/doi/10.1103/PhysRevLett.128.024801.

[Pom+09] Daniel A Pomeranz Krummel et al. "Crystal structure of human spliceosomal U1 snRNP at 5.5 Å resolution". In: *Nature* 458.7237 (2009), pp. 475–480.

[Web] PSI Website. *About Swiss Light Source SLS*. URL: https://www.psi.ch/en/sls/about-sls.

Appendix : Miscellaneous remarks

- Un rapport doit toujours être bien numéroté;
- De préférence, ne pas utiliser plus que deux couleurs, ni un caractère fantaisiste;
- Essayer de toujours garder votre rapport sobre et professionnel;
- Ne jamais utiliser de je ni de on, mais toujours le nous (même si tu as tout fait tout seul);
- Si on n'a pas de paragraphe 1.2, ne pas mettre de 1.1;
- TOUJOURS, TOUJOURS faire relire votre rapport à quelqu'un d'autre (de préférence qui n'est pas du domaine) pour vous corriger les fautes d'orthographe et de français;
- Toujours valoriser votre travail : votre contribution doit être bien claire et mise en évidence;
- Dans chaque chapitre, on doit trouver une introduction et une conclusion;
- Ayez toujours un fil conducteur dans votre rapport. Il faut que le lecteur suive un raisonnement bien clair, et trouve la relation entre les différentes parties;
- Il faut toujours que les abréviations soient définies au moins la première fois où elles sont utilisées. Si vous en avez beaucoup, utilisez un glossaire.
- Vous avez tendance, en décrivant l'environnement matériel, à parler de votre ordinateur, sur lequel vous avez développé : ceci est inutile. Dans cette partie, on ne cite que le matériel qui a une influence sur votre application. Que vous l'ayez développé sur Windows Vista ou sur Ubuntu n'a aucune importance;
- Ne jamais mettre de titres en fin de page;
- Essayer toujours d'utiliser des termes français, et éviter l'anglicisme. Si certains termes sont plus connus en anglais, donner leur équivalent en français la première fois que vous les utilisez, puis utilisez le mot anglais, mais en italique;
- Éviter les phrases trop longues : clair et concis, c'est la règle générale !

APPENDIX : MISCELLANEOUS REMARKS

Rappelez vous que votre rapport est le visage de votre travail : un mauvais rapport peut éclipser de l'excellent travail. Alors prêtez-y l'attention nécessaire.

22

APPENDIX : MISCELLANEOUS REMARKS

23