Software Engineering for Computational Science: Tests, Modules, Domain-Specific Languages, Flows

Wilhelm (Willi) Hasselbring

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

CRC 1404 FONDA, April 27th, 2021

Kiel University
Christian-Albrechts-Universität zu Kiel

HELMHOLTZ
SCHOOL FOR MARINE
DATA SCIENCE

Agenda

1. Research Software
2. Research Software Engineering
   - Automated testing
   - Modular Software
     - Modular commercial software
     - Modular research software
   - Domain-specific software engineering
   - Flow-based programming
3. Summary & Outlook

Research Software

GESELLSCHAFT FÜR FORSCHUNGSSOFTWARE

- Research software is software
  - that is employed in the scientific discovery process or
  - a research object itself.
- Computational science (also scientific computing) involves the development of research software
  - for model simulations and
  - data analytics

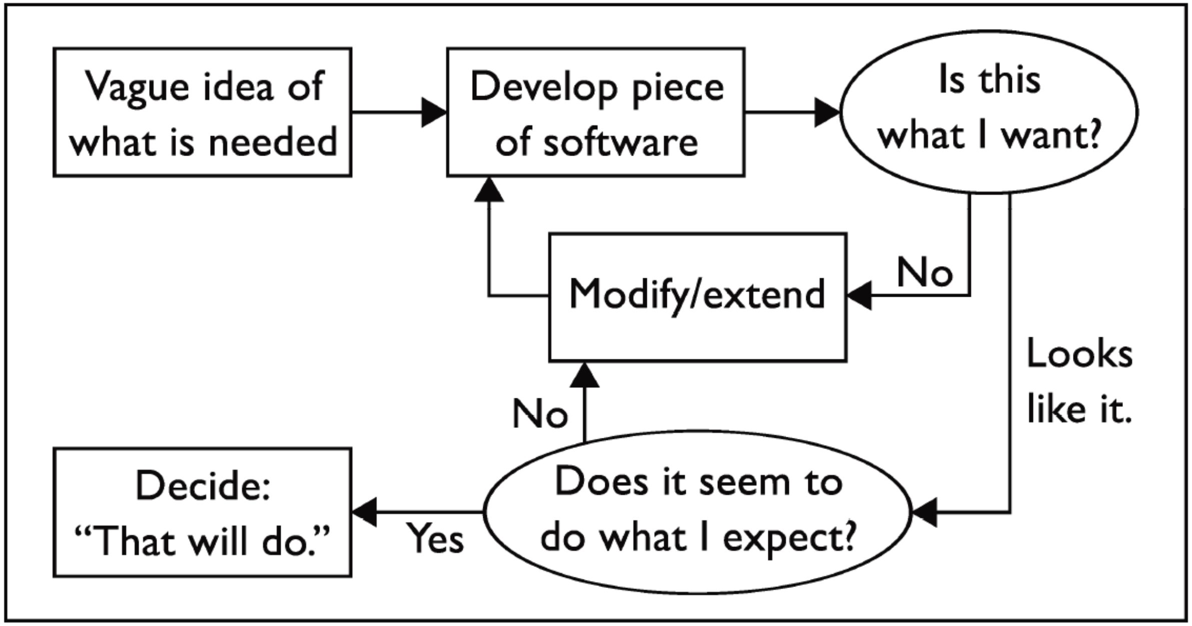
  to understand natural systems answering questions that neither theory nor experiment alone are equipped to answer.

RSE

SOCIETY OF RESEARCH SOFTWARE ENGINEERING

Characteristics of Research Software

- **Functional Requirements** are not known up front
  - And often hard to comprehend without some PhD in science
- **Verification** and validation are difficult,
  - and strictly scientific
- Overly formal software **processes** restrict research

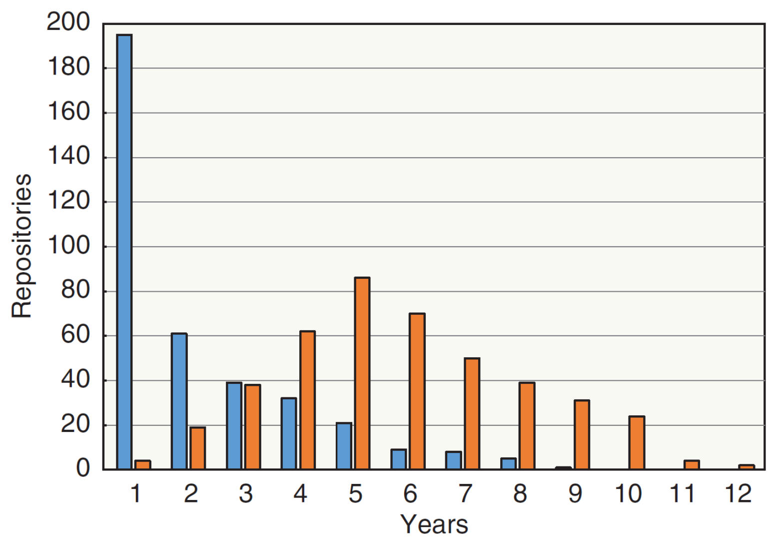


Characteristics of Research Software

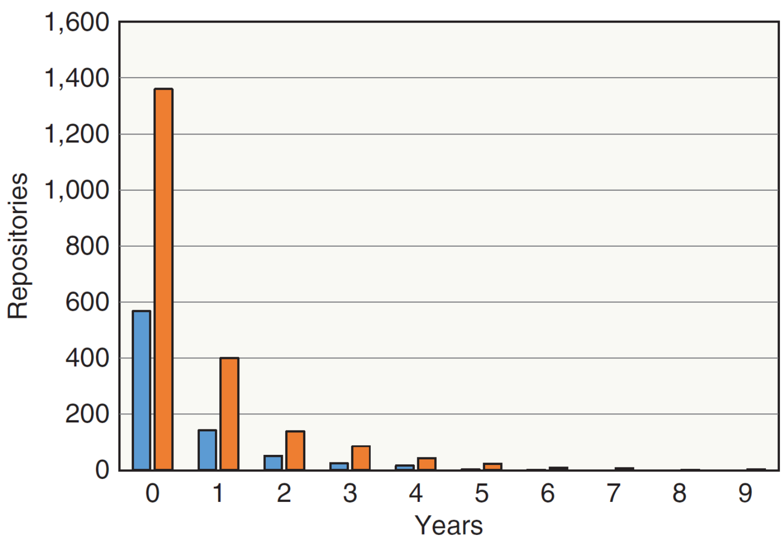
- Software **quality requirements**
  - Jeffrey Carver and colleagues[22] found that scientific software developers rank the following characteristics as the most important, in descending order [Carver et al. 2007]:
    1. functional (scientific) correctness,
    2. performance,
    3. portability, and
    4. maintainability.
- Research software in itself has **no value**
  - Not really true for community software
- Few scientists are **trained** in software engineering
  - Disregard of most modern software engineering methods and tools

Sustainability of Research Software

- Research software publishing practices in computer science and in computational science show significant differences:
  - computational science emphasizes **reproducibility**,
  - computer science emphasizes **reuse**.



Lifespan of Github repositories cited in computer science publications



Lifespan of Github repositories cited in computational science publications

[Hasselbring et al. 2020a]

SE for Research Software ?

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 ‘hassle,’ 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.
- Because of this ideal of **generality**, the question of how specifically computational scientists should develop their software in a well-engineered way would probably have perplexed a software engineer, whose answer might have been:
  - ‘Well, just like any other application software.’ ”

Software Carpentry

- Programming / Coding
  - Fortran, C++, Python, R, etc
  - Using compilers, interpreters, editors, etc
- Using version control (git etc)
- Team coordination (GitHub, Gitlab, etc)
- [Continuous integration (Jenkins, etc)]

software carpentry

Teaching basic lab skills
for research computing

https://software-carpentry.org/

So, SE 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

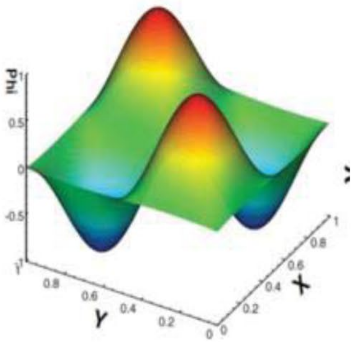
Editors:
Jeffrey Carver, carver@cs.ua.edu; Damian Rouson, damian@sourceryinstitute.org

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

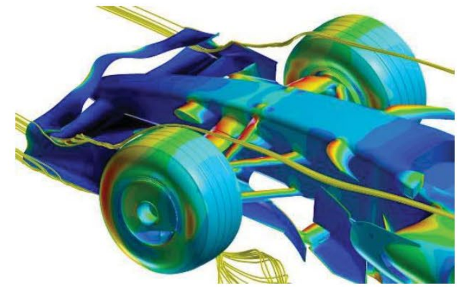
1. Research Software
2. Research Software Engineering
   - **Automated testing**
   - Modular Software
     - Modular commercial software
     - Modular research software
   - Domain-specific software engineering
   - Flow-based programming
3. Summary & Outlook

Testing the Untestable: Test Oracles?

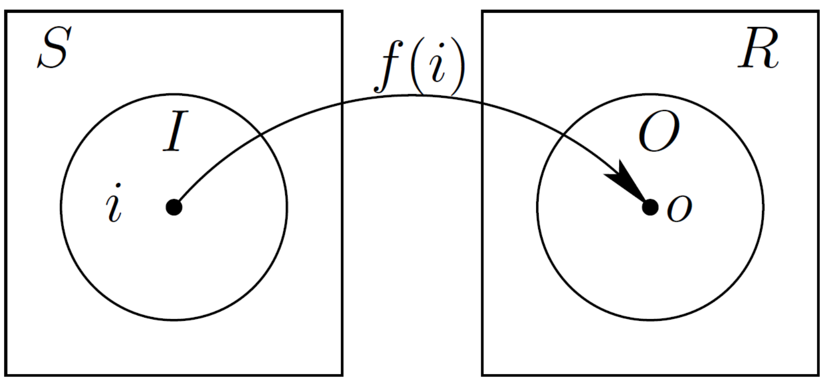


Scientific calculations

Artificial intelligence



Simulation and modelling



[Kanewala and Bieman 2014]

Stimulus and observations:

- *S* is anything that can change the observable behavior of the SUT *f*;
- *R* is anything that can be observed about the system's behavior;
- *I* includes *f*'s explicit inputs;
- *O* is its explicit outputs;
- everything not in *S* ∪ *R* neither affects nor is affected by f.

Metamorphic Testing

- The nature of research software is **exploratory**.
- Output is usually **unknown** and cost-intensive to compute.
- Hence it is challenging to validate using conventional testing methodology
- Metamorphic Testing provides an approach for testing software without test oracles
  - Validating software by comparing outputs of multiple runs with varying (morphed) input data
  - The central element of metamorphic testing is the metamorphic relation.
    - The input data is morphed based on this property
  - If the output is in accordance of the applied morphing to the input data, the test is asserted.

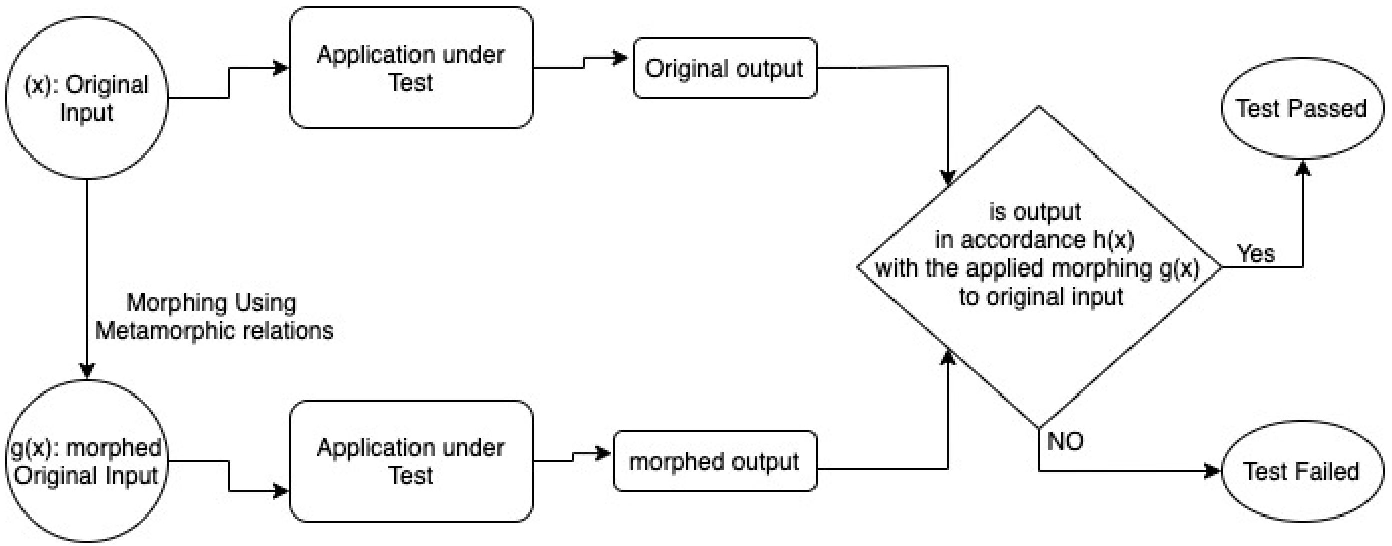
[Segura et al. 2020]

Metamorphic Testing for Ocean Models

Metamorphic testing may be defined as

$$f(g(\vec{x})) = h(f(\vec{x}))$$

- function under test $f: X \rightarrow Y$



Our Goal: To generate metamorphic test cases and metamorphic relations automatically via machine learning for verifying Ocean System Model applications [Hiremath et al. 2021]

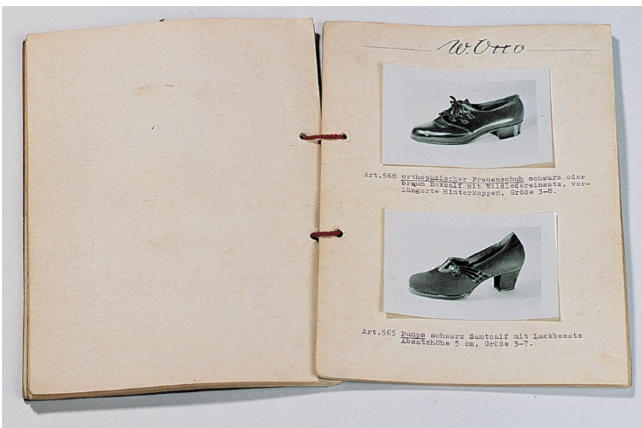
MARDATA HELMHOLTZ SCHOOL FOR MARINE DATA SCIENCE

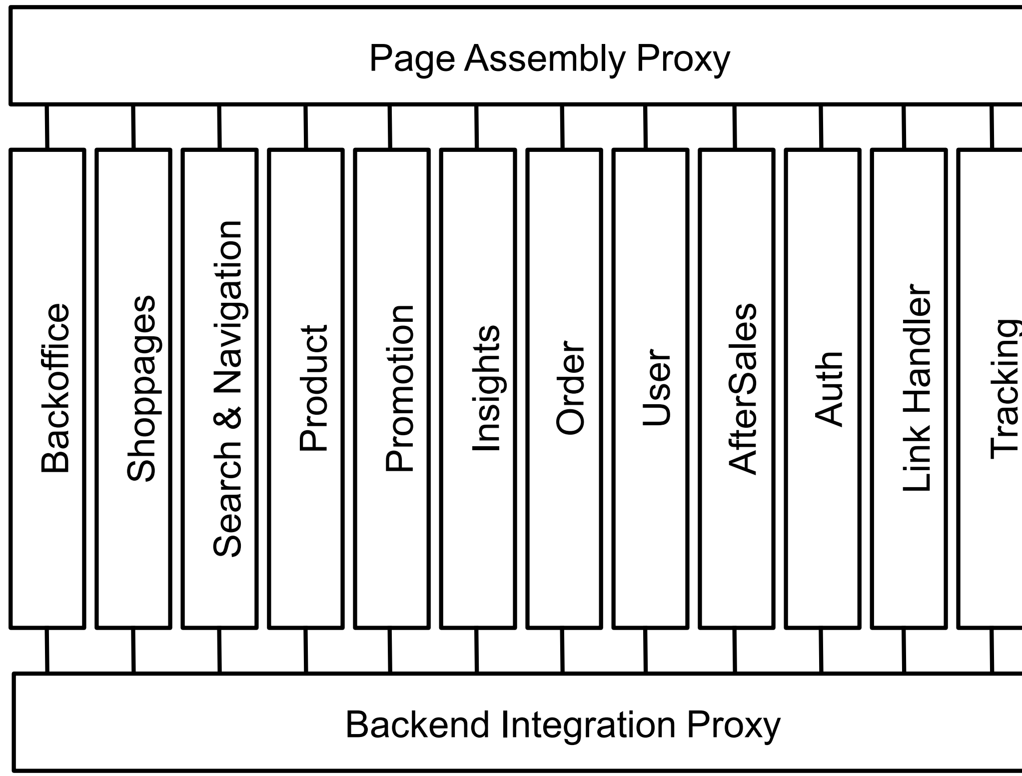
Agenda

1. Research Software
2. Research Software Engineering
   - Automated testing
   - Modular Software
     - **Modular commercial software**
     - Modular research software
   - Domain-specific software engineering
   - Flow-based programming
3. Summary & Outlook

Modular Commercial Software

Example: otto.de

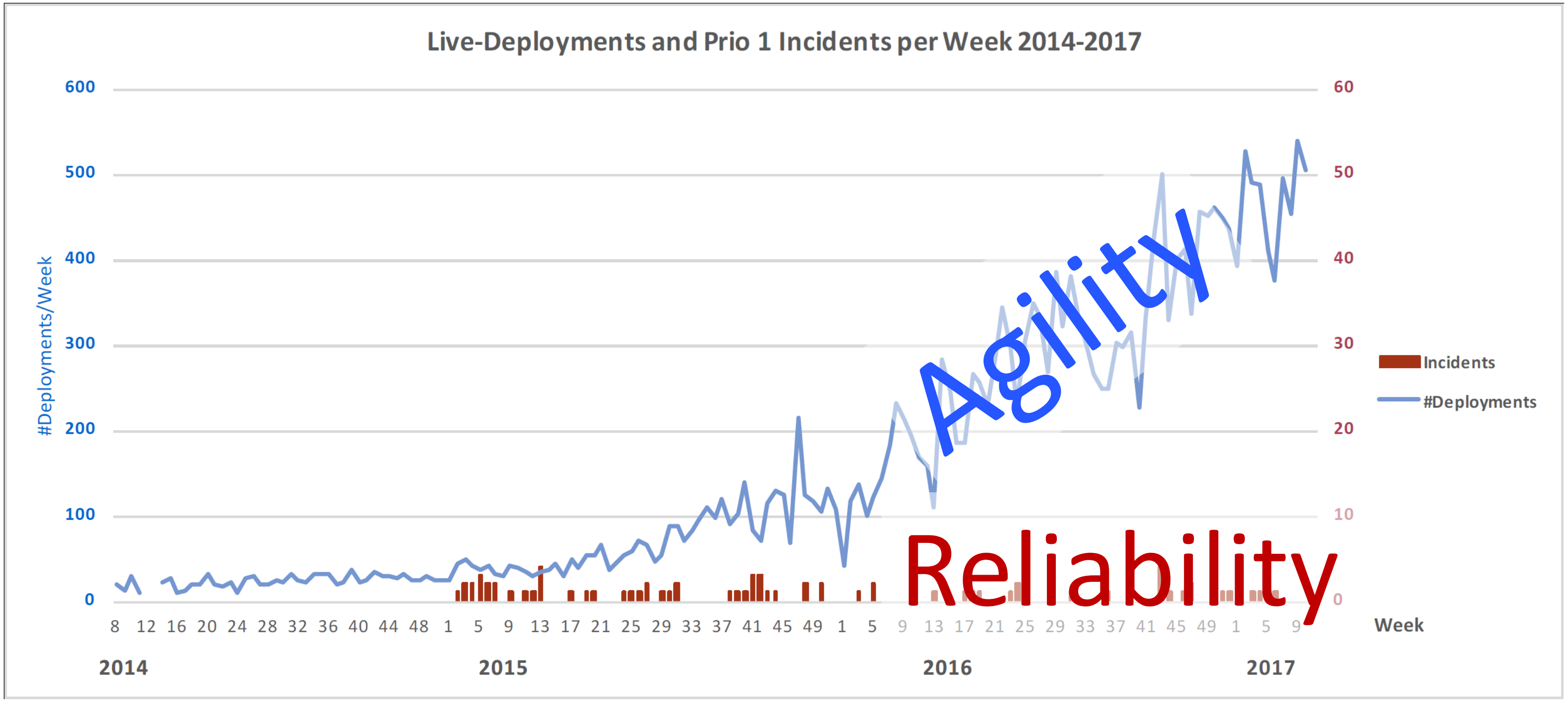




Microservices: [Hasselbring 2016, 2018, Hasselbring & Steinacker 2017, Knoche & Hasselbring 2019]

Modular Commercial Software

Example: otto.de



Scalability, Agility and Reliability [Hasselbring & Steinacker 2017]

Agenda

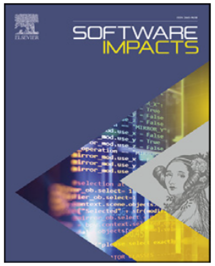
1. Research Software
2. Research Software Engineering
   - Automated testing
   - Modular Software
     - Modular commercial software
     - **Modular research software**
   - Domain-specific software engineering
   - Flow-based programming
3. Summary & Outlook

Modular Scientific Code

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts



Eulerian-Lagrangian fluid dynamics platform: The `ch4-project`

Enrico Calzavarini

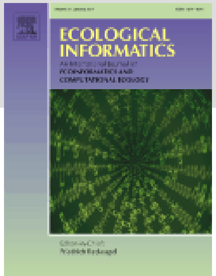
Highlights

- Ch4-project is a fluid dynamics code used in academia for the study of fundamental problems in fluid mechanics.
- It has contributed to the understanding of global scaling laws in non-ideal turbulent thermal convection.
- It has been used for the characterisation of statistical properties of bubbles and particles in developed turbulence.
- It is currently employed for a variety for research projects on inertial particle dynamics and convective melting.
- **Its modular code structure allows for a low learning threshold and to easily implement new features.**

Modular Scientific Code

[Calzavarini 2019]:

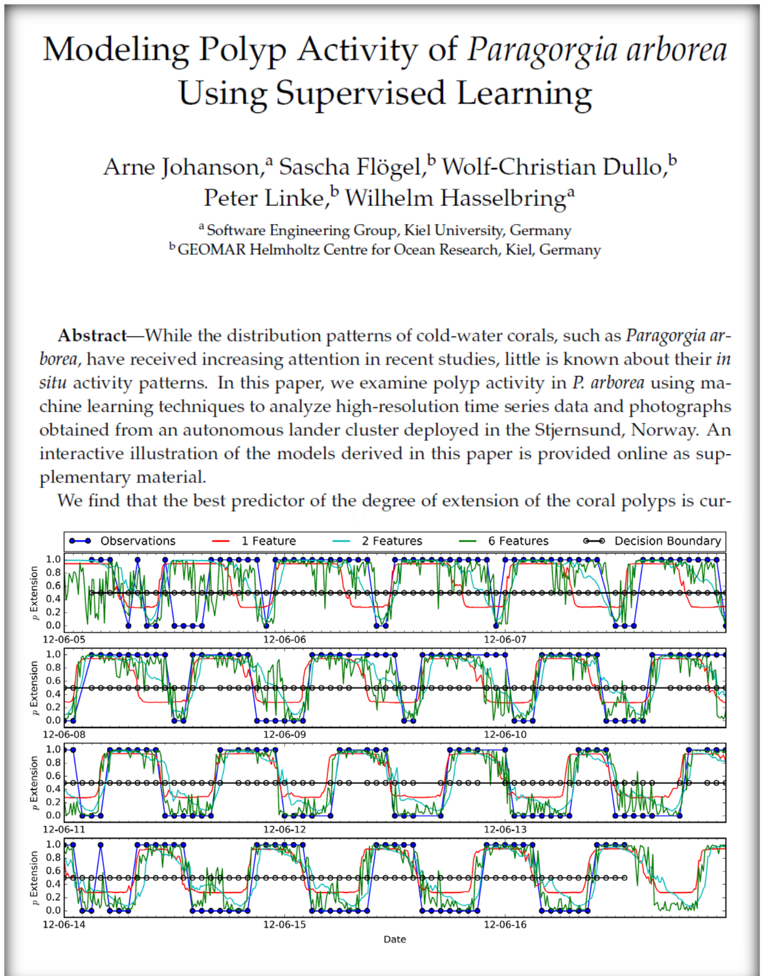
- “A dream for principal investigators in this field is to not have to deal with different (and soon mutually incompatible) code versions for each project and junior researcher in his/her own group.
- In this respect an **object-oriented modular** code structure would be the ideal one,
  - but this makes the code less prone to modifications by the less experienced users.
- The choice made here is to rely on a systematic use of **C language preprocessing** directives and on a **hierarchical naming convention** in order to configure the desired simulation setting in a module-like fashion at compiling time.”



Publishing Ocean Observation Data & Analytics

- Paper: [Johanson et al. 2017b]
- Code: https://github.com/cau-se/oceantea/
- Software service with data: https://oceantea.uni-kiel.de/

future ocean
KIEL MARINE SCIENCES



Modeling Polyp Activity of *Paragorgia arborea* Using Supervised Learning

Arne Johanson,[a] Sascha Flögel,[b] Wolf-Christian Dullo,[b] Peter Linke,[b] Wilhelm Hasselbring[a]

[a] Software Engineering Group, Kiel University, Germany
[b] GEOMAR Helmholtz Centre for Ocean Research, Kiel, Germany

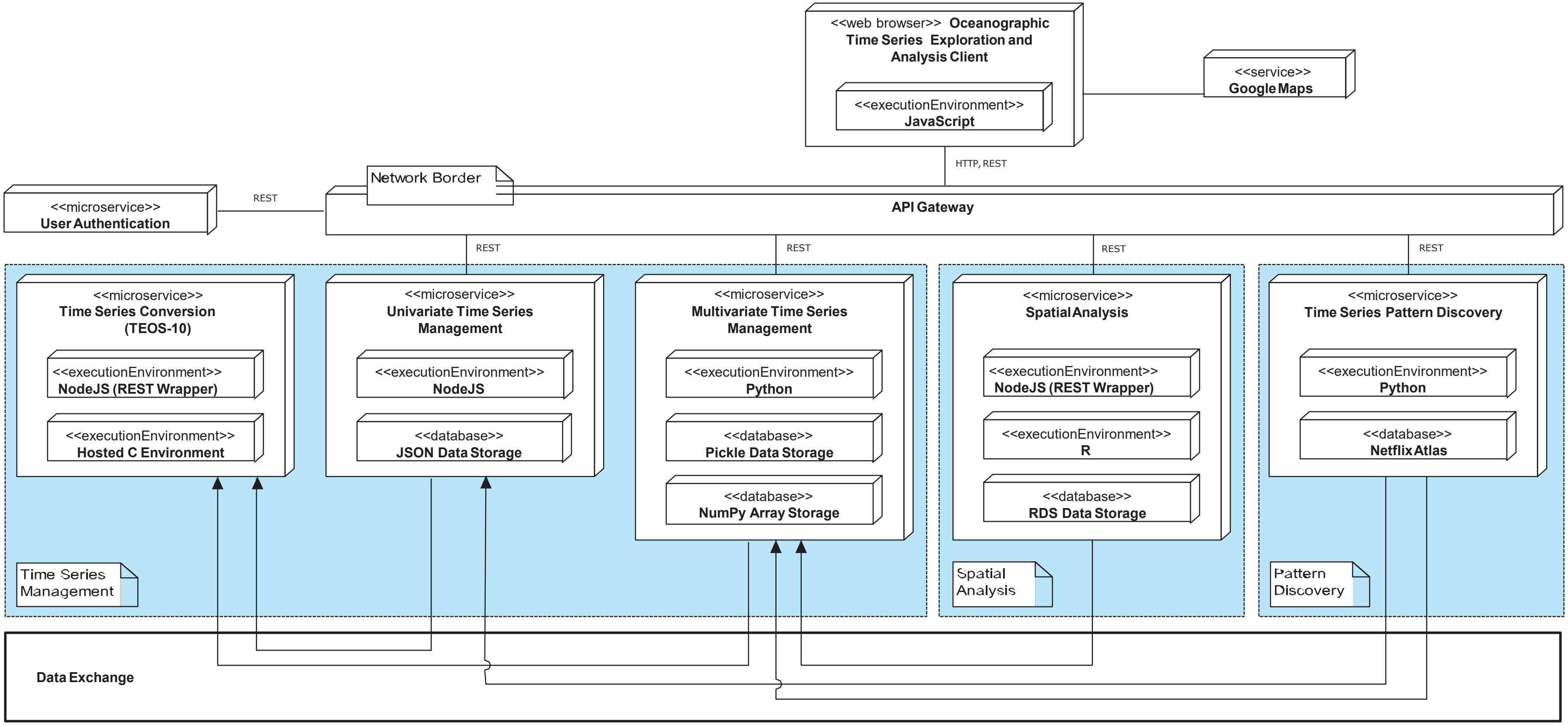
Abstract—While the distribution patterns of cold-water corals, such as *Paragorgia arborea*, have received increasing attention in recent studies, little is known about their *in situ* activity patterns. In this paper, we examine polyp activity in *P. arborea* using machine learning techniques to analyze high-resolution time series data and photographs obtained from an autonomous lander cluster deployed in the Stjernsund, Norway. An interactive illustration of the models derived in this paper is provided online as supplementary material.

We find that the best predictor of the degree of extension of the coral polyps is cur-

[Johanson et al. 2017b]

[Johanson et al. 2016a]

OceanTEA: Microservice-based Architecture



OceanTEA: [Johanson et al. 2016a]

Migrating toward Microservices

FOCUS: MICROSERVICES

IEEE SOFTWARE

Using Microservices for Legacy Software Modernization

Holger Knoche and Wilhelm Hasselbring, Kiel University

// *Microservices promise high maintainability, making them an interesting option for software modernization. This article presents a migration process to decompose an application into microservices, and presents experiences from applying this process in a legacy modernization project.* //

reduce coordination effort and improve team productivity.

It is therefore not surprising that companies are considering microservice adoption as a viable option for modernizing their existing software assets. Although some companies have succeeded in a complete rewrite of their applications,[2] incremental approaches are commonly preferred that gradually decompose the existing application into microservices.[3] Other approaches to modernization—e.g., restructuring and refactoring of existing legacy applications—are also valid options.[4] However, decomposing a large, complex application is far from trivial. Even seemingly easy questions like "Where should I start?" or "What services do I need?" can actually be very hard to answer.

In this article, we present a process to modernize a large existing software application using microservice principles, and report on experiences from implementing it in an ongoing industrial modernization project. We particularly focus on the process of actually decomposing the

[Knoche & Hasselbring 2018, Krause et al. 2020]

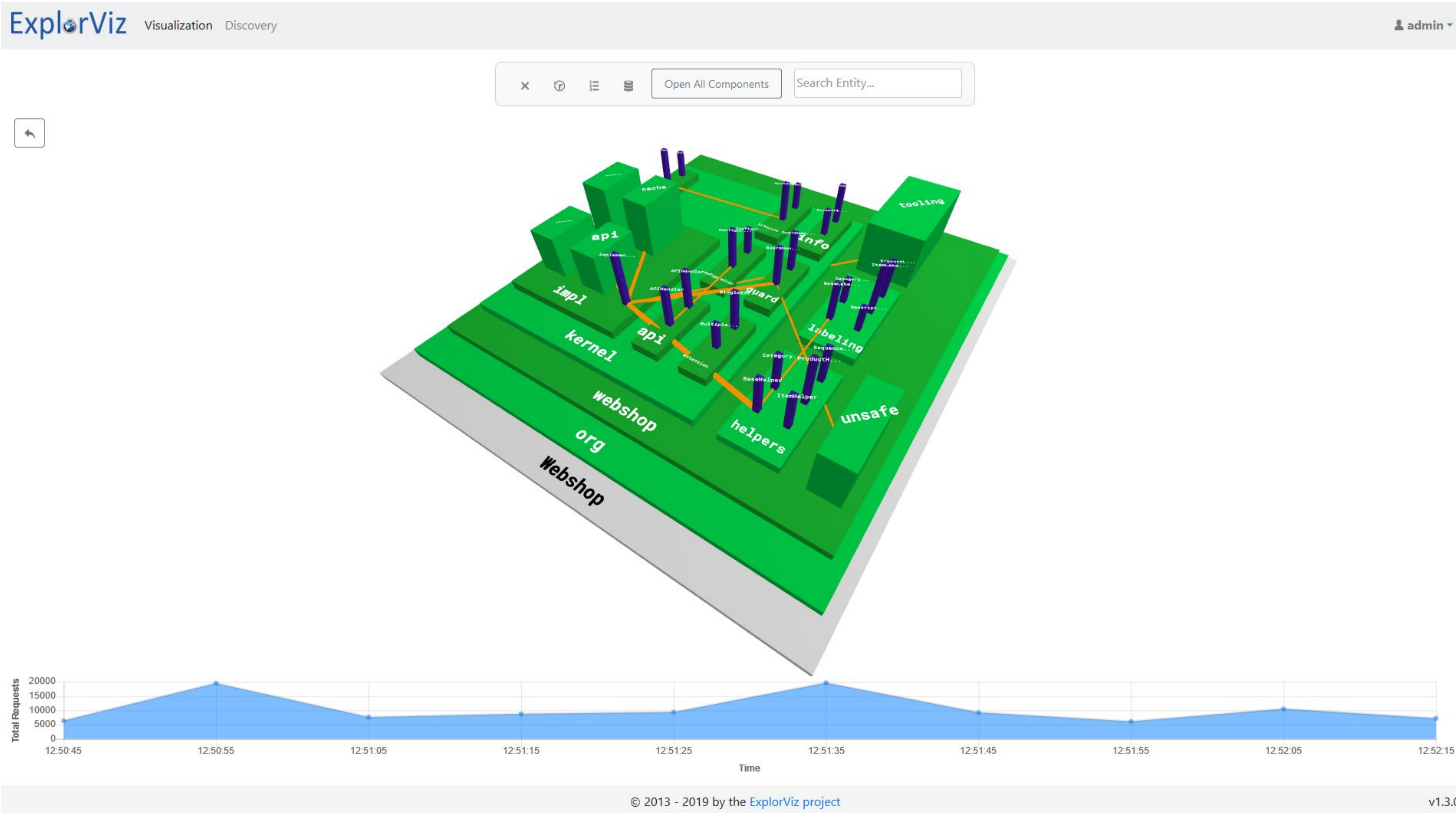
ExplorViz Live Trace Visualization Tool

https://www.explorviz.net
https://github.com/ExplorViz

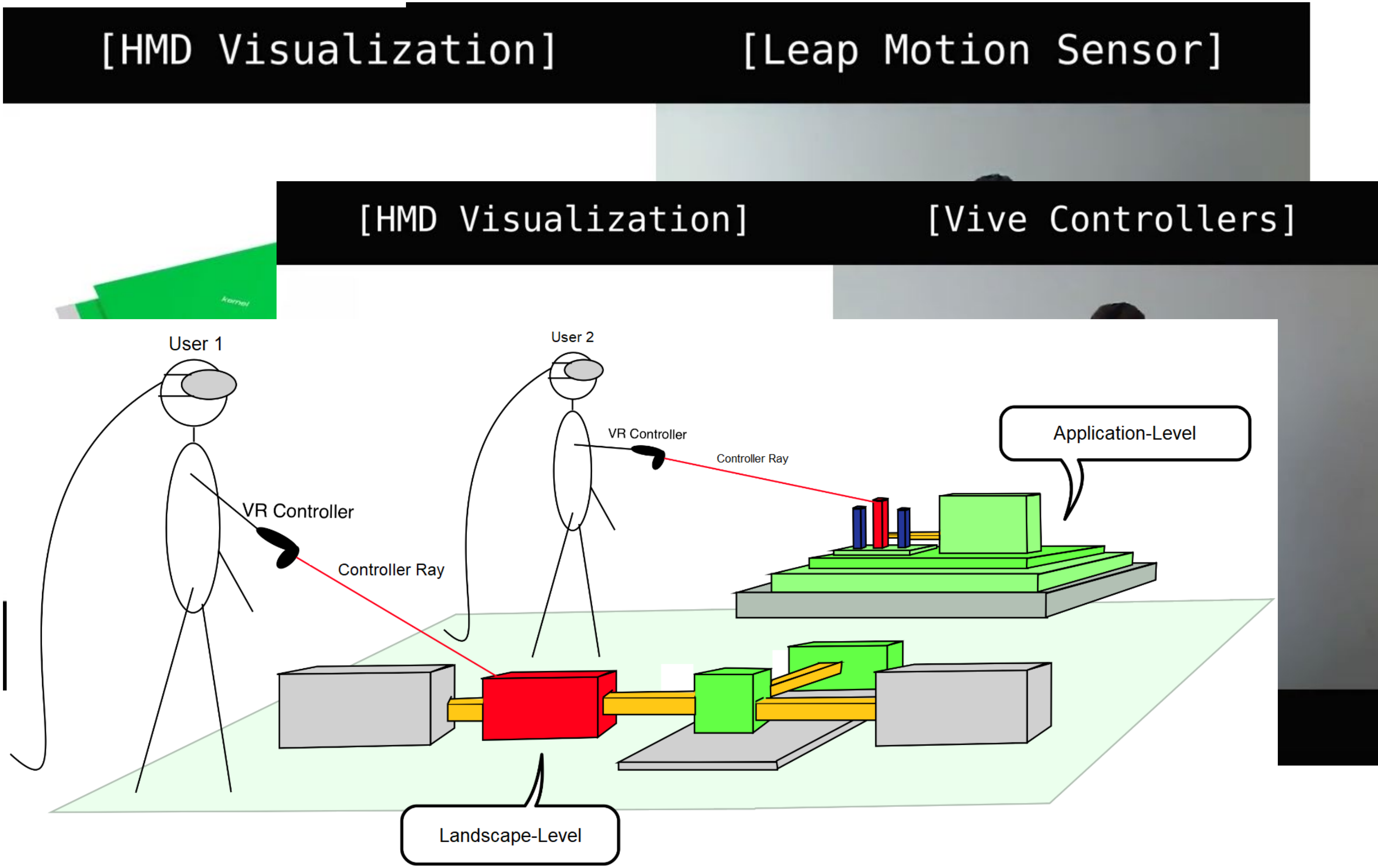
- Program- and system comprehension for software engineers
- Started as a Ph.D project in 2012
- Open Source from the beginning (Apache License, Version 2.0)
- Continuously extended over the years
- [Fittkau et al. 2013, 2015a-d, 2017;
  Krause et al. 2018, 2020;
  Zirkelbach et al. 2019, 2020;
  Hasselbring et al. 2020c]

ExplorViz

3D Application Visualization

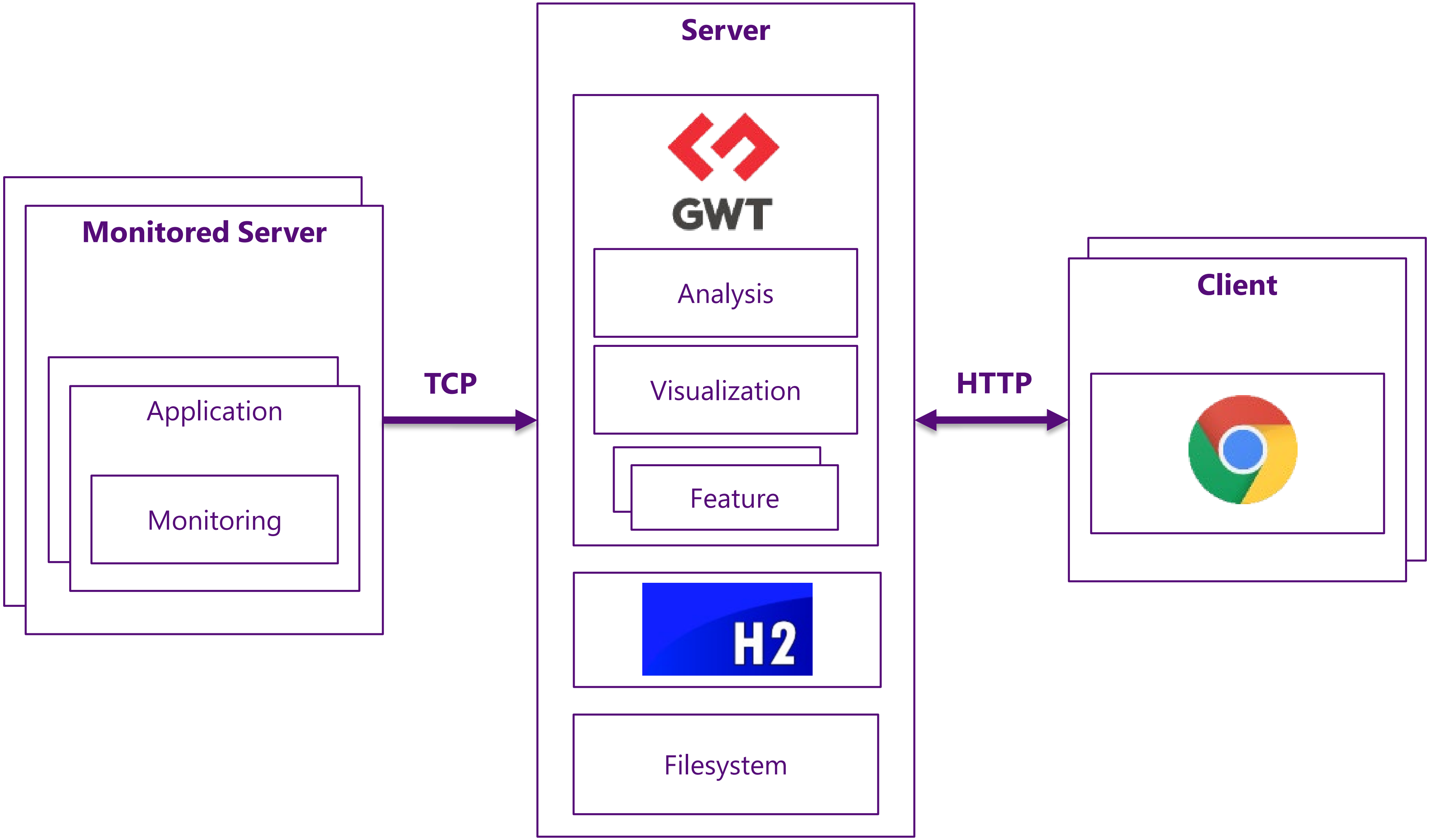


Some VR Extensions

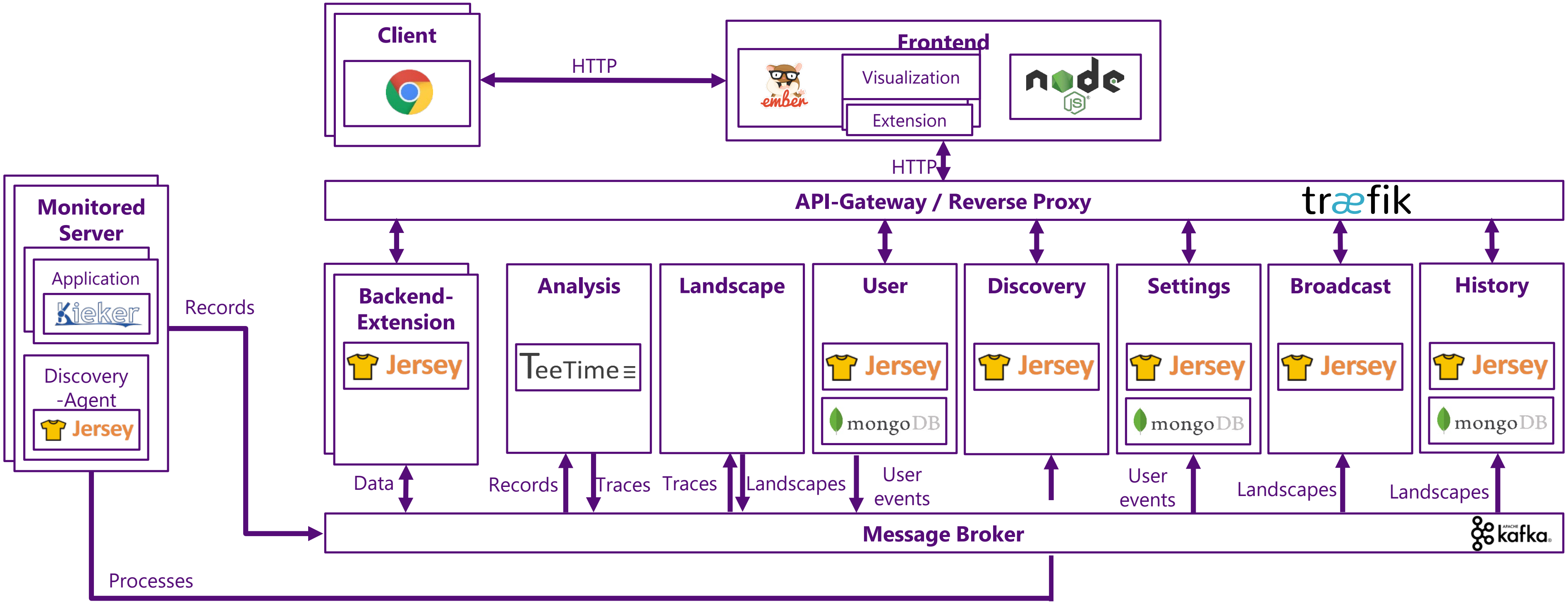


ExplorViz

Legacy Layered Architecture



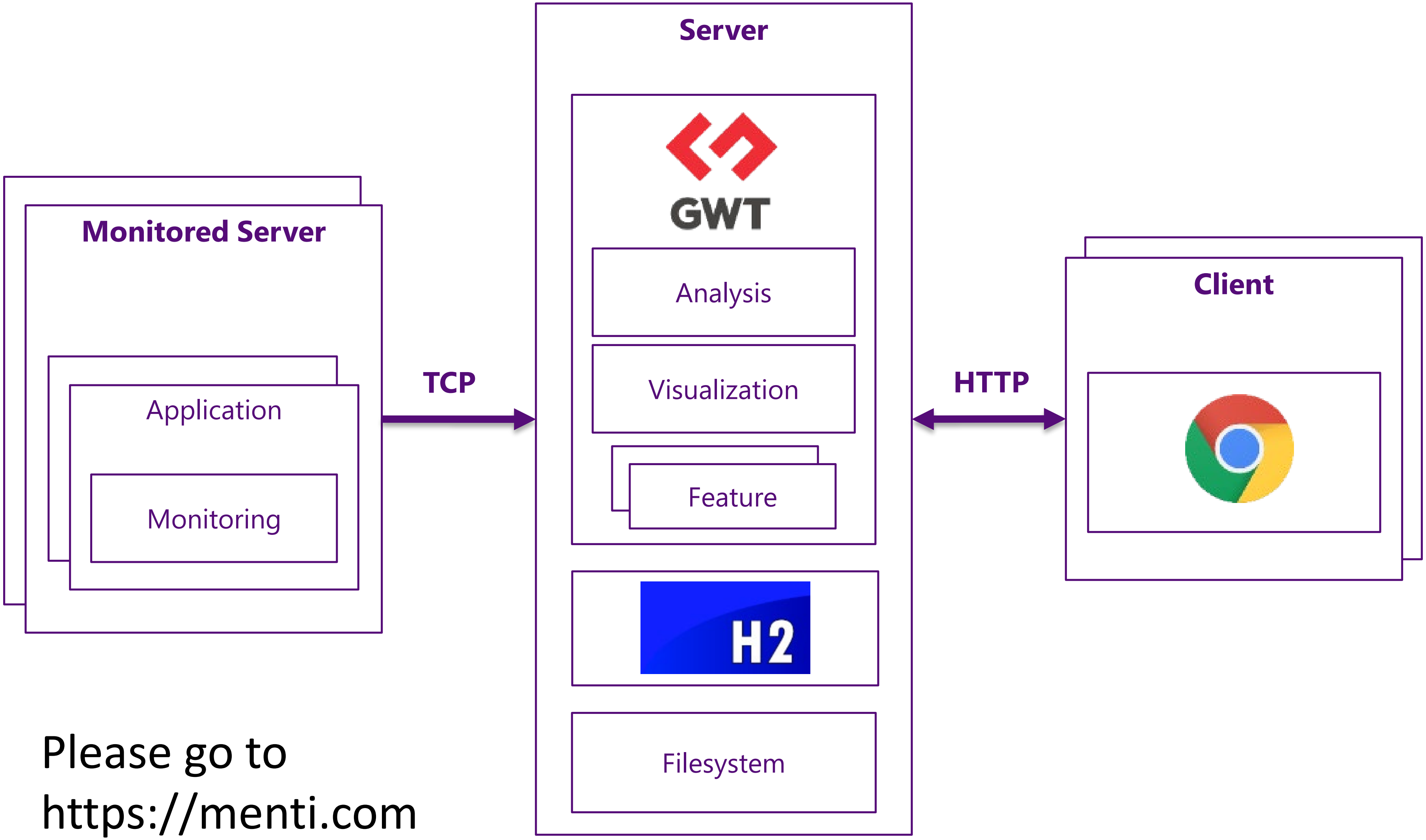
ExplorViz New Modular Architecture



More details in [Zirkelbach et al. 2019, 2020]

ExplorViz

Legacy Layered Architecture



Please go to https://menti.com

Migrating Computational Science Models ?

The software architecture of climate models [Alexander & Easterbrook 2015]

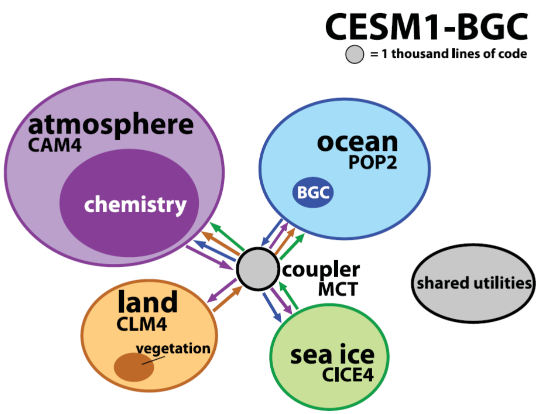


Figure 1. Architecture diagram for CESM1-BGC.

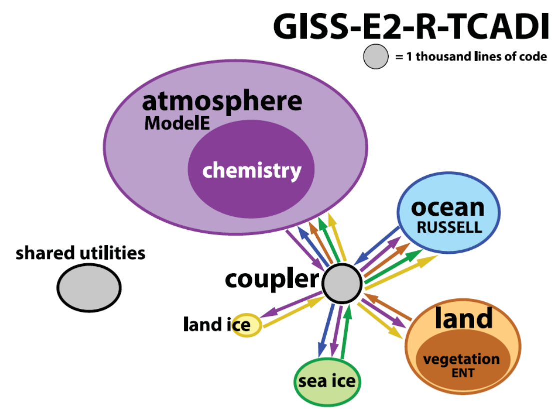


Figure 3. Architecture diagram for GISS-E2-R-TCADI.

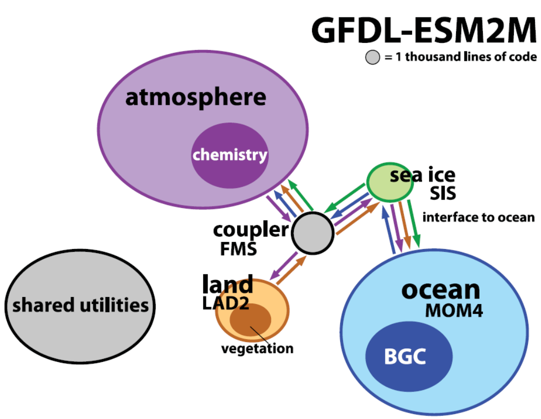


Figure 2. Architecture diagram for GFDL-ESM2M.

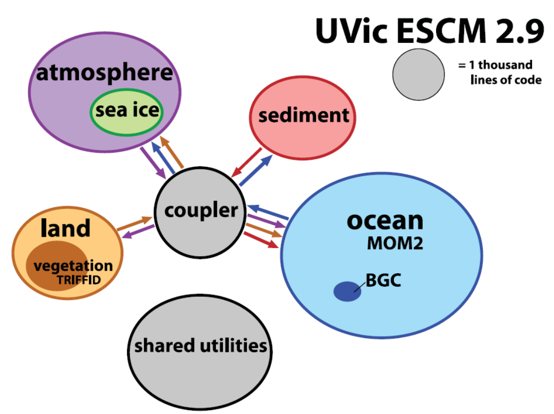
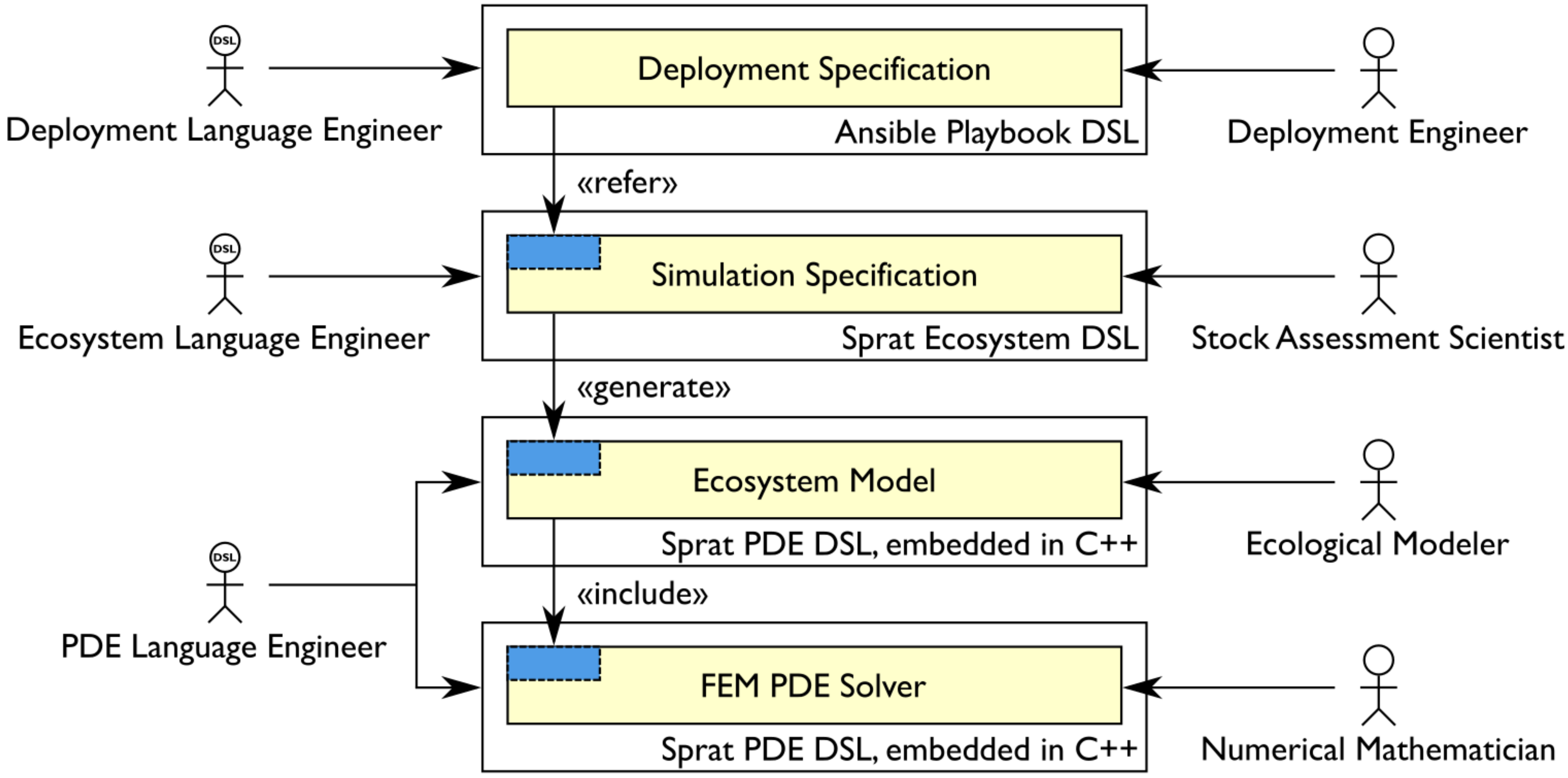


Figure 4. Architecture diagram for UVic ESCM 2.9.

Agenda

1. Research Software
2. Research Software Engineering
   - Automated testing
   - Modular Software
     - Modular commercial software
     - Modular research software
   - **Domain-specific software engineering**
   - Flow-based programming
3. Summary & Outlook

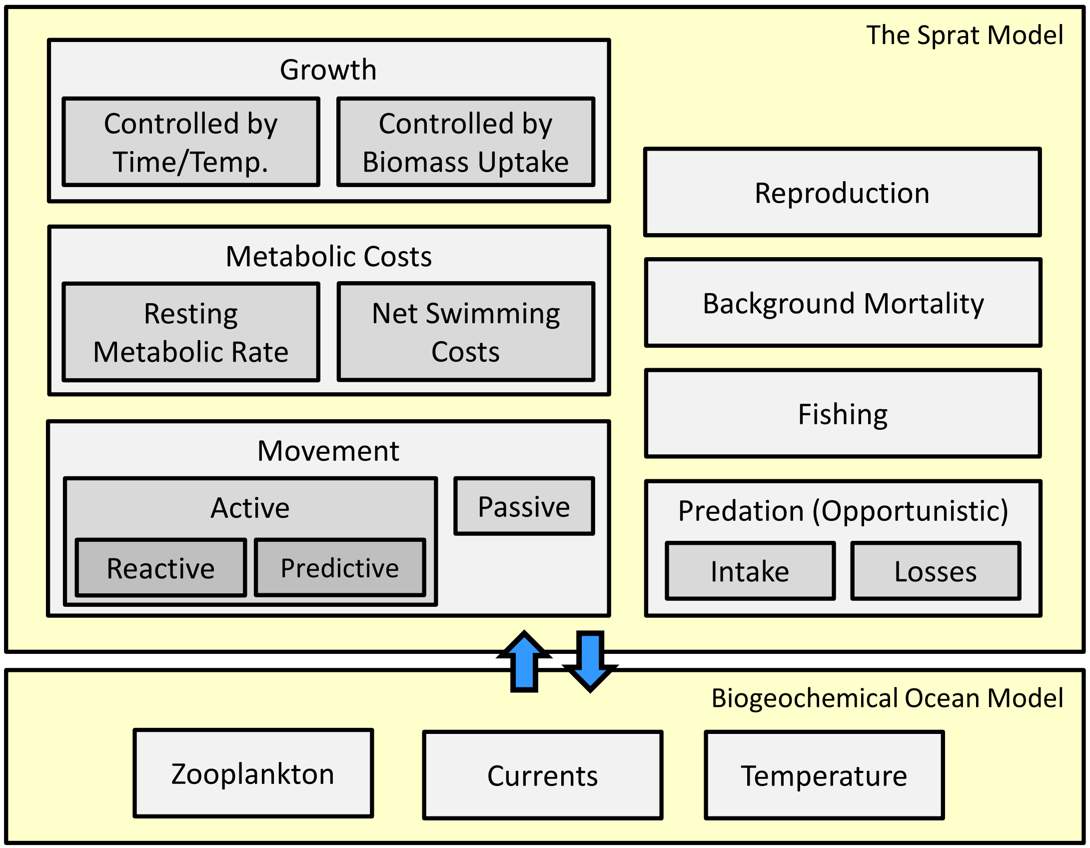
The Sprat Approach: Hierarchies of DSLs



[Johanson & Hasselbring 2014a,b, 2016b]

Evaluation of the Sprat

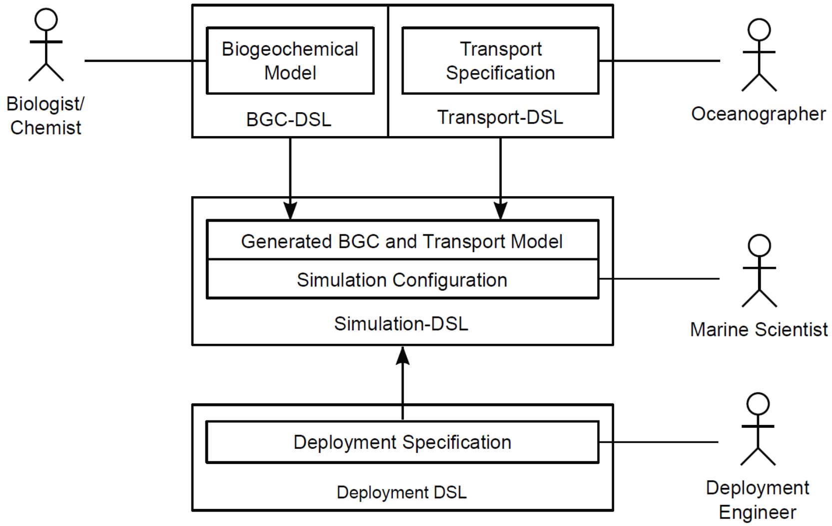
- Controlled experiments with domain scientists [Johanson & Hasselbring 2017]
- Expert interviews and benchmarks [Johanson et al. 2016b]
- The Sprat Marine Ecosystem Model:
  Original scientific contributions to Ecological Modeling [Johanson et al. 2017a]



Outlook: OceanDSL

- OceanDSL – Domain-Specific Languages for Ocean Modeling and Simulation
- Provide an infrastructure for building **modular and testable** ocean modeling and simulation software
- Initial focus on configuration and parametrization DSLs [Jung et al. 2021]

DFG



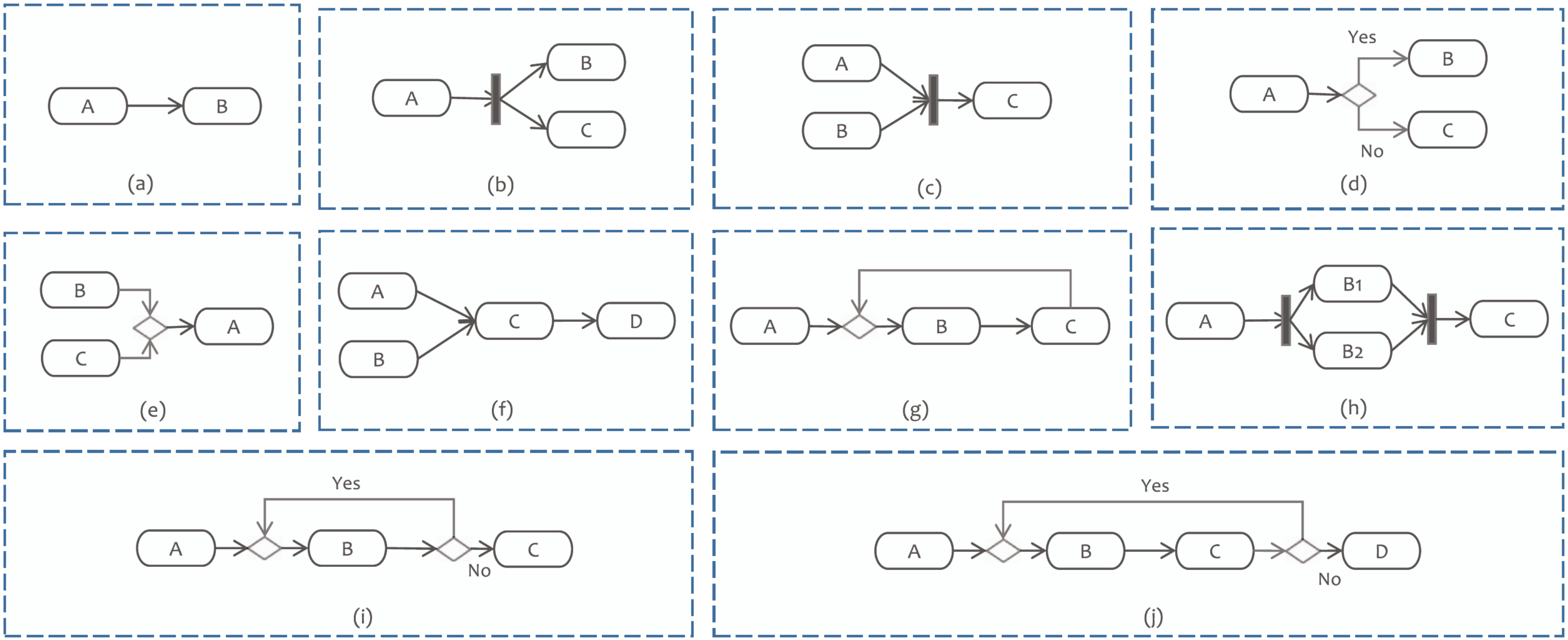
C A U
Kiel University
Christian-Albrechts-Universität zu Kiel

GEOMAR

Agenda

1. Research Software
2. Research Software Engineering
   - Automated testing
   - Modular Software
     - Modular commercial software
     - Modular research software
   - Domain-specific software engineering
   - **Flow-based programming**
3. Summary & Outlook

Workflow Control-Flow Patterns

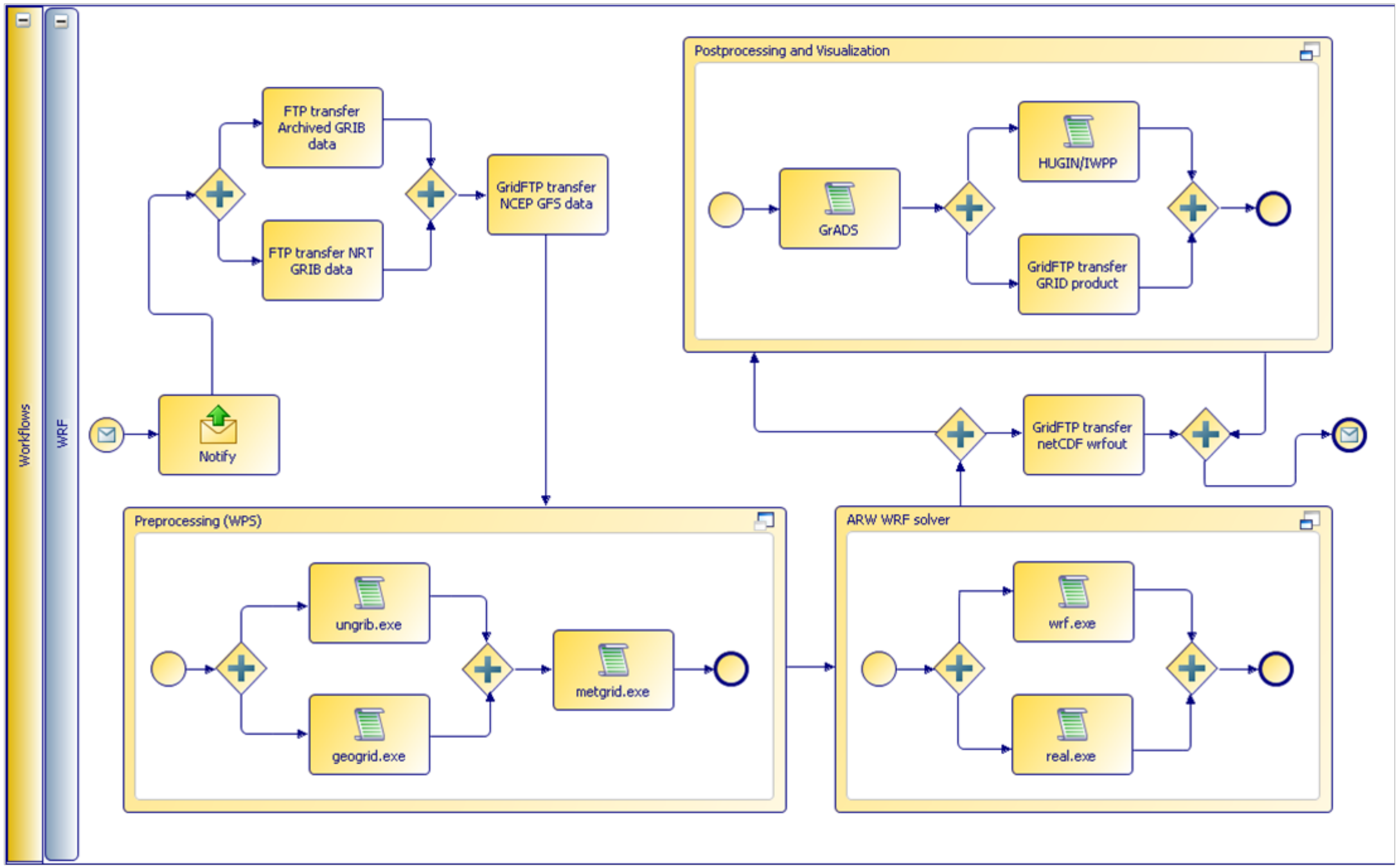


Control-flow patterns UML activity diagram:
(a) Sequence; (b) Parallel Split; (c) Synchronisation; (d) Exclusive Choice; (e) Simple Merge; (f) Multi Merge; (g) Arbitrary Cycle; (h) Multiple Instance with a prior design-time knowledge; (i) Multiple Instance with a prior run-time knowledge; and (j) Milestone.

[Butt and Fitch 2021]

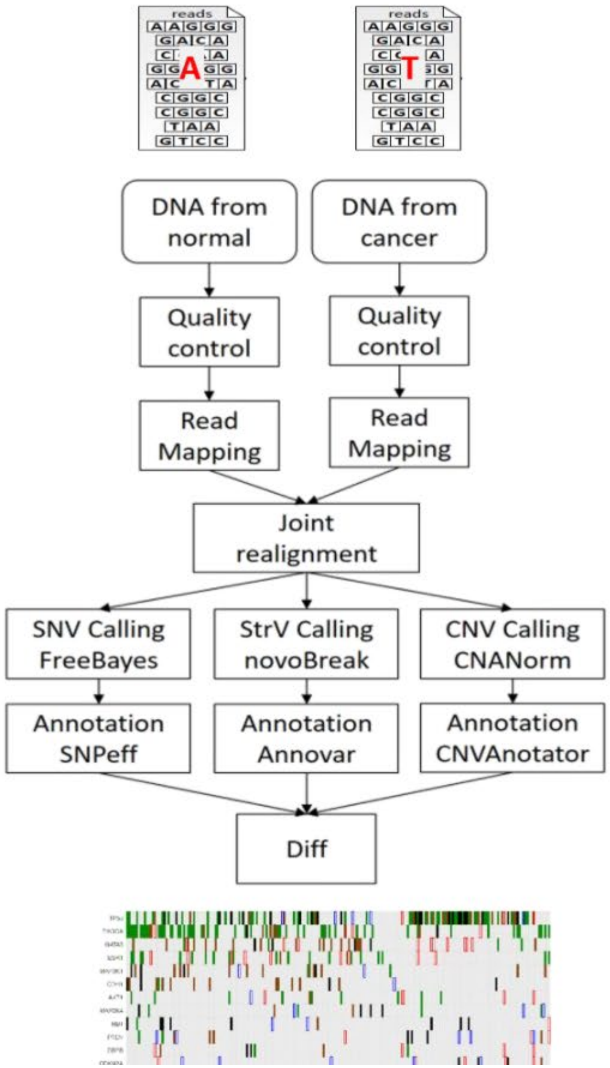
Scientific Workflows

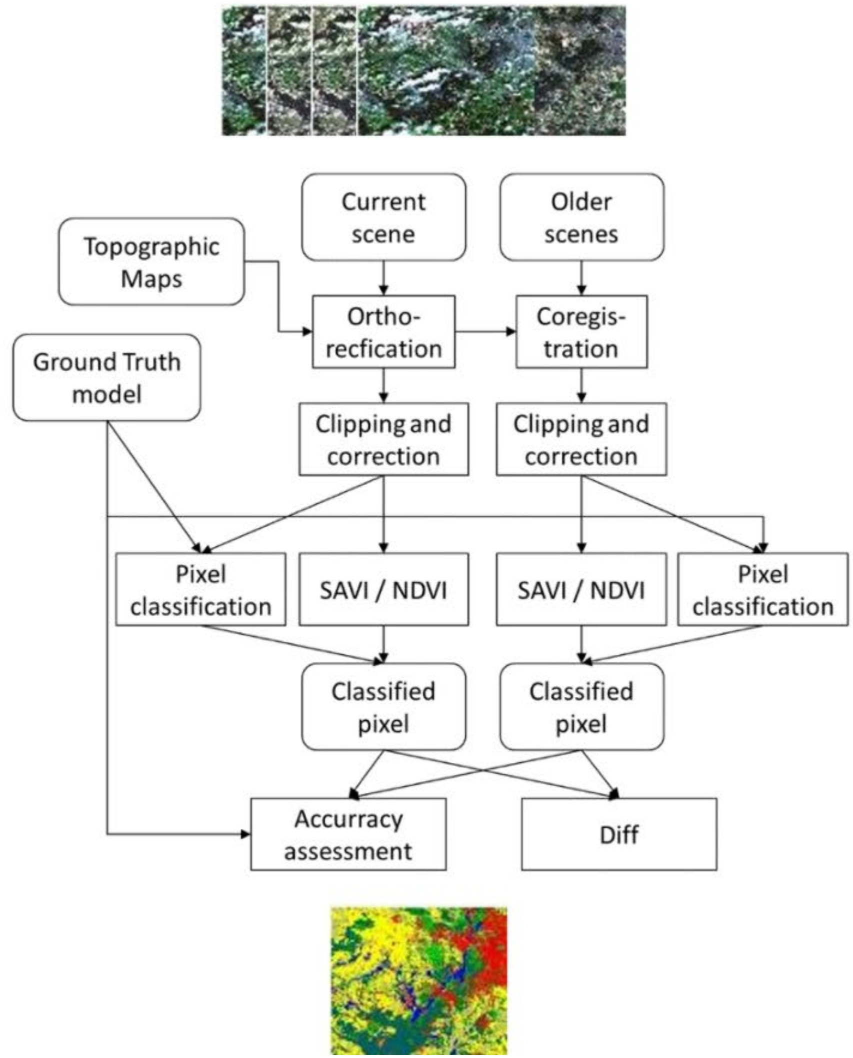
From the D-Grid project WISENT on e-Science for Energy Meteorology
[Hasselbring et al. 2006]



From the control-flow patterns, only Parallel Split and Synchronisation (aka Fork/Join).
No Exclusive Choices or Loops.

Data Analysis Workflows in FONDA





Ulf Leser @ GIBU 2021

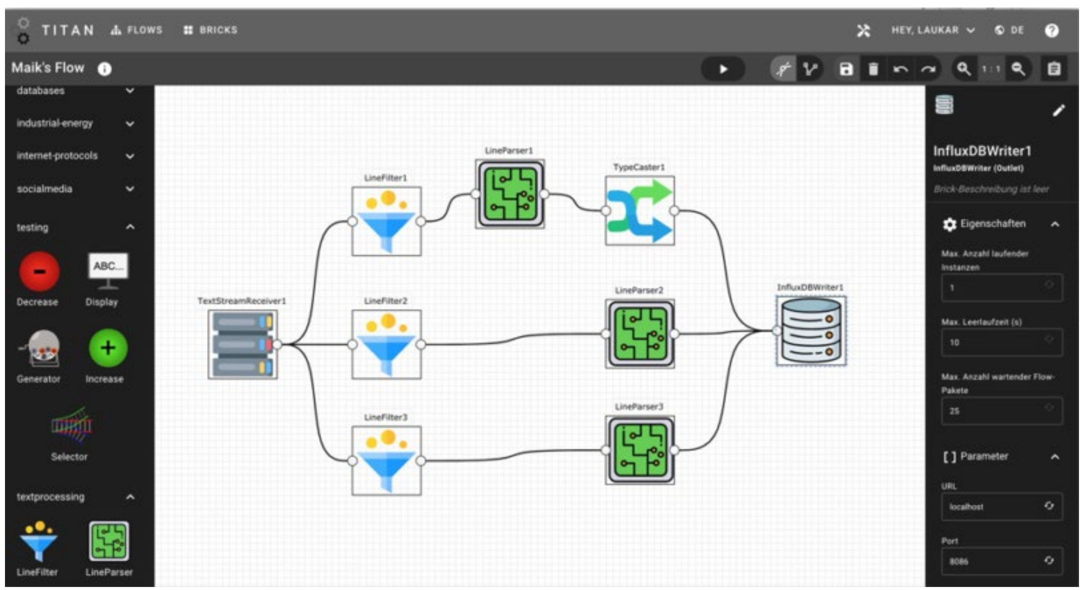
Control Flow Versus Data Flow in Distributed Systems Integration: Revival of Flow-Based Programming for the Industrial Internet of Things

Wilhelm Hasselbring, *Kiel University, 24118 Kiel, Germany*

Maik Wojcieszak, *CTO Wobe-Systems GmbH, 24145 Kiel, Germany*

Schahram Dustdar, *TU Wien, 1040 Vienna, Austria*

IEEE Internet Computing



[Hasselbring et al. 2021], see also https://www.industrial-devops.org/

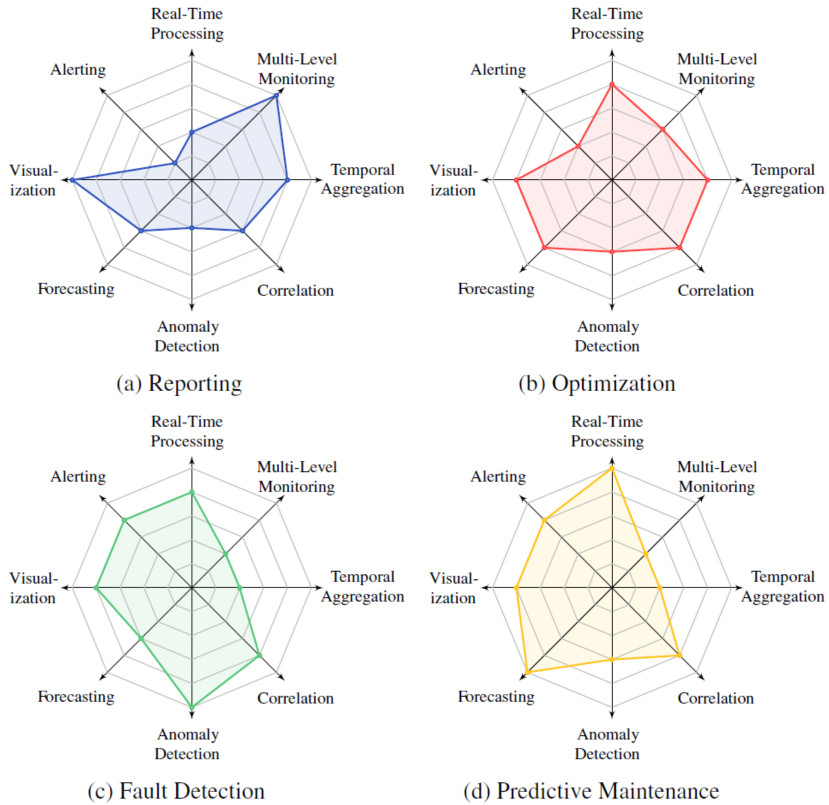
Journal of Data, Information and Management
https://doi.org/10.1007/s42488-021-00043-5

ORIGINAL ARTICLE

Check for updates

Goals and measures for analyzing power consumption data in manufacturing enterprises

Sören Henning[1] · Wilhelm Hasselbring[1] · Heinz Burmester[2] · Armin Möbius[3] · Maik Wojcieszak[4]



[Henning et al. 2021]

Developing Analysis Workflows in FONDA

- Like software in the 70ties!
  - No standardized architectural components
  - No established abstractions with common APIs
- Programs tightly tied to software infrastructure
- Low productivity – "Software crisis"

- FONDA's overall goal

How can we increase human productivity in the creation, maintenance, and execution of DAWs for large-scale scientific data analysis?

How can we increase portability, adaptability, and dependability of DAWs and DAW infrastructures?

Ulf Leser @ GIBU 2021

Summary

- Modularity is essential for maintainability, scalability and agility
  - also for **reusability**
  - also for **testability**
  - So, **microservices** could be a beneficial architectural style for research software, too.
- However, **domain-specific** software engineering approaches are required for computational science
  - Implausible to modernize legacy scientific code
- When researching data analysis workflows in FONDA,
  - I suggest to emphasize data flow over control flow [Hasselbring et al. 2021]
- **Open Science** also for Computer Science / Software Engineering research itself
  - “Eat your own dog food”
  - Follow the FAIR principles [Hasselbring et al. 2020b]

References

[Alexander & Easterbrook 2015] K. Alexander and S. M. Easterbrook: “The software architecture of climate models”, In: Geosci. Model Dev., 8, 1221–1232, 2015. DOI http://doi.org/10.5194/gmd-8-1221-2015

[Butt and Fitch 2021] A.S. Butt, P. Fitch: “A provenance model for control-flow driven scientific workflows”. In: Data & Knowledge Engineering, 131–132, 2021, DOI https://doi.org/10.1016/j.datak.2021.101877

[Calzavarini 2019] E. Calzavarini: “Eulerian–Lagrangian fluid dynamics platform: The ch4-project”. In: Software Impacts 1, 2019. DOI https://doi.org/10.1016/j.simpa.2019.100002

[Carver et al. 2007] J.C. Carver et al., “Software Development Environments for Scientific and Engineering Software: A Series of Case Studies,” Proc. 29th Int'l Conf. Software Eng. (ICSE 07), 2007, pp. 550–559. DOI https://doi.org/10.1109/ICSE.2007.77

[Fittkau et al. 2013] F. Fittkau, J. Waller, C. Wulf, W. Hasselbring: “Live Trace Visualization for Comprehending Large Software Landscapes: The ExplorViz Approach“, In: 1st IEEE International Working Conference on Software Visualization (VISSOFT 2013). DOI https://doi.org/10.1109/VISSOFT.2013.6650536

[Fittkau et al. 2015a] F. Fittkau, S. Roth, W. Hasselbring: “ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes“, In: 23rd European Conference on Information Systems (ECIS 2015). DOI https://doi.org/10.18151/7217313

[Fittkau et al. 2015b] F. Fittkau, A. Krause, W. Hasselbring: “Hierarchical Software Landscape Visualization for System Comprehension: A Controlled Experiment”. In: 3rd IEEE Working Conference on Software Visualization, 2015. DOI https://doi.org/10.1109/VISSOFT.2015.7332413

[Fittkau et al. 2015c] F. Fittkau, A. Krause, W. Hasselbring: “Exploring Software Cities in Virtual Reality”, In: 3rd IEEE Working Conference on Software Visualization, 2015. DOI https://doi.org/10.1109/VISSOFT.2015.7332423

[Fittkau et al. 2015d] F. Fittkau, S. Finke, W. Hasselbring, J. Waller: “Comparing Trace Visualizations for Program Comprehension through Controlled Experiments”, In: 23rd IEEE International Conference on Program Comprehension (ICPC 2015), May 2015, Florence. DOI

[Fittkau et al. 2017] F. Fittkau, A. Krause, W. Hasselbring: “Software landscape and application visualization for system comprehension with ExplorViz”, In: Information and Software Technology. DOI 10.1016/j.infsof.2016.07.004

References

[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 2006] W. Hasselbring, and others: “WISENT: e-Science for Energy Meteorology”. In: Proceedings of 2nd IEEE International Conference on e-Science and Grid Computing (e-Science'06). pp. 93-100. DOI https://doi.org/10.1109/E-SCIENCE.2006.156

[Hasselbring 2016] W. Hasselbring, “Microservices for Scalability (Keynote Presentation),” In: 7th ACM/SPEC International Conference on Performance Engineering (ACM/SPEC ICPE 2016), March 15, 2016 , Delft, NL. DOI https://doi.org/10.1145/2851553.2858659

[Hasselbring 2018] W. Hasselbring, “Software Architecture: Past, Present, Future,” In: The Essence of Software Engineering. Springer, pp. 169-184. 2018. DOI 10.1007/978-3-319-73897-0_10

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

[Hasselbring et al. 2021] W. Hasselbring, M. Wojcieszak, S. Dustdar: “Control Flow Versus Data Flow in Distributed Systems Integration: Revival of Flow-Based Programming for the Industrial Internet of Things”. In: IEEE Internet Computing, 2021. DOI https://doi.org/10.1109/MIC.2021.3053712

[Hasselbring & Steinacker 2017] W. Hasselbring, G. Steinacker: “Microservice Architectures for Scalability, Agility and Reliability in E-Commerce”, In: Proceedings of the IEEE International Conference on Software Architecture (ICSA 2017), April 2017, Gothenburg, Sweden. DOI

[Henning et al. 2021] S. Henning, W. Hasselbring, H. Burmester, A. Möbius, M. Wojcieszak: “Goals and measures for analyzing power consumption data in manufacturing enterprises”. In: Journal of Data, Information and Management, 2021. DOI https://doi.org/10.1007/s42488-021-00043-5

References

[Hiremath et al. 2021] D.J. Hiremath, M. Claus, W. Hasselbring, and W. Rath: “Towards Automated Metamorphic Test Identification for Ocean System Models”. In: Proceedings of the 6th International Workshop on Metamorphic Testing. IEEE, June 2021.. (in press)

[Johanson & Hasselbring 2014a] A. Johanson, W. Hasselbring: “Hierarchical Combination of Internal and External Domain-Specific Languages for Scientific Computing”. In: International Workshop on DSL Architecting & DSL-Based Architectures (DADA'14), 2014, pp. 17:1-17:8. DOI https://doi.org/10.1145/2642803.2642820

[Johanson & Hasselbring 2014b] A. Johanson, W. Hasselbring: “Sprat: Hierarchies of Domain-Specific Languages for Marine Ecosystem Simulation Engineering”. In: Spring Simulation Multi-Conference (SpringSim 2014), April 2014, Tampa, Florida, USA, pp. 187-192. DOI https://dl.acm.org/doi/abs/10.5555/2665008.2665034

[Johanson et al. 2016a] 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), September 2016, Boulder, Colorado. DOI https://doi.org/10.5065/D6K072N6

[Johanson et al. 2016b] A. Johanson, W. Hasselbring, A. Oschlies, B. Worm: “Evaluating Hierarchical Domain-Specific Languages for Computational Science: Applying the Sprat Approach to a Marine Ecosystem Model”. In: Software Engineering for Science. CRC Press. 175-200.

[Johanson et al. 2017a] A. Johanson, A. Oschlies, W. Hasselbring, A. Worm: “SPRAT: A spatially-explicit marine ecosystem model based on population balance equations”, In: Ecological Modelling, 349, pp. 11-25, 2017. DOI https://doi.org/10.1016/j.ecolmodel.2017.01.020

[Johanson et al. 2017b] 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

References

[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/

[Kanewala and Bieman 2014] U. Kanewala and J.M. Bieman, “Testing Scientific Software: A Systematic Literature Review,” Information and Software Technology, 56(10), 2014, . DOI https://doi.org/10.1016/j.infsof.2014.05.006

[Knoche and Hasselbring 2018] H. Knoche and W. Hasselbring, “Using Microservices for Legacy Software Modernization IEEE Software, 35 (3). pp. 44-49. 2018. DOI https://doi.org/10.1109/MS.2018.2141035.

[Knoche and Hasselbring 2019] H. Knoche and W. Hasselbring, “Drivers and Barriers for Microservice Adoption - A Survey among Professionals in Germany,” Enterprise Modelling and Information Systems Architectures (EMISAJ) - International Journal of Conceptual Modeling, 14 (1). pp. 1-35. 2019. DOI https://doi.org/10.18417/emisa.14.1.

[Krause et al. 2020] A. Krause, C. Zirkelbach, W. Hasselbring, S. Lenga, D. Kröger: “Microservice Decomposition via Static and Dynamic Analysis of the Monolith”. In: IEEE International Conference on Software Architecture (ICSA 2020). pp. 9-16 . DOI https://doi.org/10.1109/ICSA-C50368.2020.00011.

[Randell 2018] B. Randell: 50 years of Software Engineering. May 2018, https://arxiv.org/abs/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

[Segura et al. 2020] S. Segura, D. Towey, Z. Q. Zhou, and T. Y. Chen. “Metamorphic Testing: Testing the Untestable”, IEEE Software, 37(3):46-53,May 2020. DOI https://doi.org/10.1109/MS.2018.2875968.

[Zirkelbach et al. 2019] Zirkelbach, C., Krause, A. und Hasselbring, W.: “Modularization of Research Software for Collaborative Open Source Development”, In: The Ninth International Conference on Advanced Collaborative Networks, Systems and Applications (COLLA 2019), June 30 - July 04, 2019, Rome, Italy.

[Zirkelbach et al. 2020] Zirkelbach, C., Krause, A. und Hasselbring, W.: “The Collaborative Modularization and Reengineering Approach CORAL for Open Source Research Software”. In: International Journal On Advances in Software, 13 (1&2). pp. 34-49.