Software Engineering for Computational Science

Arne Johanson[1] Wilhelm Hasselbring[2]

Abstract: Despite the increasing importance of in silico experiments to the scientific discovery process, state-of-the-art software engineering practices are rarely adopted in computational science. To understand the underlying causes for this situation and to identify ways to improve it, we conducted a literature survey on software engineering practices in computational science. We identified recurring key characteristics of scientific software development that are the result of the nature of scientific challenges, the limitations of computers, and the cultural environment of scientific software development. Our findings allow us to point out shortcomings of existing approaches for bridging the gap between software engineering and computational science and to provide an outlook on promising research directions that could contribute to improving the current situation.

Keywords: Computational Science; Model-driven software engineering; Software architecture

Computational science (also scientific computing) involves the development of models and simulations to understand natural systems answering questions that neither theory nor experiment alone are equipped to answer [Rü18]. Computational science is a multidisciplinary field lying at the intersection of mathematics and statistics, computer science, and core disciplines of science. Despite the increasing importance of so-called in-silico experiments to the scientific discovery process, well-established software engineering practices are rarely adopted in computational science [JH18]. However, meanwhile the computational science community starts to appreciate that software engineering is central to any effort to increase computational science's software productivity [Rü18, page 737]. Among the methods and techniques that software engineering can offer to computational science are

- model-driven software engineering with domain specific languages [JH14; JH17; Jo16b; Jo17],
- modular software architectures [Ha18; HS17; Jo16a],
- specific requirements engineering techniques [Th09], and
- testing without test oracles [KB14].

Computational science requires maintainable, long-living software [Go15], enabled by domain-specific software engineering methods and techniques.

[1] XING Marketing Solutions GmbH, Hamburg arj@informatik.uni-kiel.de
[2] Kiel University, Software Engineering Group, Kiel hasselbring@email.uni-kiel.de

doi:10.18420/se2019-08

Literatur

[Go15] Goltz, U.; Reussner, R.; Goedicke, M.; Hasselbring, W.; Märtin, L.; Vogel-Heuser, B.: Design for future: managed software evolution. Computer Science – Research and Development 30/3, S. 321–331, Aug. 2015.

[Ha18] Hasselbring, W.: Software Architecture: Past, Present, Future. In: The Essence of Software Engineering. Springer, S. 169–184, 2018, URL: https://doi.org/10.1007/978-3-319-73897-0_10.

[HS17] Hasselbring, W.; Steinacker, G.: Microservice Architectures for Scalability, Agility and Reliability in E-Commerce. In: Proceedings 2017 IEEE International Conference on Software Architecture Workshops (ICSAW). S. 243–246, 2017.

[JH14] Johanson, A.; Hasselbring, W.: Hierarchical Combination of Internal and External Domain-Specific Languages for Scientific Computing. In: Proceedings of the 2014 European Conference on Software Architecture Workshops - ECSAW '14. ACM, S. 1–8, Aug. 2014, URL: https://doi.org/10.1145/2642803.2642820.

[JH17] Johanson, A.; Hasselbring, W.: Effectiveness and efficiency of a domain-specific language for high-performance marine ecosystem simulation: a controlled experiment. Empirical Software Engineering 22/4, S. 2206–2236, Aug. 2017, URL: http://rdcu.be/urXK.

[JH18] Johanson, A.; Hasselbring, W.: Software Engineering for Computational Science: Past, Present, Future. Computing in Science & Engineering 20/2, S. 90–109, März 2018, URL: https://doi.org/10.1109/MCSE.2018.021651343.

[Jo16a] Johanson, A.; Flögel, S.; Dullo, C.; Hasselbring, W.: OceanTEA: Exploring Ocean-Derived Climate Data Using Microservices. In: Proceedings of the Sixth International Workshop on Climate Informatics (CI 2016). S. 25–28, 2016.

[Jo16b] Johanson, A.; Hasselbring, W.; Oschlies, A.; Worm, B.: Evaluating Hierarchical Domain-Specific Languages for Computational Science: Applying the Sprat Approach to a Marine Ecosystem Model. In: Software Engineering for Science. Chapman und Hall/CRC, S. 175–200, 2016.

[Jo17] Johanson, A. N. et al.: SPRAT: A spatially-explicit marine ecosystem model based on population balance equations. Ecological Modelling 349/, S. 11–25, 2017, URL: http://doi.org/10.1016/j.ecolmodel.2017.01.020.

[KB14] Kanewala, U.; Bieman, J. M.: Testing scientific software: A systematic literature review. Information and Software Technology 56/10, S. 1219–1232, 2014.

[Rü18] Rüde, U.; Willcox, K.; McInnes, L. C.; Sterck, H. D.: Research and education in computational science and engineering. Siam Review 60/3, S. 707–754, 2018.

[Th09] Thew, S.; Sutcliffe, A.; Procter, R.; De Bruijn, O.; McNaught, J.; Venters, C. C.; Buchan, I.: Requirements engineering for E-science: experiences in epidemiology. IEEE Software 26/1, S. 80–87, 2009.