TELEMAC-MASCARET SYSTEM

Software Quality Plan

Yoann Audouin

Version v7p3
March 16, 2018

Contents

1. Goal, domain of application and responsibilities

1.1 Introduction

TELEMAC-MASCARET 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. TELEMAC-MASCARET SYSTEM is managed by a Consortium of core organisations: Artelia (formerly Sogreah, France), BundesAnstalt für Wasserbau (BAW, Germany), Centre d'Etudes et d'expertise sur les risques, l'environement, la mobilité et l'aménagement (CEREMA, France), Daresbury Laboratory (United Kingdom), Electricité de France R&D (EDF, France), and HR Wallingford (United Kingdom).
TELEMAC-MASCARET 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 the access to TELEMAC-MASCARET SYSTEM for the whole community of consultants and researchers, the choice of open source has been made. Anyone can thus take advantage of TELEMAC-MASCARET SYSTEM and assess its performances, and will find necessary resources on this website. However the quality of assistance, maintenance and hotline support are also very important to professional users, and a special effort has been made to offer alternatively a broad range of fee-paying services.
This Quality Plan describes the general disposition in place to assure the Quality of TELEMAC-MASCARET SYSTEM and its services. The procedure associated with this Quality Plan are described in complementary documents.

1.2 Description of the software

TELEMAC-MASCARET SYSTEM is composed of several modules including numerical models and resolution algorithms useful for its utilisation.
TELEMAC-MASCARET SYSTEM and its tools are open source, under GPL license, with the BIEF finite element library under LGPL. It is available from the TELEMAC-MASCARET SYSTEM website (`www.opentelemac.org`). This website is the main access to TELEMAC-MASCARET SYSTEM information and documentation.
TELEMAC-MASCARET SYSTEM can use external software and libraries. This Quality Plan does not cover those external elements (MPI, Metis, MED...), it only covers the function using them in the code.
This quality plan should guarantee the inter-operability between TELEMAC-MASCARET SYS-

TEM modules and software used for pre- and post-processing (such as Fudaa, Salomé...). The minimum requirements about inter-operability are to follow the same standard as any development in TELEMAC-MASCARET SYSTEM (Appendix C).

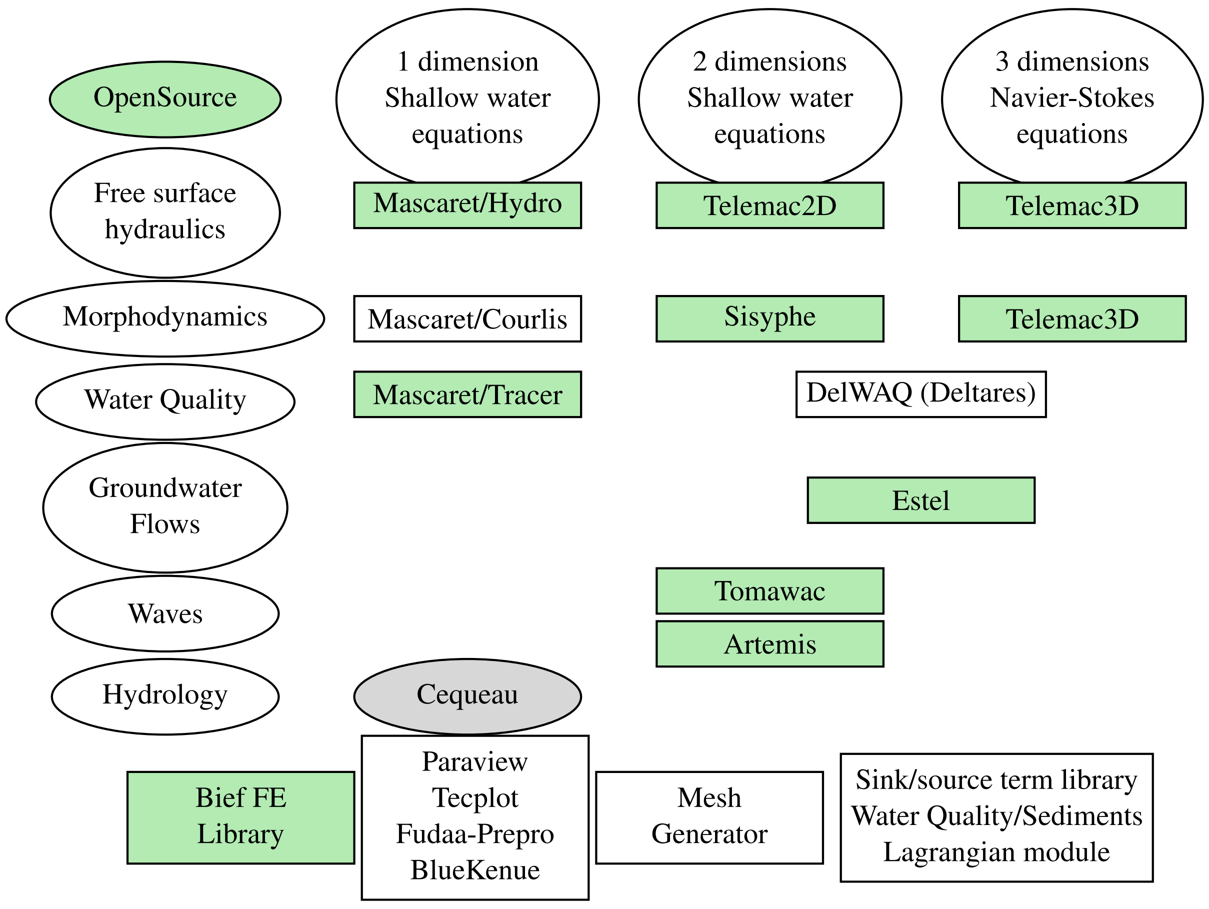

Figure 1.1: Codes in TELEMAC-MASCARET SYSTEM

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

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

1.4.1 Level 1 handler

The level 1 is for the people in charge of the project. The Level 1 Quality Handler (QH1) is the TELEMAC-MASCARET 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 changes necessary. He has to verify that the action applied complies with the SQP.

1.4.2 Level 2 handler

The level 2 is for the preservation of the quality of the products and activities of TELEMAC-MASCARET SYSTEM. The Level 2 Quality Handler (QH2) is the Head of the Department (HD) hosting the TELEMAC-MASCARET SYSTEM project. In particular, he verifies and validates the new versions of the SQP. He can delegate this role to the Delegate Head of Department (DHD) or to the Head of the Group (HG) hosting the TELEMAC-MASCARET SYSTEM project.

1.4.3 Level 3 handler

The level 3 is for people with an external role to the project. The Level 3 Quality Handler is the Strategy Handler (SH) of the TELEMAC-MASCARET SYSTEM project.

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.

Any modification must be validated by the Consortium.

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 both the QH1 and the QH2. This derogation is traced in a document with the Software Quality Plan. Otherwise a modification of the SQP can be asked by the QH2 or initiated by the QH1 so the SQP could be applicable again.

2. Terminology

2.1 Glossary

Module: It is one of the element of the code TELEMAC-MASCARET SYSTEM. There are 10 of them: TELEMAC-2D, TELEMAC-3D, TOMAWAC, ARTEMIS, SISYPHE, POSTEL-3D, BIEF, STBTEL, ESTEL-3D and MASCARET.
Post processor: Tools to visualize results from a TELEMAC-MASCARET SYSTEM run.
Source file: List of the file used for a feature. Those files can be Fortran code, Python, Perl...

2.2 Abbreviation

CIS: Continuous Integration System.
ADEPHE: TELEMAC-MASCARET SYSTEM developers meeting.
GUI: Global User Interface.
SQP: Software Quality Plan.
QH2: Level 2 Quality Handler.
SH: Strategy Handler.
HD: Head of department.
DHD: Delegate head of department.
HG: Head of the Group.
PM: Project Manager.
CH: Commit Handler.

3. The proceeding in 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 (Reference [2]) states the requirements needed in 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 steering process in EDF. 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." (in French in the text). It also describes the processes and actions keeping the quality of the software and its continuous amelioration.
The document [4] was also used as a guideline to write this Software Quality Plan.
The document [3] helps categorize numerical developments by dividing them in 4 categories and giving for each of them the steps to follow to capitalize them.
The document [1] if not specific for TELEMAC-MASCARET SYSTEM gives good guidelines on how to handle Numerical Softwares.

3.1 Documents specific to TELEMAC-MASCARET SYSTEM

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

- The version sheet, which contains the list of the elements associated with a version;
- All the documents referenced in the SQP given in appendix: the development plan, the organisation of the TELEMAC-MASCARET SYSTEM activity, the nominative list of people in TELEMAC-MASCARET SYSTEM.

4. Organisation of the TELEMAC-MASCARET SYSTEM activity

4.1 Structure of the TELEMAC-MASCARET SYSTEM activity

4.1.1 General Organisation

The TELEMAC-MASCARET 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 TELEMAC-MASCARET SYSTEM activities

The TELEMAC-MASCARET 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 Appendix C for definition of those notions):

- **Development activity**: concerns any modification on the code, its documentation and the associated tools, either it is an evolution, a correction, a reorganisation.
- **Verification and validation activity**: every development must be verified by the verification and validation tests. A major version or production version (See Appendix C) must run all the cases.
- **Exploitation activity**: concern the distribution, formation, support, maintenance of the different version and the 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-MASCARET SYSTEM " in Appendix A.

4.3 Development activity

A development represent any intervention on the source code of TELEMAC-MASCARET SYSTEM, its documentation or its tools (i.e. any element under the configuration, see the development plan in Appendix C). 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 he had (performance, strength, precision, adaptation to demands ...) or to add new functionalities to the software to answer new demands or requests.
- **Corrective maintenance**: The corrective maintenance removes a bug in the code or the documentation.
- **Adaptive maintenance**: The adaptive maintenance adapts the software when its environment changes (Operating system, compilers, new type of clusters...) to insure the software is still running on that environment.

Those three categories imply an integration process in the development version of TELEMAC-MASCARET 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 Appendix C

4.3.1 Processes, tools and rules

The processes to follow for each development activities are described in the development plan in Appendix C. Following those processes ensure traceability, non-regression, portability, verification, validation and re-usability of the functions developed and integrated in TELEMAC-MASCARET 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 a ticket on the TELEMAC-MASCARET SYSTEM project manager CUE (`http://cue.opentelemac.org`) may it be a evolution, corrective or adaptive maintenance in order to organize and prioritize development made in TELEMAC-MASCARET SYSTEM.
All those development 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 Appendix C.
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 Appendix C

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 (even a bit redundant) with the test realised during the implementation (See the development plan in Appendix C). We distinguish them as follow:

- **Verification**: The verification aims to establish that the code solves efficiently the mathematical problem he 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 the verification is using an analytical solution of the model or a solution based on measurements with the possible addition of source terms to the equation. 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 an 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 simplify to get access to experimental measurements) composed of multiple physical phenomenon.

- **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 TELEMAC-MASCARET 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 Appendix C
The release of a new major version of TELEMAC-MASCARET SYSTEM is decided by the chief of department hosting the TELEMAC-MASCARET SYSTEM project. The list of what is concerned by this new version is written by the CHs in the version note. This note describes:

- The code elements that compose the software,
- The documentation,
- The code environment,
- The major modifications since version $n-1$.

This version note formalises the quality insurance of the major version.
For the minor version release of TELEMAC-MASCARET SYSTEM the creation of this version note is not necessary instead the information about the modifications in the code are written in the release note. Even though the minor version still has to follow the quality processes followed by a major version.
The development version of TELEMAC-MASCARET SYSTEM is not formally released. Never the less it is compiled and a few test cases are run nightly. The last development version is available on the TELEMAC-MASCARET SYSTEM svn (`http://svn.opentelemac.org/svn/opentelemac/trunk`).
The process for the removal of older stable version is described in the development plan in Appendix C.

4.5.2 Distribution

Distribution in EDF R&D
The major and minor versions of TELEMAC-MASCARET SYSTEM are installed on the NOE server on an area dedicated to the project. The DSPIT handles the backup of those data.
The test cases are available (at least in reading) on the NOE server. The validation and verification is made on an external server based in H.R.Wallingford (UK). The environments on which the code was verified and validated are listed in the version note.
Distribution in EDF
The distribution outside of the R&D is under the responsibility of the TELEMAC-MASCARET SYSTEM Chain Handler. The environment guide written by the Chain Handl. which contains an help for the installation of the code.
Open source distribution
All the stable version of TELEMAC-MASCARET SYSTEM, the development version and the tools are under open source license (GPL for TELEMAC-MASCARET 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 TELEMAC-MASCARET SYSTEM.
The open source version are tested with on Windows 7 and on the following Linux Dristribution: Opensuse, Fedora, Ubuntu, Debian. It is also tested using the NAG Compiler, the intel Compiler and the gnu compiler.
Protection
The software is protected by the label TELEMAC since 05/02/96 with the number INPI 96/623 748.

4.5.3 Relation with the TELEMAC-MASCARET SYSTEM users

The services handled by the TELEMAC-MASCARET 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-MASCARET 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. A ticket must then be created on the development project manager CUE (`http://cue.opentelemac.org`). The formalisation of those demands allows:

  - A better repartition of resources by the project.
  - The follow up by the user of the demand and allows him to interact with the developer handling the demand.

- **Second level assistance**
  This assistance is made by the development team of the project TELEMAC-MASCARET SYSTEM. It is an anticipated action aimed to create documentation and didactic software (user and developer documentation, online documentation, tutorials, FAQs...).

The assistance activity can lead to a ticket on the CUE that must follow the development rules established in the development plan in Appendix C, if the problem is deemed worth it.
The actors of this activity are:

- The 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.
- The people allowed to submit development demands more specifically the development team (See the development plan in Appendix C): They answer to the forum, the CUE. They subcontract some of the work then they handle the dialog with the subcontract.
- The Kernel Handler: handles the CUE, its maintenance and its good use.

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

- **Organisation of the TELEMAC-MASCARET SYSTEM User Conference**: The organisation of the conference is explained in Appendix B.6.2 and in the document describing the organisation of the TELEMAC-MASCARET SYSTEM activity in Appendix B.

4.5.4 Relation with partners

The partners (like for example the members of the TELEMAC-MASCARET SYSTEM consortium) must follow the same rules for the development as described in the development plan in Appendix C.

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 Appendix C. 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. Documentation

The documentation handled by this Software Quality Plan is divided in two part:

- The managing documentation of the TELEMAC-MASCARET SYSTEM activity;
- The technical documentation associated with the different version of TELEMAC-MASCARET SYSTEM.

5.1 Software Quality Plan and additional documents

They are the documents defining the organisation and the processes followed in the TELEMAC-MASCARET SYSTEM project. They are available in Eureka and on the TELEMAC-MASCARET SYSTEM website (`http://www.opentelemac.org`). It includes the Software Quality Plan, the development plan, the nominative list of the people in TELEMAC-MASCARET SYSTEM, the organisation document.

5.2 Technical documentation

The documentation is usually divided into 5 parts:

- The reference manual, which describes every keyword of the dictionary of one code,
- The principle note which describes the physical phenomena modelled by the module and the numerical methods used to solve the equations modelling these phenomena.
- The validation document, which presents a list of test cases; they validate the conceptual model, algorithms and software implementations,
- The user manual, which describes how to use the code,
- The release note which lists the new features of the version,

Usually, every manual is to be written for every module of the code.
Another documentation exists for all sources of the code, that is the Doxygen documentation. It contains informations on all the elements of the source code. It is generated by the Python environment and is available on the website `docs.opentelemac.org`.
A documentation listing all the environment commands and giving a short description is also available.

6. Configuration handling

6.1 Elements of the configuration

The elements under configuration regroups all the elements mandatory to the construction, development, maintenance, verification and validation and a part of the technical documentation in a version of TELEMAC-MASCARET SYSTEM.
In addition to those elements, the configuration of a version takes also into account the environment on which it was verified and validated.
A version note is given for each major version of TELEMAC-MASCARET SYSTEM and is available on the website (`http://www.opentelemac.org`).

6.2 Identification of the software elements

The software element composing a version of TELEMAC-MASCARET SYSTEM are listed in the version note for each major version.
The different types of version, how they are created and the planning are defined in the development plan in Appendix C.

6.3 Identification of the documentation elements

The elements of the documentation that are included in the configuration are listed in the development plan in Appendix C.
The documentation associated with a version of TELEMAC-MASCARET SYSTEM contains the name of the version "*vxpy*" in its name.

6.4 Handling software configuration

This process is described in the development plan in Appendix C.

6.5 Handling documentation configuration

This process is described in the development plan in Appendix C.

7. Application of the Software Quality Plan

7.1 Principle

The guarantee of quality is made continuously through the action the **Level 1** and **Level 2** Quality Handlers without the intervention of an outside Quality handler. The actions they apply rely on the Software Quality Plan and its additional documents.

7.2 Quality intervention on the development step

Level1 interations are on the creation and application of the processes defined in the development plan. **Level1** interactions give technical advice and can ask for additional information, occasionally with the help of experts in the domain concerned.
Level2 interactions concern the application of those processes and can ask for internal verification.

7.2.1 Quality insurance qualification

The qualification of the quality of a major version is insured by the version note signed by the CHs and the chief of department hosting the TELEMAC-MASCARET SYSTEM project.

A. Nominative list of handlers

Name of the project TELEMAC-MASCARET SYSTEM for the period 2010-2014: PHE (Hydro-environmental platform), E961/P10VU0

Main responsibilities			
Name	**Function**	**First name, Last name**	**Unit**
CMD	Sponsor of the TELEMAC-MASCARET SYSTEM project	Francois VERDIEL	R&D/ED
SH	Strategy Handler of the TELEMAC-MASCARET SYSTEM project	Didier LE-REVEREND	R&D/ED
PM	Manager of the TELEMAC-MASCARET SYSTEM project	Nicole GOUTAL	R&D/LNHE
QH2	Level 2 Quality Handler of the TELEMAC-MASCARET SYSTEM project	Jean-Daniel MATTEI	R&D/LNHE
CH1D	Commit Handler for the 1D part of the TELEMAC-MASCARET SYSTEM project	Fabrice ZA-OUI	R&D/LNHE
CH2D	Commit Handler for the 2D part of the TELEMAC-MASCARET SYSTEM project	Riadh ATA	R&D/LNHE
CH3D	Commit Handler for the 3D part of the TELEMAC-MASCARET SYSTEM project	Chi-Tuân PHAM	R&D/LNHE

Main responsibilities			
Name	**Function**	**First name, Last name**	**Unit**
Chain hand.	Software chain handler	Yoann AUDOUIN	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	Nicole GOUTAL	R&D/LNHE
Com hand.	Communication handler	Nicole GOUTAL	R&D/LNHE
Salome hand.	SALOME handler	Yoann AUDOUIN	R&D/LNHE
UI hand.	User Interface handler	To be defined	

B. Organisation of the TELEMAC-MASCARET SYSTEM activity

B.1 Organisation

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

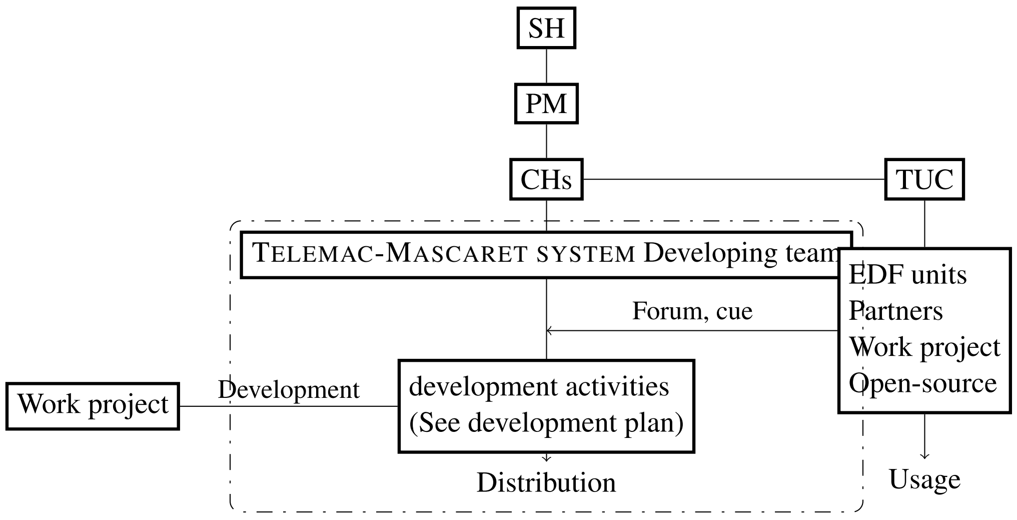

Figure B.1: General Organisation of the TELEMAC-MASCARET SYSTEM activity

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

B.2 Quality handler

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

The Table B.2 and the Table B.3 are describing the role of the different actors of TELEMAC-MASCARET SYSTEM. "Project" here represents "the TELEMAC-MASCARET SYSTEM project".

B.4 Coordination

Job	Roles
SH	• Strategy handler for the project. • Role given by the project sponsor. • In charge of the project strategical decisions. • Must keep the coherence of the project with the rest of the firm activities.
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 (CASP2). • 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 SH and the resources manager). For example organises the COPIL (Steering committee) of the project where are shown the technical and budget advancement 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.
QH2	• Level 2 Quality Handler of TELEMAC-MASCARET SYSTEM. • Signs the version note and the configuration note.
CH1D CH2D CH3D	• Organises the software development. • Keeps informed the SH and PM of the activities evolution. • In charge of the Level 1 Quality by delegation from the PM.

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

B.5 TELEMAC-MASCARET 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 CUE tickets. • In charge of the ADEPHE meetings. • Has the final decision on the integration of developments. • Delegates those tasks to the CHs if necessary.
CH1D CH2D CH3D	• 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-MASCARET 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 CUE, 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-MASCARET 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-MASCARET SYSTEM documentation. • Controls the modifications made to the documentation. • Insures the coherence of the documentation.
Com hand.	• In charge of the communication plan of TELEMAC-MASCARET SYSTEM. • In charge of the TELEMAC-MASCARET SYSTEM website and the coherence of its content. • In charge of the presentation, posters and fliers.
Salome hand.	• In charge of the integration of the TELEMAC-MASCARET SYSTEM module in the SALOME platform. • Makes the link with the SALOME community, goes to the ADS (Salome developer meetings).
UI hand.	• In charge of the Global User Interface.

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

B.6 Meeting of the TELEMAC-MASCARET SYSTEM project

The steering of the TELEMAC-MASCARET SYSTEM project is discussed in the ADEPHE meetings.

The TELEMAC-MASCARET SYSTEM User Conference does not have a steering function but gathers the TELEMAC-MASCARET SYSTEM community.

B.6.1 ADEPHE

Developer Meeting (ADEPHE)	
Actions	• Presentation of the news and evolution of TELEMAC-MASCARET SYSTEM. • Validation by the PM and the CHs of the integration of a development.
Actors	• PM, CHs, all the Hand. (Mandatory). • Members from the development team. • Any candidate for a development. • Anyone willing to keep in touch with TELEMAC-MASCARET SYSTEM evolution.
Agenda	• **In charge**: PM and CHs. • **Diffusion**: The agenda is e-mailed by the PM.
Animation	• PM and CHs.
Report	• PM and CHs.
Periodicity	every two month

B.6.2 TUC

TELEMAC-MASCARET SYSTEM User Conference (Club-U)	
Actions	• Promote study with high added value for the code TELEMAC-MASCARET SYSTEM. • Present the new versions and their new functionalities. • Share of experience between users. • Share of information between users and developers.
Actors	• PM (mandatory). • CHs (mandatory). • SH. • Members from the development team. • Any user interested by TELEMAC-MASCARET SYSTEM.
Agenda	• **In charge**: A member of the consortium (rotating every year). • **Diffusion**: A member of the consortium (rotating every year).
Animation	• The member of the consortium in charge.
Proceedings	• The member in charge of the consortium in charge.
Periodicity	Annually

C. TELEMAC-MASCARET SYSTEM development plan

The development plan describes explicitly all the processes on the software during the life cycle of the TELEMAC-MASCARET SYSTEM code and insures its Quality of the software.
The TELEMAC-MASCARET SYSTEM activity is decomposed into three main activities from the creation of a version, to its release and finally its removal:

- **Development activity**: Concerns any intervention on the source code of TELEMAC-MASCARET SYSTEM, its documentation, its associated tools may it be for a corrective, adaptive or evolution maintenance.
- **Verification and Validation activity**: Every development must go through the verification and validation process. A major version must go through the full validation before its release.
- **Release activity**: Concerns the diffusion, support and maintenance of the different versions of the code.

The development plan describes the action having a direct impact on the source code. The other action are described in the SQP.

C.1 Rights to commit new developments

People intervening in the development of TELEMAC-MASCARET SYSTEM can be divided in the following groups:

- **The TELEMAC-MASCARET SYSTEM development team (TDT)**: To be a part of the TDT you need to have writing access on the subversion repository. To be in the development team, the developer must have followed the TELEMAC-MASCARET SYSTEM workshop and know 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-MASCARET SYSTEM project. It is recommended that they follow the TELEMAC-MASCARET 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-MASCARET SYSTEM workshop. The follow-up of the development will have to be made by a members of the TDT.

- **Open source community, academic and industrial partners**: TELEMAC-MASCARET SYSTEM is an open source code under GNU-GPL. Anyone can get the source and develop his own version. But for a development to be integrated in the official version, it must be studied by the TDT and follow the process described in C.3. If the integration is validated, the delivery of the evolution for the code must go through a member of the TDT (See C.5).

C.2 Life cycle of a development

C.2.1 Definition of a development

A development represents a modification on the source code of TELEMAC-MASCARET SYSTEM, its documentation and its dedicated tools. (i.e. everything that is under the configuration, see C.6). The developments are categorised in 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...).
  - Develops new functionalities to satisfy new needs or demands.

  Those 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. Every maintenance must be joined with a CUE ticket. This ticket will allow the PM to have a general view of the work in progress in the code. According to the size of the development, it can even be presented by the developer in an ADEPHE meeting, this will allow the TDT do evaluate the impact of the development and maybe advice 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 the documentation. A bug 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 CUE ticket.

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

Each of those activities will be tied to a CUE ticket. The Type of CUE Ticket will be determined by the kind of activity (Bug, Feature, Documentation...). Every bug ticket should be resolved before the release of a new version.

C.2.2 Life cycle of a TELEMAC-MASCARET SYSTEM development

The life cycle of a TELEMAC-MASCARET SYSTEM development follows a list of 7 steps shown on the Figure C.1. Those steps will be described in the next section.

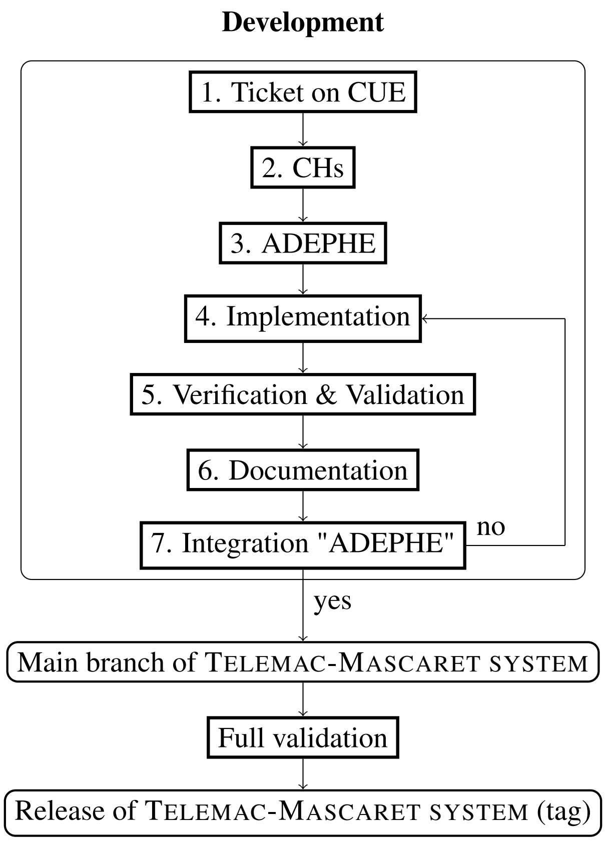

Figure C.1: Life cycle of a TELEMAC-MASCARET SYSTEM development

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. Job repartition step.
3. Examination and impact study step.
4. Implementation step.
5. Verification step.
6. Documentation step.
7. 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.

C.3 Description of the development steps

C.3.1 Proposal step

Every demand must go through the proposal step may it be a corrective, adaptive, evolution maintenance (See section C.2.1 for a definition).

This proposal is given to the TDT with a ticket on the CUE (`cue.opentelemac.org`).
The goal of this proposal is to:

- Inform the CHs of the development proposed.
- Give more information on the impact study.
- Plan and organise the resources for the development.
- To prepare the validation in ADEPHE if necessary.

In charge: The proposer.
Documents: The CUE ticket.

C.3.2 Job repartition step

The job repartition step is made by the CHs in collaboration with the PM.

- If the proposal is not in a specific project, the CHs can propose to have it billed by the TELEMAC-MASCARET SYSTEM project. If the proposal is specific to a work project the CHs can transfer the proposal to that project, if the proposal is transfered the same tools (CUE, CIS...) will be used by the new project.
- If the proposal is bound to a project, the resources for the development will come from that project, the TELEMAC-MASCARET SYSTEM project can still allocate some resources in order to help, guide and give advice to the developer.

In charge: The CHs.
Documents: Update of the CUE ticket.

C.3.3 Examination and impact study step

When the cue ticket is published, the CHs have to determine if the development must be followed by the TDT and presented in ADEPHE. This mainly concerns evolution maintenance but can also concern corrective maintenance if it has a major impact on the code.
The goal of the study impact is:

- If it is a 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 GUI and the validation and Verification process.
- To help the developer in its choice of algorithm and implementation.
- To discuss the test cases for Validation and Verification.
- To estimate the impact on the documentation.

If the development is heavy on the code, multiple ADEPHE can be used to follow the development.
In charge: CHs, TDT, handler of the development.
Documents:

- INPUT: The CUE ticket.
- OUTPUT: Reports after each ADEPHE.

C.3.4 Implementation step

This is the coding part of what was specified in the CUE ticket, with the modifications that the discussion during the ADEPHE could have generated.
The developer must follow the coding convention given in the developer guide of TELEMAC-MASCARET SYSTEM (Available in Appendix D). All the developments are made under a configuration handling tool (SVN), for each development a branch is created dedicated to that development.
The control of the implementation is made through tests:

- Evolution maintenance:
  - Unit tests (Non-exhaustive list):
    - Parallelism testing.
    - Continued computation testing.
    - Post-treatment testing.
    - GUI testing.
  - Verification test: one or more depending of what is proposed by the developer (seen in ADEPHE, if there was one).
  - Validation test: one or more depending of what is proposed by the developer (seen in ADEPHE, if there was one).
- Corrective or adaptive maintenance:
  - Old unit test if impacted.
  - Verification and validation test.

The Unit tests are for the developer to create.
The development must be made on a branch and follow the development version (or trunk, see Section C.6 for a definition) using a configuration handling tool (SVN is used in Edf). 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 doing it at the end of the development. The Figure C.2 shows what the evolution of the development should look like.

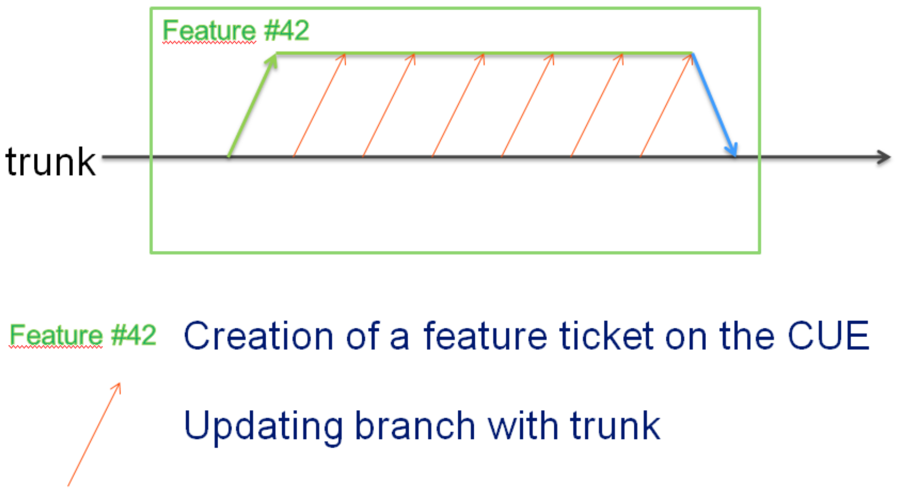
Feature #42
trunk
Feature #42
Creation of a feature ticket on the CUE
Updating branch with trunk

Figure C.2: Life cycle of a development branch

It is also recommend to use developing tools during the development like a debugger (gdb, totalview), valgrind for memory leak.
Every development must be tested both in serial and parallel configuration and the GUI must be updated accordingly.
The CUE ticket is updated as the implementation goes on.
In charge: The developer.
Documents:

- INPUT: cue ticket, developer manual, verification and validation test cases.
- OUTPUT: source code, test cases (with xml), result of tests.

C.3.5 Verification step

The Verification and Validation process is deeply linked with the implementation process. This process aims to verify the code on the algorithm level and the whole model using the validation and verification test cases (even if a small part of the code was 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 as distinguish as describe below:

- **Verification**: The verification aims to establish that the code solves efficiently the mathematical problem he 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 the verification is using an analytical solution of the model or a solution based on measurements with the possible addition of source terms to the equation. The process is then most of the time to compared 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 phenomenon.

As mentioned in the previous paragraph every development must contain validation and verification test cases:

- Evolution maintenance: For an evolution of the code at least one verification and one validation case must be added if existing cases can not be used. The choice of those cases are discussed in ADEPHE.
- Corrective and adaptive maintenance: All the test cases must be rerun.

In charge: The developer.
Documents:

- INPUT: cue ticket, validation manual, ticket for each case.

- OUTPUT: update of cue ticket, LaTeX documentation follow the validation standard for each test case.

C.3.6 Documentation step

The TELEMAC-MASCARET SYSTEM technical documentation is composed of the following manuals:

- **Reference Manual**: describes every keyword in the dictionary of one module.
- **Principle Note**: 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.
- **Release Note**: lists the new features of the version.
- **Environment Guide**: gives a list of the environment commands.
- **Online Manual(Doxygen)**: This documentation is linked with the source code (it is build by using special comment made in the source code), it describes the functions and structures of the source code.
- **Installation Manual**: describes the process to follow to install the TELEMAC-MASCARET SYSTEM code.
- **Developer Manual**: describes all the actions a developer might have to do during a development it also defines the programming convention.

The first five manuals are duplicated for each of the modules of TELEMAC-MASCARET SYSTEM. The documentations are under configuration handler (svn) in LaTeX format. The last two documentation are available on the TELEMAC-MASCARET SYSTEM website. 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 principle note describing the new physics and in the user manual showing how to use that functionality. The Online documentation must be updated as well.
- **Corrective, adaptive Maintenance**: If the development has an impact on the documentation, it must be updated by the developer.

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 Hand.
Documents:

- INPUT: All the documentation.
- OUTPUT: The updated documentation.

C.3.7 Integration step

The integration step concerns the restitution of a development into the main branch of TELEMAC-MASCARET SYSTEM (trunk, see C.6).
The integration step is composed of at most 4 steps, the presentation in ADEPHE being optional:

- Demand for a presentation in ADEPHE, the demand is made through the CUE ticket.
- Presentation of the development and the verification case in ADEPHE.
- 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 Hand.

Not following those steps allows the CHs to refuse the integration and even remove it from the development version. We describe below the presentation in ADEPHE and the integration process.
Presentation of the development in ADEPHE
The ADEPHE meeting are bi-monthly. Must go to an approbation in ADEPHE:

- Any evolution maintenance out of the TELEMAC-MASCARET SYSTEM project.
- Important modification in the code with the approbation of the CHs.

To be allowed in ADEPHE, a developer must have:

- Synchronize its branch with the latest version of the main branch (trunk).
- Verify that the code follows the programming rules and that the unit test are working.
- The new test cases added by the development must pass through the validation tool (CIS).
- Update the CUE ticket.

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

- The development and how it was implemented: the functionalities if it is a evolution maintenance, in case of a corrective, adaptive maintenance: the data structure, the new keywords, how to use it.
- The modifications in the GUI if they are any.
- 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 trunk.
- 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 CHs have the final word.
In charge: CHs, TDT.
Documents:

- INPUT: CUE ticket, documentation.
- OUTPUT: ADEPHE report, update of cue ticket.

Integration
After the ADEPHE meeting the person in charge of the integration has to do the following actions:

- Integrate the verification and verification cases in the cases database.
- Push the development and the documentation modifications into the TELEMAC-MASCARET SYSTEM main branch.
- Close the CUE ticket.

In charge: TDT.
Documents:

- INPUT: CUE ticket.
- OUTPUT: update of CUE ticket, documentation updated.

C.4 Maintenance

In this part the maintenance concerns only the corrective and adaptive maintenance.
The TELEMAC-MASCARET SYSTEM project is in charge of the non-regression of the TELEMAC-MASCARET SYSTEM functionalities on the Verification and Validation test base.
The TELEMAC-MASCARET SYSTEM project is also in charge of the adaptive maintenance (see Section C.2.1).
The project in charge of other developments is in charge of the corrective maintenance for their developments. But an 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 C.2.2 and summarised in Figure C.1.
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 trunk, it is not retroactive.
The removal of a functionality concern:

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

C.5 Integration of development from the open-source community

The TELEMAC-MASCARET SYSTEM project (CHs and PM) is in charge of the evaluation of development from the open source community.
Two cases can occur:

- A CUE ticket is posted.
- A member or entity of the open source community has developed a new functionality and wants it to be included in the official version maintained by EDF. A CUE ticket could have been made.

To be a potential candidate for an integration in the development version of TELEMAC-MASCARET SYSTEM, a development from the open source community must:

- Be generic enough to be useful to the entire community. For example the development must not be bound to an industrial application.
- Impact as little as possible the structure of the code. And have a good modularity so it can be deleted when the maintenance is not possible anymore.
- Not be bound to a copyright, the code being under GNU GPL the development must be under the same license.

If the development passes those three criteria then the integration is given to a member of the TDT who becomes in charge of that development.
Then the development must follow the same processes, described in section C.2.2, as any other developments.

C.6 Configuration handling

C.6.1 Tools of configuration handling

The different versions of the elements of the TELEMAC-MASCARET SYSTEM software are handled by a version control tool. Those tools allow to track the evolution and ease the collaboration between multiple developers. Nonetheless, in addition to those tools the source are saved on the EDF network NOE.
The version control tool used is Subversion (SVN) which allows to rebuild any version. The subversion repository is based at the following address: `http://svn.opentelemac.org/svn/opentelemac`.
For each stable version a configuration note signed by the CHs and the RQ2 is published. It precises:

- The elements concerned.
- The history of the configuration.
- The organisation of spaces.
- The creation of the stable version.
- The saving and storing processes.

C.6.2 Elements of the configuration

The elements of the TELEMAC-MASCARET SYSTEM software concerned by the configuration control are given in the configuration note.
The source code, the environment script, the documentation ...

C.6.3 Type of version

TELEMAC-MASCARET SYSTEM official or reference version

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

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

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

Creation of the reference version for TELEMAC-MASCARET SYSTEM

The creation of the reference version is described in the configuration note. The process will be describe here as well:

The TELEMAC-MASCARET SYSTEM version are handled using "tags".

- Creation a tag is decided by the Chain Hand. and the CHs.
- Tags are created for every minor version.
- Tags are named in the format **x.y.z** where:
  - **x** is the major version index, incremented when a major modification is made 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 version are divided into branches and tags. There is:

- **The main branch** (trunk), also called the development version. It represents the work in progress between two stable version. It is where the different maintenance are integrated. Figure **??** shows the evolution of the main branch between two stable versions.
- **The version branch** ($x.y$), they are created every new major or minor version. They are used only for corrective and adaptive maintenance. They have been through the complete validation process.
- **The stable version tag** ($x.y.z$), the tags are made from the version branch ($x.y$) those tags are fixed they will not be modified. If a corrective or adaptive maintenance is necessary the changes will be made in the version branch and a new tag ($x.y.(z+1)$) will be created. Figure C.4 shows the creation of the $x.y$ and the tags created.

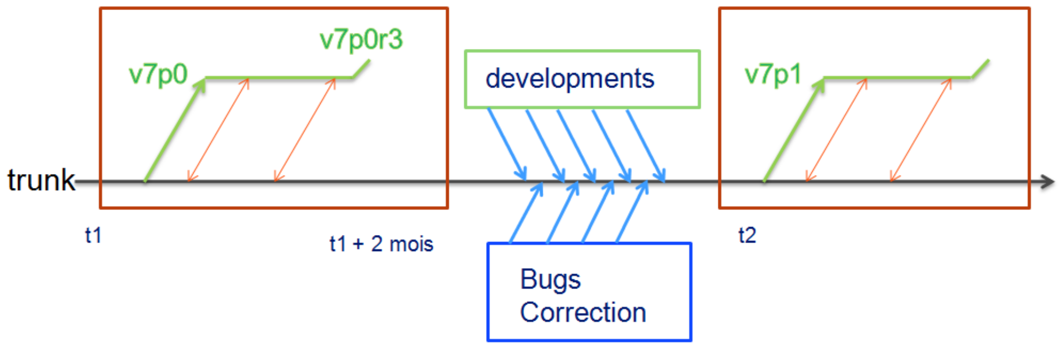

Figure C.3: Life cycle of the TELEMAC-MASCARET SYSTEM version

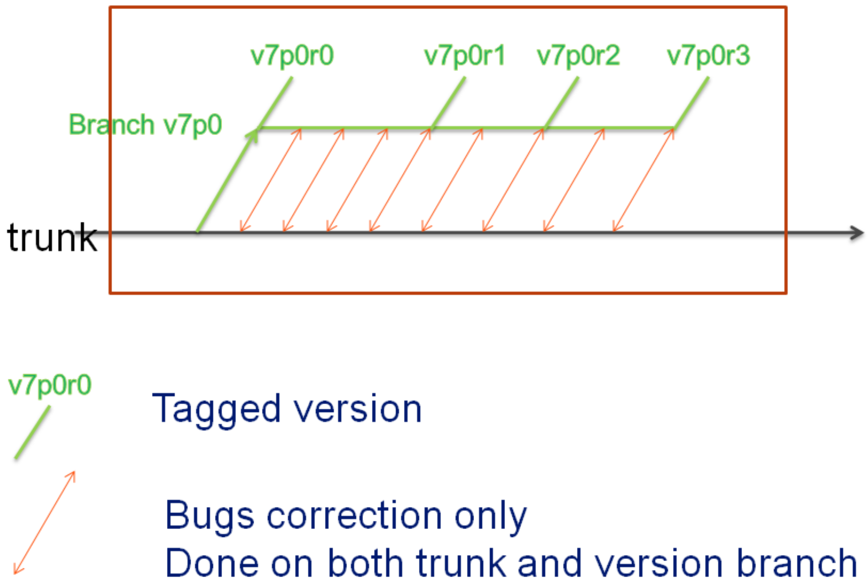

Figure C.4: Life cycle of a version branch

Evolution maintenances are not merged on the version branch they are only merged on the trunk.

C.6.4 Version schedule

Version release

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

- The major version are out every 4-5 years.
- The minor release are out every year, if the amount of new development is considered acceptable, and is maintained until the release

Finally at an instant *t* in time we have the following version:

- The development version: It is the main branch (trunk).
- The latest stable version $x.y.z$

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.

Version maintained
Only last two minor version are maintained.
Exploitation version
To do a study with TELEMAC-MASCARET SYSTEM it is recommended to use the latest stable version. It should be the most advanced and least bugged version.

D. TELEMAC-MASCARET SYSTEM Coding Convention

D.1 Main rules

We give hereafter a number of safety rules that will avoid most common disasters. It is however highly recommended to THINK before implementing. The structure of your code and the choice of the algorithms will deeply influence: the manpower requested, the number of lines to write, the memory requested, the computer time. For a given task, differences of a factor 10 for these 4 items are common and have been documented (see e.g. "the mythical man-month, essays on software engineering" by Frederick Brooks). These differences will eventually result in "success" or "failure". So let the power be with you and just follow Yoda's advice: "when you look at the dark side, careful you must be".

D.2 Subroutine header

```
!                     ****************
                      SUBROUTINE METEO
!                     ****************
!
     &(PATMOS,WINDX,WINDY,FUAIR,FVAIR,X,Y,AT,LT,NPOIN,VENT,ATMOS,
     & HN,TRA01,GRAV,ROEAU,NORD,PRIVE)
!
!***********************************************************************
! TELEMAC2D   V6P2                                   27/07/2012
!***********************************************************************
!
!brief    COMPUTES ATMOSPHERIC PRESSURE AND WIND VELOCITY FIELDS
!+               (IN GENERAL FROM INPUT DATA FILES).
!
!warning  MUST BE ADAPTED BY USER
!
!history  J-M HERVOUET (LNHE)
!+        02/01/2004
!+        V5P4
!+
!
```

```
!history  N.DURAND (HRW), S.E.BOURBAN (HRW)
!+        13/07/2010
!+        V6P0
!+   Translation of French comments within the FORTRAN sources into
!+   English comments
!
!history  N.DURAND (HRW), S.E.BOURBAN (HRW)
!+        21/08/2010
!+        V6P0
!+   Creation of DOXYGEN tags for automated documentation and
!+   cross-referencing of the FORTRAN sources
!
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
!| AT,LT          |-->| TIME, ITERATION NUMBER
!| ATMOS          |-->| YES IF PRESSURE TAKEN INTO ACCOUNT
!| FUAIR          |-->| VELOCITY OF WIND ALONG X, IF CONSTANT
!| FVAIR          |-->| VELOCITY OF WIND ALONG Y, IF CONSTANT
!| GRAV           |-->| GRAVITY ACCELERATION
!| HN             |-->| DEPTH
!| NORD           |-->| DIRECTION OF NORTH, COUNTER-CLOCK-WISE
!|                |   | STARTING FROM VERTICAL AXIS
!| NPOIN          |-->| NUMBER OF POINTS IN THE MESH
!| PATMOS         |<--| ATMOSPHERIC PRESSURE
!| PRIVE          |-->| USER WORKING ARRAYS (BIEF_OBJ BLOCK)
!| ROEAU          |-->| WATER DENSITY
!| TRA01          |-->| WORKING ARRAY
!| VENT           |-->| YES IF WIND TAKEN INTO ACCOUNT
!| WINDX          |<--| FIRST COMPONENT OF WIND VELOCITY
!| WINDY          |<--| SECOND COMPONENT OF WIND VELOCITY
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
!
      USE BIEF
!
      IMPLICIT NONE
      INTEGER LNG,LU
      COMMON/INFO/LNG,LU
!
!+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!
      INTEGER, INTENT(IN)                :: LT,NPOIN
      LOGICAL, INTENT(IN)                :: ATMOS,VENT
      DOUBLE PRECISION, INTENT(IN)       :: X(NPOIN),Y(NPOIN),HN(NPOIN)
      DOUBLE PRECISION, INTENT(INOUT)    :: WINDX(NPOIN),WINDY(NPOIN)
      DOUBLE PRECISION, INTENT(INOUT)    :: PATMOS(NPOIN),TRA01(NPOIN)
      DOUBLE PRECISION, INTENT(IN)       :: FUAIR,FVAIR,AT,GRAV,ROEAU,NORD
      TYPE(BIEF_OBJ), INTENT(INOUT)      :: PRIVE
!
!+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!
```

```
!         LOCAL  DECLARATIONS
!
          DOUBLE  PRECISION  P0 ,Z(1)
!
```

D.3 The coding convention

- The code must pass Fortran 2003 Standard,
- A file must contain only one program/module/subroutine/function and must have the same name as that program/module/subroutine/function,
- All subroutines and functions must conform to the subroutine header given in the previous paragraph,
- All subroutines and functions must be protected by an IMPLICIT NONE statement. Their arguments types must be given with their INTENT,
- The order in declarations is free except than some compilers will not accept that an array has a dimension that has not been declared before, hence:

  ```
  INTEGER,  INTENT(IN)  ::  N
  DOUBLE  PRECISION,  INTENT(INOUT)  ::  DEPTH(N)
  ```

  is correct and:

  ```
  DOUBLE  PRECISION,  INTENT(INOUT)  ::  DEPTH(N)
  INTEGER,  INTENT(IN)  ::  N
  ```

  is not correct.
- Error messages: they must be given in French and English, using logical unit LU taken in COMMON block INFO. LNG = 1 is French, LNG = 2 is English. Parameterizing the listing logical unit is necessary because it is not always 6 in parallel, as the listing of slave processors does not appear on the screen but is redirected to files.
- Lines must be limited to a size of 72 characters, and only in UPPERCASE. Spaces must be only one blank, for example between a CALL and the name of a subroutine. This is to facilitate research of character string in source code,
- Indents in IF statements and nested loops are of 2 blanks,
- Tabs for indenting are forbidden. The reason is that depending on compilers they represent a random number of blanks (6, 8, etc.) and that it is not standard Fortran,
- Blank lines are better started by a “!”.
- Comments line should begin with a "!".
- Names of variables: a name of variable should not be that of an intrinsic function, e.g. do not choose names like MIN, MAX, MOD, etc., though possible in theory this may create conflicts in some compilers, for example the future Automatic Differentiation Nag compiler.
- Functions: intrinsic functions must be declared as such. Use only the generic form of intrinsic functions, e.g. MAX(1.D0,2.D0) and not DMAX(1.D0,2.D0). It is actually the generic function MAX that will call the function DMAX in view of your arguments, you are not supposed to do the job of the compiler.

D.4 Defensive programming

When programming, one has always to keep in mind that wrong information may have been given by the user, or that some memory fault has corrupted the data. Hence when an integer OPT may only have 2 values, say 1 and 2 for option 1 and option 2, always organise the tests as follows:

```
IF(OPT.EQ.1) THEN
  ! here option 1 is applied
ELSEIF(OPT.EQ.2) THEN
  ! here option 2 is applied
ELSE
  ! here something wrong happened, it is dangerous to go further, we stop.
  IF(LNG.EQ.1) THEN
    WRITE(LU,*) 'OPT=',OPT,' OPTION INCONNUE DANS LE SOUS-PROGRAMME...'
  ENDIF
  IF(LNG.EQ.2) THEN
    WRITE(LU,*) 'OPT=',OPT,' IS AN UNKNOWN OPTION IN SUBROUTINE...'
  ENDIF
  CALL PLANTE(1)
  STOP
ENDIF
```

D.5 Over-use of modules

Modules are very useful but used in excess, they may become very tricky to handle without recompiling the whole libraries. For example the declaration modules containing all the global data of a program cannot be changed without recompiling all subroutines that use it.
A common way of developing software in the TELEMAC-MASCARET SYSTEM system is to add modified subroutines in the user FORTRAN FILE. This will sometimes be precluded for modules as some conflicts with already compiled modules in libraries will appear.
A moderate use of modules is thus prescribed (though a number of inner subroutines in BIEF would deserve inclusion in modules).

D.6 Allocating memory

For optimisation no important array should be allocated at every time step, it is better to use the work arrays allocated once for all in TELEMAC-MASCARET SYSTEM programs, like $T1$, $T2$, etc., in TELEMAC-2D (note that they are $BIEF_OBJ$ structures, which brings some protection against misuses). If it cannot be avoided, an array allocated locally should be clearly visible and:

- Either allocated once and declared with a command SAVE,
- Or if used once only, deallocated at the end of the subroutine.

Automatic arrays are strictly forbidden. For example the array X in the following line:

```
DOUBLE PRECISION X(NPOIN)
```

If it is not in the subroutine arguments, while $NPOIN$ is. In this case it is a hidden allocation of size $NPOIN$, which may change from one call to the other. It is not standard Fortran, in worst cases it will cause a compiler to crash after a number of calls..

D.7 Test on small numbers

Always think that computers do truncation errors. Tests like: $IF(X.EQ.0.D0)THEN\ldots$ are very risky if X is the result of a computation. Allow some tolerance, like: $IF(ABS(X).LT.1.D-10)THEN\ldots$, especially if divisions are involved.

D.8 Optimisation

Optimisation is a key point, a badly written subroutine may spoil the efficiency of the whole program. Optimisation is a science and even an art, but it can be interesting to have a few ideas or tricks in mind. Here are a few examples:
Example 1: powers
The following loop:

```
DO I=1,NPOIN
   X(I)=Y(I)**2.D0
ENDDO
```

is a stupid thing to do and should be replaced by:

```
DO I=1,NPOIN
   X(I)=Y(I)**2
ENDDO
```

As a matter of fact, $Y(I)**2$ is a single multiplication, $Y(I)**2.D0$ is an exponential ($exp(2.D0*Log(Y))$), it costs a lot, and moreover will crash if $Y(I)$ negative.
Example 2: intensive loops with useless tests
Case 1: the following loop

```
DO I=1,NPOIN
   IF(OPTION.EQ.1) THEN
      X(I)=Y(I)+2.D0
   ELSE
     X(I)=Y(I)+Z(I)
   ENDIF
ENDDO
```

Should be replaced by:

```
IF(OPTION.EQ.1) THEN
   DO I=1,NPOIN
      X(I)=Y(I)+2.D0
   ENDDO
ELSE
DO I=1,NPOIN
      X(I)=Y(I)+Z(I)
   ENDDO
ENDIF
```

In the first case the test on $OPTION$ is done $NPOIN$ times, in the latter it is done once.
Case 2: the following loop

```
DO I=1,NPOIN
    IF(Z(I).NE.0.D0) X(I)=X(I)+Z(I)
ENDDO
```

seems a good idea to avoid doing useless additions, but forces a lot of tests and actually spoils computer time, prefer:

```
DO I=1,NPOIN
    X(I)=X(I)+Z(I)
ENDDO
```

Example 3: strides
Declaring an array as $XM(NELEM,30)$ or $XM(30,NELEM)$ for storing 30 values per element is not innocent with respect to optimisation. The principle in Fortran is that in memory the first index varies first. If you want to sum values number 15 of all elements, the first declaration is more appropriate. If you want to sum the 30 values of element 1200 the second declaration is more appropriate. The principle is that the values that are summed should be side by side in the memory.
A lot remains to be done in TELEMAC-MASCARET SYSTEM on strides. Sometimes it brings an impressive optimisation (case of murd3d.f in library TELEMAC-3D, with XM declared as $XM(30,NELEM)$ unlike the usual habit), sometimes it makes no change, e.g. the matrix-vector product in segments seems to be insensitive to the declaration of $GLOSEG$ as $(NSEG,2)$ or $(2,NSEG)$. This can be compiler dependent.
Example 4: the use and abuse of subroutine OS
Using subroutine OS is meant for simple operations like $X(I) = Y(I) + Z(I)$. Do not combine long lists of successive calls of OS to compute a complex formula, do it in a simple loop.
Thus the following sequence taken from bedload_seccurrent.f in SISYPHE:

```
CALL OS( 'X=YZ    ' , T2 , QU    , QU   , C   ) ! QU**2
CALL OS( 'X=Y/Z   ' , T2 , T2    , HN   , C   ) ! QU**2/HN
CALL OS( 'X=Y/Z   ' , T2 , T2    , HN   , C   ) ! QU**2/HN**2
CALL OS( 'X=YZ    ' , T3 , QV    , QV   , C   ) ! QV**2
CALL OS( 'X=Y/Z   ' , T3 , T3    , HN   , C   ) ! QV**2/HN
CALL OS( 'X=Y/Z   ' , T3 , T3    , HN   , C   ) ! QV**2/HN**2
CALL OS( 'X=X+Y   ' , T2 , T3    , T3   , C   ) ! QU**2+QV**2/HN**2
```

should be better written (once the discretization of $T2$ is secured, for example by $CALL\ CPSTVC(QU,T2)$):

```
DO I=1,NPOIN
  T2%R(I)=(QU%R(I)**2+QV%R(I)**2)/HN%R(I)**2
ENDDO
```

D.9 Parallelism and tidal flats

Parallelism and tidal flats are VERY demanding for algorithms. For example parallelism often doubles the time of development. It is also the case of tidal flats that bring many opportunities of divisions by zero and a number of extra problems. New algorithms must then be duly tested against parallelism and tidal flats or, in 3D, cases where elements are crushed.

E. Documentation

The documentation handled by this Software Quality Plan is divided in two part:

- The managing documentation of the TELEMAC-MASCARET SYSTEM activity;
- The technical documentation associated with the different version of TELEMAC-MASCARET SYSTEM.

E.1 Software Quality Plan and additional documents

They are the documents defining the organisation and the processes followed in the TELEMAC-MASCARET SYSTEM project. They are available in Eureka and on the TELEMAC-MASCARET SYSTEM website (`http://www.opentelemac.org`). It includes the Software Quality Plan, the development plan, the nominative list of the people in TELEMAC-MASCARET SYSTEM, the organisation document.

E.2 Technical documentation

The documentation is usually divided into 5 parts:

- The reference manual, which describes every keyword of the dictionary of one code,
- The principle note which describes the physical phenomena modelled by the module and the numerical methods used to solve the equations modelling these phenomena.
- The validation document, which presents a list of test cases; they validate the conceptual model, algorithms and software implementations,
- The user manual, which describes how to use the code,
- The release note which lists the new features of the version,

Usually, every manual is to be written for every module of the code.
Another documentation exists for all sources of the code, that is the Doxygen documentation. It contains informations on all the elements of the source code. It is generated by the Python environment and is available on the website `docs.opentelemac.org`.
A documentation listing all the environment commands and giving a short description is also available.

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

[2] GUISNEL Francoise. Les actions de management de projet applicables à edf r&d et leur modulation. *Eureka H-D03-2011-01748-FR*.

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

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