Procedia Computer Science

Volume 51, 2015, Pages 1088–1097

ICCS 2015 International Conference On Computational Science

A Survey of Open Source Multiphysics Frameworks in Engineering

Önder Babur[1], Vit Smilauer[2], Tom Verhoeff[1], and Mark van den Brand[1]

[1] Eindhoven University of Technology, The Netherlands
{o.babur, t.verhoeff, m.g.j.v.d.brand}@tue.nl
[2] Czech Technical University, Czech Republic
smilauer@cml.fsv.cvut.cz

Abstract
This paper presents a systematic survey of open source multiphysics frameworks in the engineering domains. These domains share many commonalities despite the diverse application areas. A thorough search for the available frameworks with both academic and industrial origins has revealed numerous candidates. Considering key characteristics such as project size, maturity and visibility, we selected Elmer, OpenFOAM and Salome for a detailed analysis. All the public documentation for these tools has been manually collected and inspected. Based on the analysis, we built a feature model for multiphysics in engineering, which captures the commonalities and variability in the domain. We in turn validated the resulting model via two other tools; Kratos by manual inspection, and OOFEM by means of expert validation by domain experts.

Keywords: Multiphysics; Multiscale; Modelling and Simulation; Domain Analysis; Feature Model

1 Introduction

Numerical simulations have been used in industry and academia for a few decades as a computational paradigm for research and development. Recently, and notably after the 2000s, it has become increasingly important that knowledge about various physical phenomena involving distinct space/time scales and scientific disciplines are integrated in an efficient way to promote further advancement [2]. This integrative paradigm is called multiscale/multiphysics modelling and simulation (MMS). Due to its cross-disciplinary and collaborative nature, there are many groups from different domains of engineering, such as mechanical, materials and aerospace engineering, working on an open source MMS framework [6, 7].

Related work. There has been considerable effort to survey different MMS methods and tools from an engineering point of view, which sheds some light on the systematic understanding of MMS techniques and approaches [7, 5]. From the perspective of the software architect,

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.273

there are potentially many commonalities among these tools as well. Yet there seems to be little interaction between the individual developer communities (with minor exceptions such as CAE Linux[1]). Moreover little comprehensive/comparative research has been done to analyse the domain focusing on the software architecture and domain-independent features. [1] presents a comprehensive bibliography of MMS frameworks and integration approaches in multiple disciplines, but lacks in-depth analysis for engineering. A comparative account of some of the MMS tools in engineering together with other disciplines is given in [6], involving features such as distributed execution and user interface. [9] shortly discusses several tools in a variety of domains in terms of parallelisation, coupling mechanism, etc. Modelling technology and infrastructure technologies are compared with pointers to important commercial and open source tools [12]. [18] presents a conceptualisation of the fundamental MMS problem and discusses various tools revolving around a conceptualisation focus. Furthermore in the individual papers of the frameworks, several comparisons are made to outline related work [13, 15, 19]. Although scattered bits of comparisons and architectural overviews of MMS frameworks are present in these papers, there is a lack of in-depth investigation in the engineering domain. The frameworks that are particularly interesting are the open source ones as they have key characteristics such as open access to source code repositories or collaborative and transparent nature [16], which can enhance the depth and objectiveness of analysis.

Objective. The purpose of this study is to answer the following questions:

1. What are the commonalities and differences between the open source multiphysics frameworks in the engineering domains in terms of software architecture, user-visible and domain-independent functionalities?
2. How can we model these common and distinctive features in a unified way?
3. How can we use this model to increase understanding of the domain, promote communication, future development, and technology adoption in engineering communities?

2 Methods

In general, we adopted a qualitative screening approach [10] to conduct our research, i.e. most of the work including data collection, selection and analysis was done manually. We followed a similar methodology that is used for systematic literature review [11] with slight modifications.

Domain analysis and modelling. Domain analysis is an important activity in Software Product Lines which analyses related software systems in a domain to find their common and variable parts [3]. Among several techniques is Feature-Oriented Domain Analysis (FODA), a well established technique that has been validated by the community [8] and has good tool support (such as FeatureIDE[2] and SPLOT[3]). FODA allows a systematic analysis of a domain in order to understand the domain and create reusable resources (e.g. design, components). A relevant formalism in FODA is the feature model, which captures the common, alternative and optional features and their interdependencies. A feature is defined as an operating environment, user-visible capability, domain technology or an implementation technique. Figure 1 demonstrates a slightly modified version of the classic example in [4]. A car can be modelled as consisting of a mandatory body, transmission and engine. It can optionally pull a trailer

[1] http://www.caelinux.com

[2] http://wwwiti.cs.uni-magdeburg.de/\iti_db/research/featureide/

[3] http://www.splot-research.org/

as well. Furthermore, the transmission feature is in turn either manual or automatic, but not both (alternative). Finally a car can have gasoline and electric engine, or even both (hybrid). The last line in the figure is a constraint in a propositional formula, meaning that if an electric car (not a hybrid) must have automatic transmission.

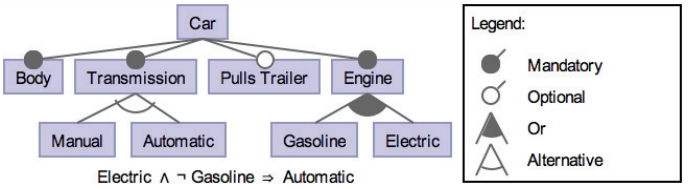


Figure 1: A simple feature model of a car

An advantage of using FODA over other analysis techniques is that it naturally captures the variability together with the commonality in a domain, and focuses on external user-visible features instead of the internal building blocks of a domain. FODA specifies several sources of information that can be used for domain analysis: textbooks, standards, existing applications and domain experts. Among these, existing applications are considered to be the most important source of domain knowledge for directly determining the domain features.

Search process. There are many open source multiphysics tools that have been developed independently by different communities both in academia and industry. In order not to ignore any potential framework with industry origins and less of an academic footprint, we used both Google and Google Scholar search to look for the keywords. We selected the keywords *multiphysics* and *multiscale* combined with *framework*, *platform*, *coupling [toolkit]* and *environment* to cover only the explicitly advertised tools of interest. We do not claim to have covered all of the market/literature, but we believe we cover a great deal of the prominent open source multiphysics tools in engineering.

Selection criteria. As FODA suggests, a domain analysis activity typically involves a minimum of three tools for analysing and another one or two for validating the analysis results. For driving the selection process among many potential tools, we implemented a two-step approach. First we eliminated the tools that are experimental, paper-only, at an early development stage, too small sized, abandoned or vaporware to obtain a first set. Then we collected initial data on the first set with some key characteristics such as source code size (SLOC), number of development years, size of the developer team, level of documentation, size and activity of the community and the number of citations in the literature. Eventually we aimed to identify the tools that are relatively more mature and have an active community, while restricting ourselves with the engineering domain for the scope of this paper.

Data collection. Once we determined the target tools to be analysed, we collected all the documentation we could find in the public domain through the tool websites, repositories and publications. An advantage of using user-level documentation such as manuals and presentations is that the user-visible features are easier to identify in comparison with lower level information sources such as source code where one has to perform additional techniques to extract those features. The informal pieces of information found in these documents were collected and closely inspected to identify relevant characteristics as features. Note that we

completely ignored proprietary extensions (e.g. GiD integration for Kratos) and did not include corresponding features in our model.

Domain modelling. We modelled the collected features from several tools in the form of a feature model in an iterative manner. We integrated the output of the data collection activity, i.e. the list of individual features for each tool into an empty feature model, while using various modelling techniques such as composition and generalisation to further refine the models. We followed a protocol to extract features from the vast amount of documentation. For each tool:

1. Inspect each document manually to extract the set of keywords, structures and concepts to build a condensed summary while ignoring low level technical and domain-specific concepts. Potential targets for extraction are:
   (a) elements in architectural diagrams,
   (b) chapter and section titles in user documentation,
   (c) package and module names in API documentation,
   (d) overview lists, FAQ items in the tool websites,
   (e) comparison with other tools in various presentations and papers.
2. Identify potential features in the condensed summary and iteratively add to the feature model. Consider to:
   (a) merge synonymous or very similar features into a single feature for simplification,
   (b) add non-standard or experimental features as *Optional*,
   (c) possibly transform multiple features in the form of noun clauses as *Or* features,
   (d) possibly transform multiple features in the form of adjective clauses as *Alternative* or *Or* features.
3. Refine the model further, consulting external sources for additional merging and grouping of features.
4. Identify additional constraints (e.g. dependencies and exclusion) implied by the relations embedded in the concepts.

Validation. The feature model resulting from the analysis of the first set of tools was validated twofold: once by manually inspecting another tool and once by consulting the developer of a further tool, who is a also domain expert. The objective for both, although performed by different stakeholders, was to try to fit the new tool's features on the feature model, and report additional feature requests and objections to the model structure. We performed product configuration, i.e. selected the appropriate features on the domain model which are present in the validating tool. FeatureIDE provided the tool support in terms of the graphical editor and consistency checking functionality. The percentage of conformance to the model (new features/total features) was documented as an indicator of the feature model's genericness and coverage/representation of the whole multiphysics domain.

3 Results

In this section we present our main results. Through an extensive search for open source multiphysics frameworks, we were able to find 80+ tools. The first selection step resulted in a set of 30+ major tools where we focused to collect the key characteristics as specified in the methods section. To determine the main application domain, we used a similar scheme as in [6] considering various computational engineering subdomains such as mechanical engineering, aerospace engineering and additionally materials science/engineering to limit our scope.

Selection of tools. As a further refined set of interesting tools in engineering, we selected 9 major tools among the first set, which are outlined as advertised by their developers:

- Elmer is a finite element multiphysics package from Helsinki CSC-IT for mechanical engineering, fluid dynamics, electromagnetics, heat transfer and acoustics[4].
- OOFEM is a object-oriented finite element package from Czech Technical University for mechanical engineering, transportation and fluid mechanics[5].
- Moose is a multiphysics object oriented simulation environment from Idaho National Lab for microstructure, chemistry, geomechanics and superconductivity[6].
- OpenFOAM is a computational fluid dynamics package from OpenFOAM Foundation for fluid dynamics[7].
- SU2 (Stanford University Unstructured) is a computational fluid dynamics package from Stanford University for aerospace[8].
- Kratos is a framework for parallel multiphysics calculation from Cimne UPC for solid mechanics, fluid dynamics and thermodynamics[9].
- CoolFluid3 is a collaborative simulation environment for complex multiphysics from Von Karman Institute for fluid dynamics, hydrodynamics and aerodynamics[10].
- MBDyn is a multi body dynamics simulation package from Politechnico di Milano for aerospace, wind, automotive and mechanics[11].
- OpenCASCADE Technology (OCCT) is a software development platform from the company OPEN CASCADE S.A.S. for automotive, aeronautics and space, naval, mechanics and medical science[12].
- Salome is an open-source software from the company OPEN CASCADE S.A.S. that provides a generic platform for pre- and postprocessing for numerical simulation[13].

We would like to mention some of the noteworthy tools that we eliminated, either because of their small size and community or lack of source code access/discontinued development: MuPIF[14], AixVipMap[15], CHEOPS[14] and CouPE[16]. We omitted also Trilinos[17] because of its huge size and focus on low level technical implementation (e.g. solver algorithms); it is out of scope for this manual analysis.

Key characteristics of the selected tools. Here we classified two categories of characteristics, one related to development aspects and the other to the developer/user community. Table 1 shows the development start date, latest release date, latest commit date in the source code repository, the number of developers and the size of the tool in source lines of code (SLOC); all data being collected in November 2014. Bug/issue tracker on the other hand indicates how systematic the communication of the bugs is. The findings indicate that there are quite sizeable projects which have a long-lived development process and a relatively large team of developers. Note that it is not always possible to retrieve exact numbers, for instance the number of developers for OpenFOAM and OCCT are presumed to be in the scale of 10s based on access to a part of their development history.

The second list of characteristics, as shown in Table 2, is related to the communication and visibility of the tool community. The first and second columns show how many results are

[4] https://csc.fi/web/elmer
[5] http://www.oofem.org/
[6] http://mooseframework.org/
[7] http://www.openfoam.com/
[8] http://su2.stanford.edu/
[9] http://www.cimne.com/kratos/
[10] http://coolfluid.github.io/
[11] https://www.mbdyn.org/
[12] http://www.opencascade.org/
[13] http://www.salome-platform.org/
[14] http://www.oofem.org/wiki/doku.php?id=mupif:mupif
[15] http://aixvipmap.iehk.rwth-aachen.de/
[16] https://sites.google.com/site/coupempf/
[17] http://trilinos.org/

Table 1: Tool Characteristics - Development

Tool Name	Start Date	Latest Release	Latest Commit	# Devs	SLOC	Bug Tracker
Elmer	1995	2010	2013	10	800k	medium
OOFEM	2007	2014	2014	25	220k	good
Moose	2008	2014	2014	70	64k	good
OpenFOAM	2004	2014	2014	10x	650k	good
SU2	2012	2014	2014	20	100k	-
Kratos	2009	2012	2014	42	1600k	good
CoolFluid3	2010	2012	2014	12	250k	medium
MbDyn	2001	2014	2014	25	200k	-
OCCT	1993	2014	2014	10x	1000k	good
Salome	2000	2014	2014	20	1300k	good

Table 2: Tool Characteristics - Community

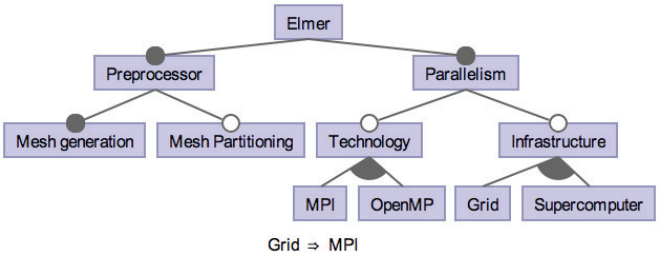
Tool Name	# Google	# Scholar	# Doc	Support	Community
Elmer	69k	11k	good	medium	good
OOFEM	12k	300	good	low	medium
Moose	150k	30k	good	low	medium
OpenFOAM	443k	11k	good	good	good
SU2	2k	60	medium	low	low
Kratos	72k	112	medium	low	low
CoolFluid3	175k	359	medium	low	medium
MbDyn	10k	258	low	low	low
OCCT	210k	1,8k	good	good	good
Salome	350k	7,6k	good	good	good

found in Google and Google Scholar respectively. Documentation quality is a measure of how extensive the user documentation is and whether it is maintained on a par with development or might be lagging. Any support services offered to the users in the form of developer support or professional contracted support is given in the support column. Finally community activity includes the activity of the tool developers and users, including in forums, mailing lists, blogs and at workshops/meetings. The results agree with those reported in previous work such as [6], indicating that open source multiphysics tools have achieved high popularity with active communities and visibility.

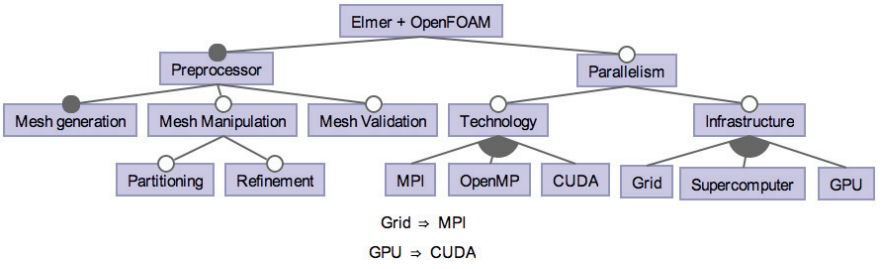
From all of this gathered information, we chose a final set of five tools, three for modelling and two for validation. The three tools for modelling were selected to be Elmer, OpenFOAM and Salome, while we used Kratos and OOFEM for validation of our model. Salome is a special case as it originates from the energy domain but we preferred it over OCCT because OCCT is already integrated into Salome as a more generic environment.

A feature model of the multiphysics frameworks. We modelled the frameworks iteratively using Elmer, OpenFOAM and Salome. The whole design process involved multiple iterations and refinements. We show here a small example of an iteration of integration. In

Figure 2a, an initial model is built using Elmer. Elmer consists of a preprocessor feature with mesh generation and optionally mesh partitioning. In terms of parallelism, it supports potentially Grid and Supercomputer using technologies of MPI and OpenMP. The additional constraint makes sure that Grid support requires MPI technology to work. Next in Figure 2b we demonstrate the feature model of OpenFOAM integrated into Elmer's model. OpenFOAM supports additionally mesh refinement, which is translated into a feature Mesh Manipulation with sub-features from both frameworks. Furthermore, OpenFOAM supports GPU computing via CUDA, which is added in corresponding locations in the Parallelism feature.



(a) Feature model of Elmer



(b) Feature model of Elmer and OpenFOAM integrated

Figure 2: A small example of our iterative modelling

The final feature model, including features from all three frameworks, consists of more than 150 features. The whole model is too big to put in this paper and can be accessed in our repository[18]. See Figure 3 for a small section of our domain model.

Validation of the feature model. As mentioned above, we validated our feature model twofold. For Kratos, we could detect 5 changes (new features, removals, etc.) out of 80 features in total, giving ∼94% of conformance to our model. For OOFEM, our domain expert suggested 7 changes while selecting 85 features in total, i.e. ∼92% conformance to our model. The relatively high conformance ratings indicate that our domain model has a considerable representativeness of the whole engineering multiphysics domain, of course within the limited scope of our study. The configuration files for both tools can be publicly accessed[19].

4 Discussion

In this section we summarise the implications of our study. Our extensive survey reveals that there are many open-source multiphysics frameworks in the market. Even solely for the

[18] `http://www.win.tue.nl/~obabur/mms/fm.xml` [19] `http://www.win.tue.nl/~obabur/mms/config/`

Figure 3: A small part of the final multiphysics feature model

engineering domain, many communities are actively developing and maintaining sizeable frameworks that are being used by a large user base. A considerable number of teams are apparently using software engineering tools such as source code repositories and bug trackers, which are considered to be among the best practices for scientific computing [17]. Although the selected tools cover a variety of computational domains (e.g. fluid dynamics versus solid mechanics) and modelling paradigms (e.g. finite element versus finite volume) there are significant commonalities among them. This is presumably due to the fact that the fundamental problems are the same: numerical simulation and software integration. Some examples of these commonalities are Computer Aided Design (CAD) import, mesh operations, data post-processing, visualisation of results and object-oriented code extension. Besides the commonalities, we found also many differences among the frameworks. It is natural that each framework considers a different set of features as relevant and of high priority and chooses to implement them. For instance, one tool deals with larger meshes and complex physics which need more computational resources, and hence prioritises advanced parallelisation mechanisms and HPC infrastructure support. On the other hand, another tool has a better and extendible GUI, and aims at higher usability.

Our feature modelling technique allows modelling the whole engineering multiphysics domain, while retaining the tool-specific information as well. Thanks to the tool support for graphical editing and consistency checking, we were able to focus on the actual modelling. We provided the feature model and configuration profiles in machine-processable XML format, allowing for future extension and model transformation.

Possible uses for the model. We believe that our formalised modelling effort with feature models leads to a better understanding and communication of the multiphysics engineering domain in terms of user-visible features and key architectural issues. Visualising the interesting components of the system, their hierarchy, variability and interdependence as a whole, while at the same time being able to inspect the particular configuration of each individual tool aids the user to compare the tools and select one that fits their requirements. Furthermore, the feature model would also be beneficial to framework developers as a reference architecture, where they might find weak spots of their products and potential improvements.

Threats to validity. There are several threats to validity for our study. First of all, the relatively small number of tools analysed might lead to a less accurate model of the whole engineering domain. A possible way of overcoming this shortcoming is to extend the study including a larger set of tools. Secondly, we considered the public documentation of tools to analyse the features, but any missing or lagging (belonging to the previous release) documentation would result in an incomplete feature set for that particular tool. Feature location techniques based on source code would arguably help but also introduce much extra work as they do not guarantee a fully automated and safe location. A similar threat is due to the qualitative analysis; our study might include subjectivity or erroneous judgement. Extensive validation by more domain experts, both framework developers and users, would compensate for this disadvantage.

5 Conclusions and Future Work

In this paper we report an extensive survey that we conducted on open-source multiphysics frameworks in engineering. We searched rigorously for the available frameworks in the market and provided objective selection mechanisms to identify the most prominent ones for our study. Based on a final set of frameworks, we modelled the commonalities and differences of these tools in a unified way as feature models; this way we achieved a preliminary domain model for engineering multiphysics. We in turn validated this model with two other prominent tools. The machine-processable models and validating configurations are available to be used with tool support. Finally we explained how this domain model would increase our understanding of the domain, help the framework developers to compare and improve their products, and act as a product configuration/decision support system for end-users of these frameworks.

Several directions for future work would include increasing the number of tools analysed, possibly from other domains such as energy or environment to achieve a cross-disciplinary domain model for multiphysics. Another possible extension point is to analyse commercial multiphysics frameworks (such as Comsol Multiphysics[20]) with the help of their public user documentation and compare the domain model for open-source tools with the commercial one to see whether the former offers similar features or is lagging behind. With the help of collaborators from the engineering domain, we could aim to model the technical engineering features of the multiphysics domain as well. A more extensive validation effort from domain experts from engineering could help achieve a more accurate and sound domain model. Furthermore, with proper tool support such as already provided in SPLOT, a product recommendation/decision support system could be developed, where users start selecting the features they want, and the system suggests them suitable tools they can use (e.g. a research group needs a CFD tool with Grid support, CAD import, and VTK export; which tool can they use?).

[20] `http://www.comsol.com/comsol-multiphysics`

Acknowledgements

The research leading to these results has been funded by EU programme FP7-NMP-2013-SMALL-7 under grant agreement number 604279 (MMP). We would like to thank the participants of EU Multiscale Materials Modelling cluster, European Materials Modeling Council (EMMC[21]), especially the Open Simulation Platform working group, and Integrated Computational Materials Engineering expert group (ICMEg[22]) for providing invaluable information and motivation on our work.

References

[1] Ö. Babur, T. Verhoeff, and M. van den Brand. Multiphysics and multiscale software frameworks: An annotated bibliography. Technical Report 15-01, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, 2015.

[2] D. L. Brown et al. *Applied Mathematics at the U.S. Department of Energy: Past, Present and a View to the Future*. Feb 2008.

[3] P. Clements and L. Northrop. *Software product lines: practices and patterns*, volume 59. Addison-Wesley Reading, 2002.

[4] K. Czarnecki and U. W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[5] R. de Borst and S. J. Hulshoff. Multiscale Methods in Computational Fluid Dynamics and Solid Mechanics. *European Conference on Computational Fluid Dynamics*, pages 1–18, 2006.

[6] D. Groen, S. J. Zasada, and P. V. Coveney. Survey of multiscale and multiphysics applications and communities. *Computing in Science & Engineering*, 16(2):34–43, 2014.

[7] M. F. Horstemeyer. Multiscale Modeling: A Review. In Jerzy Leszczynski and Manoj K Shukla, editors, *Practical Aspects of Computational Chemistry*, pages 87–135. Springer Netherlands, 2010.

[8] K. C. Kang et al. Feature-Oriented Domain Analysis (FODA) Feasibility Study. November 1990.

[9] D.E. Keyes et al. Multiphysics Simulations: Challenges and Opportunities. *International Journal of High Performance Computing Applications*, 2(1):4–83, February 2013.

[10] B. Kitchenham. Desmet: a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal*, 8:120–126(6), June 1997.

[11] B.A. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. 2007.

[12] J. G. Michopoulos, C. Farhat, and J. Fish. Modeling and Simulation of Multiphysics Systems. *Journal of Computing and Information Science in Engineering*, 5(3):198, 2005.

[13] B. Patzák, D. Rypl, and J. Kruis. MuPIF - A distributed multi-physics integration tool. *Advances in Engineering Software*, 60-61:89–97, June 2013.

[14] G. Schopfer et al. Cheops: A tool-integration platform for chemical process modelling and simulation. *International Journal on Software Tools for Technology Transfer*, 6(3):186–202, 2004.

[15] E. De Sturler and J. Hoeflinger. A new approach to software integration frameworks for multiphysics simulation codes *. *The Architecture of Scientific Software*, pages 87–104, 2001.

[16] S. Weber. *The Success of Open Source*. Harvard Univ. Press, 2004.

[17] G. Wilson et al. Best practices for scientific computing. *PLoS biology*, 12(1):e1001745, 2014.

[18] A. Yang and W. Marquardt. An ontological conceptualization of multiscale models. *Computers & Chemical Engineering*, 33(4):822–837, April 2009.

[19] S. F. Portegies Zwart et al. Multi-physics simulations using a hierarchical interchangeable software interface. *Computer Physics Communications*, 184(3):456–468, March 2013.

[21] `http://emmc.info`

[22] `http://icmeg.info`