EMMC-CSA

European Materials Modelling Council

Training material for standards in software development and where to find it

by

Kurt Stokbro, Synopsys

Gerhard Goldbeck and Alexandra Simperler, Goldbeck Consulting

Volker Eyert and Erich Wimmer, Materials Design

TABLE OF CONTENT

The EMMC-CSA project has received funding from the
European Union's Horizon 2020 research and innovation programme
under Grant Agreement No 723867

1. Executive summary

1.1 Description of the content and objectives

This documentation is an Annex to the "White paper for standards of modelling software development" providing links to and training material to be used for developers of modelling software.

Different groups of software developers have to be considered, including

- Graduate students and post-doctoral researchers developing new algorithms and computational approaches and writing new stand-alone software to achieve their research goals. This group tends to focus on the domain-specific aspects and may have little experience in writing computer programs. These will be referred to as "beginners".
- Academic software developers who are building on existing computer programs with the goal to make these codes available to a wider user community. This group has both deep domain-specific knowledge and may be experienced in the techniques of writing computer programs, but should be made aware of all issues related to software development as addressed in the White Paper. These will be referred to as "Research Software Engineers"
- Professional software developers who are integrating software components into comprehensive modeling environments for large-scale distribution for a wide range of applications. This group typically has deep knowledge in software engineering, but may struggle with domain-specific issues. These will be referred to as "experts".

The present document should provide an orientation to training material for each of the groups. Clearly, the experts are is rather a resource for training (secondments) than being in need for training material. However, some advanced topics should be of interest also to this group.

Disclaimer: In the training materials section there will be often referred to American or UK sources. The intention is for the trainees to take this as a thought-provoking impulse and find the respective counterparts in their country of residence.

2. Training Materials and Information online

This will be published on the EMMC webpage with all links live, and social media of the authors shall attract people. Our linked-in group, Software for Materials Modeling (https://www.linkedin.com/groups/8684365/) will engage the community and invite them to add online resources.

2.1 For beginners

Often, beginners start coding, when they are confronted with large amount of data and an analysis by hand becomes unfeasible. They seek books or attend courses provided by their universities' IT departments to learn a programing language and start coding. Often this approach does not nurture how to think like a programmer and makes coding rather clumsy.
Here we would recommend the Software Carpentry (https://software-carpentry.org/). Since 1998, Software Carpentry has been teaching researchers the computing skills they need to get more done in less time and with less pain. These courses have found their way to Europe and would come to any place that can pay a moderate

fee. Hence, beginners would either have to persuade their departments to aid with organizing a carpentry or try to join one by a different host. Some IT departments are doing more and more outreach to attract beginners into IT, for example Imperial College London, https://www.imperial.ac.uk/admin-services/ict/self-service/research-support/rcs/training/. Some universities do subscribe to online course sites such as https://www.lynda.com/, and offer these for free to their staff.
Very popular are online courses https://www.udemy.com and https://www.coursera.org which have a number of courses related to the topics covered in the white paper. For instance, to learn Python we can highly recommend the course: https://www.coursera.org/course/interactivepython1. There will be similar courses if another programming language is of interest. These courses do charge a fee but come with a certification.
On Future Learn, https://www.futurelearn.com/using-futurelearn, there are some free courses, but one has to wait until they are available, to get access to the course for its duration plus some additional 14 day. However, without a fee there will be no access to course tests and to certificates.

Even a small piece of code can be important to the research community. A good place to put it could be https://github.com/ or https://sourceforge.net/. These repositories are also a very useful back-up in case code gets lost due some technical failures or a laptop gets stolen or lost. The community would also appreciate documentation of your code and the provision of an input file and an output file and explanations thereof.

Documentation is not only useful for the user but also for the developer themselves to remind them what exactly they were thinking when they wrote particular lines of code. There is Doxygen, https://www.stack.nl/~dimitri/doxygen/, and Sphinx, http://www.sphinx-doc.org/en/stable/, for Python. Also, very popular is the Jupyter Notebook, http://jupyter.org/, which is an open-source web application that allows to create and share documents that contain live code, equations, visualizations and narrative text.

It is always good practice to tell the community who owns the software and how it is licensed. Here, one needs to look at the status of employment of the developer, i.e. self-employed or employed. In the latter case, one may not be free to make the decision to open source or commercialize ones' software. In general, software that one creates as an employee belongs to the employer and this is also true if a university is the employer. We would recommend to look into employment contracts and seek advice from management/supervisor how to proceed.
A good overview can be found here: https://software.ac.uk/resources/guides/adopting-open-source-licence. This is written for the UK, but can be adapted to other countries. Once it is established who owns the software (Intellectual Property) and the copyright, a license should be selected. The Institute of Formal and Applied Linguistics at Charles University in Prague provide an open license selection tool, https://ufal.github.io/public-license-selector/.
More information about licenses can be found in our White Paper, https://emmc.info/emmc-csa-white-paper-for-standards-of-modelling-software-development/ and https://choosealicense.com/licenses/, for example.

It is very important, that each individual investigates the law of their respective country to be able to make the correct decisions.

Finally, some tips and tricks for software development can be found here:

- Software Sustainability Institute (SSI): https://www.software.ac.uk/resources/top-tips
- Guide to software development and projects at the Netherlands eScience Center: https://legacy.gitbook.com/book/nlesc/guide/details

- The Molecular Science Software Institute (MolSSI)
  https://molssi.org/education/best-practices/
- a good book on scientific software development in general:
  https://www.amazon.com/Introduction-Performance-Computing-Scientists-Computational/dp/143981192X
- Some Blog Sites and Forums
  http://best-practice-software-engineering.blogspot.com/
  https://softwareengineering.stackexchange.com/

Once the beginners become more familiar with software developing, they may embark also on the materials in Section 2.2., especially version control and testing.

2.2 For Research Software Engineers (RSEs)

This group has both deep domain-specific knowledge and experience in the techniques of writing computer programs. As programming is their main profession, they are already very familiar with many topics discussed in section 2.1. However, RSEs tend to work on much larger software projects and also in teams which requires additional considerations such as being more careful with licensing, more formal project management, thinking software sustainability, version control, software testing, and code parallelization.

The comments about intellectual property, copyright and licensing are still the same as discussed in section 2.1. RSEs embark usually on more complex software projects and may have to deal with the incorporation of existing code, libraries and database are used. It is very important to gather information about licensing of these existing parts as that may affect the type of licensing that is permitted for the final software.

RSEs who work in a team may consider establishing project management if there is none in place. The RSEs should only consider taking on Project Manager roles if they are either trained as such or will be provided with coaching as part of their continuous professional development. The most common style for managing software development is Agile (http://agilemanifesto.org/) and more information can be found here https://www.atlassian.com/agile. This webpage offers some insight into the Scrum and Kanban methods (https://www.atlassian.com/agile/kanban/kanban-vs-scrum) with tutorials and background information. However, as one can see after further reading all these methods require a skilled person to manage a project successfully and project manager and software developer for large projects should be ideally two different roles. An interesting concept in Agile software development are Acceptance Criteria (AC). Microsoft Press defines Acceptance Criteria as "*Conditions that a software product must satisfy to be accepted by a user, customer or other stakeholder*." Google defines them as "*Pre-established standards or requirements a product or project must meet.*" AC are very useful in aiding with what should be developed first. It starts with the "user story" where a user, the community, or the RSEs have an innovative idea about software and what it should be able to do. Then the MoSCoW method can be used which is a prioritization technique used in management, business analysis, project management, and software development to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement; it is also known as MoSCoW prioritization or MoSCoW analysis. The term MoSCoW itself is an acronym derived from the first letter of each of the four prioritization categories (Must have, Should have, Could have, and Won't have (this time)).
Some further reading can be found here:
https://www.softwaretestinghelp.com/what-are-the-quality-attributes/

https://hackernoon.com/quality-attributes-in-software-architecture-3844ea482732
http://prince2.wiki/Quality

If the RSE works in a commercial environment there are often corporate solutions to ensure that the software is reliable and reproducible, has long-term maintenance, has professional documentation, quality control in place, support, training, etc. For the RSE in academia there is often no such elaborate sustainability framework in place. That does not mean they cannot produce sustainable software. The SSI offers an evaluation tool, https://software.ac.uk/resources/online-sustainability-evaluation, and an RSE could assess their software there and get some tips. The SSI would email a report with sustainability advice that is tailored to the RSE's project.

For every software developer version control is very important it should go beyond saving the code as code_v1, code_v2, etc. There are special version control systems out that can keep a complete work history, and allow moves between past versions and commit messages that describe each added change. This is greatly beneficial for both a sole developer or a team of developers. https://git-scm.com/docs/gittutorial is a tutorial on how to use git, which is the mostly used version control system. Of interest could be also:
https://www.software.ac.uk/blog/2016-10-06-top-tips-version-control
https://molssi.org/education/best-practices/
http://swcarpentry.github.io/git-novice/
https://github.com/trein/dev-best-practices/wiki/Git-Commit-Best-Practices
https://www.atlassian.com/git/tutorials/comparing-workflows

Software should undergo testing and this happens on different levels:

- Unit testing: hereby, the functionality of a specific section of code is tested.
- Integration testing: hereby, interfaces between and interaction of modules are tested.
- Component testing: the handling of data between components are tested, usually a range of data and data types are tested.
- System Testing: an input will be created and edited, a calculation/operation will be performed and the output will be checked.
- Operational Acceptance tests/ User Acceptance tests: will be performed to check if operations can work and if users would accept the actual software product.

The level of testing which has to be performed regularly by the RSEs is unit testing and more information can be found here:
https://www.guru99.com/unit-testing-guide.html, a guide to test driven development with links to different unit testing software libraries
https://software.ac.uk/resources/guides/testing-your-software
https://gitlab.kitware.com/cmake/community/wikis/home
https://www.atlassian.com/continuous-delivery/different-types-of-software-testing

The RSEs are also responsible that their software performs on different operating systems. They can of choose one over the other but may want to document quite clearly which ones (Linux, Win, Mac, …) and which versions are supported. Often software will profit from running on HPC systems which requires the RSE to parallelize their code if it was not written in that fashion first place. Some supercomputing centers would offer courses and information for that:
http://www.archer.ac.uk/training/

http://www.prace-ri.eu/training/
http://www.prace-ri.eu/best-practice-guides/
https://molssi.org/education/summer-schools/
Further, a good tutorial on parallel computing using OpenMP and MPI is:
http://mpitutorial.com/tutorials/

Whereas a software developer in industry are expected to do all these things listed above and get professional credit for it, academic RSEs get judged by the amount of papers they produce – not by how good their code is. High Standards in Software Development require however a person to find themselves in an environment that nurtures their intention to deliver such software. The Research Software Engineers' Association, https://rse.ac.uk/about/, is attempting this and in Spring 2019 a Society will be launched, https://www.society-rse.org/. Some international efforts can be found here: https://rse.ac.uk/community/international-rse-groups/.
It is also very important to get credit for the actual software:
https://software.ac.uk/blog/2018-11-26-credit-and-recognition-research-software-current-state-practice-and-outlook

2.3 For Experts:

Experts do have Standards in Software development and this is evidenced by great software that has emerged over the last 30 years or so. The standards always have followed the needs of the users and the hardware developments. The first users were experts and often skilled in software development themselves. In the nineties of last millennium visualizers and graphical user interfaces emerged more widely and non-programmers could more easily handle software and the traditional userbase became wider. Hardware, which was very expensive, became cheaper and changed its architecture, so parallel computing became possible. Experts had to adapt and rewrite their software to run in parallel. Especially the materials manufacturing industry became interested in adding electronic and atomistic models to their research additionally to longer established FEM solvers. Also, standard workflows were not enough anymore, and users wanted to create their own workflows. Experts added APIs (application programming interfaces) to their software to answer to the users' request of more flexibility. graphics processing units (GPU)s which where intended for the purpose of rendering images (gaming industry) found their way into materials modelling and again, the experts had to react to this.

Nicholas Grundy of Thermocalc illustrated in his talk (IntOP2018 Freiburg) how the user behaviour can change using the Blockbuster/Netflix example. Blockbuster would offer stores full with video cassettes and later DVDs but went defunct in 2013 after Netflix and similar providers began to allow streaming of movies, series, etc on demand. The experts have to ask themselves right now if their users are also on a verge to change their behavior of how to use software and also prefer on-demand services rather than a full version of a software suite.

Experts, who are working on their software in the framework of a large enterprise may already be aware of such trends and look into the development of APPs and how to deploy their software in the cloud. The APPs are very suitable for users who want access to a particular automatized workflow only and the cloud becomes more attractive to users who do not want to or cannot deal with running their own hardware, such as SMEs.

The user community becomes more interested in multiscale workflows and some experts offer software that has one user interface and offers different materials models. However, sometimes users want to "mix" software of their choice (open source, in-house developed, commercial) and not of the vendors' choice. Thus, Open

Simulation Platforms (OSP) enter the stage. An OSP is formulated as a set of common standards and related tools that form the basic environment on top of which compatible and compliant simulation workflows can be developed and run. Thus, the experts who wish to take part must look into writing wrappers and learning about ontologies to permit their software to be deployed on an OSP.

Instead of providing actual training materials a selection of sources shall be provided. The expert may want to follow these sources, learn about new trends and decide if there is indeed a change in how their users want to consume the software.

The EMMC, the European Materials Modelling Council, https://emmc.info/, is an active community of relevant stakeholders and hosts high-caliber events with strong emphasis to get everyone who is interested in Materials Modelling involved.

The Software Sustainability Institute (UK), states "*The Software Sustainability Institute cultivates better, more sustainable, research software to enable world-class research. We help people build better software, and we work with researchers, developers, funders and infrastructure providers to identify key issues and best practice in scientific software.*" https://www.software.ac.uk/

The Molecular Sciences Software Institute (MolSSI, USA) states "*The Molecular Sciences Software Institute serves as a nexus for science, education, and cooperation serving the worldwide community of computational molecular scientists – a broad field including of biomolecular simulation, quantum chemistry, and materials science.*" https://molssi.org/

The European Materials Modelling marketplaces will host all stakeholders relevant to materials modelling, i.e. modellers, software owners, service providers, service consumers, experimentalists, industry, translators, database owners etc. to provide a holistic ecosystem for materials modelling.
https://www.the-marketplace-project.eu/
https://www.vimmp.eu

3. Materials provided by the EMMC

3.1 White Paper for standards of modelling software development 2.0

https://emmc.info/emmc-csa-white-paper-for-standards-of-modelling-software-development/

This EMMC-CSA White Paper was authored by Volker Eyert (Materials Design S.A.R.L) and Kurt Stokbro (Synopsys – former QuantumWise A/S)

It provides a basis for the standards of modelling software development and addresses areas such as method description, assumptions, accuracy and limitations; testing requirements; issue resolution; version control; user documentation and continuous support and resolution of issues.

The document is based on the work already carried out in the context of the EMMC to drive the adoption of software quality measures, and to ensure sustainable implementation of this EMMC initiative. Given the high level of sophistication of each of the developments which solve particular aspects of the multi-physics/chemistry spectrum of materials modelling, the industrial usefulness of individual achievements

requires integration into larger software systems. Thus, guidelines and standards are needed, which will enable the exploitation of these codes.

The major outcome are guidelines for academic software developers creating materials modelling codes. In many cases, design decisions taken at an early stage have unforeseeable consequences for many years ahead. In this context, the white paper gives academic researchers a framework, which paves the way for successful integration and industrial deployment of materials modelling. This goal is achieved by addressing a range of topics including model descriptions and software architectures, implementation, programming languages and deployment, intellectual property and license considerations, verification, testing, validation, and robustness, organization of software development, metadata, user documentation, and support.

3.2 EMMC-CSA: Webinar on Best Practices for Software Development

Slides and recording are available:
https://emmc.info/events/emmc-csa-webinar-on-best-practices-for-software-development/

This webinar was presented by Dr. Kurt Stokbro (SYNOPSYS) and Dr. Volker Eyert (Materials Design SARL) on the 29th of May 2018 and organized by Natalia Konchakova (Helmholtz-Zentrum Geesthacht).

93 persons registered and 55 attended the webinar.

The webinar was based on the White Paper for standards of modelling software development, where the EMMC-CSA provides guidelines for standards of modelling software development. The issues addressed in the paper include method description, assumptions, accuracy and limitations, testing requirements, issue resolution, software licensing, version control and user documentation as well as continuous support.

The aim of the webinar was to extend awareness of these topics which are an integral part of professional software development but often overlooked by academic software developers. The goal was to increase the value of academic software through adaption of these guidelines in the academic community.

In this webinar EMMC software experts introduced the guidelines of the white paper. After the presentation there will was a Q&A for the detail discussion of the recommendations and applicability of the recommendations.

The community learned more about:

- model descriptions and software architectures,
- implementation, programming languages and deployment,
- intellectual property and license considerations,
- verification, testing, validation, and robustness,
- organization of software development,
- metadata,
- user documentation, and support.

3.3 Trainings Materials provided by the EMMC for beginners

Herewith, we would like to facilitate the journey from the drawing board towards Software.

We provide a set of 38 slides, “Introductions to Standards in Software Development for Beginners”, to be held as webinar on 18th of January 2019 (https://www.youtube.com/watch?v=TnmpgYWVHLg). It should be inclusive to every person who would like to get started with software development.

Disclaimer: This facilitation workshop invites persons who would like to write simple to medium complex software and aims to provide some thought-provoking impulses to get them started. The participants are responsible for verifying the quality of all listed sources and make their own choice on how to proceed with their software endeavours.

3.3.1 Innovative Ideas

New software comes from the fact that somebody needs something that has not been written yet and the creation thereof is a very innovative process. The participants will be reminded on their innovation potential and how to improve it. Also, during the innovation process documentation is key. Finally, we will share trades of successful software developers.

3.3.2 Scoping

It is important to check if one will write stand-alone code of additional functionality to existing software. The latter requires effort to learn more about the architecture of software as additions can break it. Also, a programming language has to be chosen which is suitable for the task at hand. Documentation of the software is strongly suggested as a must and not as an option. Finally, the software is supposed to run on a particular operating system or hardware architecture and professional advice for the latter is highly recommended.

3.3.3 Documentation

Good software should be understood by peers and likewise by users. Software needs a technical documentation so developers can see what it does. Also, a description of its functionality using an agreed taxonomy (CEN, RoMM) can make software more findable by technology scouts, i.e. people often hired to “scout” newest developments and the usefulness thereof. Users should be provided with a good manual and tutorials to be able to install the software and run it.

3.3.4 Ownership and Licensing

Before embarking on software development, it is key to find out who will own the code/software that is produced. Sustainable software has clear owner/s and an appropriate licensing. Three main types of licensing are introduced and some suggestions to choose one.

3.3.5 Testing

The importance of testing code is discussed on a commercial level and unit testing, function testing and validation is strongly recommended for academic codes as well.

3.3.6 Version Control

This is needed to keep track of different stages of code development and also to enable developing teams working simultaneously on software.

3.3.7 Project Management

This is important even for small projects important and a must for large projects. The preferred approach Agile with Scrum and Kanban. Also, acceptance criteria are discussed, following MoSCoW. Risk and Change Management are introduced as a good to have.

3.3.8 The Technical Debt

If one does not make software sustainable from its humble beginnings as a code, one occurs "technical debt". The phrase was coined by Ward Cunningham, one of the authors of the Agile Manifesto.
"*Shipping first-time code is like going into debt. A little debt speeds development so long as it is paid back promptly with refactoring. The danger occurs when the debt is not repaid. Every minute spent on code that is not quite right for the programming task of the moment counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unfactored implementation, object-oriented or otherwise*." Ward Cunningham, 1992 OOPLSA conference.
https://www.agilealliance.org/introduction-to-the-technical-debt-concept/
Technical Debt can naturally occur if a SWO has to push to the market and therefore uses short cuts to create a viable but not necessarily a perfect piece of software that still delivers business value. The issues may be related to coding practices and style, test coverage, missing documentation, potential bugs, etc. The idea is here to monitor where technical debt will occur and then be prepared to gradually pay it back.

3.3.9 Legacy

What happens when you stop developing? If software was developed sustainable, it may live on to be used by the next generation.

Authors	Kurt Stokbro (SYNOPSYS), Gerhard Goldbeck, Alexandra Simperler (both GCL), Erich Wimmer, Volker Eyert (both MDS)
Contributing EMMC-CSA partners	SYNOPSYS, GCL and MDS

EC-Grant Agreement	723867
Project acronym	EMMC-CSA
Project title	European Materials Modelling Council - Network to capitalize on strong European position in materials modelling and to allow industry to reap the benefits
Instrument	CSA
Programme	HORIZON 2020
Client	European Commission
Start date of project	01 September 2016
Duration	36 months

Consortium		
TU WIEN	Technische Universität Wien	Austria
FRAUNHOFER	Fraunhofer Gesellschaft	Germany
GCL	Goldbeck Consulting Limited	United Kingdom
POLITO	Politecnico di Torino	Italy
UU	Uppsala Universitet	Sweden
DOW	Dow Benelux B.V.	Netherlands
EPFL	Ecole Polytechnique Federale de Lausanne	Switzerland
DPI	Dutch Polymer Institute	Netherlands
SINTEF	SINTEF AS	Norway
ACCESS e.V.	ACCESS e.V.	Germany
HZG	Helmholtz-Zentrum Geesthacht Zentrum für Material- und Küstenforschung GMBH	Germany
MDS	Materials Design S.A.R.L	France
Synopsys / QW	Synopsys (former QuantumWise A/S)	Denmark
GRANTA	Granta Design LTD	United Kingdom
UOY	University of York	United Kingdom
SINTEF	SINTEF AS	Norway
UNIBO	ALMA MATER STUDIORUM – UNIVERSITA DI BOLOGNA	Italy
SYNOPSYS	Synopsys Denmark ApS	Denmark

Coordinator – Administrative information	
Project coordinator name	Nadja ADAMOVIC
Project coordinator organization name	TU WIEN
Address	TU WIEN | E366 ISAS | Gusshausstr. 27-29 | 1040 Vienna | Austria
Email	nadja.adamovic@tuwien.ac.at emmc-info@tuwien.ac.at
Project web-sites & other access points	https://emmc.info/