Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 1369

pSTAIX - A Process-Aware Architecture to Support Research Processes

Marius Politze[1], Bernd Decker[2] and Thomas Eifert[3]

Abstract: Universities IT service providers are faced with rising demands on existing IT systems and higher degrees of individualization. The challenge thus is to provide services that researchers can use today but that are flexible and sustainable enough to also support tomorrows' research processes. Emerging from previous projects supporting administrative and learning processes, a reference architecture is proposed that aims at providing a general guideline to build process-aware services supporting eResearch. The proposed architecture gives guidance on structuring development and operation of services and formalizes how existing IT systems transition into process-aware services.

Keywords: eResearch; OAuth2; process support; process modelling; research data management; SOA; software architecture

1 Introduction

To cope with rising numbers of students, administrative processes are often heavily supported by centralized university IT systems. Many initiatives focus on eLearning to develop, streamline and standardize technology enhanced teaching and learning. Even though teaching and research are the two core processes used at all universities which are mostly carried out by the same user group and the same organizations. Whereas Supporting technology for scientific research processes still needs significant investments in order to build a foundation for eResearch services.

At university level, there is an initiative to set up an integrated Research Data Management (RDM) system over the next years. A project group focuses on consulting and training as well as on the development of technical solutions for RDM. Since managing data requires extra effort from researchers, usability and seamless integration into existing processes are key to establishing an integrated RDM. Technical solutions need to cover all domains of the research process: the private and collaborative domain, in which researchers actively work with the data, as well as the archive and publication domain, in which data access is less frequently [EMS16].

[1] RWTH Aachen University, IT Center, Seffenter Weg 23, 52080 Aachen, Germany politze@itc.rwth-aachen.de
[2] RWTH Aachen University, IT Center, Seffenter Weg 23, 52080 Aachen, Germany decker@itc.rwth-aachen.de
[3] RWTH Aachen University, IT Center, Seffenter Weg 23, 52080 Aachen, Germany eifert@itc.rwth-aachen.de

doi:10.18420/in2017_137

On the one hand, researchers have rising demands to existing and IT systems and infrastructure in order to gain short-term value. On the other hand, many organizational issues need to be addressed such as defining when and how RDM systems are used within the research process. The challenge for central IT service providers thus is to implement services that researchers want to use today and that are flexible enough to support tomorrows' processes.

In contrast to research processes, learning processes at the university have been supported by central IT services for almost a decade. While processes themselves may be very different, the number of individuals and participating organizations within the university are comparable. Based on experience from previous projects, which support administrative and learning processes with centralized and decentralized IT services, the proposed infrastructure aims to provide a general guideline to tailor process-aware services to support eResearch.

Also digital literacy increases among researchers and changes the way IT services are used. While a set of ready-to-use services are required, individualization and adoption of the processes is becoming more important to heavy users. Institutes can join forces to build an application that enhances common workflows within processes and thus increase their short-term business value. Transferring applications that meet the needs of many researchers into centralized services, in turn increases sustainability and long-term benefits for all users.

2 Related Work

Over more than a decade centralized and decentralized IT services have been established by universities for eLearning and, more general, student lifecycle management (SLM). This development is in line with the general trend in our society. Juling describes that mobility and ubiquity of information technology and therefore global accessibility, reachability have become part of our daily life [Ju09]. Students rightfully have been expecting the digitization of university services as Barkhuus and Dourish have discussed [BD04]. While the technological basis has severely changed since the study in 2004, the social roles, relationships and responsibilities of the students are mostly comparable. Furthermore, former students returned to universities as researchers and are expecting the same kind of support they got when they were students.

Of course this made eResearch and especially RDM a focus of research groups in the past years. With RADAR (Research Data Repository) Kraft et al. established a software as a service (SaaS) that scientists may use to preserve and publish their research data, providing long term storage, persistent object identifiers and metadata [Kr16]. Other approaches such as ProjektRepository [PD16] or TR32DB [Cu14] additionally introduce the researchers' needs for collaboration within a research repository.

In a more general approach research groups are focusing on the design of service oriented architectures (SOA) for arbitrary business processes. The dawn of so called cloud services changed the focus and clearly separates the operation of IT services and SOAs building upon

these as described by Zimmermann et al. [Zi13]. Micro service architecture as described by Namiot et al. [NSS14] and new models for assessing maturity of SOAs as presented by Rathfelder et al. [RG08] and Welke et.el. [WHS11] show how these developments influence the design and implementation of SOAs.

3 Conceptual Model

Based on the existing architecture for supporting eLearning services [PSD16], the conceptual model adds an application proxy, cache and a uniform application programming interface (API) as a SOA as a basis for the development of more complex IT services. The proxy service organizes the access to other systems using vendor specific or legacy APIs of back end systems. Basing the proxy API on the processes rather than the technology of underlying services effectively reduces the impact of vendor lock in. Furthermore, this design allows adding new services and distributing requests between multiple instances or implementations based on defined sets of rules. This actively decouples the systems into smaller functional units, which in turn increases maintainability.

Among ready-to-use applications, the model allows users to extend and integrate processes in their own applications using the provided API. Exposing a programmable interface to all users, however, poses very strict requirements to the implementation: (1) The interface has to be explained to the users, (2) it must not allow abusing the supported workflows within the processes and (3) it has to be secure.

With every supported process in the interface, the complexity of an API increases. Explaining a complex interface to users thus becomes more challenging. Appropriate and consistent use of names, matching the definition and wording within the defined process, therefore is crucial and allows users to transfer their knowledge from taking part in the processes to their implementation. Existing IT services, however, often do not offer such a process-aware view but provide a much more general interface.

If the API is not specific, it may invite users to use it for other workflows than intended. While this may be beneficial for the short-term interests of the user, IT service providers usually desire the usage only within the specified parameters, having to meet technical bounding conditions such as CPU load, disk sizes, and the like. Enforcing these boundaries within the interfaces is crucial for failure-free operation of back end services but may also limit the possible degree of individualization.

Allowing users programmatic access to services, potentially exposes personal and confidential data in a machine readable format. As strictly reviewing all applications implemented by users usually is infeasible, security becomes another design principle of the API. Nevertheless, developers have to register prior to the implementation of their solutions. Applications furthermore often work on behalf or within the context of a user. Commonly, a user supplies credentials such as a user name and a password to authenticate. This allows the back end

services to distinguish between different user contexts and associated rights and roles. However, by giving user credentials to an application it gains full power to perform every possible action within the users' context. Thus, the API should allow to limit applications to access defined subsets of processes.

4 Abstraction to Process Level

Basing on the conceptual model, pSTAIX (Process oriented Software Tiers for Application Interfaces and eXtension) aims to be a reference SOA, abstracting system and technology depended application interfaces to process-aware services. Therefore, pSTAIX uses tiers to allow separation of concerns; especially in terms of separating technology and process dependent parts of the implementation. Each of the tiers is based on the interfaces offered by the previous tiers and combines them to process-aware services. Figure 1 shows an overview of the relationships of different tiers in the reference architecture.

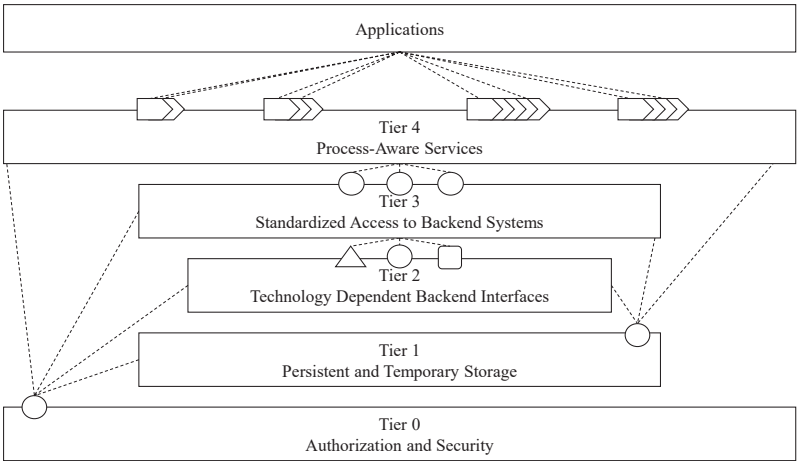

Fig. 1: Overview of the pSTAIX reference architecture

Apart from technologically structuring services, pSTAIX also formalizes the transitions of IT systems into process-aware services offered to application developers and end users. As technology dependencies decrease and services orient more towards defined processes, they can be easier understood and require less technology dependent knowledge about back end systems. Apart from providing a technical structure to elevate services to process-aware services, pSTAIX also provides guidance on how to structure software development and IT operation teams in order to successfully implement and provide a certain kind of service. As a result each team is responsible for providing a defined set of interfaces and can therefore focus on development and implementation in their area of responsibility. Clearly defined interfaces between the tiers allow teams to use different technologies and even to exchange technology stacks if interfaces remain.

4.1 Tier 0: Authorization and Security

Services in this tier are the basis for all other services. For almost every process in university context, it is necessary to identify the users of the services. Once processes span across multiple system boundaries, it needs to be assured that users can access all spanned systems. A centralized identity management (IdM) is crucial for the provisioning of necessary user information to systems. Users' identities and sessions then need to be conveyed between the systems.

This tier, however, only supplies and manages information about the users' identity and current session. Backend systems are generally very well capable of handling their own set of permissions. Whenever possible, tier 0 services should only provide sufficient context about the user such that services in the higher tiers can decide about their permissions.

Development and operation of this tier are crucial for other services. Aggregation of information about identities, groups and roles for higher level tiers makes tier 0 services crucial for data security and data privacy. Services in this tier should focus on state-of-the-art technology to protect and govern usage of personal data.

4.2 Tier 1: Persistent and Temporary Storage

Process-aware services created in the higher tiers often require saving session states or small amount of user data, such as settings, to be preserved between steps of the process. To reduce the need of individual storage solutions, tier 1 introduces persistent storage for small quantities of data. This is crucial, since some back end systems are not designed to be accessed interactively by users and may be slow when put under high load. Also, lifting services to process level may sacrifice efficiency in a way that requests to back end systems need to be repeated multiple times. Short term or temporary storage can be used for caching of such requests and therefore increases the performance of the overall architecture.

Both data stores can operate on either a user or process basis for private or public data respectively. Tier 1 services are likely to use different technologies to separate temporary from persistent data. If temporary storage needs to cache files prior to processing it may need to handle medium quantities of data in the range of a few gigabytes per user. For this kind of general storage, it should, however, be kept in mind that services are generally intended for small quantities of data, in the range of several kilobytes per user. If back end systems need to store more complex or higher volume data they should integrate storage solutions in upper tiers.

4.3 Tier 2: Technology Dependent Backend Interfaces

Opposed to tier 0 and 1, services in tier 2 are very specific to existing systems and depend heavily on the technological details of back end systems. Furthermore, they are generic since they are not designed to support specific processes but simply allow programmatic access to the back end system.

Especially in case of legacy systems, this tier often provides administrative or root access to the back end system or allows impersonating arbitrary users. These interfaces are not commonly available to end users but are used by technical personnel for managing purposes. Most modern systems likely allow resolving the users' session using tier 0 services by using plug-ins or specific extensions.

4.4 Tier 3: Standardized Access to Backend Systems

As technology advances, services in this tier are likely to change and move between internal or external organizations or contractors. The tier architecture, however, limits the impact by this kind of changes by introducing tier 3 services. While tier 2 services change abruptly when the back end system changes, services in this tier can change using evolutionary processes, paving the way for more elaborate software engineering methodologies.

To form a basis for process-aware services, tier 3 introduces standardized protocols and semantics spreading across system boundaries. If not supported by tier 2, these interfaces enforce personalized use and therefore do no longer offer the ability of impersonating arbitrary users. Instead, they work in the current users' context. The services should also integrate the back end systems into the organizational structure of the university by taking identities, groups and roles into account.

4.5 Tier 4: Process-Aware Services

After standardizing the transport protocols, this tier aims to integrate interfaces from previous tiers into process-aware services. It therefore lifts the semantics of the services from technical solutions to organizational and process level. Most importantly, this tier introduces consistent naming and semantics across processes and bundles mandatory or dependent calls to different back ends. Services from this tier are likely to be provided to heavy users as they can be used for individualization, automation and integration of processes in their own applications.

4.6 End User Applications

Last but not least, end user applications are on top of the reference architecture. The previous tiers focused on programmatic access to processes, services and back end systems. Applications should focus on using tier 4 services providing business value for the users. Since services are process-aware, it is easy to implement applications guiding the user through a single process. Furthermore, applications should make use of consistent semantics offered by tier 4 services to span across process boundaries.

5 Key Technologies

Before implementing actual processes on top of the pSTAIX reference architecture, some supporting key technologies were selected. Due to easy use and ubiquitous availability, web technologies have emerged to a defacto standard for process supporting services in many business areas. Even though architecture agnostic, the implementation of the pSTAIX reference architecture makes use of web technologies to build process-aware services.

Representational State Transfer (REST) [Fi00] is probably one of the most used paradigms in modern internet applications. The support of many programming languages and frameworks is very mature. Due to its wide spread in internet applications, also many software developers know how to use REST APIs. In addition, various platforms, ranging from smart devices and sensors via smartphones and desktop computers to compute clusters, are able to access HTTP resources. HTTP has thus emerged as a standard protocol in communication and is therefore used for standardized access in tiers 3 and 4.

Managing universities users, their roles and associations are key for many local services [EB13]. Identity federations are well established and widely used to secure educational internet applications. These federations offer single sign on capabilities and allow exchanging user information between participating services [Gr16]. National federations organize hierarchically to inter-federations such as eduGAIN. This does not only provide a standardized way to authenticate users, but also allows easy sharing and reuse of services across universities and other organizations. The underlying Security Assertion Markup Language (SAML) protocol [Ca05], however, is technically restricted to interactive sessions in a web browser. Other applications, such as apps installed on a smartphone, require additional protocols. Instead of authentication, the OAuth2 protocol allows users to authorize an application to use resources on their behalf [Ha12]. It thus allows secured, personalized access to REST APIs and handles the users' authorization without supplying credentials to the application itself. This also paves the way for third party developers accessing central IT services. The OAuth workflow has various extension points to integrate it further into existing infrastructures. This complex of identity management forms tier 0 in the reference architecture with OAuth as the main workflow to secure APIs for the users.

While not a technology in classical sense, the programmers of the applications are crucial to the success of the conceptual model. Especially researchers from technical disciplines tend to own implementations. As digital literacy increases and APIs are easier to use, it is likely that this trend becomes more popular. Crotch et al. point out that many small supporting implementations may lead to issues in terms of software quality and reproducibility for future researchers [Cr13]. Within the UK, the Software Sustainability Institute addresses these issues by trainings and further support for research oriented software. As IT specialists are a rare resource, recruiting junior staff becomes even more important. Apart from handling basic web technologies, it is mandatory that they understand the business needs of the institutions. Integrated degree programs as presented by Küppers et al. join theory and practice from the very beginning of a study program in the field of computer science [Kü16] and as such aim to solve this problem.

6 Case Study: simpleArchive

The project *simpleArchvie* serves as a prototype scenario to validate how pSTAIX architecture influences the development of an IT service supporting the research process.

Looking at the processes in figure 2, the general idea of *simpleArchive* is as follows: The applications presents a web interface to the researcher, that allows uploading small amounts of research data artifacts (up to two gigabyte). After the data is uploaded, it is assigned a persistent identifier (PID) that uniquely defines the uploaded data and can be used by the researcher as a reference. According to requirements e.g. posed by funding or publishing organizations, *simpleArchive* retains a copy of the data for at least ten years. Researchers and their peers can retrieve stored data from *simpleArchive* if they want to share it at a later point in time.

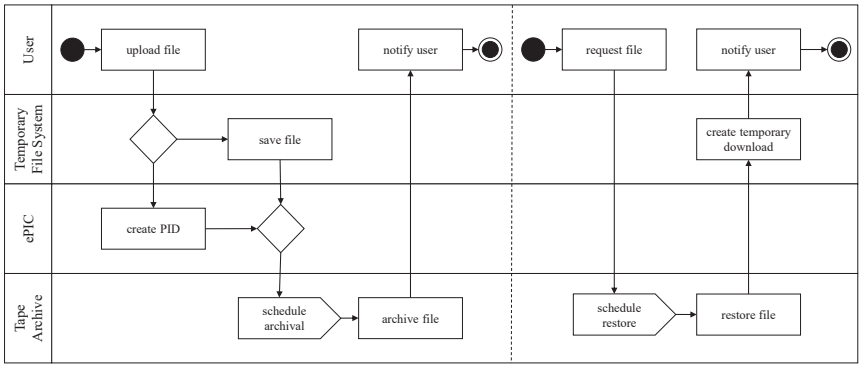

Fig. 2: Simplified overview of *simpleArhive* processes

Obviously, archiving files also poses technical challenges to IT systems. In order to provide long term storage for files, an existing TSM tape archive provides a cost-efficient solution. However, access requires using specialized software and is generally not suitable to be

used by non-technical personnel. This specialized software and a C++ API are therefore technology dependent services from tier 2. The same applies for the system used to register or query PIDs: ePIC, and a temporary file system to store data while being processed by the archive or retrieval queues: *gigaMove* [BBH11]. All the mentioned IT systems were already available and have various system dependent interfaces forming tier 2.

Services in tier 0 and 1 are not specific to eResearch and are therefore shared with already established structures from eLearning as previously described in [PSD16]. Shibboleth and OAuth2 are used to establish a users context and to pass authorizations between different back end systems. The distributed cache and a database storage infrastructure to manage archive queues and store meta information about uploaded files for users were also reused.

Building upon the services from tiers 0 to 2, allowed to develop a REST API to access the different back end systems. Typical services in this part of tier 3 are

- Creation of an archive node,
- Creation of a PID,
- Storing a file to the temporary file system,
- Saving file meta information to the database,
- Writing and reading a file in the archive or
- Making a temporary file available for download.

Most of these services are already working in the users' context. Also, by looking at the described methods, it is fairly possible to resemble the process above. However, from these methods the processes powering *simpleArchive* are not quite obvious: The archive node is not coupled to the PID, nor is the file meta information; writing a file to the archive requires saving it to the temporary storage et cetera. In different contexts, these methods can support a multitude of processes. Therefore, using these methods as building blocks for process-aware services allows developers without knowledge of the exact inner relationships of back end systems to build applications using the process.

In tier 4 *simpleArchive* therefore defines services such as

- Upload a file for archival and return a PID
- List all PIDs and meta information created in *simpleArchive*
- Queue a file for restore and generate a download link

These services use clear naming conventions and identify all uploaded files with the assigned PID. Besides *simpleArchive* already offering a basic web interface for researchers, application developers may then use these services to integrate them into a specialized application, which in turn could be integrated into an institutional intranet or other applications.

7 Future Challenges

To support future processes with centralized IT services, service providers need to gain more insight into actual research processes, find common approaches and how to effectively support them. Experiences from already established approaches, e.g. for didactic coaching in academics and eLearning or business process analysis and consulting in companies, can be reused and applied to enhance research processes.

Performing a good process analysis is key for tier 4 services. If these appeal to the user and match their expectation and experiences, they can be used correctly, integrated into researchers workflows and provide short-term business value. The architecture of pSTAIX allows reusing services from tiers 0 to 3 within future processes. The users, however, are limited to processes-aware services in tier 4. When publishing interfaces for users, consistent naming and semantics are most important and have to be enforced throughout all process-aware services.

Building a process-aware SOA implies connecting different existing back end systems. Thus, an error in one of these systems often leads to a failure of the whole process. To keep the services accessible for users, it is therefore important to isolate different pieces of the service from one another. Micro service architectures help to solve this, but add additional complexity to software engineering and application lifecycle processes during development of APIs and applications. Process-aware APIs are therefore much more complex and require establishing governance and change management systems.

As back end systems are connected, it needs to be assured that they can identify users across system boundaries. On the one hand user accounts and data needs to be accessible and provisioned into back end systems. On the other hand, it needs to be assured that personal data is shared across system and process boundaries only if the user desires to do so. Establishing a centralized identity and group management as well as shared authentication and authorization services is therefore key to spread processes across system boundaries.

8 Conclusion

The presented pSTAIX architecture provides a general guideline for implementing process-aware IT services based upon existing back end systems. The reference architecture can be used to structure developed services but also teams of software developers and IT operators into units with clear responsibilities. While some challenges arise from this newly created setting in organizational and technical context, it was possible to successfully apply the reference architecture in the development of *simpleArchive*.

Building a reference architecture for process-aware services is a bottom up approach from a technical perspective. From point of view of the whole organization there are still many uncertainties on how to supply the created services to the organizational units within the

university. In depth analysis of the research processes needs further attention to successfully deliver short- and long-term business value to researchers.

Even though intended to support research processes, this methodology bears the risk that APIs cannot be used by researchers directly: Basic applications need to be available in order to support research processes without requiring programming capabilities. APIs are provided to adopt or integrate supported processes into existing tools at the institutes. To allow fast adoption in custom scenarios, it is therefore crucial that users' knowledge about existing processes can be reused, when using the APIs.

As digital literacy and ubiquity of computing resources increases, most basic IT services can be leased of bought from specialized companies easily. Institutes will no longer require IT service providers at universities to run these services. Process analysis, identification and implementation of key technologies such as centralized layers for identity management, authorization, caching and continuous development of process-aware and integrated APIs to off-the-shelf IT components is an emerging competence that existing university IT service providers need to build up in order to keep their right to exist.

References

[BBH11] Bischof, Christian; Bunsen, Guido; Hinzelmann, Sebastian: Gigamove – Einfach und schnell große Dateien austauschen. In (Verein zur Förderung eines Deutschen Forschungsnetzes e.V., ed.): DFN Mitteilungen (Ausgabe 80), pp. 30–32. Berlin, 2011.

[BD04] Barkhuus, Louise; Dourish, Paul: Everyday Encounters with Context-Aware Computing in a Campus Environment. In (Hutchison, David; Kanade, Takeo; Kittler, Josef; Kleinberg, Jon M.; Mattern, Friedemann; Mitchell, John C.; Naor, Moni; Nierstrasz, Oscar; Pandu Rangan, C.; Steffen, Bernhard; Sudan, Madhu; Terzopoulos, Demetri; Tygar, Dough; Vardi, Moshe Y.; Weikum, Gerhard; Davies, Nigel; Mynatt, Elizabeth D.; Siio, Itiro, eds): UbiComp 2004: Ubiquitous Computing, volume 3205 of Lecture Notes in Computer Science, pp. 232–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[Ca05] Cantor, Scott; Kemp, John; Philpott, Rob; Maler, Eve, eds. Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 2005.

[Cr13] Crouch, Stephen; Hong, Neil Chue; Hettrick, Simon; Jackson, Mike; Pawlik, Aleksandra; Sufi, Shoaib; Carr, Les; de Roure, David; Goble, Carole; Parsons, Mark: The Software Sustainability Institute: Changing Research Software Attitudes and Practices. Computing in Science & Engineering, 15(6):74–80, 2013.

[Cu14] Curdt, Constanze: Design and Implementation of a Research Data Management System: The CRC/TR32 Project Database (TR32DB). Dissertation, Universität zu Köln, Cologne, 2014.

[EB13] Eifert, Thomas; Bunsen, Guido: Grundlagen und Entwicklung von Identity Management an der RWTH Aachen. PIK - Praxis der Informationsverarbeitung und Kommunikation, 36(2), 2013.

[EMS16] Eifert, Thomas; Muckel, Stephan; Schmitz, Dominik: Introducing Research Data Management as a Service Suite at RWTH Aachen University. In (Müller, Paul; Neumair, Bernhard; Reiser, Helmut; Dreo Rodosek, Gabi, eds): 9. DFN-Forum Kommunikationstechnologien, volume 257 of GI Edition Lecture Notes in Informatics Proceedings (LNI), pp. 55–66. Köllen, Bonn, 2016.

[Fi00] Fielding, Roy Thomas: Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.

[Gr16] Grabatin, Michael; Hommel, Wolfgang; Metzger, Stefan; Pöhn, Daniela: Improving the Scalability of Identity Federations through LevelofAssurance Management Automation. In (Müller, Paul; Neumair, Bernhard; Reiser, Helmut; Dreo Rodosek, Gabi, eds): 9. DFN-Forum Kommunikationstechnologien, volume 257 of GI Edition Lecture Notes in Informatics Proceedings (LNI), pp. 67–76. Köllen, Bonn, 2016.

[Ha12] Hardt, D.: The OAuth 2.0 Authorization Framework. RFC Editor, 2012.

[Ju09] Juling, Wilfried: Vom Rechnernetz zu e-Science. PIK - Praxis der Informationsverarbeitung und Kommunikation, 32(1):33–36, 2009.

[Kr16] Kraft, Angelina; Razum, Matthias; Potthoff, Jan; Porzel, Andrea; Engel, Thomas; Lange, Frank; van den Broek, Karina; Furtado, Filipe: The RADAR Project - A Service for Research Data Archival and Publication. ISPRS International Journal of Geo-Information, 5(3):28, 2016.

[Kü16] Küppers, Bastian; Dondorf, Thomas; Willemsen, Benno; Pflug, Hans Joachim; Vonhasselt, Claudia; Magrean, Benedikt; Müller, Matthias S.; Bischof, Christian: The Scientific Programming Integrated Degree Program – A Pioneering Approach to Join Theory and Practice. Procedia Computer Science, 80:1957–1967, 2016.

[NSS14] Namiot, Dmitry; Sneps-Sneppe, Manfred: On Micro-services Architecture. International Journal of Open Information Technologies, 2014.

[PD16] Politze, Marius; Decker, Bernd: Ontology Based Semantic Data Management for Pandisciplinary Research Projects. In (Curdt, Constanze; Wilmes, Christian, eds): Proceedings of the 2nd Data Management Workshop, volume 96 of Kölner Geographische Arbeiten. Cologne, Germany, 2016.

[PSD16] Politze, Marius; Schaffert, Steffen; Decker, Bernd: A secure infrastructure for mobile blended learning applications. In (Bergström, Johan, ed.): European Journal of Higher Education IT 2016-1. Umeå, 2016.

[RG08] Rathfelder, Christoph; Groenda, Henning: iSOAMM: An Independent SOA Maturity Model. In (Meier, René; Terzis, Sotirios, eds): Distributed Applications and Interoperable Systems, volume 5053 of Lecture Notes in Computer Science, pp. 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[WHS11] Welke, R.; Hirschheim, R.; Schwarz, A.: Service-Oriented Architecture Maturity. Computer, 44(2):61–67, 2011.

[Zi13] Zimmermann, Alfred; Sandkuhl, Kurt; Pretz, Michael; Falkenthal, Michael; Jugel, Dierk; Wissotzki, Matthias: Towards an integrated service-oriented reference enterprise architecture. In (Knauber, Peter; Knodel, Jens; Lungu, Mircea Filip, eds): Proceedings of the 2013 International Workshop on Ecosystem Architectures. pp. 26–30, 2013.