TELEMAC SYSTEM

Software Quality Plan

Version v8p5

December 1, 2023

Contents

1. Goal, domain of application and responsibilities

1.1 Introduction

The TELEMAC SYSTEM is an integrated suite of solvers for use in the field of free-surface flow. Having been used in the context of many studies throughout the world, it has become one of the major standards in its field. The TELEMAC SYSTEM is managed by a Consortium of core organisations: Artelia (France), BundesAnstalt für Wasserbau (BAW, Germany), Centre d'Études et d'expertise sur les Risques, l'Environnement, la Mobilité et l'Aménagement (CEREMA, France), Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique (CERFACS, France), Daresbury Laboratory (United Kingdom), Électricité de France R&D (EDF, France), HR Wallingford (United Kingdom), International Marine and Dredging Consultants (IMDC, Belgium) and École des Ponts ParisTech (France).

The TELEMAC SYSTEM is used by most partners for dimensioning and impact studies, where safety is prevailing and, for this reason, reliability, validation and a worldwide recognition of our tools are of utmost importance. As a consequence and to improve access to the TELEMAC SYSTEM for the whole community of consultants and researchers, the choice of the open source has been made. Anyone can thus take advantage of the TELEMAC SYSTEM and assess its performances.

This Quality Plan describes the general dispositions in place to ensure the quality of the TELEMAC SYSTEM and its services.

1.2 Description of the software

The TELEMAC SYSTEM is composed of several modules including numerical models and resolution algorithms useful for its utilisation.

The TELEMAC SYSTEM and its tools are open source, under GPL license, with the BIEF finite element library under LGPL. It is available from the TELEMAC SYSTEM website (`www.opentelemac.org`). This website is the main access to the TELEMAC SYSTEM information and documentation.

The TELEMAC SYSTEM can use external software and libraries. This Quality Plan does not cover those external elements (MPI, METIS, MED...), it only covers the functions using them in the code.

This quality plan should guarantee the interoperability between the TELEMAC SYSTEM modules and software used for pre- and post-processing (such as Fudaa, Salome...). The minimum requirements for interoperability are to follow the same standard as any development in the TELEMAC SYSTEM (Chapter 5).

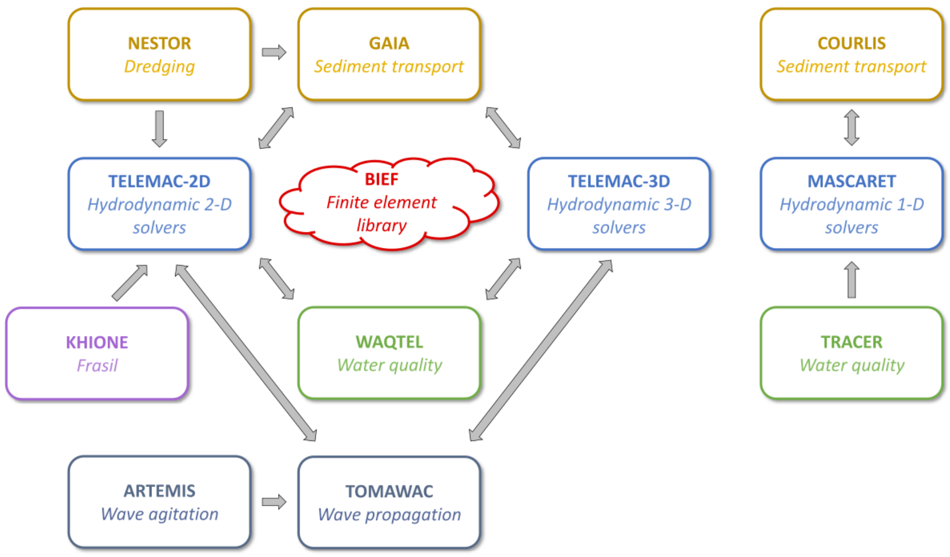

Figure 1.1: Modules of the TELEMAC SYSTEM

1.3 Versions concerned

This Software Quality Plan (SQP) is applied since version 7.0 of TELEMAC and version 8.0 of MASCARET.

1.4 Quality Plan Maintainers

The purpose of this paragraph is to identify the people in charge of the creation, verification, validation and follow-up of the Software Quality Plan. The document 'Organisation of the TELEMAC SYSTEM activity' in Appendix B gives more details on the organisation and on the action in their charge. The Appendix A gives a nominative list for those roles.

1.4.1 Quality Handler

The Quality Handler (QH) is the TELEMAC SYSTEM Project Manager (PM). He is in charge of the creation and the follow-up of the Software Quality Plan. He can be helped by the Code Handlers (CH) to whom he can delegate the implementation of the necessary changes. He has to verify that the applied action complies with the SQP.

1.5 Evolution of the Software Quality Plan

The evolution of the Software Quality Plan and its complementary documents can result from the following actions:

- Evolution of the Quality procedure in EDF R&D.

- Evolution of the organisation inside the Consortium.
- Evolution of the handling procedure for the software.
- An extension of the applicable domain.

1.6 Case of non application of the Software Quality Plan

In case the SQP or part of it cannot be applied, a derogation is possible, if validated by the QH. This derogation is traced in a document with the Software Quality Plan. Otherwise a modification of the SQP can be asked by the QH so the SQP could be applicable again.

2. Terminology

2.1 Glossary

Module: It is one of the core elements of the TELEMAC SYSTEM. There are 13 of them: TELEMAC-2D, TELEMAC-3D, TOMAWAC, ARTEMIS, GAIA, KHIONE, POSTEL-3D, BIEF, STBTEL, WAQTEL, MASCARET and COURLIS.
Post processor: tools to visualize results from a TELEMAC SYSTEM run.
Source files: list of the files used for a feature. Those files can be Fortran code, Python scripts, Jupyter Notebooks...

2.2 Abbreviation

ADEPHE: the TELEMAC SYSTEM EDF developers meeting
CIS: Continuous Integration System
CH: Code Handler
DHD: Delegate head of department
GUI: Graphical User Interface
HD: Head of department
HG: Head of the Group
PM: Project Manager
QH: Quality Handler
SQP: Software Quality Plan

3. The procedure at EDF R&D

The procedure `H-D03-2011-01748-FR` ”Les actions de management de projet applicables à EDF R&D et leur modulation” from the R&D Quality system division [2] states the requirements needed at EDF R&D to submit, define, contractualize, report the state and conclude a R&D project, as said in the referential 1999 of the EDF project management and following the EDF steering process. Especially, it gives the following definition of a Software Quality Plan: ”The Quality plan is a document which describes the standard for technical, administrative, financial, timetable order as well as the actions to apply for organisation, steering and managing off the project. It describes the timetable of the project, the product expected and all the elements about handling problems.” (translated from the original in French). It also describes the processes and actions aimed at maintaining the quality of the software and its continuous improvement.

The document [3] was also used as a guideline to write this Software Quality Plan.

The document [4] helps to classify numerical developments by dividing them into four categories and outlining the steps to take to capitalise on each one.

The document [1], if not specific to the TELEMAC SYSTEM, gives good guidelines on how to handle Numerical Software.

3.1 Documents specific to the TELEMAC SYSTEM

The practical documents for the respect of the Software Quality Plan are the following:

- All the documents referenced in the SQP given in appendix: the development plan, the organisation of the TELEMAC SYSTEM activity and a nominative list of the people in charge of the TELEMAC SYSTEM.

4. Organisation of the TELEMAC SYSTEM activity

4.1 Structure of the TELEMAC SYSTEM activity

4.1.1 General Organisation

The TELEMAC SYSTEM is managed by a Consortium composed of two main instances:

- The Steering Committee,
- The Technical and Scientific Committee.

The main functions of the Steering Committee are as follows:

- To seek advice from the Technical and Scientific Committee and so to decide on the main strategic technical, technological and scientific focuses of the Consortium for the medium-to-long term.
- To seek advice from the Technical and Scientific Committee and so to decide, choose and prioritise action items to be included in the Development Plan, whether the action items are proposed by the Members of the Consortium or by third Parties.
- To define modes of implementation of the Development Plan and modes of integration in the SOFTWARE of the deliverables of the Development Plan calling on resources either within the Members of the Consortium or on the Experts in Confidence,
- To make sure that the deliverables of the Development Plan are of desired quality, including but not limited to compatible source code, documentation, cases for benchmarking and technical approval.
- To identify, define and validate the strategy for extracting added value from the Prior Knowledge and promote the inclusion of Prior Knowledge in the software whether it resides within the Consortium or not.
- To ensure that the present Membership Agreement is performed properly and, if necessary, propose endorsements aimed at improving operation of the Membership Agreement.
- To grant/revoke voting rights to Member Representatives.
- To include/exclude a Member to/from the Consortium subject to the duration and termination of a Member's membership.

The main functions of the Chairperson of the Steering Committee are as follows:

- To gather action items, feature wish lists and developments provided by Member Representatives and third Parties and to seek development ideas from other open source software communities.
- To convene and preside over the meetings of the Steering Committee.
- To formalise and report minutes of the meetings of the Steering Committee.
- To formalise and report the Development Plan.

The main functions of the Technical and Scientific Committee are as follows:

- To advise the Steering Committee in its scientific and technical focuses.
- To diagnose for the Steering Committee the admissibility of the developments supplied to the Steering Committee by the Members of the Consortium or by third Parties.
- To enlighten the Steering Committee on the prioritisation to be done and the benchmarking and technical approval plans to be implemented.

The main functions of the Chairperson of the Technical and Scientific Committee are as follows:

- To convene and preside over the meetings of the Representative Members of the Technical and Scientific Committee.
- To formalise and report minutes of the meetings of the Representative Members of the Technical and Scientific Committee and gather appropriate references, technical notes or scientific evidences supporting its conclusions.
- To synthesise answers to requests submitted via the Chairperson of the Steering Committee.

4.2 Step in the TELEMAC SYSTEM activities

The TELEMAC SYSTEM activity is divided in three main categories directing the life of a version of the software, from its creation to its exploitation and finally to its archiving (See Development Plan in Chapter 5 for definition of those notions):

- **Development activity**: concerns any modification on the code, its documentation and the associated tools, whether it is an evolution, a correction or a reorganisation.
- **Verification and validation activity**: every development must pass through the verification and validation tests. A major version or production version (see Chapter 5) must run all the cases.
- **Exploitation activity**: concerns the distribution, workshops, support, maintenance of the different versions and removal of exploitation, as well as the archiving.

4.2.1 Responsibilities

The affectation for the different responsibilities are given in the document "Nominative list of the people of TELEMAC SYSTEM" in Appendix A.

4.3 Development activity

A development represents any intervention on the source code of the TELEMAC SYSTEM, its documentation or its tools (i.e. any element under the configuration, see the development plan in Chapter 5). The development on the code can be subdivided into different categories:

- **Evolution maintenance**: the evolution maintenance changes the software to break some of the limitation it has (performance, strength, precision, adaptation to demands. . .) or to add new functionalities to answer new demands or requests.
- **Corrective maintenance**: the corrective maintenance removes a bug in the code, the examples or the documentation.
- **Adaptive maintenance**: the adaptive maintenance adapts the software when its environment changes (operating system, compilers, new type of clusters. . .) to ensure that the software still works in that environment.

Those three categories imply an integration process in the development version of the TELEMAC SYSTEM.
The main actors are:

- The development team from each member of the consortium.
- The developer from outside firms.
- The Open-source community, academic and industrial partners.

The process attached to those activities are described in the development plan in Chapter 5.

4.3.1 Processes, tools and rules

The processes to follow for each development activities are described in the development plan in Chapter 5. Following those processes ensure traceability, non-regression, portability, verification, validation and re-usability of the functions developed and integrated in TELEMAC SYSTEM.
Those rules are based on the following principles:

- Keep the programming coherent with the existing one.
- Always base yourself on the latest version of the code (the development version) when creating new files.
- Communicate with the developer community on the evolution of the code structure.

Every new development must have an issue and/or merge request opened on the TELEMAC SYSTEM GitLab project page (`https://gitlab.pam-retd.fr/otm/telemac-mascaret`), may it be an evolution, corrective or adaptive maintenance in order to organize and prioritize the developments done in the TELEMAC SYSTEM.

All these developments must be validated by the associated CH and the PM.

4.3.2 Evolution maintenance

The developments realised for any maintenance are under the responsibility of the developer who was given this assignment by the CHs. They are handled by following the processes described in the development plan in Chapter 5.

The final integration of the development of a maintenance is in charge of the associated CH and must follow the process described in the development plan in Chapter 5.

4.4 Verification and validation activity

The verification and validation activity aims to verify the implementation on an algorithmic level or on the complete model using verification and validation test cases (even if a small part of the code was impacted as it can be for the corrective and adaptive maintenance). It is complementary with the test realised during the implementation (See the development plan in Chapter 5). We distinguish them as follow:

- **Verification**: The verification aims to establish that the code solves efficiently the mathematical problem it is supposed to resolve (and that is described in the specification of the concern algorithm). The verification focuses on the quality of the implementation from a computer science and theoretical point of view. The answer to the verification process is a binary one: the code does or does not respond to the demands. To do that verification, an analytical solution of the model or a solution based on measurements with the possible addition of source terms to the equation is used. The process is then most of the time to compare the numerical solution with a reference solution on meshes with increasing mesh refinement. This verifies that the algorithm converges in mesh and gives the level of precision associated.
- **Validation**: The validation aims to see to what level the code is able to answer to a physical problem in a given application domain. The validation focuses on the ability of the code to predict event from a physical point of view. The answer to this process is not binary. The evaluation is made through a comparison between a numerical solution and an experimental solution. We distinguish the validation with separated effects, which focuses on an isolated physical phenomenon and the full validation which focuses on an industrial problem (which can be simplified to get access to experimental measurements) composed of multiple physical phenomena.
- **Qualification**: The qualification aims to establish the perimeter of action of the code by using heavily validated complexed cases. This action is not done by the developers, it is in charge of main users. The creation of a methodology note is part of the qualification process. The qualification can keep track of the performance of the code as well as the non-regression of the test cases.

4.5 Release and distribution activity

4.5.1 Release, reception, removal

Between the different versions of the TELEMAC SYSTEM, we distinguish the release of a major version or production version from the other versions (minor, development release).

The different types of version are given in the development plan in Chapter 5.

The release of a new major version of the TELEMAC SYSTEM is decided by the chief of department hosting the project.

The development version of the TELEMAC SYSTEM is not formally released. Nevertheless it is compiled and test cases are run periodically. The latest development version is available on the official GitLab repository (`https://gitlab.pam-retd.fr/otm/telemac-mascaret`).

The process for the removal of older stable version is described in the development plan in Chapter 5.

4.5.2 Distribution

Distribution in EDF R&D

The major and minor versions of the TELEMAC SYSTEM are installed on the MNEMO data storage solution on an area dedicated to the project. The DSIT handles the backup of these data.

The test cases are available (at least in read-only mode) on MNEMO. The verification and validation is done on HPC clusters based in EDF R&D (France). The environments on which the code is verified and validated are listed in the release note.

Distribution in EDF

The distribution outside of the R&D is under the responsibility of the TELEMAC SYSTEM Chain Handler. The environment guide written by the Chain Handler contains a help for the installation of the code.

Open source distribution

All the stable versions of the TELEMAC SYSTEM, the development version and the tools are under open source license (GPL for the TELEMAC SYSTEM modules except for the BIEF which is under LGPL) and are available on the website `http://www.opentelemac.org` which is the main access for all information about the TELEMAC SYSTEM.

Protection

The software is protected in France by the label TELEMAC since 05/02/96 with the number INPI 96/623 748.

4.5.3 Relation with the TELEMAC SYSTEM users

The services handled by the TELEMAC SYSTEM project and available to users are:

- A technical assistance: forum, bug tracker, guides, documentation.
- Meeting with users: internally in EDF or externally (User Conference)

All those services are available on the TELEMAC SYSTEM website (`http://www.opentelemac.org`). The developers responding to the forum are not bound to respond to the forum, they do that on their available time.

Technical assistance

The organisation of the technical assistance is divided into two levels:

- **First level assistance**
  This assistance can concern either the installation or the usage of the software. This assistance can lead to a demand for an evolution of the code or the discovery of a bug. The formalisation of those demands allows:
  - A better repartition of resources by the project.
  - The follow-up by the user of the demand and allowing him to interact with the developer handling the demand.
- **Second level assistance**
  This assistance is done by the development team of the TELEMAC SYSTEM. This is an anticipated action to create documentation and didactic software (user and developer documentation, online documentation, tutorials, FAQs. . .).

The assistance activity can lead to a development if the problem is deemed worth it.

The actors of this activity are:

- Users: they are telling us their needs, helping in the specification and the validation of the code. They can also respond to some questions on the forum.
- People allowed to submit development demands, more specifically the development team (see the development plan in Chapter 5): they answer to the forum and GitLab issues. They can subcontract some of the work.
- The Kernel Handler: handles the GitLab repository, its maintenance and its good use.

Meeting with users
The meeting with user orbits around the following element:

- **the TELEMAC SYSTEM User Conference**

4.5.4 Relation with partners

The partners (like for example the members of the TELEMAC SYSTEM consortium) must follow the same rules for the development as described in the development plan in Chapter 5.

4.6 Control of sub-contract

Any developer from the development team can sub-contract some of his development like:

- The corrective, adaptive, evolution maintenance.
- Action on the documentation.
- Action of verification and validation.

In this case the process and the rules to follow are the same as an internal development described in the development plan in Chapter 5. The developer ordering this sub-contract is in charge of the follow-up of the work and its future maintenance.
Other actions can be sub-contracted:

- Action on the website.
- Action of support.

5. The TELEMAC SYSTEM development plan

The development plan describes explicitly all the processes on the software during the life cycle of the TELEMAC SYSTEM code and ensures its Quality.

The TELEMAC SYSTEM activity is decomposed into three main activities that are described in Chapter 4.3:

- **Development activity**.
- **Verification and Validation activity**.
- **Release activity**.

5.1 Rights to commit new developments

People involved in the development of the TELEMAC SYSTEM can be divided into the following groups:

- **The TELEMAC SYSTEM development team (TDT)**: to be a part of the TDT you need to have writing access on the GitLab repository. To be in the development team, the developer must have read the developer guide and the SQP in order to apply it to its future developments. The PM is a member of the TDT.
- **Developers from EDF units**: they are developers from one of the group of EDF and are developing for the TELEMAC SYSTEM project. It is recommended that they follow the TELEMAC SYSTEM workshop. They will need to contact a member of the TDT to integrate their development.
- **Non EDF developers working with EDF**: they can be subcontractors, interns, PhD students or any other partners working with EDF. It is also recommended that they follow the TELEMAC SYSTEM workshop. The follow-up of the development will have to be done by a members of the TDT.
- **Open source community, academic and industrial partners**: The TELEMAC SYSTEM is an open source code under GNU-GPL. Anyone can get the source and develop their own version. But for a development to be integrated into the official version, it must be studied by the TDT and follow the process described in 5.3. If the integration is validated, the delivery of the evolution for the code must go through a member of the TDT (See 5.5).

5.2 Life cycle of a development

5.2.1 Definition of a development

A development represents a modification on the source code of the TELEMAC SYSTEM, its documentation and its dedicated tools. (i.e. everything that is under the configuration, see 5.6). The developments are classified into three types of maintenance:

- **Evolution maintenance**: this maintenance aims to upgrade the software in order to:
  - Break limits of some existing functionalities (performance, strength, precision...).
  - Develop new functionalities to satisfy new needs or demands.

  These evolutions can be developments of new features made by one of EDF Units. They can also originate from an outside developer and been brought in by the PM. Each evolution must be associated with a GitLab issue and merge request. This allows the PM to get a general overview of the work in progress. Depending on the size of the development, it can even be presented by the developer in an ADEPHE meeting. This allows the TDT to evaluate the impact of the development and maybe advise the developer on what to do. The development must be validated by one of the CHs.

- **Corrective maintenance**: the corrective maintenance is the removal of an anomaly in the code or documentation. Such an anomaly can be:
  - A problem in the code, a mistake in the documentation, an error in the GUI.
  - An error in a test case detected during the Verification and Validation process.
  - A problem in the implementation detected by user.

  Corrective maintenance must also be coupled with a GitLab issue and merge request if the fix is also to be applied to the current stable version.

- **Adaptive maintenance**: the adaptive maintenance concerns evolution in the code in order to comply with a change of environment (Operating System, compiler, new cluster...).

5.2.2 Life cycle of a TELEMAC SYSTEM development

The life cycle of a TELEMAC SYSTEM development follows a list of steps shown on the Figure 5.1. Those steps will be described in the next sections.

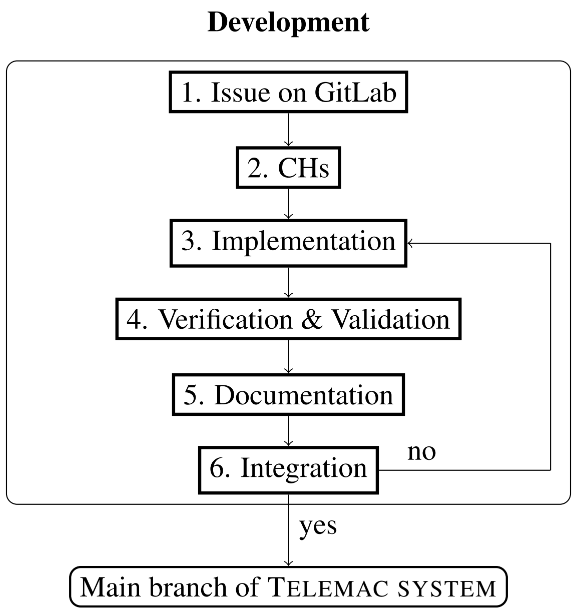

Figure 5.1: Life cycle of a TELEMAC SYSTEM development

Depending on the kind of development (corrective, adaptive, evolution maintenance) some of those steps can be skipped. The different steps are:

1. Proposal step.
2. Examination and impact study step.
3. Implementation step.
4. Verification step.
5. Documentation step.
6. Approbation and validation step.

This development cycle is based on a V scheme life cycle, it contains those different steps and the different types of testing (see next paragraph). To be more precise, it is more of spiral cycle in which the V cycle is read iteratively in order to improve the quality gradually.

5.3 Description of the development steps

5.3.1 Proposal step

Every demand must go through the proposal step, may it be a corrective, adaptive, evolution maintenance (see Section 5.2.1 for a definition).

This proposal is given to the TDT by creating a new GitLab issue (`https://gitlab.pam-retd.fr/otm/telemac-mascaret/-/issues`).

The goal of this proposal is to:

- Inform the CHs of the proposed development.
- Give more information on the impact in the software.

- Plan and organise the resources for the development.
- To prepare the validation in ADEPHE if necessary.

In charge: the proposer.
Documents: the GitLab issue.

5.3.2 Job repartition step

When the GitLab issue is published, the CHs have to determine if the development must be followed by the TDT. This mainly concerns evolution maintenance but can also concern corrective maintenance if it has a major impact on the code.

The goal of this step is:

- If it is an evolution maintenance, to verify the coherence of the evolution with: the kernel, the existing, in development or planned functionalities, the functionality itself, the modifications needed in the data structure, the architecture of the code or the GUI, the name of the new keywords and structure. In particular guiding the developer towards a reusing of existing routines.
- In case of a corrective maintenance it studies the impact on the Verification and Validation process.
- To help the developer in its choice of algorithm and implementation.
- To discuss the test cases for Verification and Validation.
- To estimate the impact on the documentation.

In charge: CHs, TDT, handler of the development.
Documents:

- Input: the GitLab issue.
- Output: update of evolution on the issue.

5.3.3 Implementation step

This is the coding part of what was specified in the GitLab issue, with the modifications that the discussion during the ADEPHE could have generated.

The developer must follow the coding conventions given in the developer guide of the TELEMAC SYSTEM. All the developments are done under a version control tool. For for each development, a branch dedicated to that development needs to be created and a GitLab merge request (https://gitlab.pam-retd.fr/otm/telemac-mascaret/-/merge_requests) must be opened to keep track of the development updates.

The control of the implementation is made by running the TELEMAC SYSTEM validation. When possible, in the case of evolutionary maintenance, it is preferable to add a test case that checks the new functionalitiy, if one does not exist.

The development must be done on a branch and follow the development version (or main, see section 5.6 for a definition). It is strongly recommended to keep up to date with the development version as often as possible. This lessens the workload of that process compared to what would need to be done at the end of the development.

It is also recommended to use developer tools during the development, such as a debugger (GDB, TotalView), a memory leak detector (valgrind) and a code editor (Atom, Emacs, Visual Studio Code...) or an IDE (Code::Blocks, Visual Studio...)

The GitLab merge request is updated as the implementation goes on.

In charge: the developer.
Documents:

- Input: GitLab issue and merge request, developer manual, verification and validation test cases.
- Output: source code, test cases, results of tests.

5.3.4 Verification step

The Verification and Validation process is deeply linked with the implementation process. This process aims to verify the code at the level of the algorithm and the whole model using the Verification and Validation test cases (even if a small part of the code has been modified, like during a corrective and adaptive maintenance). This is then complementary (even redundant) with the verification done during the implementation. The Verification and Validation process is explained in Chapter 4.4.

As mentioned in the previous paragraph, every development must contain Verification and Validation test cases:

- Evolution maintenance: for an evolution of the code at least one Verification and one Validation case must be added if existing cases cannot be used. The choice of those cases are discussed with the TDT.
- Corrective and adaptive maintenance: All the test cases must be rerun.

In charge: the developer.
Documents:

- Input: GitLab issue and merge request, validation manual.
- Output: update of GitLab issue and merge request, LaTeX the documentation follows the validation standard for each test case.

5.3.5 Documentation step

The TELEMAC SYSTEM technical documentation is composed of the following manuals: The five below are duplicated for each the TELEMAC SYSTEM module.

- **Reference Manual**: describes every keyword in the dictionary of one module.
- **Theory guide**: describes the physical phenomena modelled by the module and the numerical methods used to solve the equations modelling these phenomena.
- **Validation Manual**: presents a list of test cases which validate the conceptual model, algorithms and software implementations. It is a concatenation of all the test case documentation for one module.

- **User Manual**: describes how to use the code.
- **Online Manual (Doxygen)**: this documentation is linked with the source code (it is built by using special comment written in the source code). It describes the functions and structures of the source code.
- **Developer Manual**: describes all the actions a developer might have to do during a development. It also defines the programming convention.
- **NEWS.txt**: for every release, lists the new functionalities and corrections.

The documentations are under a configuration handler (Git) in LaTeX format. The documentation is under the responsibility of the Doc Handler and the CHs.
Depending on the type of development, a few cases can occur:

- **Evolution Maintenance**: for a new functionality of the code, a new section must be added in the theory guide describing the new physics and in the user manual showing how to use that functionality. The Online documentation as well as the **NEWS.txt** file must also be updated.
- **Corrective, adaptive Maintenance**: if the development has an impact on the documentation, it must be updated by the developer. If the development is to be merged on the version branch, the **NEWS.txt** file should also be updated.

In any case, the modification must go through the Doc Handler and must be validated by the CHs and the PM.

In charge: the Doc Handler.
Documents:

- Input: all the documentation.
- Output: the updated documentation.

5.3.6 Integration step

The integration step concerns the restitution of a development into the main branch of the TELEMAC SYSTEM (main, see 5.6).

The integration step is composed of the following steps:

- Designing the member of the TDT in charge of the integration of the development.
- Integration of the Verification and Validation cases under the supervision of the V&V Handler.

Not following those steps allows the CHs to refuse the integration and even remove it from the development version.

Presentation of the development in ADEPHE
The ADEPHE meetings are monthly. To be allowed in ADEPHE, a developer must have:

- Synchronized their branch with the latest version of the main branch.
- Verified that the code follows the programming rules.

- Added the new test cases for the development.
- Validated their branch using the validation tools (CIS).
- Updated the GitLab merge request.

During the ADEPHE, the person in charge of the development must present:

- The development and how it was implemented: the functionalities if it is an evolution maintenance, in case of a corrective, adaptive maintenance: the data structure, the new keywords, how to use it.
- The Verification and Validation cases and their results.
- The impact on the documentation.

At the end of the presentation the TDT must decide:

- The integration or the dismissal of the development in the main branch.
- The integration or the dismissal of the test case.
- The integration or the dismissal of the modifications on the documentation.
- The person in charge of the integration.

In case of a disagreement in the TDT the PM has the final word.
In charge: CHs, TDT.
Documents:

- Input: GitLab merge request, documentation.
- Output: update of the GitLab merge request.

Integration
After the green light of the TDT, the person in charge of the integration has to do the following actions:

- Accept the merge request.
- Close the GitLab issue.

In charge: TDT.
Documents:

- Input: GitLab merge request.
- Output: update of GitLab merge request, documentation, sources.

5.3.7 Checklists for Developer and for Integrator

To summarise the step that were explained before here is a checklist of the actions he will have to do:

1 ☐ Create a GitLab issue (on https://gitlab.pam-retd.fr/otm/telemac-mascaret/-/issues).
2 ☐ Discuss with TDT.
3 ☐ Update the main branch to the latest version.

4 ☐ Create a development branch to work on.

5 ☐ Do the work.

6 ☐ Update the documentation.

7 ☐ Update `NEWS.txt` to reflect the changes.

8 ☐ Add test cases to test the development (with documentation, graphics and validation).

9 ☐ Run `compile_telemac.py --check` (to check Fortran coding conventions).

10 ☐ Run `pylint` (to check Python coding conventions).

11 ☐ Run `compile_telemac.py --clean --rescan` and update cmdf files accordingly.

12 ☐ Run `compile_telemac.py --clean` for normal and debug configuration.

13 ☐ Run `validate_telemac.py` for normal and debug configuration.

14 ☐ Run `validate_telemac.py --notebook` for normal and debug configuration.

15 ☐ Run `doc_telemac.py` if there are any modifications in the documentation.

16 ☐ Run `damocles.py --eficas` if there are any modifications in the dictionaries.

17 ☐ Submit a merge request and assign it to the integrator.

The following checklist describes the actions that the integrator must follow:

1 ☐ Check that the GitLab issue and merge request are properly filled.

2 ☐ Check that the code follows the coding conventions.

3 ☐ Check that the test cases are appropriate.

4 ☐ Check that the implementation of the solution is good.

5 ☐ Check that the development is up to date with the main branch.

6 ☐ Check that the `NEWS.txt` file has been updated.

7 ☐ Check that the branch has been cleaned of redundant commits using Git interactive rebase.

8 ☐ Accept the merge request.

5.4 Maintenance

In this part, the maintenance concerns only the corrective and adaptive maintenance.

The TELEMAC SYSTEM project is in charge of the non-regression of the TELEMAC SYSTEM functionalities on the Verification and Validation test base.

The TELEMAC SYSTEM project is also in charge of the adaptive maintenance (see Section 5.2.1).

The project in charge of other developments is in charge of the corrective maintenance for their developments. But a help from the TDT is possible in case of big difficulties. If the bug is easy to correct the TDT can, with the authorisation of the person on charge, correct the bug themselves.

In any case the development must follow the development step described in Section 5.2.2 and summarised in Figure 5.1.

If the bug is also in the latest version branch, it should also be merged there and an issue and merge request must be created.

Removal of a functionality
The CHs can decide to remove a functionality if it has become obsolete or irrelevant. This decision must be validated by the PM. The CHs must warn the user community.

The removal of a functionality can only happen on the main branch, it is not retroactive.

The removal of a functionality concerns:

- The source code and the linked library.
- If there are any, the impacted verification and validation.
- The documentation.

5.5 Integration of development from the open-source community

An integration from the open-source community must follow the same process as an internal development.

5.6 Configuration management

5.6.1 Version control

The different versions of the elements of the TELEMAC SYSTEM software are handled by a version control tool. Such a tool allows to track the evolutions of the source code and ease the collaboration between multiple developers.

The version control tool that is used for the TELEMAC SYSTEM is Git, which allows to clone any revision of the source code with the entire modification history. The official Git repository of the software is hosted on a GitLab server and is accessible at the following URL: `https://gitlab.pam-retd.fr/otm/telemac-mascaret`.

5.6.2 Elements under version control

The elements of the TELEMAC SYSTEM software concerned by the version control are the source code, the environment scripts, the documentation and the test cases.

5.6.3 Type of version

The TELEMAC SYSTEM official or reference version
An official (or reference) version is a version handled by the TELEMAC SYSTEM project. The official versions are:

- The version of exploitation.
- The latest stable version, minor or major release.
- The development version.

For example, the development branches of the TDT are not considered official.

Creation of the reference version for the TELEMAC SYSTEM

The creation of the reference version is described here.

The TELEMAC SYSTEM version are handled using “tags”.

- The creation of a tag is decided by the Chain Handler and the PM.
- Tags are named in the format **vXpYrZ** where:
  - **X** is the major version index, incremented when a major modification is done to the code (ex: makeover of the interface, modification of the code structure. . .),
  - **Y** is the minor version index, incremented when new minor functionalities are added to the software,
  - **Z** is the patch index, incremented when a group of bug corrections are implemented in the software.

The versions are divided into branches and tags. There is:

- **The main branch** (**main**), also called the development version. It represents the work in progress between two stable versions. It is where the different kind of maintenances are integrated. Figure 5.2 shows the evolution of the main branch between two stable versions.
- **The version branch** (**vXpY**), created for each new major and minor version release. It is used only for corrective and adaptive maintenance and has passed the full validation process. A new version branch is created each year in September.
- **The stable version tag** (**vXpYrZ**), created on the version branch (**vXpY**) at each new release. Such a tag is fixed and is never modified afterwards. If a corrective or adaptive maintenance is necessary, the changes are first made on the main branch and then merged into the version branch, before the next tag (**vXpYrZ+1**). Figure 5.2 shows the creation of a branch **vXpY** and its corresponding tags. The first tag is created (pending validation) two months after the creation of the version branch.

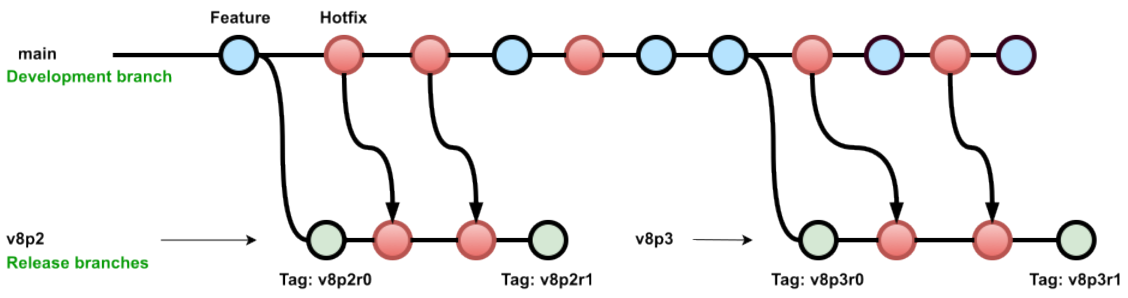

Figure 5.2: Development life cycle of the TELEMAC SYSTEM version

5.6.4 Version schedule

Version release

The different versions, major and minor, are separated by a flexible interval:

- Major versions are released every 4 to 5 years.
- Minor versions are released every year, if the amount of new developments is considered acceptable.

Finally, at any time we have the following versions:

- The development version: it is the **main** branch.
- The latest stable version **vXpYrZ**.

This timetable can be modified due to a big change in the code that would require longer validation to leave time for the normal integration to adapt.

Removal of stable version
The removal of a stable version is automatic when:

- A corrective version **Z > 0** is released.
- It reaches its expiration date.

Maintained versions
Only the last two minor versions are maintained.

Exploitation version
To do a study with the TELEMAC SYSTEM, it is recommended to use the latest stable version as it should be the most advanced and has the fewest bugs.

5.6.5 Checklist for version release

Here are the actions that need to be done when creating a new version **vXpY**:

1 ☐ Create the branch **vXpY** from the main.
2 ☐ Update the `NEWS.txt`. Replace main section by section **vXpY**.
3 ☐ In `documentation/data/TelemacDoc.cls` change the command `\telmaversion` to **vXpY**.
4 ☐ In `sources/utils/special/declarations_special.f` change the parameter VERSION to **vXpY**.
5 ☐ In `configs/systel.edf.cfg` change version to **vXpY**.
6 ☐ Run `compile_telemac.py --clean --rescan`. Update cmdf on main if necessary.
7 ☐ Run `validate_telemac.py`.
8 ☐ Run `validate_telemac.py --notebook --notebook-update`. Update content on the branch and also on main if necessary.
9 ☐ Run `doc_telemac.py`.
10 ☐ Add all the generated documentation to repository.
11 ☐ Run `damocles.py --eficas`.
12 ☐ Add the `scripts/python3/eficas` folder to the repository.

Here are the actions that need to be done when creating a new version **vXpYrZ**:

1 ☐ In the branch **vXpY** update `NEWS.txt` with release date.
2 ☐ Rerun Validation, documentation, eficas if some modifications where done to examples, documentation, dictionaries.
3 ☐ Create the tag **vXpYrZ**.

A. Nominative list of handlers

Main responsibilities

Name	**Function**	**First name, Last name**	**Unit**
PM	Manager of the TELEMAC SYSTEM project	Mathieu COUPLET	R&D/LNHE

Code Handlers (CHs)

Module	**First name, Last name**	**Unit**
TELEMAC-2D	Chi-Tuân PHAM	R&D/LNHE
TELEMAC-3D	Chi-Tuân PHAM	R&D/LNHE
WAQTEL	Chi-Tuân PHAM	R&D/LNHE
TOMAWAC	Thierry FOUQUET	R&D/LNHE
GAIA	Sara PAVAN	R&D/LNHE
KHIONE	Fabien SOUILLE	R&D/LNHE
NESTOR	Rebekka KOPMANN	BAW
ARTEMIS		
MASCARET		

Main responsibilities

Name	**Function**	**First name, Last name**	**Unit**
Chain hand.	Software chain handler	Boris BASIC	R&D/LNHE
Kernel hand.	Kernel handler	CHs	R&D/LNHE
V&V hand.	Verification and Validation handler	CHs	R&D/LNHE
Doc hand.	Documentation handler	CHs	R&D/LNHE
Com hand.	Communication handler	Mathieu COUPLET	R&D/LNHE

B. Organisation of the TELEMAC SYSTEM activity

B.1 Organisation

The general organisation of the TELEMAC SYSTEM activity is described in Figure B.1.

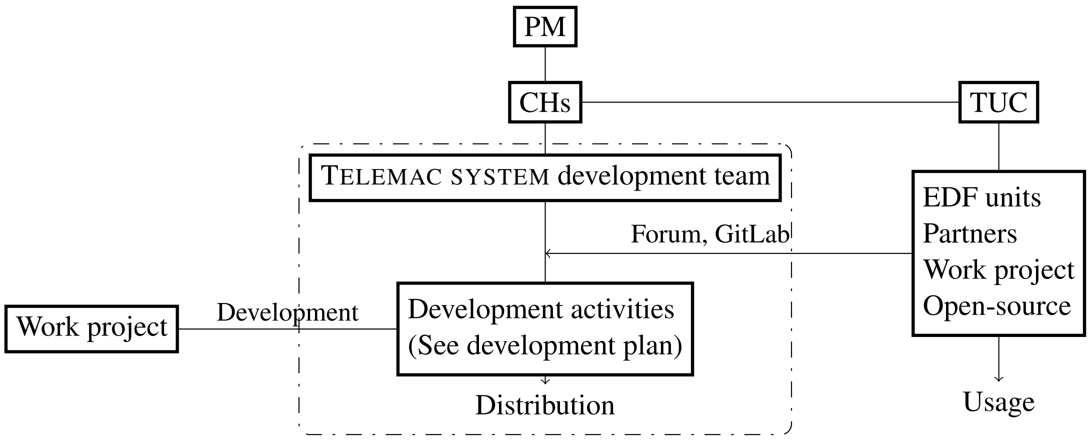

Figure B.1: General Organisation of the TELEMAC SYSTEM activity

The development activities are described in the development plan in Appendix 5.

B.2 Quality handler

In Table B.1 the level 1 and level 2 Quality handler for every TELEMAC SYSTEM activities. The nominative list is given in the "Nominative list of people in TELEMAC SYSTEM" in A.

Level	Quality Handling	Configuration Handling	Integration	Maintenance	Verification and Validation	Documentation	Diffusion
N1	PM	Chain hand.	Kernel hand.	Chain hand.	V&V hand.	Doc hand.	Chain hand.
N2	CHs	CHs	CHs	CHs	CHs	CHs	CHs

Table B.1: Quality Handlers for each activity

B.3 Coordination

Job	Roles
PM	• Project Manager. • Handles the project team in coordination with the CHs. • Establishes the annual action plan from the statement made by the Referring and project following committee. • Defines with the group chief the repartition of resources to complete the objectives. • In charge of the reporting (communication on the project evolution to the Strategy Handler and the resources manager). For example organises the COPIL (Steering committee) of the project where are shown the technical and budget advancements of the project. • In charge of the quality aspect of the project. Prepares at the end of the project the instrumentation of a new cycle.

Table B.2: Role of each actor in the coordination of TELEMAC SYSTEM

B.4 TELEMAC SYSTEM activities for each role

Job	Roles
EDF Units (i.e. EDF Groups) Work project	• Apply the Quality process defined in the SQP. • In charge of the implementation and the corrective maintenance of their development. • In charge of the Verification and Validation of their development.
PM	• Organises and dispatches the tasks between the developers, in accordance with their hierarchy. • Dispatches the corrective actions coming from the Issues tickets. • In charge of the ADEPHE meetings. • Has the final decision on the integration of developments. • Delegates those tasks to the CHs if necessary.
CHs	• Level 2 handlers of the coherence of integrated developments. • In charge of the configuration and the documentation. • In charge of the creation of branches and of the release and stabilisation of versions. • Has the final decision on the integration of developments. • Certifies the quality of the released versions.
Chain hand.	• In charge of the full chain of execution of TELEMAC SYSTEM. • In charge of the coherence and evolution of the data structure, the parallelism, the coupling, the pre/post-treatment, scripts and tools. • In charge of the installation of the released versions.
Kernel hand.	• In charge of the integration of new development and of their coherence. • In charge of the Issue, of its application and its good use. • In charge of the source code and its coherence.
V&V hand.	• In charge of the Verification and Validation of the TELEMAC SYSTEM code. • Verifies the usefulness of test cases and their redundancies. • Defines the list of test cases to be used. • In charge of the content of the Validation Manual.
Doc hand.	• Establishes the content of the TELEMAC SYSTEM documentation. • Controls the modifications made to the documentation. • Insures the coherence of the documentation.
Salome hand.	• In charge of the integration of the TELEMAC SYSTEM module in the SALOME platform. • Makes the link with the SALOME community, goes to the ADS (Salome developer meetings).

Table B.3: Role of each actor in the conducting of TELEMAC SYSTEM activities

C. Checklists

Development:

1 ☐ Create a GitLab issue (on https://gitlab.pam-retd.fr/otm/telemac-mascaret/-/issues).
2 ☐ Discuss with TDT.
3 ☐ Update the main branch to the latest version.
4 ☐ Create a development branch to work on.
5 ☐ Do the work.
6 ☐ Update the documentation.
7 ☐ Update `NEWS.txt` to reflect the changes.
8 ☐ Add test cases to test the development (with documentation, graphics and validation).
9 ☐ Run `compile_telemac.py --check` (to check Fortran coding conventions).
10 ☐ Run `pylint` (to check Python coding conventions).
11 ☐ Run `compile_telemac.py --clean --rescan` and update cmdf files accordingly.
12 ☐ Run `compile_telemac.py --clean` for normal and debug configuration.
13 ☐ Run `validate_telemac.py` for normal and debug configuration.
14 ☐ Run `validate_telemac.py --notebook` for normal and debug configuration.
15 ☐ Run `doc_telemac.py` if there are any modifications in the documentation.
16 ☐ Run `damocles.py --eficas` if there are any modifications in the dictionaries.
17 ☐ Submit a merge request and assign it to the integrator.

Integration:

1 ☐ Check that the GitLab issue and merge request are properly filled.
2 ☐ Check that the code follows the coding conventions.
3 ☐ Check that the test cases are appropriate.
4 ☐ Check that the implementation of the solution is good.
5 ☐ Check that the development is up to date with the main branch.
6 ☐ Check that the `NEWS.txt` file has been updated.
7 ☐ Check that the branch has been cleaned of redundant commits using Git interactive rebase.
8 ☐ Accept the merge request.

Major release:

1 ☐ Create the branch **vXpY** from the main.

2 ☐ Update the `NEWS.txt`. Replace main section by section **vXpY**.

3 ☐ In `documentation/data/TelemacDoc.cls` change the command `\telmaversion` to **vXpY**.

4 ☐ In `sources/utils/special/declarations_special.f` change the parameter VERSION to **vXpY**.

5 ☐ In `configs/systel.edf.cfg` change version to **vXpY**.

6 ☐ Run `compile_telemac.py --clean --rescan`. Update cmdf on main if necessary.

7 ☐ Run `validate_telemac.py`.

8 ☐ Run `validate_telemac.py --notebook --notebook-update`. Update content on the branch and also on main if necessary.

9 ☐ Run `doc_telemac.py`.

10 ☐ Add all the generated documentation to repository.

11 ☐ Run `damocles.py --eficas`.

12 ☐ Add the `scripts/python3/eficas` folder to the repository.

Minor release:

1 ☐ In the branch **vXpY** update `NEWS.txt` with release date.

2 ☐ Rerun Validation, documentation, eficas if some modifications where done to examples, documentation, dictionaries.

3 ☐ Create the tag **vXpYrZ**.

[1] CHAULIAC Christian and MONDON Christian. Guide d'application de la révision 1 de la décision commune 2005-03. *SEPTEN Reference D305913000993*.

[2] GUISNEL Françoise. Les actions de management de projet applicables à EDF R&D et leur modulation. *Eureka H-D03-2011-01748-FR*.

[3] S. MANDELKERN. Guide pour l'écriture d'un plan qualité logiciel. *HP-77-02-021-A*.

[4] ISSA Reza. Capitalisation des éléments numériques. *Veol*.