

Microservice Architectures for Data Science Applications

Prof. Dr. Wilhelm (Willi) Hasselbring

Software Engineering Group, Kiel University

<http://se.informatik.uni-kiel.de/>

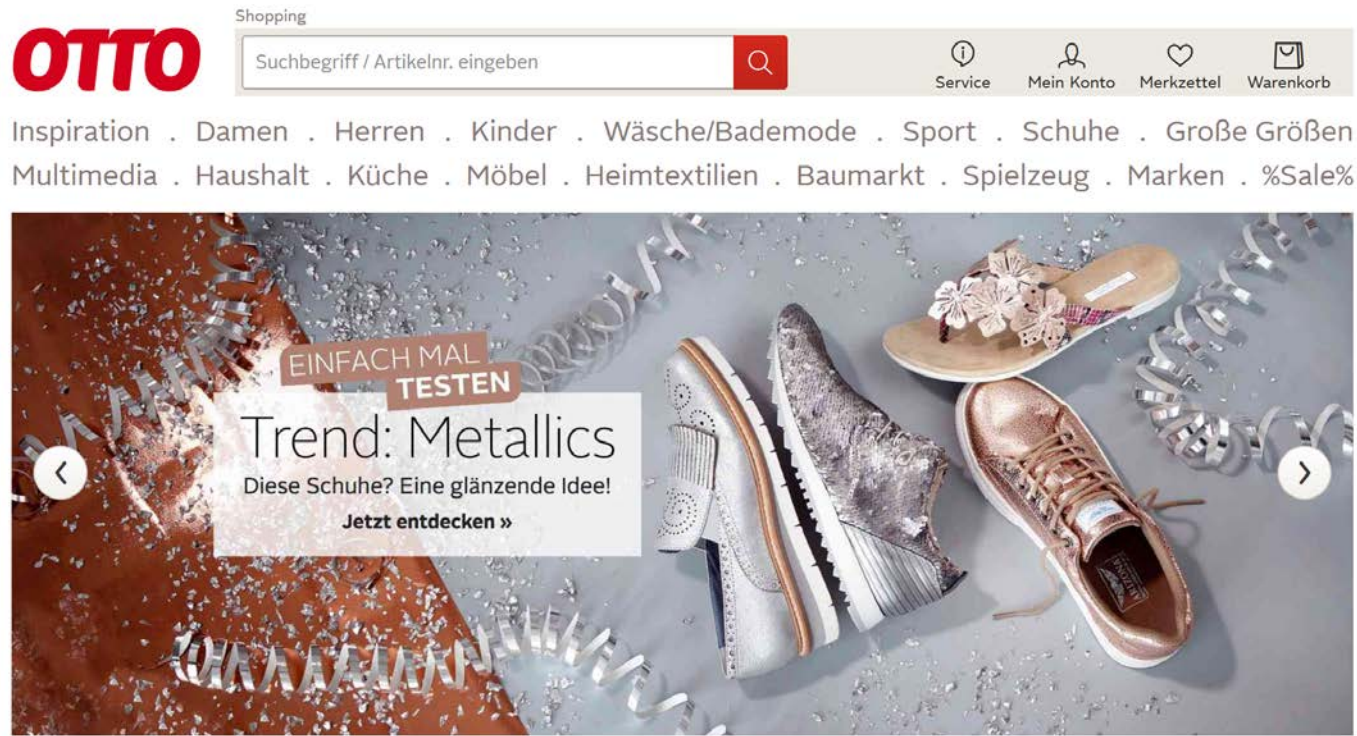
GEOMAR Data Science Symposium, April 19th, 2018



Agenda

1. A look at industrial, Internet-scale Software Systems
2. Software Engineering for Computational Science
3. OceanTEA: Platform for Repeatable Ocean Observation Data Processing
4. GeRDI: Generic Research Data Infrastructure
5. Summary & Outlook

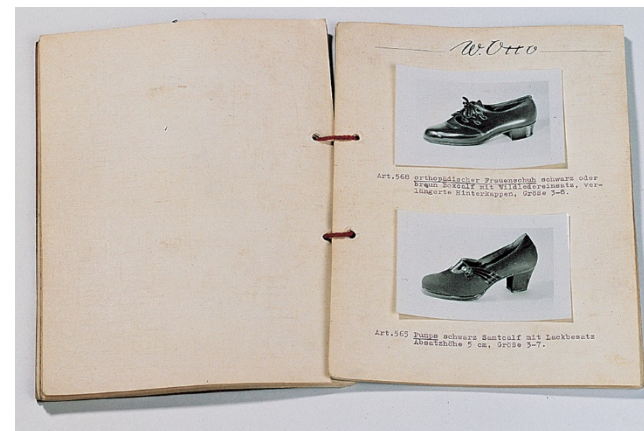
Internet-Scale Software Example: otto.de



Facts and figures about OTTO

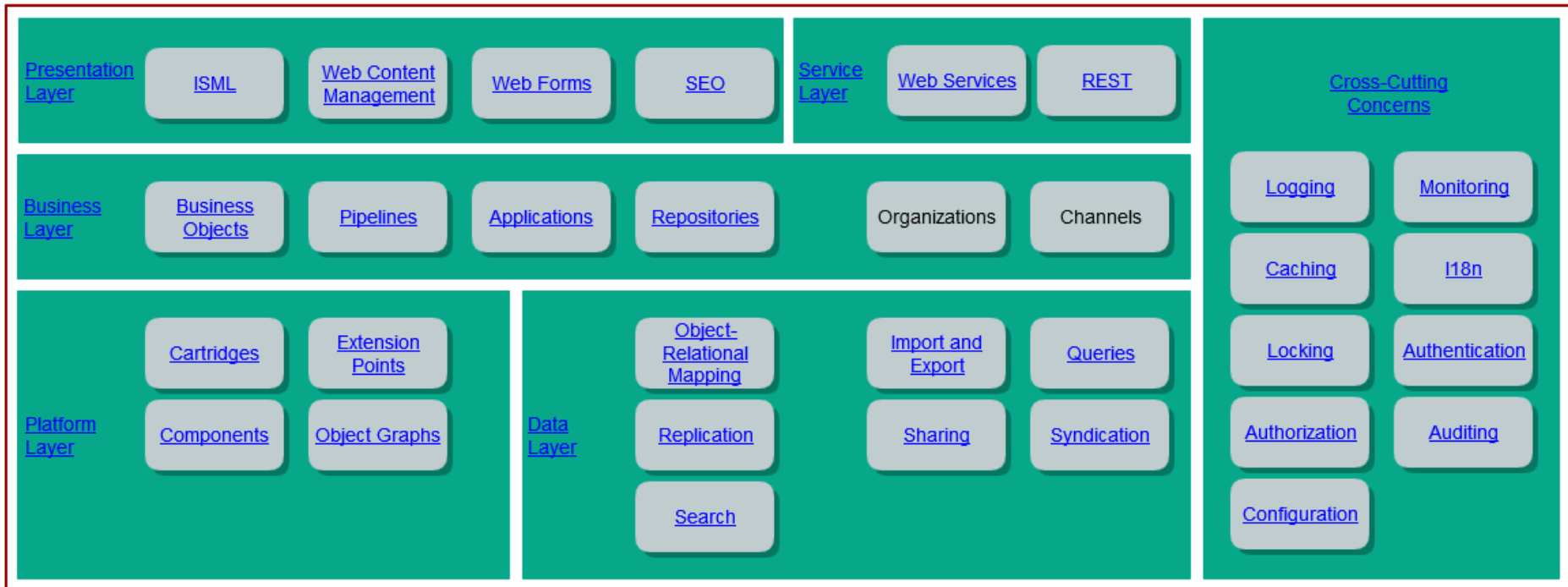
Profile

Founded	August 17th, 1949
No. of employees	4,350
Revenue in 2015/16 FY	2.563 billion Euros
Online revenue share	round 90 percent



<http://www.ottogroup.com/de/presse/material.php>

Otto Web Shop until 2013



<https://support.intershop.com/kb/index.php/Display/276B90>

Otto: Project Lhotse 2011-2013

- In 2011, Otto started a complete re-implementation of their ecommerce software from scratch.
- The drivers for this decision were diverse, but had mostly to do with **non-functional requirements** like scalability, performance and fault tolerance.
 - Regarding scalability, they were not only thinking about technical **scalability** in terms of load or data.
 - They needed a solution that was scaling with respect to the **number of teams and/or developers** working on the software at a given time.
 - In addition to that, they planned to practice **DevOps** including continuous deployment, in order to deliver features quickly to the customer.

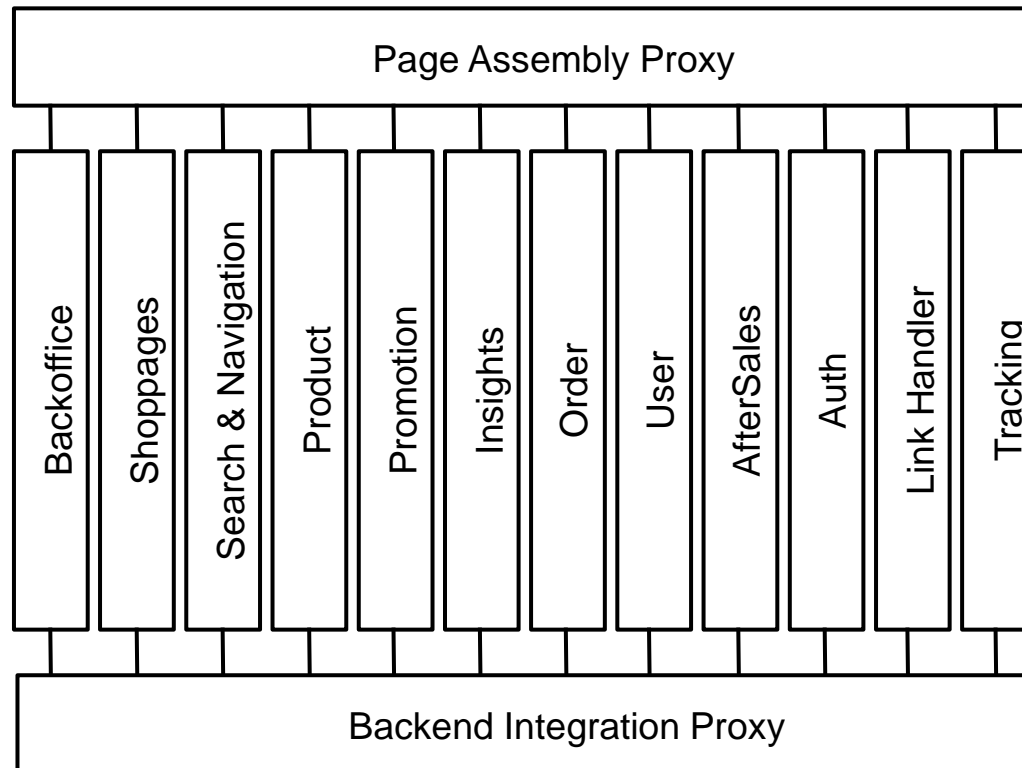
<https://www.otto.de/unternehmen/de/newsroom/dossiers/lhotse.php>

Modernization Strategy

- What they have found was in the first place a little bit unusual, but in the end highly successful:
 - Instead of setting up a single development team to create a new platform for the shop, they were actively employing Conway's Law by starting development **with initially four separate teams** with four loosely coupled applications (a.k.a. microservices):
 - **Product**, being responsible for products and their presentation.
 - **Order** for shopping carts and the order process.
 - **Promotion**, serving product recommendations and promotions for assortments, brands, and so on.
 - **Search and Navigation** for search and navigation in the shop.
- In the following years, they founded several more teams and systems.

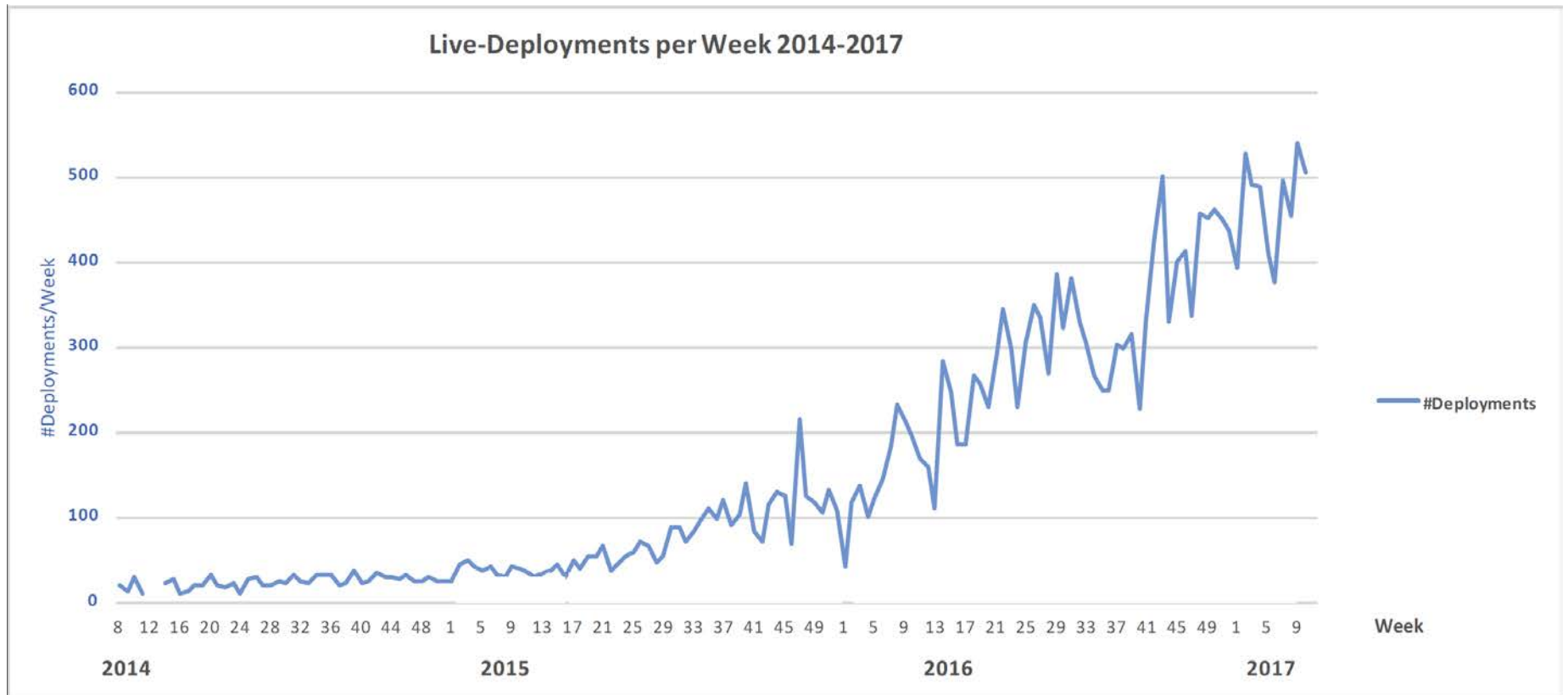
Verticals for Business Functions

Example: otto.de

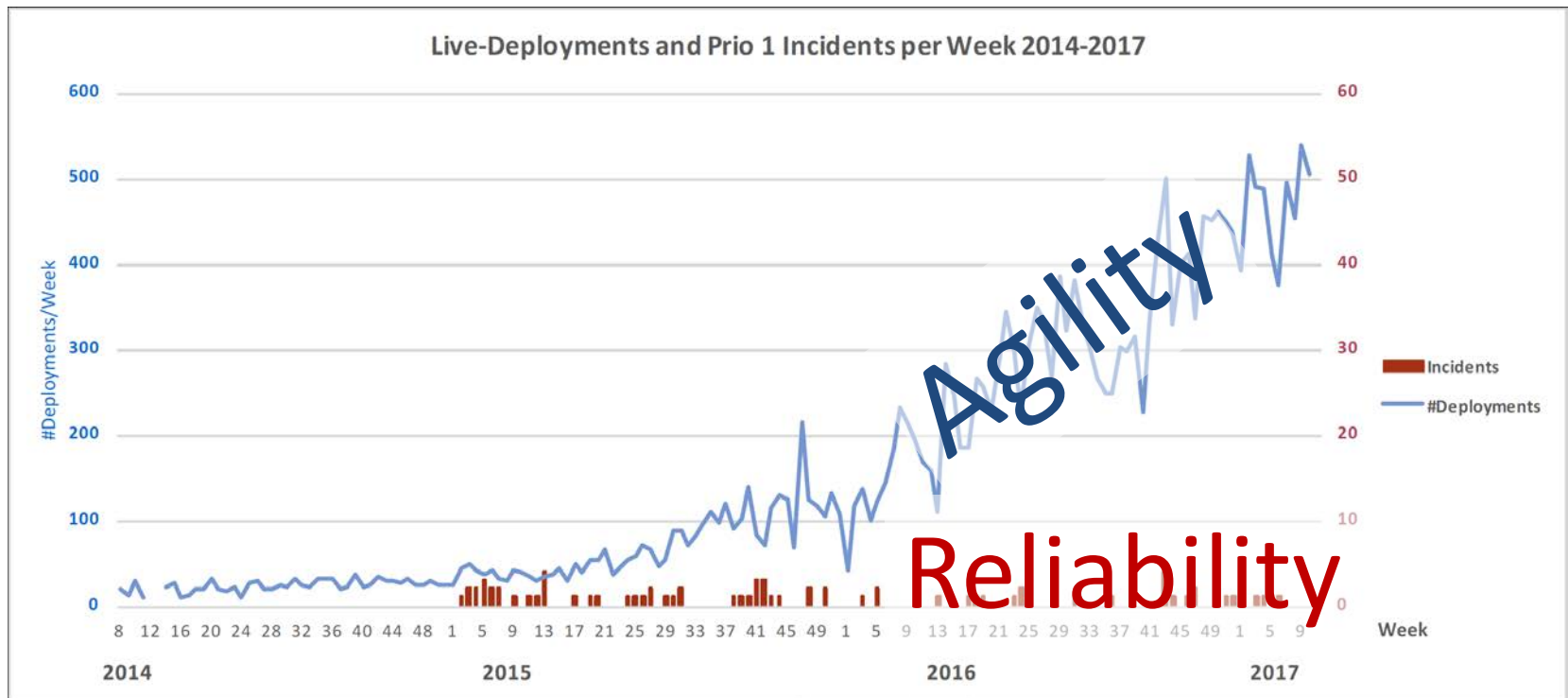


[Hasselbring & Steinacker 2017]

Life Deployments @ Otto.de



Life Deployments & Incidents @ Otto.de



[Hasselbring & Steinacker 2017]

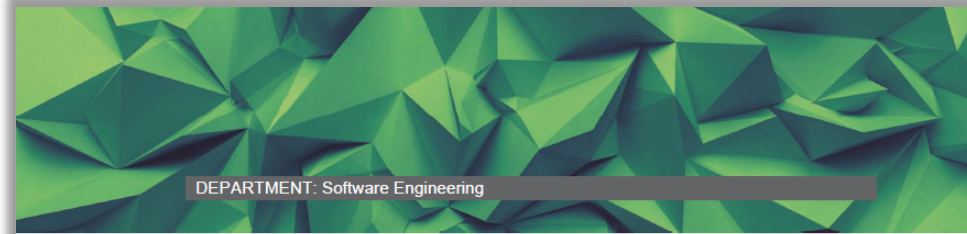
Agenda

1. A look at industrial, Internet-scale Software Systems
- 2. Software Engineering for Computational Science**
3. OceanTEA: Platform for Repeatable Ocean Observation Data Processing
4. GeRDI: Generic Research Data Infrastructure
5. Summary & Outlook

Message:
**Adapt existing
software engineering
techniques for
computational science**

<https://doi.org/10.1109/MCSE.2018.108162940>

[Johanson & Hasselbring 2018]



DEPARTMENT: Software Engineering

Software Engineering for Computational Science:

Past, Present, Future

Arne N. Johanson
XING Marketing Solutions
GmbH

Wilhelm Hasselbring
Kiel University

Editors:
Jeffrey Carver,
carver@cs.uu.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.

With the constantly increasing capabilities of modern computers, in silico experiments are becoming more complex and playing a more important role in the scientific discovery process.¹ As a consequence, the complexity and lifespan of scientific software are growing, as well as the necessity for its output to be reproducible and verifiable. This increases the importance of employing sound software engineering practices in the development of scientific software to guarantee reliable and accurate scientific results. However, surveys show that state-of-the-art software engineering methods are rarely adopted in computational science.^{2,3} To understand the underlying causes for this and to identify ways to improve the current situation, in this article, we survey literature on software engineering in computational science and identify key characteristics that are unique to scientific software development.

To provide a basis for our survey, we outline the historical development of the relationship between the disciplines of software engineering and computational science. This relationship is, to a large extent, characterized by an isolation between the two disciplines that has resulted in

Agenda

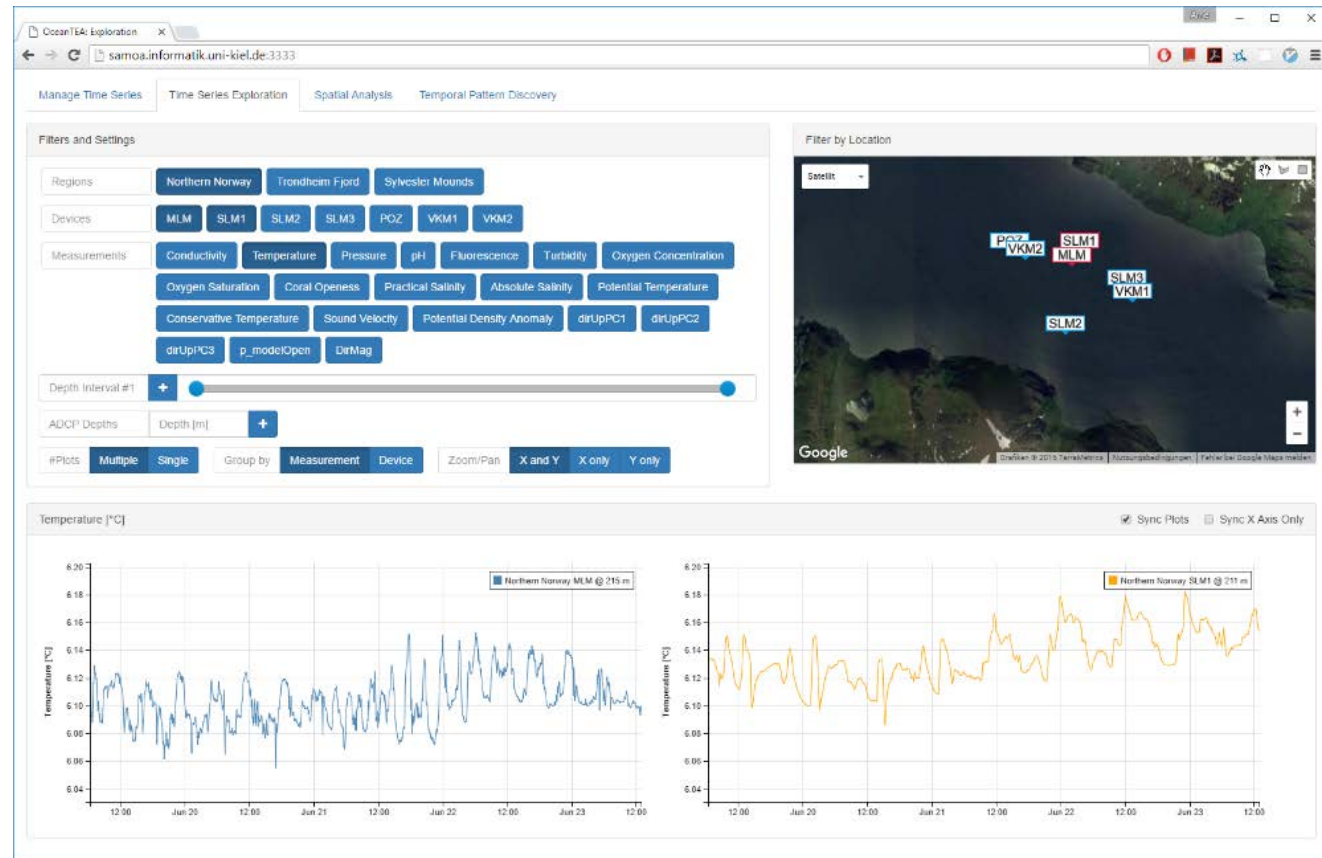
1. A look at industrial, Internet-scale Software Systems
2. Software Engineering for Computational Science
- 3. OceanTEA: Platform for Repeatable Ocean Observation Data Processing**
4. GeRDI: Generic Research Data Infrastructure
5. Summary & Outlook

Cloud-Based Platform for Repeatable Ocean Observation Data Processing

OceanTEA

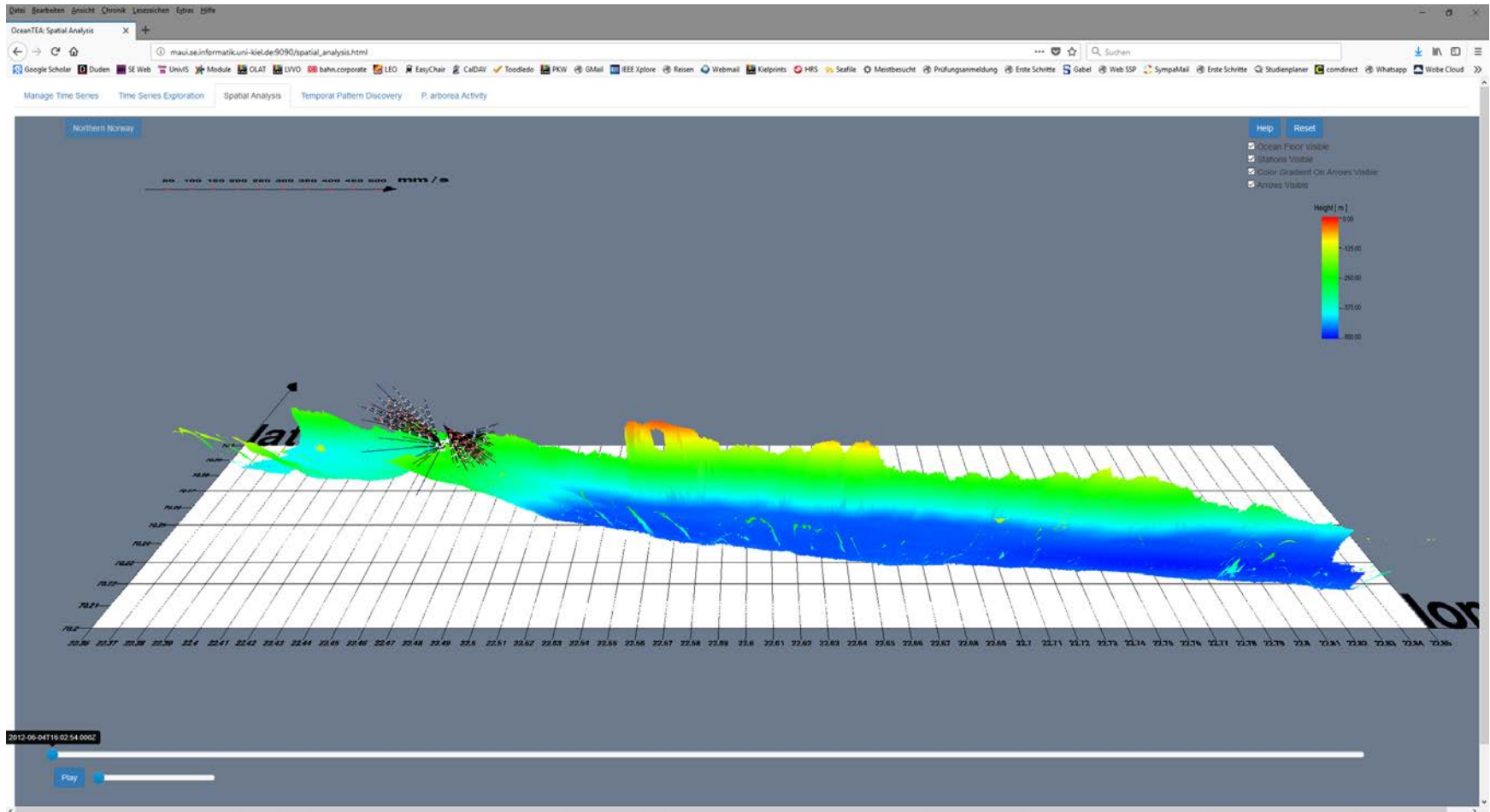


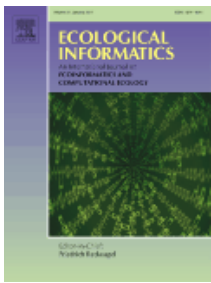
future ocean
KIEL MARINE SCIENCES



[Johanson et al. 2016]

4D Spatial Analysis with OceanTEA





Machine Learning on Ocean Observation Data with OceanTEA

- Paper: <http://dx.doi.org/10.1016/j.ecoinf.2017.02.007>
- Source code: <https://github.com/a-johanson/oceantea>
- Software service with data: <http://maui.se.informatik.uni-kiel.de:9090/> (URL will change, refer to the GitHub repository for updates)

Contents lists available at ScienceDirect

Ecological Informatics

journal homepage: www.elsevier.com/locate/ecoinf

Modeling polyp activity of *Paragorgia arborea* using supervised learning

Arne N. 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


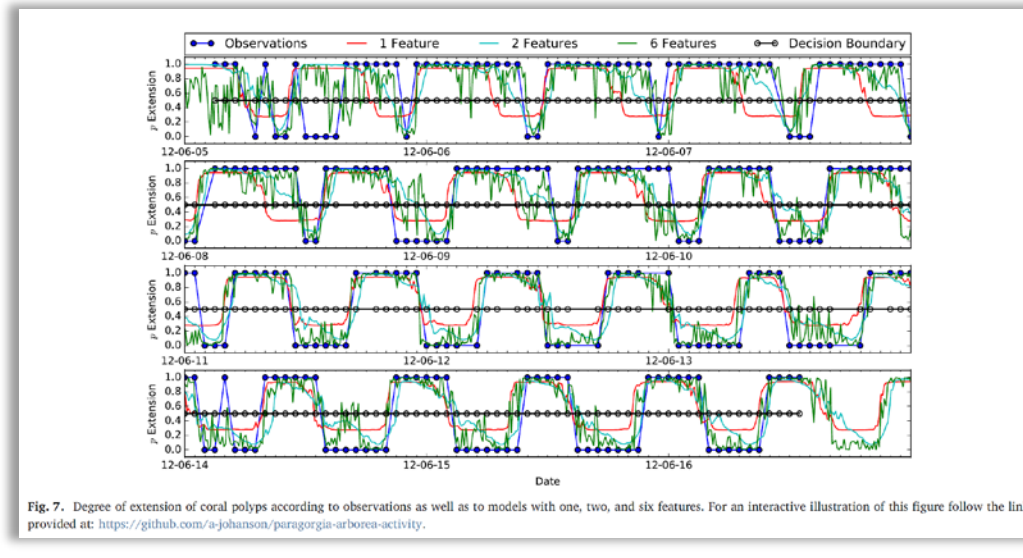



Fig. 7. Degree of extension of coral polyps according to observations as well as to models with one, two, and six features. For an interactive illustration of this figure follow the link provided at: <https://github.com/a-johanson/paragorgia-arborea-activity>.

Model Properties

Features

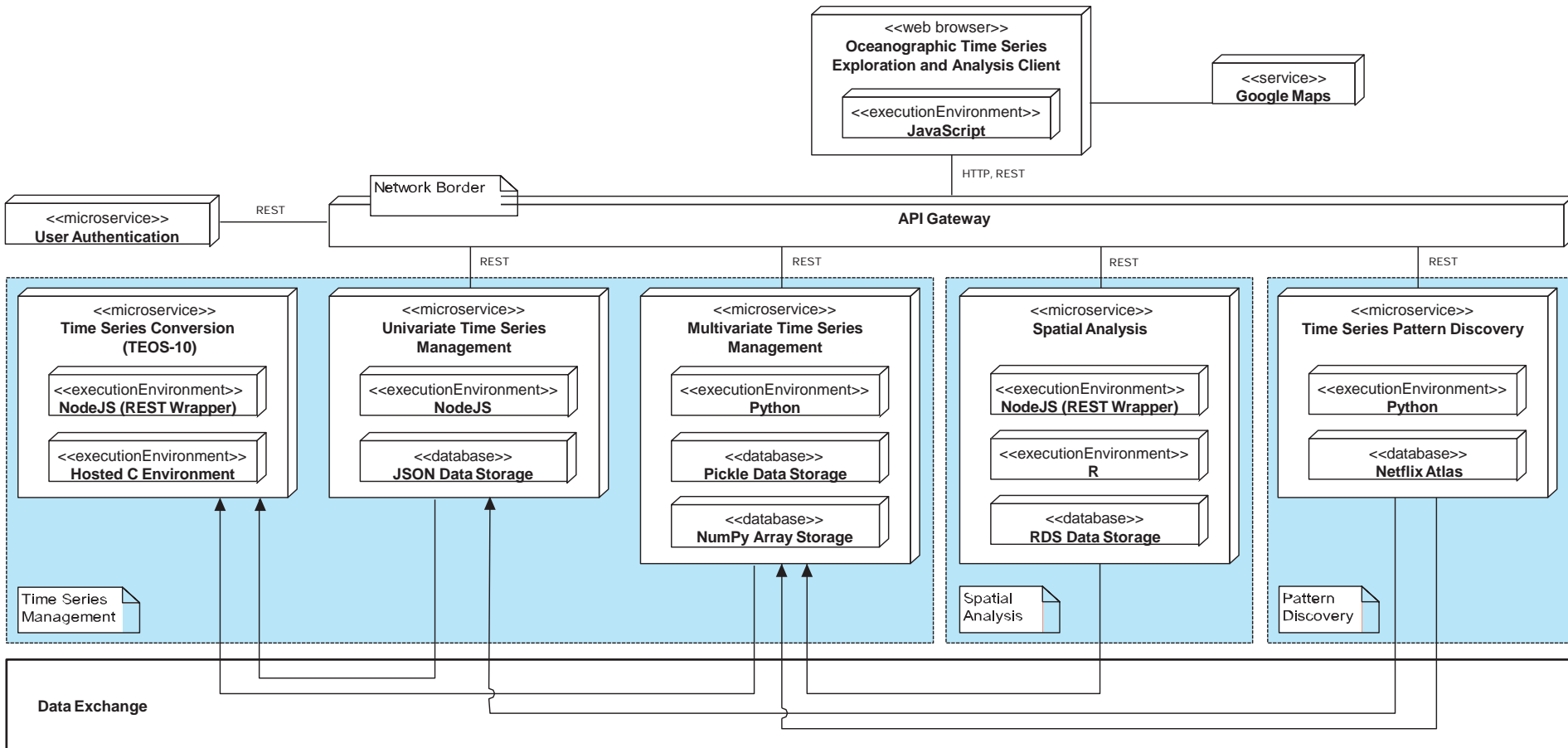
<input type="checkbox"/> Temperature (core)	<input type="checkbox"/> Direction down, PC1	<input type="checkbox"/> Direction down, PC2	<input type="checkbox"/> Velocity down, PC1
<input type="checkbox"/> Temperature (core), 2h lag	<input type="checkbox"/> Direction up, PC1	<input type="checkbox"/> Direction down, 2h lag, PC2	<input type="checkbox"/> Velocity down, 2h lag, PC1
<input type="checkbox"/> Temperature (core), 4h lag	<input type="checkbox"/> Direction up, 2h lag, PC1	<input type="checkbox"/> Direction down, 2h lag, PC1	<input type="checkbox"/> Velocity down, 2h lag, PC2
<input type="checkbox"/> Salinity (core)	<input type="checkbox"/> Direction up, 4h lag, PC1	<input type="checkbox"/> Direction down, 2h lag, PC2	<input type="checkbox"/> Velocity down, 2h lag, PC1
<input type="checkbox"/> Salinity (core), 2h lag	<input type="checkbox"/> Velocity up, PC1	<input type="checkbox"/> Direction down, 2h lag, PC2	<input type="checkbox"/> Velocity down, 2h lag, PC1
<input type="checkbox"/> Salinity (core), 4h lag	<input type="checkbox"/> Velocity up, 2h lag, PC1	<input type="checkbox"/> Direction down, 4h lag, PC1	<input type="checkbox"/> Velocity down, 2h lag, PC2
<input type="checkbox"/> ρ_{θ} density	<input type="checkbox"/> Velocity up, 4h lag, PC1	<input type="checkbox"/> Direction down, 4h lag, PC2	<input type="checkbox"/> Velocity down, 4h lag, PC1
<input type="checkbox"/> ρ_{θ} density, 2h lag	<input type="checkbox"/> Direction down, PC1	<input type="checkbox"/> Direction down, 4h lag, PC3	<input type="checkbox"/> Velocity down, 4h lag, PC2
<input type="checkbox"/> ρ_{θ} density, 4h lag	<input type="checkbox"/> Direction down, PC2	<input type="checkbox"/> Velocity down, PC1	<input type="checkbox"/> Velocity down, 4h lag, PC3

Model Statistics

```

Model with 2 Features:
Test accuracy: 0.8853846533846534
Overall accuracy: 0.8827899222222222
Coefficients:
Intercept: 1.553234838018897
Direction up, 2h lag, PC1: -2.873892088080226
Velocity up, 2h lag, PC1: -1.265067028878017
-----
Model with 6 Features:
Test accuracy: 0.8841573465133846
Overall accuracy: 0.89348873893841
Coefficients:
Intercept: 1.35053489539535
Direction up, 2h lag, PC1: -3.443382821848379
Velocity up, 2h lag, PC1: -0.891929955133846
Direction up, 4h lag, PC1: -1.581923368317865
Velocity down, 2h lag, PC1: 2.127384832289482
Velocity down, 2h lag, PC2: 0.917091228997738
Velocity down, 2h lag, PC3: 0.74358452886868
    
```

OceanTEA: Microservice-based Architecture

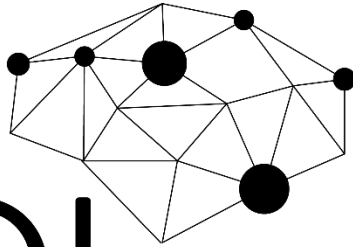


[Johanson et al. 2016, Johanson et al. 2017, Hasselbring 2016, Knoche & Hasselbring 2018]

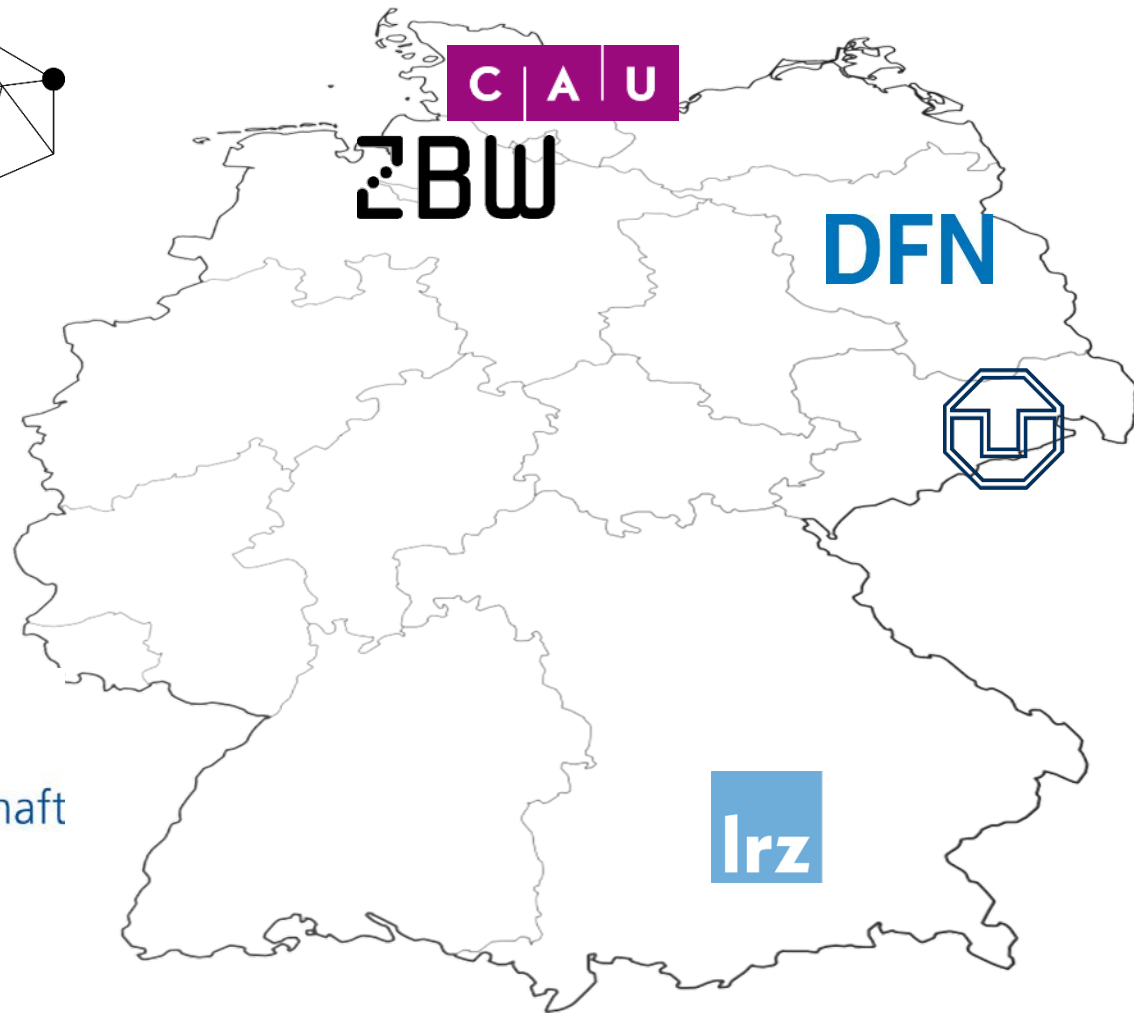
Agenda

1. A look at industrial, Internet-scale Software Systems
2. Software Engineering for Computational Science
3. OceanTEA: Platform for Repeatable Ocean Observation Data Processing
4. **GeRDI: Generic Research Data Infrastructure**
5. Summary & Outlook

Generic Research Data Infrastructure



GeRDI

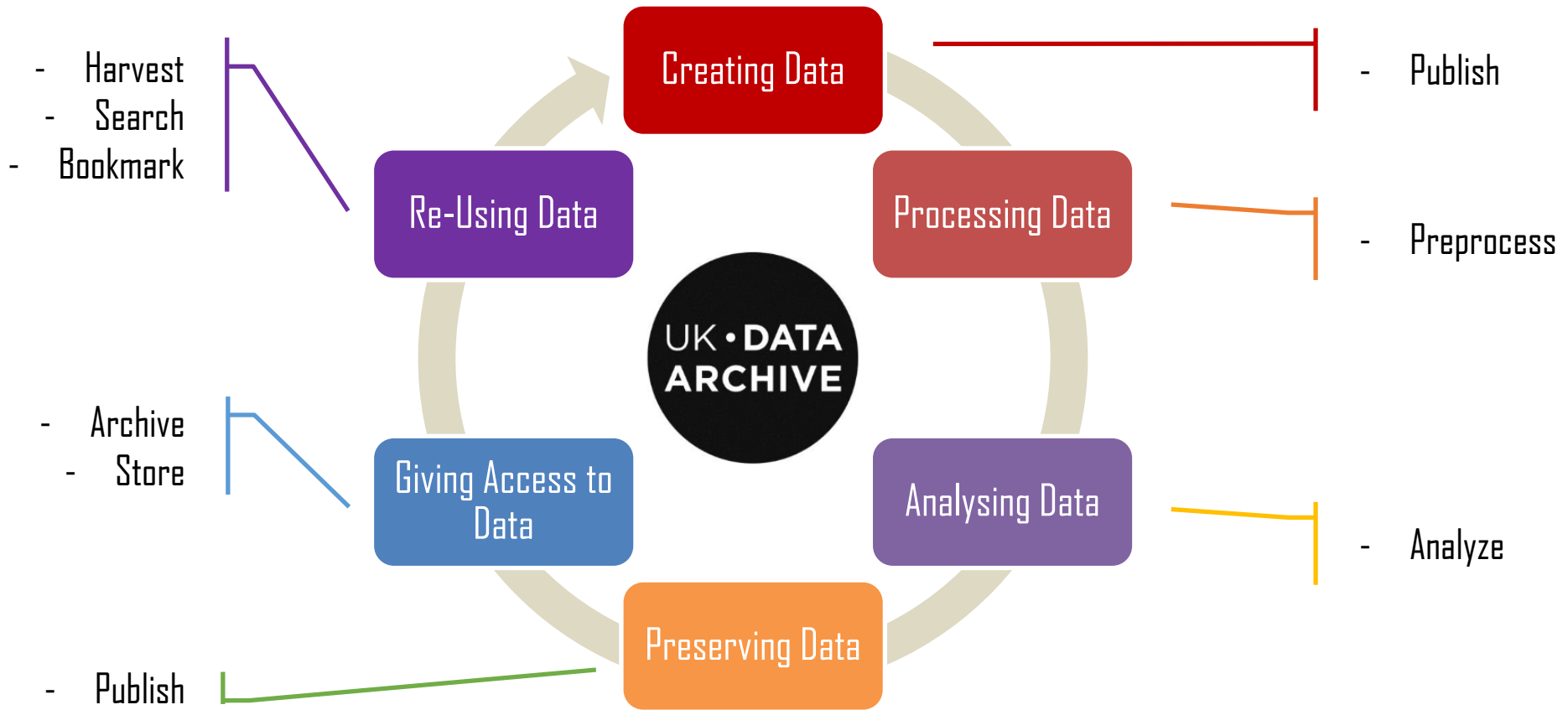


Gefördert durch

DFG Deutsche
Forschungsgemeinschaft

<http://www.gerdi.-project.de/>

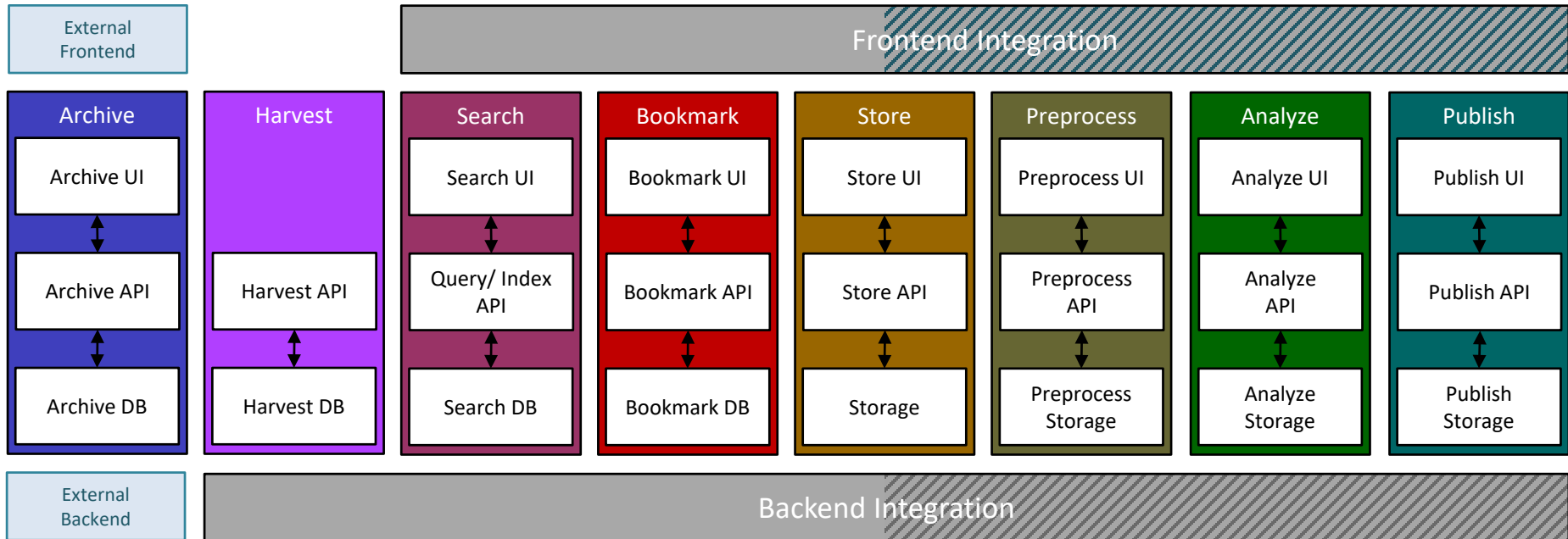
Research Data Lifecycle





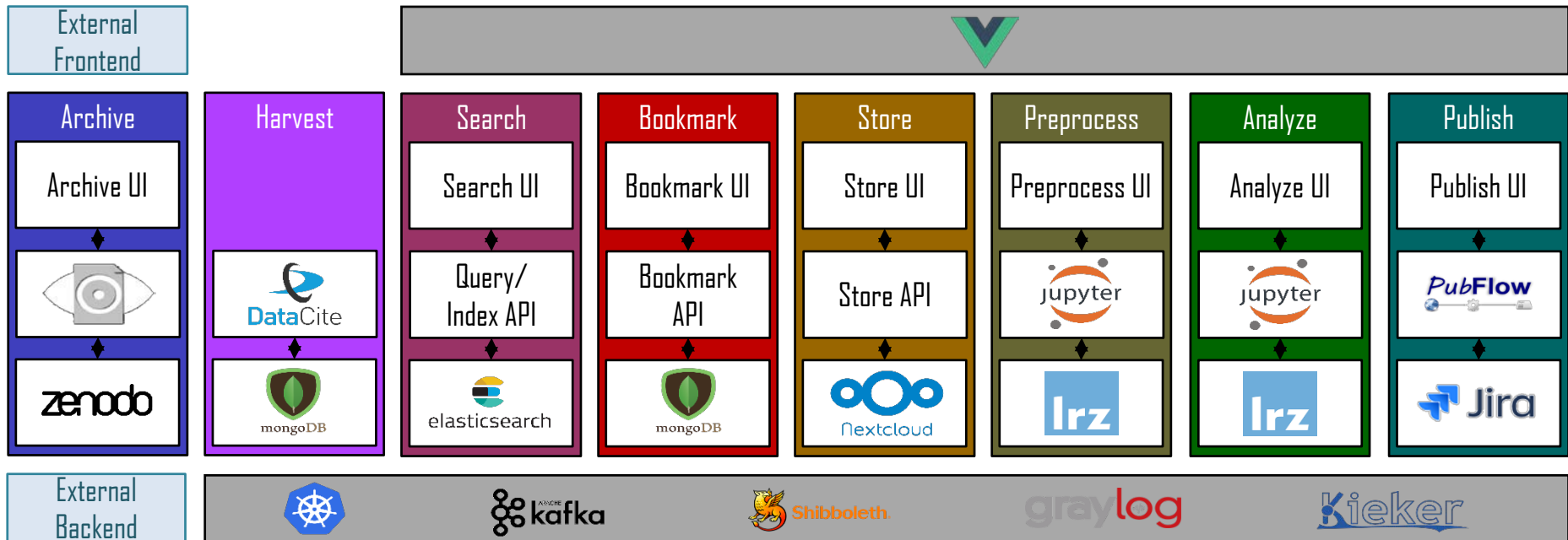
GeRDI

Generic Research Data Infrastructure



[Tavares de Sousa et al. 2018]

Implementation View



Design Rationale

- Self-contained Systems (SCS)
 - A.k.a. microservices, <http://scs-architecture.org>
 - Each SCS is an autonomous (web) application.
 - For the SCS's domain all data, the logic to process that data and all code to render the web interface is contained within the SCS.
 - Communication with other SCSs or 3rd party systems is asynchronous wherever possible.
 - This decouples the systems, reduces the effects of failure, and thus supports autonomy.
 - To avoid tight coupling an SCS should share no business code with other SCSs.
- Entry / Exit Options for users at various points in the workflow

Summary & Outlook

- Modularity is essential for sustainable software systems
 - Including data science applications
- Microservices provide a disruptive architectural style for modular systems
 - Successful for Internet-scale systems
 - However, non-trivial to achieve
- Transfer to research software
 - OceanTEA: Platform for Repeatable Ocean Observation Data Processing
 - GeRDI: Generic Research Data Infrastructure

References

- [Hasselbring 2016] W. Hasselbring: “Microservices for Scalability: Keynote Talk Abstract”. In: International Conference on Performance Engineering (ICPE 2016), March 15, 2016, Delft, Netherlands. DOI 10.1145/2851553.2858659
- [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 10.1109/ICSAW.2017.11
- [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), September 2016, Boulder, Colorado.
- [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, Elsevier, 2017. DOI 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. DOI 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 10.1109/MCSE.2018.108162940
- [Knoche & Hasselbring 2018] H. Knoche, W. Hasselbring: “Using Microservices for Legacy Software Modernization”, In IEEE Software, 2018
- [Tavares de Sousa et al. 2018] N. Tavares de Sousa, W. Hasselbring, T. Weber, D. Kranzlmüller: “Designing a Generic Research Data Infrastructure Architecture with Continuous Software Engineering”, In: 3rd Workshop on Continuous Software Engineering (CSE 2018), March 2018, Ulm, Germany.