RSE Research

Wilhelm (Willi) Hasselbring

Software Engineering
http://se.informatik.uni-kiel.de

GI-Fachgruppe RSE, Berlin 08. Juni 2023

Research Software

RDA FAIR for Research Software (FAIR4RS) WG [Chue Hong 2022] :

- Research software includes source code files, algorithms, scripts, computational workflows, and executables that are created **during** the research process or **for** a research purpose.
- Software components (e.g., operating systems, programming languages, libraries, etc.) that are used for research but were not created during or with a clear research intent should be considered `**software in research**´ and not `research software´.
- Thus, research software is a separate metaphor of software in research.

Research software should be **FAIR** [Hasselbring et al. 2020b, Lamprecht et al. 2020] and **open** [Hasselbring et al. 2020a].

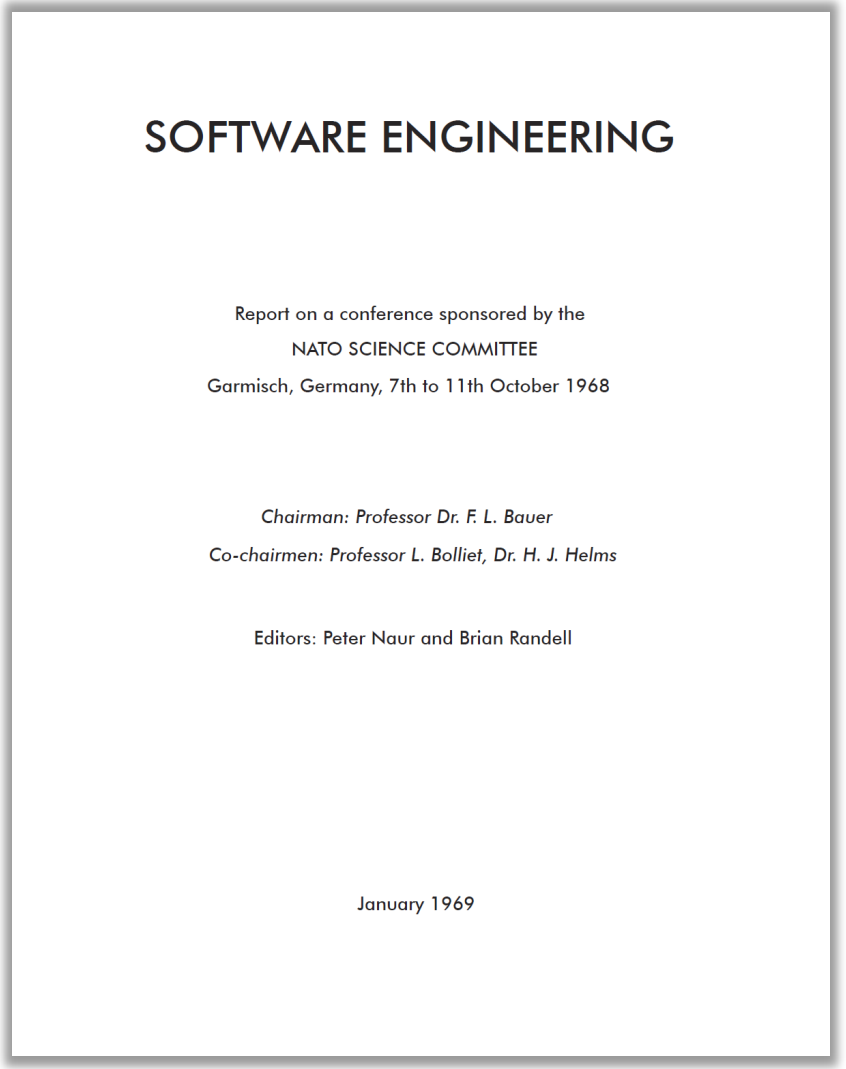
Agenda

1. Software Engineering und Forschungssoftware
2. Kategorien von Forschungssoftware
3. Ausblick: RSE Research

Software Engineering: Origin

Software Engineering and Computer Science for generality [Randell 2018]:

- That **NATO** was the sponsor of this conference marks the relative **distance** of software engineering from computation in the academic context.
- The perception was that while **errors** in scientific data processing applications might be a nuisance, they are all in all **tolerable**.
- In contrast, failures in **mission-critical** military systems might cost lives and substantial amounts of money.
- Based on this attitude, software engineering—like computer science as a whole— aimed for generality in its methods, techniques, and processes and focused almost exclusively on **business** and **embedded** software


SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

January 1969

http://homepages.cs.ncl.ac.uk/brian.randell/NATO/index.html

Computational Science: Challenges

The **Productivity Crisis** in Computational Science

- As early scientific software was developed by small teams of scientists primarily for their own research, **modularity, maintainability**, and team coordination could often be neglected without a large impact.

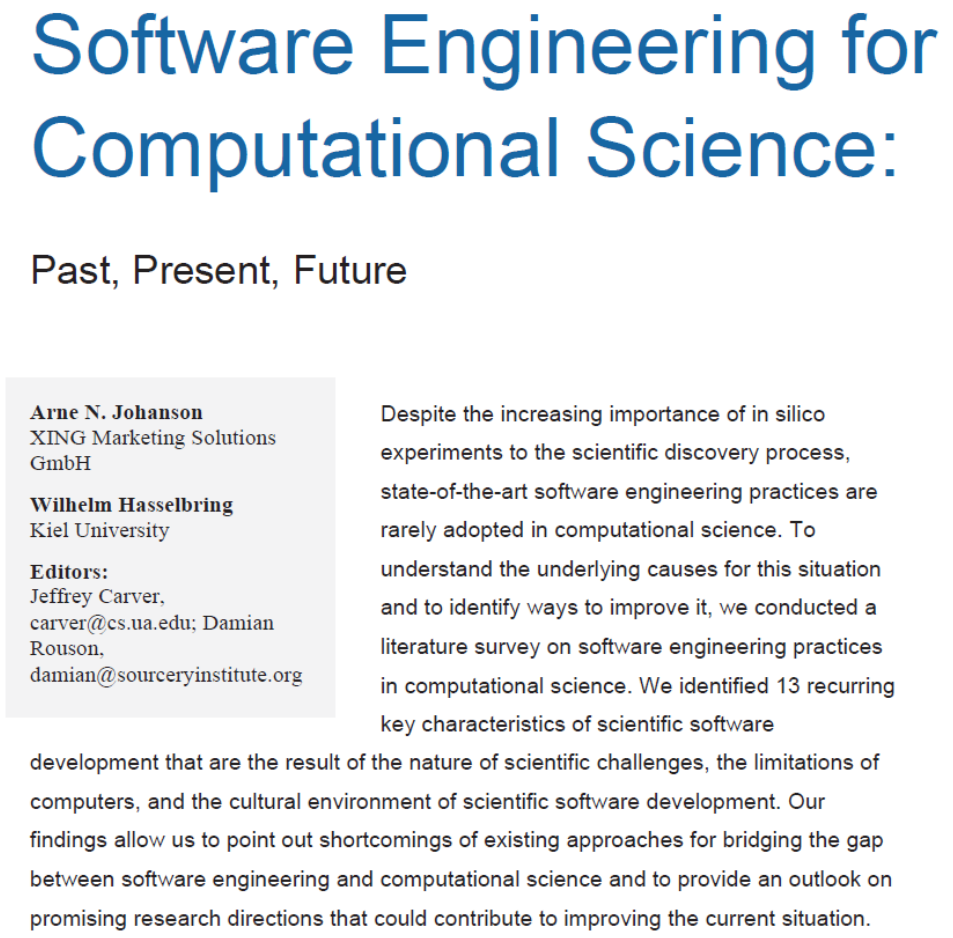
The **Credibility Crisis** in Computational Science:

- **Climategate.** The scandal erupted after hackers leaked the email correspondence of scientists just before the 2009 United Nations Climate Change Conference.
- The emails uncovered a **lack of programming skills** among the researchers and exposed to a large public audience the widely applied practice in climate science of **not releasing simulation code and data** together with corresponding publications [Merali 2010].
- This in itself was, of course, enough to **undermine the scientists' work**, as the predictive capabilities of simulations are only as good as their code quality and their code was not even available for peer review—not to mention public review [Fuller and Millett 2011].

So, Software Engineering for Computational Science

[Johanson & Hasselbring 2018]:

- Among the methods and techniques that software engineering can offer to computational science are
  - **testing without test oracles,**
  - **modular software architectures, and**
  - **model-driven software engineering with domain-specific languages.**
- This way, computational science may achieve **maintainable**, long-living software [Goltz et al., 2015; Reussner et al. 2019],
  - in particular for community software.



Software Engineering for Computational Science:

Past, Present, Future

Arne N. Johanson
XING Marketing Solutions GmbH

Wilhelm Hasselbring
Kiel University

Editors:
Jeffrey Carver, carver@cs.ua.edu; Damian Rouson, damian@sourceryinstitute.org

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 13 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.

Agenda

1. Software Engineering und Forschungssoftware
2. **Kategorien von Forschungssoftware**
3. Ausblick: RSE Research

Categories of Research Software

Research software mainly falls into one of the following categories (and sometimes combinations):

1. **Modeling, simulation and data analytics** of, e.g., physical, chemical, social, or biological processes in spatio-temporal contexts.
   - Numerical and agent-based modeling and simulation (in silico experiments)
   - Data assimilation
   - Data science and data engineering
2. **Embedded control software** for complex physical or chemical experiments and instruments, including many forms of sensor-based data collection.
3. **Software prototypes** in engineering research.
4. **Infrastructure** and platform software, such as research data and software management systems.

These categories have varying quality requirements!

(in collaboration with Michael Felderer, Michael Goedicke, Lars Grunske, Anna-Lena Lamprecht, Bernhard Rumpe)

Example for Category 1 (Modeling and simulation): Modularization of Earth-system simulation software as basis for domain-specific languages

Software Modularization

How to

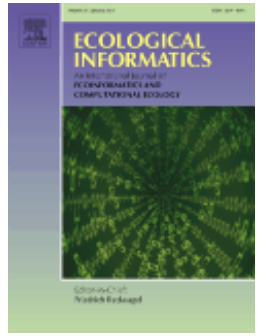
- improve maintainability, stability, reusability, reproducibility, … ?
- enable scalable execution in the Cloud?
- parallelize for high performance computing?
- test for higher quality?
- achieve higher flexibility?

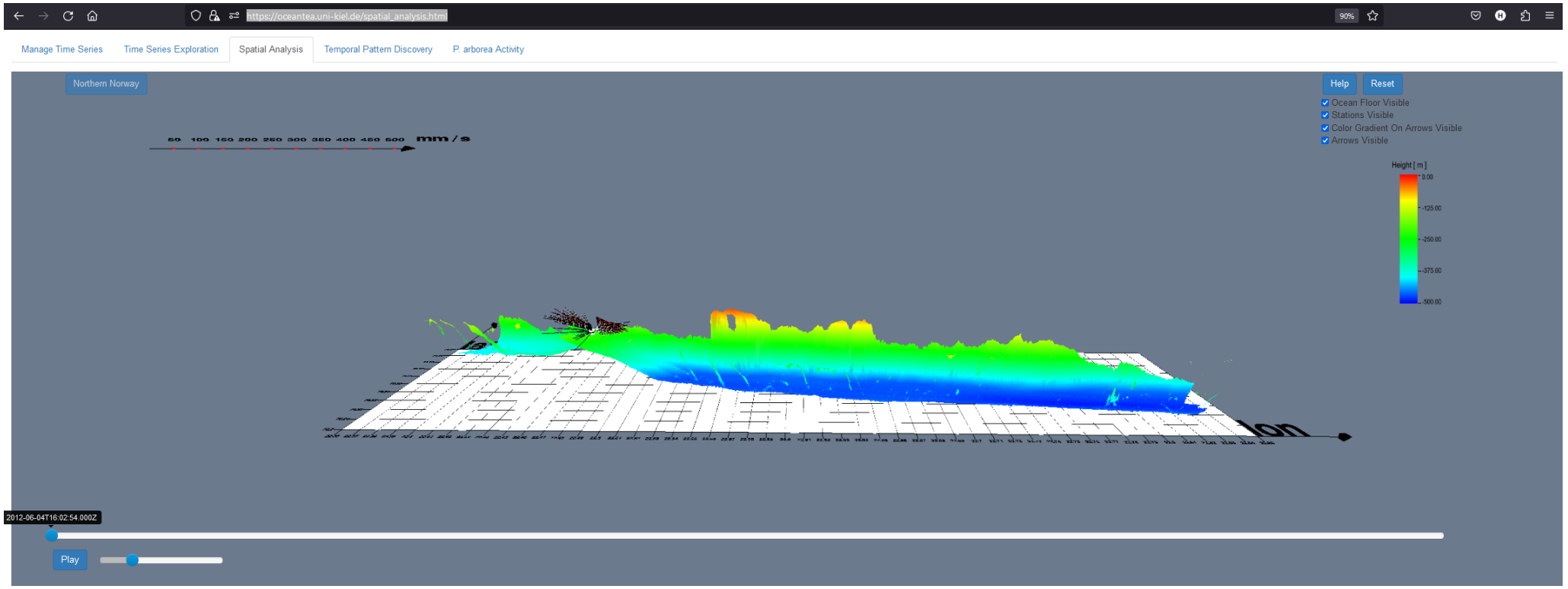
[Johanson & Hasselbring 2017, Claus et al. 2022, Jung et al. 2021, 2022a, 2022b]

OceanDSL

Funded by
DFG Deutsche Forschungsgemeinschaft
German Research Foundation

Example for Category 1 (Data analytics): OceanTEA



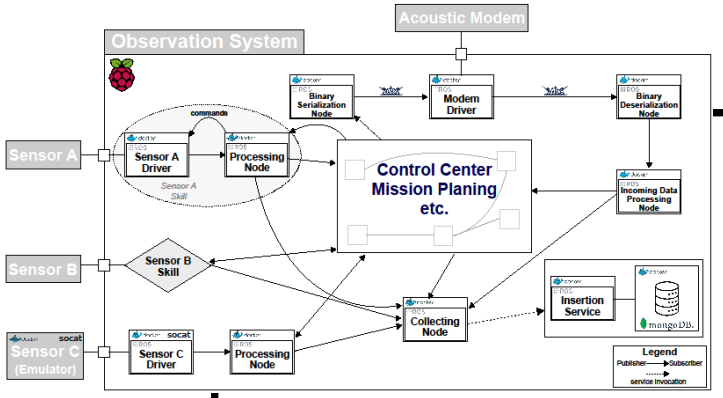


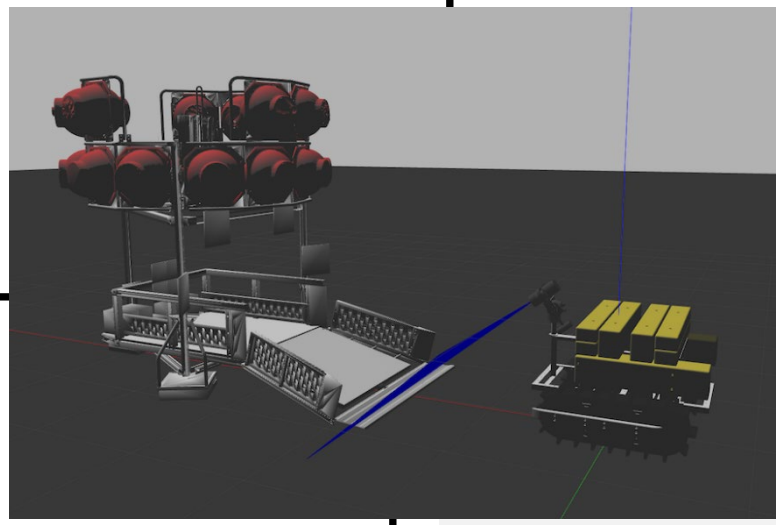
Paper on the analysis results: [Johanson et al. 2017]
Paper on the software architecture: [Johanson et al. 2016]
Code: https://github.com/cau-se/oceantea
Software service with data: https://oceantea.uni-kiel.de/

ozean der zukunft
DIE KIELER MEERESWISSENSCHAFTEN

Example for Category 2 (Embedded control software): Entwicklung von Software für Unterwasser-Roboter

Digital Twin Prototype





Digital Twin

Physical Twin

[Barbie et al. 2021]

GEOMAR

Examples for Category 3 (Software prototypes): Software Impacts

https://github.com/kieker-monitoring

Kieker: A monitoring framework for software engineering research
[Hasselbring and van Hoorn 2020]

https://github.com/ExplorViz

ExplorViz: Research on software visualization, comprehension and collaboration
[Hasselbring et al. 2020c]

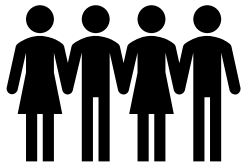
https://github.com/cau-se/titan-ccp

The Titan Control Center for Industrial DevOps analytics research
[Henning and Hasselbring 2021]

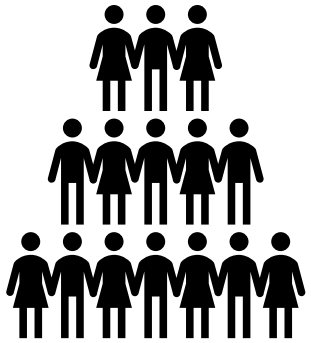
Examples for Category 4 (Infrastructure):

Another Categorization: Stages of Research Software

Individual Researcher



Local Research Group



Community (incl. Non-Researchers)

Outlook: RSE Research

Research Software Engineering

Software Engineering Research

Research Software Engineering Research
aims at understanding and improving how software is developed for research.

RSE Research, in short.

See also: https://github.com/NLeSC/RSE-research [Lamprecht et al. 2022]

References

[Barbie et al. 2021] Barbie, A., Pech, N., Hasselbring, W., Flögel, S., Wenzhöfer, F., Walter, M., Shchekinova, E., Busse, M., Türk, M., Hofbauer, M. und Sommer, S.: “Developing an Underwater Network of Ocean Observation Systems with Digital Twin Prototypes - A Field Report from the Baltic Sea.” IEEE Internet Computing. 2021. DOI https://doi.org/10.1109/MIC.2021.3065245.

[Chue Hong 2022] N. P., Chue Hong, et al. (2022). FAIR Principles for Research Software version 1.0. (FAIR4RS Principles v1.0). Research Data Alliance. DOI https://doi.org/10.15497/RDA00068

[Claus et al. 2022] Claus, M., Gundlach, S., Hasselbring, W., Jung, R., Rath, W. und Schnoor, H. (2022) Modularizing Earth system models for interactive simulation. Informatik Spektrum, 45. pp. 300-303. DOI https://doi.org/10.1007/s00287-022-01490-z.

[Fuller and Millett 2011] S.H. Fuller and L.I. Millett, “Computing Performance: Game Over or Next Level?,” Computer, vol. 44, no. 1, 2011, pp. 31–38. DOI https://doi.org/10.1109/MC.2011.15

[Goltz et al.,2015] U. Goltz et al., “Design for Future: Managed Software Evolution,” Computer Science - Research and Development, vol. 30, no. 3, 2015, pp. 321–331. DOI https://doi.org/10.1007/s00450-014-0273-9

[Hasselbring et al. 2020a] W. Hasselbring, L. Carr, S. Hettrick, H. Packer, T. Tiropanis: “Open Source Research Software”. In: Computer, 53 (8), pp. 84-88. 2020. DOI https://doi.org/10.1109/MC.2020.2998235

[Hasselbring et al. 2020b] W. Hasselbring, L. Carr, S. Hettrick, H. Packer, T. Tiropanis: “From FAIR Research Data toward FAIR and Open Research Software”, it - Information Technology, 2020. DOI https://doi.org/10.1515/itit-2019-0040

[Hasselbring et al. 2020c] Hasselbring, W., Krause, A., Zirkelbach, C.: “ExplorViz: Research on software visualization, comprehension and collaboration.” In: Software Impacts, 6, 2020. DOI https://doi.org/10.1016/j.simpa.2020.100034.

[Hasselbring and van Hoorn 2020] Hasselbring, W., van Hoorn, A.: “Kieker: A monitoring framework for software engineering research.” In: Software Impacts, 5, 2020. pp. 1-5. DOI https://doi.org/10.1016/j.simpa.2020.100019

[Henning and Hasselbring 2021] Henning, S., Hasselbring, W.: “The Titan Control Center for Industrial DevOps analytics research,” In: Software Impacts, 7, 2021 . DOI https://doi.org/10.1016/j.simpa.2020.100050

[Johanson et al. 2016] A. Johanson, S. Flögel, C. Dullo, W. Hasselbring: “OceanTEA: Exploring Ocean-Derived Climate Data Using Microservices”. In: Sixth International Workshop on Climate Informatics (CI 2016), 2016, DOI http://dx.doi.org/10.5065/D6K072N6

References

[Johanson et al. 2017] A. Johanson, S. Flögel, C. Dullo, P. Linke, W. Hasselbring: “Modeling Polyp Activity of Paragorgia arborea Using Supervised Learning”, In: Ecological Informatics, 39. pp. 109-118. 2017, DOI https://doi.org/10.1016/j.ecoinf.2017.02.007

[Johanson & Hasselbring 2017] A. Johanson, W. Hasselbring: “Effectiveness and efficiency of a domain-specific language for high-performance marine ecosystem simulation: a controlled experiment”, In: Empirical Software Engineering 22 (8). pp. 2206-2236, 2017. DOI https://doi.org/10.1007/s10664-016-9483-z

[Johanson & Hasselbring 2018] A. Johanson, W. Hasselbring: “Software Engineering for Computational Science: Past, Present, Future”, In: Computing in Science & Engineering, 2018. DOI https://doi.org/10.1109/MCSE.2018.021651343

[Jung et al. 2021] R. Jung, S. Gundlach, S. Simonov, W. Hasselbring: “Developing Domain-Specific Languages for Ocean Modeling”. In: Software Engineering 2021 Satellite Events, http://ceur-ws.org/Vol-2814/

[Jung et al. 2022a] R. Jung, S. Gundlach, W. Hasselbring: “Software development processes in ocean system modeling.” In: International Journal of Modeling, Simulation, and Scientific Computing, 13 (02). 2022, DOI https://doi.org/10.1142/S1793962322300023.

[Jung et al. 2022b] R. Jung, S. Gundlach, W. Hasselbring: “Thematic domain analysis for ocean modeling.” In: Environmental Modelling & Software, 150. p. 105323. 2022, DOI https://doi.org/10.1016/j.envsoft.2022.105323.

[Lamprecht et al. 2020] A.-L. Lamprecht et al.: “Towards FAIR principles for research software.” In: Data Science 3, 1 (June 2020), 37–59. DOI https://doi.org/10.3233/ds-190026

[Lamprecht et al. 2022] A.-L. Lamprecht et al.: “What Do We (Not) Know About Research Software Engineering?.” In: Journal of Open Research Software, 10(1). 2022. DOI https://doi.org/10.5334/jors.384

[Merali 2010] Z. Merali, “Computational Science: Error, Why Scientific Programming Does Not Compute,” Nature, vol. 467, no. 7317, 2010, pp. 775–777, DOI https://doi.org/10.1038/467775a

[Randell 2018] B. Randell: 50 years of Software Engineering - or - The View from Garmisch. May 2018, DOI https://doi.org/10.48550/arXiv.1805.02742

[Reussner et al. 2019] R. Reussner, M. Goedicke, W. Hasselbring, B. Vogel-Heuser, J. Keim, L. Märtin, L. (Eds.): “Managed Software Evolution”, Springer, 2019. DOI https://doi.org/10.1007/978-3-030-13499-0