Informatik Spektrum
https://doi.org/10.1007/s00287-022-01490-z

HAUPTBEITRAG

Modularizing Earth system models for interactive simulation

Martin Claus[1] · Sven Gundlach[2] · Wilhelm Hasselbring[2] · Reiner Jung[2] · Willi Rath[1] · Henning Schnoor[2]

Accepted: 20 July 2022

Abstract
Interactive exploration of Earth system simulations may have great potential to improve the scientific modeling process. It will allow monitoring of the state of the simulation via dashboards presenting real-time diagnostics within a digital twin world. We present the state of the art for Earth system modeling in this context. Cross-domain data handling and fusion will make it possible to integrate model and observation data in the context of digital twins of the ocean. Domain-driven modularization of monolithic Earth system models allows one to recover interfaces for such a cross-domain fusion. Reverse engineering with static and dynamic analysis enables modularization of Earth system models. The modularization does not only help with restructuring existing Earth system models, it also makes it possible to integrate additional scientific domains into the interactive simulation environment.

Introduction

Earth System Models (ESMs) play a vital role in understanding and assessing climate change and other Earth system related issues. They are complex software systems, mainly programmed in Fortran, that undergo changes as science progresses. ESMs comprise different sub-models addressing specific aspects of the Earth system, such as ocean circulation. Their code is, partly, decades old. They can start as small models, which often evolve into large, complex models or are integrated into other models [15]. The ESMs are continually modified and enhanced to provide insights for specific research questions. This leads to numerous variants and versions for an ESM, which must be maintained as other scientists intend to base their research on new features and reproduce results. Thus, ESMs are long-living software and face typical risks, e.g., blurring module boundaries and architectural erosion, due to changes in functionality, hardware, and language features [7, 17]. The resulting architectural debts can limit further research and may harm the validity of scientific results. Furthermore, the lack of documentation and the loss of knowledge, as scientists move to other positions, limit the maintainability of ESMs. Our goal is to improve the understanding of the current architecture of ESM, modernizing and modularizing them to facilitate their quality and enable us to use them in interactive simulations.

Interactive exploration of Earth system simulations may have great potential to improve the scientific modeling process. It will allow monitoring of the state of the simulation via dashboards presenting real-time diagnostics within a digital twin world. Furthermore, it will be possible to attach assessment models, or any other software, to the Earth system simulation software without the need to intermingle their codes. Specific objectives include:

- Enable researchers and stakeholders to interactively work with simulated twins of the Earth system.
- Allow software to interact with, and retrieve data from the simulated twin at run-time.
- Research methods and techniques to support restructuring Earth system simulation software with appropriate software structure visualizations.

Martin Claus
mclaus@geomar.de

✉ Sven Gundlach
sven.gundlach@email.uni-kiel.de

Wilhelm Hasselbring
hasselbring@email.uni-kiel.de

Reiner Jung
reiner.jung@email.uni-kiel.de

Willi Rath
wrath@geomar.de

Henning Schnoor
henning.schnoor@email.uni-kiel.de

[1] GEOMAR Helmholtz Centre for Ocean Research, Kiel, Germany

[2] Universität Kiel, Kiel, Germany

Published online: 08 October 2022

- Open module APIs for reusable modules and flexible runtime interaction.
- The data required for cross-domain fusion is not usually directly accessible via well-defined interfaces with ESMs; thus we enable access to this hidden data.

In the next section we start with a look at the state of the art for ESM and cross-domain data handling for ESM. Reverse engineering with static and dynamic analysis for modularizing ESMs is introduced in the following section. Employing digital twins for interactive simulation of ESMs is discussed, before we conclude the paper.

State of the art

In Earth science, the primary research activities are Earth observation and Earth modeling, followed by data analysis and data comparison [14].

Earth system models

ESMs consist of multiple components representing different subsystems of the natural Earth. These subsystems include the ocean, the atmosphere, the cryosphere, the lithosphere as well as the bio-sphere, and their respective model components are often further divided into components representing physical, chemical, or biological processes. There are multiple ESMs that differ in the specific choice of components for the different subsystems of the Earth, in the choice of parameters for each component, and in the details of the coupling between the components.

The different ESM components are typically developed by small independent groups of scientists. Development is driven by scientific requirements and hence largely follows an incremental path focused on adding features without an overall architecture concept and without clear abstractions and without a concept for formal specifications and testing [12]. Often, the individual ESM model components serve a dual purpose of being used standalone in simulations of the subsystem of the natural Earth they represent and also being used as a component in a coupled ESM. While developers of the different components typically coordinate the scientific features of their developments (which Earth system processes to represent in which way), there is little coordination on the software engineering level.

Typical ESMs are optimized for minimal execution time on high-performance computing systems. The inputs to the ESM software (parameters, input data, initial conditions, etc.) are collected in one or multiple files on disk and then read at runtime during unattended batch processing. The only way to interact with the software is to manipulate the inputs before starting execution. Because of the lack of an architectural concept and due to the absence of clear abstractions, the level of detail at which the inputs to the ESM software can be modified is very heterogeneous and depends on which configuration aspects have been important over the history of the individual ESM components. An inspection of or an active interaction/intervention with the state of the ESM at runtime is typically not possible. This is on one hand due to the fact that high-performance computing systems typically do not support interactive workflows very well, and hence there has been no effort to develop interfaces enabling inspection of or intervention with the ESM at runtime. On the other hand, the internal structure of the different ESM components was never designed to allow for interaction.

Cross-domain data handling in models

There are two classes of inputs to and outputs from typical ESM software: Configuration and data (parameter settings) [13]. Configuration determines the behavior and the capabilities of the software (simulated physical or biological processes, numerical details, etc.). ESM software sometimes also generates the configuration for a possible next cycle of the simulation. Input data determine or influence the initial or intermediate states of the simulated system or act as a pre-determined representation of a part of the Earth system that is not actively simulated in the software but still interacts with the simulated system in the real world (e.g., the atmospheric state represented by data in a simulation of the ocean circulation). Output data represents different aspects of the initial, intermediate, or final state of the simulated state. A typical ESM simulation run produces aggregated (e.g. time averaged) or instantaneous snapshots of the simulated state.

Each ESM component typically has its own requirements for file formats, metadata, and naming conventions. While there is some agreement on the use of the Network Common Data Form (netCDF) [18] and sometimes Climate and Forecast (CF) conventions [4] for metadata, this agreement does not mean that the same input data can be used for different ESM software programs. Furthermore, many ESM components resort to custom input and output formats, because achieving sufficient input-output performance with netCDF in a massively parallel setting is not trivial.

Reverse engineering of ESMs

Essential to migrating an ESM from a monolithic state into an interactive simulation is an understanding of its architecture, sub-models, and interfaces between these sub-models, as well as suggestions for restructuring an ESM. Reconstructing the software architecture [8, 9] is then the basis

for recommending restructuring options. Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction [3]. Our first step for reverse engineering is to recover the software architecture based on static source code analysis and dynamic runtime observations. The source code analysis identifies procedure calls, direct accesses to shared global variables, and dataflow traces. The runtime monitoring allows one to observe and identify call-traces, is able to inspect library calls, and provides information on how often certain functionality is used.

Static analysis ESMs are often programmed in Fortran, which uses C/C++ preprocessor directives to, e.g., include files with the #include directive. To perform the static analysis, we run the cpp-preprocessor with the same configuration options as it is run in the compilation process for the ESMs. On the resulting Fortran files, we perform a static analysis using a custom Python program built upon the *fparser2* [6] tool to discover coupling resulting from call relationships and dataflows.

Dynamic analysis We use the Kieker framework and tooling [10, 11] to obtain call traces from running ESMs. We execute the ESMs utilizing configurations derived from scientific publications. Kieker is a versatile monitoring and analysis framework and tool suite initially designed for Java, supporting a wide range of application and system monitoring features.

Combining static and dynamic analysis data Based on dynamic and static data, we recover architectures and merge this information into a combined architecture model. The second step derives interfaces from the architectural information based on shared procedure use among components, e.g., sub-models, libraries, and files. We can then assess the quality of the architecture model or inspect the model visually with different views. Finally, we can compute suggestions for restructured architectures that reduce interdependencies between components or better separate concerns within the architecture.

Digital twins for interactive simulation of ESMs

Digital twins were created for industrial production and technology engineering processes, such as digital twins for ocean observation systems [1]. Their goal is to optimize design and operation of complex processes through a highly interconnected workflow, combining a digital replica of the process with real-time observations of the physical system. The Earth system digital twin combines simulations and near-real-time observations to monitor the evolution of the Earth system [2]. For each cycle, the simulation generates a background forecast data ensemble of the Earth system, which is compared to observation data throughout a time window. The promise of digital twins of the ocean, atmosphere, and cryosphere of the Earth amounts to providing different ways of interacting with one or more Earth system simulation/assimilation systems. Today's Earth system simulations all rely on splitting the scientific process into a simulation/data production step and a subsequent data analysis process.

The promise of increased capabilities for inspection of a digital twin can be envisioned without giving up the split into data production and data analysis. However, achieving true intervention requires interaction of researchers with an Earth system simulation while it is running. Furthermore, many applications using the online output of an Earth simulation software, such as assessment models, would benefit from data at higher resolution than usually written to persistent storage, and hence the divide between data production and data analysis needs to be overcome.

Conclusions

We expect that interactive exploration of Earth system simulations and observations may have great potential to improve the scientific discovery process. Our approaches to address this include the following:

- Cross-domain data handling and fusion to integrate model and observation data in the context of digital twins of the ocean.
- Domain-driven modularization of monolithic ESMs allows one to recover interfaces for such a cross-domain fusion.
- The modularization does not only help with restructuring existing ESM, it also allows one to integrate additional scientific domains into the interactive simulation environment.

Based on the reverse engineering of ESM architectures, we intend to select bounded contexts, i.e., the foundation for decomposing the monolithic ESMs into modules when using Domain-Driven Design [5]. Such an approach was successful in the domain of information systems [16], and we intend to transfer this to the domain of ESM.

Our key goals are to answer the following research questions:

- How to design the flow of information within and into/out of Earth system simulation software to best support digital twin applications?

- How to adapt and transform Earth system simulation software to enable more interactions with scientists and stakeholders?
- How can digital-twin methods and techniques support the effort to restructure Earth system simulation software?

This twin-enabled interactive Earth system simulation software design aims at rethinking the flow of information within and into/out of Earth system simulation software by bringing together experts from the field of software engineering, ESM development, and the field of application development and user interface design.

Acknowledgement This Work is funded by KMS Kiel Marine Science—Centre for Interdisciplinary Marine Science at Kiel University.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Barbie A et al (2022) Developing an underwater network of ocean observation systems with digital twin prototypes—a field report from the baltic sea. In: IEEE Internet Computing https://doi.org/10.1109/mic.2021.3065245
2. Bauer P et al (2021) The digital revolution of earth-system science. In: Nature computational science. https://doi.org/10.1038/s43588-021-00023-0
3. Canfora G, Di Penta M (2007) New frontiers of reverse engineering. Future Softw Eng. https://doi.org/10.1109/FOSE.2007.15
4. Eaton B et al (2021) NetCDF Climate and Forecast (CF) Metadata Conventions v1.9. http://cfconventions.org/Data/cf-conventions/cf-conventions-1.9/cf-conventions.html. Accessed 30 Mar 2022
5. Evans E (2004) Domain-driven design: tackling complexity in the heart of software. Addison-Wesley
6. (2022) f2py. fparser package. https://github.com/stfc/fparser. Accessed 1 Apr 2022
7. Goltz U et al (2015) Design for future: managed software evolution. Comput Sci Dev 30(3):321–331. https://doi.org/10.1007/s00450-014-0273-9
8. Hasselbring W (2018) Software architecture: past, present, future. In: The essence of software engineering. Springer, Cham, pp 169–184 https://doi.org/10.1007/978-3319-73897-0_10
9. Hasselbring W (2006) Software-Architektur – Das aktuelle Schlagwort. Informatik Spektrum 29(1):48–52. https://doi.org/10.1007/s00287-005-0049-5
10. Hasselbring W, Hoorn A van (2020) Kieker: a monitoring framework for software engineering research. Softw Impacts. https://doi.org/10.1016/j.simpa.2020.100019
11. Hoorn A van, Waller J, Hasselbring W (2012) Kieker: a framework for application performance monitoring and dynamic software analysis. In: Proceedings of the 3rd ACM/SPEC international conference on performance engineering (ICPE 2012). ACM, pp 247–248 https://doi.org/10.1145/2188286.2188326
12. Johanson A, Hasselbring W (2018) Software engineering for computational science: past, present, future. Comput Sci 4 Eng 20(2):90–109. https://doi.org/10.1109/MCSE.2018.021651343
13. Jung R, Gundlach S, Hasselbring W (2021) CP-DSL: supporting configuration and parametrization of ocean models with UVic (2.9) and MITgcm (67w). In: Geoscientific model development https://doi.org/10.5194/gmd-2021-311
14. Jung R, Gundlach S, Hasselbring W (2022) Software development processes in ocean system modeling. Int J Model Simul Sci Computing. https://doi.org/10.1142/s1793962322300023
15. Jung R, Gundlach S, Hasselbring W (2022) Thematic domain analysis for ocean modeling. Environ Model 4 Softw 150:105323. https://doi.org/10.1016/j.envsoft.2022.105323
16. Krause A et al (2020) Microservice decomposition via static and dynamic analysis of the monolith. In: Proceedings of the IEEE international conference on software architecture com-panion (ICSA-C), pp 9–16 https://doi.org/10.1109/ICSA-C50368.2020.00011
17. Reussner R et al (2019) Managed software evolution. Springer, Cham https://doi.org/10.1007/978-3-030-13499-0
18. Rew R, Davis G (1990) NetCDF: an interface for scientific data access. IEEE Comput Grap Appl 10(4):76–82. https://doi.org/10.1109/38.56302 (Conference Name: IEEE Computer Graphics and Applications)