OPERA HPC

OPEn HPC theRmomechanical tools for the development of eAtf fuels

Deliverable D4.1 – Best practices and QA protocols for code development

Version 1 – 02/08/2023

Funded by the European Union

Disclaimer

Document type	Deliverable
Document number	D4.1 version 1
Document title	Best practices and QA protocols for code development
Authors	A. Scolaro (EPFL) B. Michel (CEA) G. Latu (CEA) D. Pizzocri (POLIMI) L. Luzzi (POLIMI) I. Clifford (PSI)
Release date	02/08/2023
Contributing partners	EPFL, CEA, POLIMI, PSI
Dissemination level	Public

Version	Short description	Main author	WP leader	Coordinator
1	First submission by authors	A. Scolaro (EPFL) 04/05/2023	A. Scolaro (EPFL) 04/05/2023	B. Michel (CEA)

Abstract

The OperaHPC project aims to improve the numerical capabilities of 3D fuel performance modelling as part of its strategic objectives. To achieve this goal, an open-source approach has been chosen for the tools developed in the framework of the project, namely MMM and OFFBEAT, the latter coupled to the SCIANTIX code. As the open-source approach is relatively new in the domain of nuclear safety studies, this document presents a framework for achieving quality assurance targets for the open-source scientific computing tools within the OperaHPC project. First, the document provides a brief review of the most common QA programs and standards employed in the field, with a particular focus to the aspects that are more relevant to OperaHPC. Then, it discusses modern software development practices to improve code quality, highlighting the importance of revision control systems, testing methodologies, and documentation. Finally, it describes the concept of governance model for regulating interactions between contributors, users, and decision-makers. The framework presented in this document provides a backbone for the verification and validation actions that will be carried out within the project and contributes to the qualification of the MMM, OFFBEAT and SCIANTIX tools for nuclear safety studies.

Table of contents

1 Introduction

One of the strategic objectives of the OperaHPC project is to improve the numerical capabilities of 3D state-of-the-art fuel performance codes by bringing the simulation of the thermomechanical behaviour of fuel rods several steps forward in terms of microstructure description, high performance computing (HPC) capabilities and quantification of uncertainties. In agreement with this objective, an open-source development approach has been selected for the three simulation tools proposed in this project: OFFBEAT for the engineering fuel rod scale, MMM for describing mechanics at the microstructure scale, SCIANTIX for modelling fission gas behaviour. The open-source approach has the potential to improve the quality, accessibility, and sustainability of fuel performance codes, but it is relatively recent compared to the standard development methodologies employed for the codes traditionally used in fuel safety studies (with or without an official licensing from regulatory authorities). For this reason, this document proposes an overview of the requirements and best-practices expected for developing open-source codes in this field. This will facilitate the efforts in Tasks 4.3 and 4.4 to develop the MMM, OFFBEAT and SCIANTIX open-source codes for fuel safety studies.

1.1 Context

Many guidance documents and industry standards offer recommendations, or even requirements to qualify scientific computing tool (SCT). In the field of nuclear reactor analysis, qualification of a SCT corresponds to recognition by the operator or designer of a nuclear software tool that this product can provide results consistent with the requirements in the context of the Nuclear Safety Studies. This report describes a set of well-established software engineering best practices and quality assurance (QA) protocols that can be put in place in the context of open-source code development. The latter is expected to follow the technical acceptance criteria associated with the modelling of fuel behaviour in normal operation or in the event of incidents or accidents affecting commercial water-cooled reactors, research reactors, spent fuel or fuel storage pools.

The qualification process typically associated with the licensing of fuel performance codes includes a key phase dedicated to verification, validation, and uncertainties quantification (VVUQ). In line with this industrial methodology, the OperaHPC project is targeting the implementation of this VVUQ phase for the two SCTs developed for the analysis of generation 2 and 3 nuclear reactors, i.e., MMM and OFFBEAT (coupled with SCIANTIX).

1.2 Objectives

Although the code development within the framework of OperaHPC will follow an open-source approach, it will still consider the well-established practices and QA protocols used worldwide for SCTs employed in nuclear safety studies. This document provides an overview of such standards and practices with the main objectives being to:

1) Ensure the traceability, quality, and reliability of the developed tools,
2) Facilitate their future maintenance,
3) Fulfil quality assurance requirements of nuclear safety authorities.

For example, these best practices and QA protocols may include, as we shall see:

- the use of a code versioning system,
- the use of continuous integration in combination with automated unit tests or regression test,
- a validation database and associated code inputs with automated launch script,

- well documented tools and models,
- the adoption of a programming paradigm to facilitate understanding and code maintenance,
- and the timely definition of coding standards.

Different practices are expected for each of the open-source tools and software developed in Work Package 4, as these will be based on their specificities and maturity. Therefore, this report will not provide specific details on each code (for which we will refer to the dedicated web pages), but will offer general and consistent guidelines and methodologies that should ensure that the provided simulation tools are developed in agreement with nuclear safety authority's requirements. As open-source development raises specific questions related to quality assurance and qualification, we will also propose rules and tailoring strategies capable to handle these questions.

The multiple processes in open-source development can be broken down into numerous tasks, requiring different skills and degrees of technical expertise. To reach the expected level of quality assurance, it is essential to define a variety of roles that allow different types of contributions to strengthen the software and prevent introducing errors. For this reason, we will also highlight the importance of selecting a governance model that dictates the exact roles and mechanisms for contributing to the open-source project.

1.3 Structure of the document

The following sections will summarize the QA standards and the software engineering practices that are traditionally used for developing codes in the scope of nuclear safety analyses. Specifically, in Section 2 the reader will have a synthetic overview of the different QA programs and standards commonly employed in the nuclear safety domain, with a particular focus to those aspects that are more relevant for the tools developed in the framework of the OperaHPC project. Then, Section 3 will outline modern software development rules to improve code quality, to ensure robustness and to ease the maintenance of the software in a sustainable manner. We will focus on those rules that are expected to be most beneficial to the open-source development planned in the project for OFFBEAT, SCIANTIX and MMM. In Section 3 we will also comment on some well-adapted governance models for open-source projects. Section 4 will conclude the report by highlighting how the development methodology proposed in the project is fully consistent with high-quality requirements for nuclear safety studies.

2 Quality Assurance in the scope of nuclear modelling

2.1 List of QA standards

The following methodologies represent a non-comprehensive summary of the QA approaches used by different organizations that supply items or services that provide a safety function for nuclear facilities.

- The International Atomic Energy Agency (IAEA) produced general guidelines [1] for the use of computer codes for deterministic safety analyses. Such guidelines clearly state the fundamental role of verification and validation while stressing the importance of model assessment and uncertainty/sensitivity analyses. The definitions and recommendations provided by IAEA represent a common ground for further indications provided by national agencies.
- The "Autorité de Sûreté Nucleaire" (ASN) produced guidelines for the qualification of software for nuclear applications [2]. Such guidelines include definition and description of the verification and validation processes (both in terms of separate effects and integral simulations) and provide recommendations for the construction of relevant safety case studies. Practical information for the application of the ASN guidelines are available by AFCEN [3].

- The American Society of Mechanical Engineers (ASME) provides guidelines and specific support for the establishment of QA for nuclear codes (the NQA-1 Certification Program [4]). The stress is on the identification and description of the technical requirements of the code to be qualified in relation to the different actors using it within the nuclear facility (design, licensing, operation and so on), as usual paired with verification and validation strategies.

These general recommendations overall highlight the importance of proper code documentation and description, with differential deep downs based on the targeted user/application, the critical role played by the verification and validation processes, paired with uncertainty and sensitivity analyses.

2.2 Actions towards QA in OperaHPC

One of the objectives of this deliverable is to identify the intersection between available QA standards and the specific goals of the open-source codes being developed in OperaHPC. It is worth clarifying from the get-go that the objective is not to qualify the codes involved in OperaHPC, but to implement development strategies in line with the qualification standards, hence facilitating further adoption of such codes in the industrial sector.

Several common features among available QA standards are in line with the current capabilities of the codes under development. In particular:

1. **Definition of the scope of the utilisation of the code**, achieved through the following formal steps
    a. **Identification of variables of interest**. This involves selecting a subset of output variables from the code that adequately represents each physical model relevant to the code's intended scope of utilization. Ranking these variables in order of importance might be beneficial for understanding their significance, but it is important to acknowledge that the determination of importance may differ based on the specific safety criteria under consideration. Expert judgment often plays a vital role in this ranking process, although it could be subjective and influenced by case-specific factors. To enhance the robustness of the assessment, numerical evaluation based on sensitivity studies can be employed to gain greater confidence in the ranking of these variables.
    b. **Identification of the principal physical phenomena**. Again, the list of phenomena is based on experiment judgement (grounded for example in available experimental evidence) and must meet the intended scope of utilisation. The code is required to include models for each physical phenomena identified. The level of description of each phenomenon may vary (e.g., some models can be based on experimental data whereas others can be physics-based) but all the important physical phenomena must be described. For example, the Phenomena Identification and Ranking Table (PIRT) process is a systematic way of gathering information from experts on a specific concept, and ranking the importance of the information, to meet some decision-making objective. It has been applied to many nuclear technology issues.
    c. **Identification of the influential parameters**. These can be either input variables, empirical parameters, or physical parameters governing the predictions of the models implemented in the codes. Their identification and ranking are to be based on expert judgement and/or sensitivity analyses.
    d. **Determination of the utilisation range**. The previous steps allow the definition of the utilisation range, obtained by the relation of the variation ranges of the influential parameters and of the variables of interest.
2. **Definition of the validation range**, which is to be intended as a subset of the utilisation range, for which it must be performed

a. **Verification**. For verification it is intended a process aimed at determining if the model equations are solved, in a broad sense, correctly. This includes the numerical methods and algorithms, and the data processing steps as well. Verification can be performed with different methods depending on the specific equation being approached and the standards of the specific application (e.g., method of exact solutions, method of manufactured solutions, comparison with reference algorithms on random datasets).
b. **Validation**. The validation activity targets the comparison of the variables of interest with available experimental data (if possible, including the associated experimental uncertainties). To minimise the occurrence of error cancellation and guarantee a satisfactory coverage of the utilisation range, ideally each model describing the identified physical phenomena must be validated as stand-alone. After this step, an integral validation is to be pursued, in which the interaction among the physical phenomena is considered.

As mentioned, these two steps are (with slightly different naming) common to several QA standards and represent the common ground on which further actions can be pursued (e.g., definition of safety cases, and so on). Recalling that in OperaHPC, given the research-oriented scope of the overall project, the objective is to perform preliminary actions towards the QA standards for the involved codes, these steps are considered sufficient within the scope of the project.

Summarizing, in the context of the OperaHPC project, the developers of the open-source codes involved will complement their model developments with

- Ranked list of variables of interest, models, and model parameters.
- Demonstration of verification for the models.
- Demonstration of stand-alone validation for each model (whenever possible).
- Demonstration of integral validation.

The details concerning how these actions are to be performed by each code developer are not outlined in this document but will be summarized in a future milestone of the project. The production of this additional material to be paired with the source code developed (in Task 4.3 and Task 6.3, mainly) is intended to be shared at least within the project, and preferably to be open source. Potential restrictions may exist for the shareability of certain material models and of many validation cases. Therefore, considerable efforts will be made to identify and utilize models available in the open literature to the fullest extend possible, while in coordination with WP5 a selection of open validation datasets is planned. As for the verification cases to be included, particular attention is connected to the HPC application of the codes involved in the project, and thus to be performed in synergy with Task 4.5.

In addition to these actions which are formally required in the qualification process, additional effort is going to be reserved for

- **Documenting the developed models**, with details connected to the variables of interest and the inputs/parameters. Ideally, the synchronization of the documentation material with the source code is to be pursued, but each code can set out its own specific ways and indicate it on the online repository.
- Implementing **automatic processes to guarantee the quality standards of the codes**. This includes the systematic use of regression test, peer review of models/code source, check of exhaustiveness of documentation, traceability of each tagged version of the code base, set of tools to automatically show the quality of the code, tools for version control, and so on.
- Performing uncertainty and sensitivity analyses (Task 5.3). These actions are not connected to the qualification process and are code dependent. Nevertheless, we mention them here since some of

the activities to be performed for the qualification process are common with the actions targeted in Task 5.3 (e.g., the sensitivity analyses that can be used to rank influential model parameters).

3 Software engineering rules to improve code quality

This section focuses on software engineering rules that can improve the quality of the fuel performance codes developed within the OperaHPC project. To this end, we will first outline some of the most renowned international software development and quality standards. Then, we will present practical rules and best practices for open-source code development, highlighting tools such as version control or regression testing that can aid in quality assurance. Finally, we will describe the concept of governance model, that clarifies the exact roles and mechanisms for contributing to the open-source project. Adhering to the rules and best practices described in this Section will guarantee that the simulations tools developed within OperaHPC in Tasks 4.3 and 4.4 meet the highest quality standards and can be used with confidence by nuclear safety authorities.

3.1 International Software Engineering Standards

The IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents [5]. Comparatively few software products are forced by law to comply with specific standards, however. This is particularly true in the area of open-source software. Standards are generally adopted voluntarily by a software engineering organization or imposed by the customer or other stakeholders. For safety critical systems, however, software needs to comply with the regulations of the country. As an example, software written for aviation control systems in the US are legally required to comply with the US Federal Aviation Administration guidelines RTCA/DO-178B [6].

Several general software development and quality standards are internationally available:

- The software engineering standard (PSS-05-0) of the European Space Agency (ESA) [7]. These standard mandates that all software shall have a lifecycle approach consisting of the following basic phases:
    - User requirements definition – The software scope and operational environment are documented in a User Requirements Document.
    - Software requirements specification – The requirements of the software are defined and documented in a Software Requirements Document.
    - Architectural design specification – The architecture and structure of the software are defined. The components, modules as well as the control and data flow between them are documented in an Architectural Design Document.
    - Detailed design and code production – In this phase the software itself is coded according to the specifications. Unit, integration, and system testing is performed according to verification plans defined in the Software Requirements and Architectural Design Documents. Once completed, a formal design review is carried out.
    - Transfer of software to operation and maintenance – This includes the activities performed for the installation, acceptance testing, transfer of the software to the operational team and monitoring.
- The MIL-STD-498 standard for software development of the US Department of Defense [8]. Similar to the ESA standard, this standard stipulates that the software engineering process shall include a

software requirements analysis, software design, software implementation, unit testing and integration and qualification testing.

- IEEE/ISO/IEC 12207 Standard for Information Technology-software life cycle processes [9][10]. The intended purpose of this international standard is to establish a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry. It provides a process framework upon which an organization can build its enterprise-level life cycle processes. It addresses the complete software engineering life cycle, from acquisition and supply, through development, to operations and maintenance. A graphical illustration of the life cycle is shown in Figure 1. One can see that this is an extensive standard that covers a broad range of topics and is actually intended for large companies. An organization, depending on its purpose, can select and apply an appropriate subset to fulfil their requirements. The scope of the developments within OperaHPC lie largely within the development, verification, and validation, along with quality assurance and training. Some relevant IEEE standards are therefore:
    - 830, Recommended Practice for Software Requirements Specifications
    - 1016, Recommended Practice for Software Design Descriptions
    - 1008, Standard for Software Unit Testing
    - 1012, Standard for Software Verification and Validation

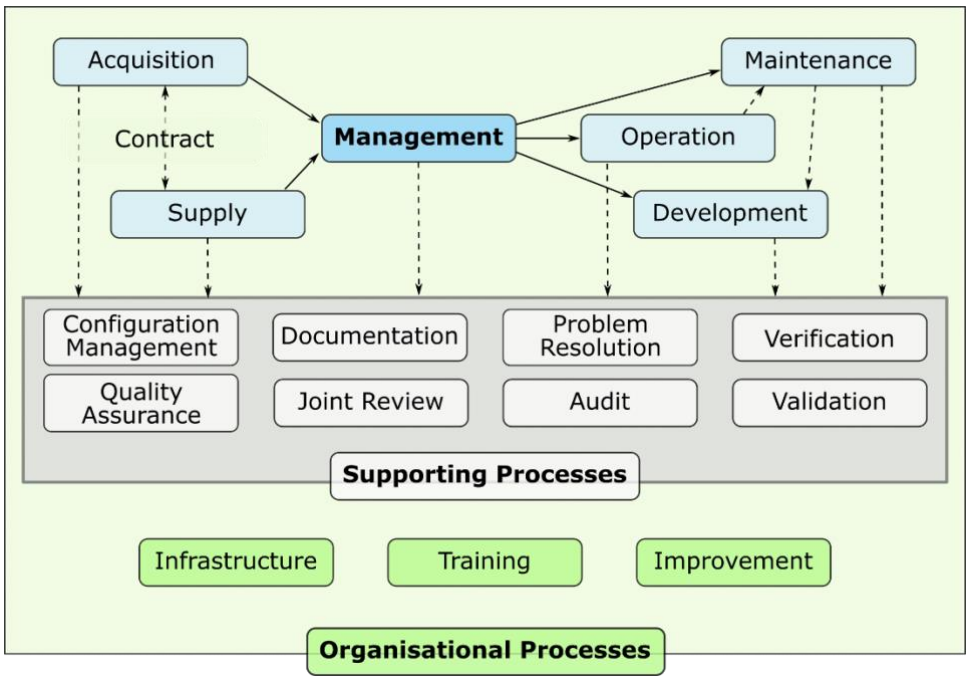

Figure 1. Overview of IEEE/EIA 12207 standard for information technology-software life cycle processes

3.2 Practical rules for open-source development

Given the nature of open-source software and developments, many informal guides are available online which relate to open-source software development best practices [11][12][13][14][15]. These guides will often touch on tools and approaches to facilitate aspects of the more comprehensive processes that are defined in the international standards. They are generally not, however, very detailed in their descriptions, and will generally highlight specific tools and approaches that are preferred by the authors but are not necessarily appropriate for all software products. Fogel [16] has attempted to provide a more comprehensive guide, which expands considerably on other informal guides and covers topics such as setting up your own

open-source community, technical infrastructure, financial aspects of open-source projects, management and communication, packaging and releasing software and legals aspects. Many of the guides mentioned above stress the following common points:

- The open-source project should be hosted online. Various public repositories are available for source code revision control (e.g., GitLab, GitHub, Sourceforge, Google Code Hosting, Gitorious).
- Code documentation should be kept up to date.
- Source code should be cleanly written, follow pre-defined guidelines and provide ample comments to understand.
- Clear rules should be provided for code contributions and best practices should be followed to manage the project, e.g., delegating work, code reviews, communication.
- Explicit or implicit hierarchy should be put in place for decision-making.
- Given that open-source users and contributors will likely be spread around the globe, tools for user and developer collaboration should be made available. This includes bug tracking systems, wiki pages, chatrooms, mailing lists, revision control systems and social networking services.
- The pros/cons of different licencing options should be considered.

From the perspective of quality assurance, several open-source tools are available to assist in building the processes for ensuring quality:

- Revision control systems (Git, Mercurial, SVN, CVS, etc.) are essential for traceability of the code. A revision control system is a software tool that helps to manage changes to source code files over time. It allows multiple developers to collaborate on a project by tracking changes to the code, providing version history, and enabling code branching and merging.
- Automated software documentation systems, which will scan the source code and produce up-to-date documentation on the structure of the software, may to some extent replace software design descriptions, providing references for software developers. Examples include Doxygen, DocUtils and Sphinx.
- Bug tracking systems will record bug reports and are therefore valuable in tracing the code issues and their resolution. Typically, they allow developers to create, track, and prioritize bugs, and provides information about their status and resolution. Many of the online revision control systems provide bug tracking tools.
- Non-regression testing tools (CTest, PyUnit, JUnit, ReFrame, etc.) as well as so-called continuous integration services (Jenkins, GitLab, etc.) are valuable for automating the process of testing new code versions. They provide relatively quick feedback on whether recent changes to the code do not break existing functionality. Continuous integration services can automate the process of building and testing the code, typically running tests automatically whenever changes are made to the codebase. Some of the more advanced tools will provide information on how comprehensive the unit tests are, i.e. how much of the software remains untested (so-called code coverage).

3.3 Management of an open-source project

Effective management is crucial for the success of an open-source project. A governance model is a set of rules and customs that define who gets to do what and how they are supposed to do it in an open-source project. It regulates and clarifies:

- The decision-making processes, including how contributors can propose changes to the code, how these are reviewed and accepted (e.g., merging of different branches).
- The open-source code license (to be chosen among GPL, MIT, BSD or others) which defines how the software can be used, modified, and distributed by others..

- The organizational structure of the project, including roles and responsibilities of contributors, users, and decision-makers.
- Guidelines for contributors, such as coding standards, documentation requirements, and testing procedures.
- Supervision of internal processes, such as code reviews, software life cycle stages (beta version delivery, release stage, tagging used to capture a point in code history).

There are several governance models that open-source projects can adopt. A few examples are the *meritocracy* model, where decision-making power is based on a contributor's track record of contributions, and the *consensus* model, where decisions are made through discussion and agreement among all contributors. The appropriate governance model for an open-source project depends on its size, complexity, and goals.

Each code developed in the framework of OPERA-HPC has the goal to indicate on its own website the chosen governance model, providing a concise but clear description of the guidelines for the contributors, of the rules that specify the way members interact within the project, of the decision-making process and of the license type chosen for the given code. If this information is not provided, each code will follow a default *founder-leader* governance model which is quite common for new projects. In this case, the individual or group who started the project also administers the project, establishes its vision, controls all permissions. This group has the final say for all the important decisions concerning the code.

4 Conclusions

In this document, we have listed a series of QA targets for the development of fuel performance codes in respect to several safety authorities' expectations. We have also outlined a set of important rules and guidelines for achieving these QA targets in the context of open-source software development, specifically for the OPERA-HPC project. Each major scientific computing tool in the project will have its own website providing tools, organization rules and reference documents to ensure that the QA targets are gradually achieved. While this document only provides the framework and organizational backbone for the open-source development of the SCT in the Opera-HPC project, specific VVQI actions, including verification and validation, are planned in the WP5 and they will be able to rely on the methodology proposed in this document. Moreover, these actions in WP5 will bring an important contribution to the qualification of the MMM, OFFBEAT and SCIANTIX tools and will provide a fundamental step towards the licensing related to Nuclear Safety Studies. As listed in [17], several initiatives are on their way to provide open-source codes for the nuclear community. In this framework, OPERA-HPC is also committed to contribute and share insights on the way to achieve open-source software that meets expected levels of quality assurance.

References

[1] INTERNATIONAL ATOMIC ENERGY AGENCY, Deterministic Safety Analysis for Nuclear Power Plants, IAEA Safety Standards Series No. SSG-2 (Rev.1), IAEA, Vienna (2019) https://www-pub.iaea.org/MTCD/Publications/PDF/PUB1851_web.pdf

[2] Autorité de Sureté Nucléaire, Qualification of scientific computing tools used in the nuclear safety case – 1st barrier, GUIDE N. 28 https://www.french-nuclear-safety.fr/asn-regulates/asn-guides/asn-guide-no.-28

[3] AFCEN, PTAN RCC-C Qualification OCS rev A - PTAN Qualification of scientific computing tools used in the nuclear safety case - 1st barrier - PTAN RC 20 001 Ind A, 2019 https://www.afcen.com/en/rcc-c/140-ptan-rcc-c-qualification-ocs-rev-a.html

[4] Nuclear Quality Assurance (NQA-1) Certification Program available at https://www.asme.org/certification-accreditation/nuclear-quality-assurance-nqa1-certification

[5] R.S. Pressman, Software Engineering; A Practitioner's Approach, Seventh Edition, Mc Graw Hill, 2010

[6] H. H. Ammar, The Software Development Standards, Presentation available online at https://community.wvu.edu/~hhammar/rts, accessed 24 March 2023

[7] ESA Software Engineering Standards, European Space Agency, ESA PSS-05-0 Issue 2, February 1991

[8] P.R. DeWeese, MIL-STD-498 Software Development and Documentation, SPC-94032-CMC, Version 01.00.00, July 1994

[9] IEEE Standards Collection: Software Engineering, IEEE Standard 610.12-1990, IEEE, 1993

[10] Systems and software engineering - Software life cycle processes, ISO/IEEE/IEC 12207:2017

[11] Open Source Development Guidelines, available at https://wiki.civiccommons.org/Open_Source_Development_Guidelines, accessed 24 March 2023

[12] Best Practices | politique-de-contribution-open-source, available at https://disic.github.io/politique-de-contribution-open-source/pratique.en.html, accessed 24 March 2023.

[13] T. Adhikary, Open Source for Developers – A Beginner's Handbook to Help You Start Contributing, available at https://www.freecodecamp.org/news/a-practical-guide-to-start-opensource-contribution, accessed 23 March 2023

[14] P. Kulkarni, How to Maintain an Open Source Project – Best Practices and Tips, available at https://www.freecodecamp.org/news/how-to-maintain-an-open-source-project, accessed 24 March 2023

[15] B. Hill, Free Software Project Management HOWTO, available at https://mako.cc/projects/howto/FreeSoftwareProjectManagement-HOWTO.html, accessed 24 march 2023.

[16] K. Fogel, Producing Open Source Software; How to Run a Successful Free Software Project, available online at https://producingoss.com, accessed 24 March 2023

[17] IAEA Open-source Nuclear Codes for Reactor Analysis, available at https://nucleus.iaea.org/sites/oncore/SitePages/List%20of%20Codes.aspx, accessed 30 March 2023