The ReSUS Project - Infrastructure for Sharing Research Software

Markus Hirsch[1], Dorothea Iglezakis[1], Frank Leymann[2] and Michael Zimmermann[2]

[1]University Library, University of Stuttgart, Germany
[2]Institute of Architecture of Application Systems, University of Stuttgart, Germany

The goal of the ReSUS project is to develop an overarching concept and an operable solution to make research software more findable, accessible, interoperable, and reusable. It will help to archive and provision research software and data, and ensure that it will be executable and usable in the future regardless of the execution environment. It will create a concept to store the software and associated research data in one place, combine it with metadata, license information, a unique identifier, and other relevant information.

1 The Motivation for the ReSUS Project

There has been a lot of progress in the last years concerning FAIR data [1] in the scientific community. Several institutions, initiatives and working groups have exchanged ideas, developed principles, standards, and created initiatives to spread those principles and ideas [2, 3]. It has also become clear that not only research data should become findable, accessible, interoperable, and reusable, but also the research software that was used in the process of generating, analyzing, and interpreting this data [4, 5].

Currently, the awareness for the importance of publishing the code of self-created scientific software is slowly rising [6]. Publishing the plain code on platforms like GitHub[1] is a first step in the direction of more FAIR research software. However, investigations show that lack of documentation leads to difficulties in executing such code [7]. Even if a description on how to run the code is available, it might be too complicated for non-tech scientists to execute. Descriptions may also be incomplete or outdated as execution environments and required components may change over time. Legal uncertainties in the licensing of software can also reduce the likelihood of publications.

Therefore, the goal of ReSUS is on the one hand to make it as convenient as possible for the creators to publish their code by supporting them in selecting a suitable license, in describing the software with appropriate metadata and in modelling dependencies. On the other hand, ReSUS will make it easier to find and use existing research software by

[1] `https://github.com/`, all links last accessed on 2021-04-29.

Publiziert in: Vincent Heuveline, Nina Bisheh (Hg.): E-Science-Tage 2021. Share Your Research Data. Heidelberg: heiBOOKS, 2022.
DOI: https://doi.org/10.11588/heibooks.979.c13737 (CC BY-SA 4.0)

offering a search index and by automatically providing the software with all its dependencies. Encapsulating the code and all necessary artifacts in a self-describing open source container format will ensure that software can be executed independently from any particular executing environment. This will increase usability regardless of future technological developments and also benefit technically less experienced users.

2 Providing Help in the License Selection Process

Software licenses are very important for the reuse of research software. They determine under which conditions and restrictions a software can be used or integrated into own code. In the ReSUS project, we will focus on free/libre open source licenses (FLOSS)[3] for software.

For the creators of scientific software, who in the most cases are not trained software developers, it is difficult and time consuming to get a general overview about license types and regulations, and select a suitable license for their own code. When their own code is based on a multitude of other code parts or libraries under different licenses, even computer scientists and software developers may struggle with the complexity of license regulations and their implications [8]. With rising complexity, the danger of license violations rises, as certain licenses, even from the same license type, might be mutually incompatible.

There is a variety of accessible information and tools that help with this. The SPDX license list[4] is widely used as a reference. It gives a comprehensive presentation of FLOSS licenses and their versions. For each of them, it provides a permanent URL and a unique identifier.

Choosealicense[5] presents a good overview and categorization of the main features of a selection of the most widely used FLOSS licenses. It also provides a brief description for each license and a very concise guide to select a license according to one's preferences. This is very useful for newly created software, but does not take into account restrictions of licenses, if software is built upon existing code.

The Joinup Licensing Assistant[6] allows to search and compare FLOSS licenses according to a large variety of criteria. It also provides a compatibility checker[7] to verify, if a certain license may be assigned if the own code is built upon code with another license. However, only two licenses can be compared at the same time (one inbound, one outbound).

[3] https://dwheeler.com/essays/floss-license-slide.html
[4] https://spdx.org/licenses/
[5] https://choosealicense.com
[6] https://joinup.ec.europa.eu/collection/eupl/solution/joinup-licensing-assistant/jla-find-and-compare-software-licenses
[7] https://joinup.ec.europa.eu/collection/eupl/solution/joinup-licensing-assistant/jla-compatibility-checker

So the case with multiple inbound licenses can not be covered easily, plus a user without knowledge about licenses may not know which licenses to compare, as no recommendation is made.

As the information and functionality of these tools is very useful and may be used in other projects, the license checker in the ReSUS project will build upon this existing knowledge. The goal is to create an automated compatibility checker integrated in the publishing process that guides the creator to a suitable license.

To do so, our knowledge about FLOSS licenses will be represented in a license ontology, which is accessible via a SPARQL endpoint. It will be based upon existing information as mentioned above. It will contain the properties of licenses, such as permissions, limitations, and the condition of reuse. The relation and mutual compatibility of the licenses will be described via these properties.

We use Fossology [9] to look for existing licenses of used components or libraries in the source code. The license checker will then get a list of used licenses as an input, access the ontology, and return only licenses that do not contradict those existing ones. This will prevent unintended license violations. In a second step, creators will be able to select their preferences guided by questions ("Are there prerequisites from your institution concerning licenses?", "Do you want to have your software used by as many people as possible?"). The questions and the compatibility check will prevent the creator from unintentionally choosing a license that might inhibit reuse. More in depth information will be supplied for those who want more explanation, while keeping the process as simple as possible.

3 Adding Software Metadata

Crucial for the citability, the findability, and the reusability of research software are structered metadata. As stated in the Software Citation Principles in [10], most important for the citation of software are a persistent identifier (PID), information about the name and version of the software, the authors, the release date, and the location or repository of the software. With the Citation File Format[7] there is an uncomplicated way to add these information in form of a structured YAML-file to software code.

For the reusability of software, there has to be some more information in the metadata. According to the redefinition of the FAIR principles for software [4], software should be described with metadata "so that it can be replicated, combined, reinterpreted, reimplemented, and/ or used in different settings". This includes (quite vaguely) "a plurality of accurate and relevant attributes", a detailed provenance and qualified references to other software. Most important for the reusability of code is the statement of all dependencies that are necessary to get this code run. CodeMeta [11] addresses these challenges with a multitude of attributes not only to citation metadata but also - among others - about the status of development, required or optional dependencies, links to help, documentation

[7] `https://citation-file-format.github.io/`

and issue trackers, linking to a funding and embargos. CodeMeta bases on schema.org[8] the ontology for structured information on the web and is defined as a JSON-LD scheme. CodeMeta files are therefore somewhat more complicated to create, but much more powerful for describing research software.

But what about the plurality of accurate and relevant attributes that metadata of FAIR software should include? So far, both the Citation File Format and CodeMeta include general bibliographic information for citing or technical information about the software part of research software, but not about the research part. What kind of research can be done with this software? Which models or procedures are implemented? What methods can be used? These are information important for searching and important for linking software to other research outputs.

Within the ReSUS platform, the component to describe software with metadata is our data (and code) repository DaRUS basing on the repository software Dataverse[9]. In Dataverse, metadata schemes are defined in the form of metadata blocks that can then be activated for a dataset collection (so called dataverse) if required to describe the contained datasets. Dataverse is maintained and developed from an international community led by the Institute of Quantitative Social Science (IQSS) of Harvard. Within this community, a working group with our participation is discussing and working on the handling of research software within Dataverse including agreeing upon a metadata block for software basing on CodeMeta. Having this metadata block for the technical description of research software, the next step will be a guide to use descriptive metadata to describe the research aspects of the software.

4 Prototypical Implementation

In this section, we present our prototypical implementation planned for the ReSUS platform. Therefore, we firstly present an overview of the overall architecture of the platform. Furthermore, we describe the individual components of the platform in more detail and show where we are building on existing software components and standards.

Figure 1 shows the five main components of our prototype: (i) the *Library System*, (ii) the *Storage Backend*, (iii) the *ReSUS Frontend*, (iv) the *ReSUS Backend*, and (v) the *ReSUS Modeling Tool*. Furthermore, on the bottom of the figure, used external services are depicted, for example, Fossology[10] for checking the licenses or a DOI Provider for creating persistent IDs. On the right side of the figure, the ReSUS Modeling Tool for creating Research Object Archives (ROARs) [12] is depicted. In summary, ROARs enable the packaging, publishing, and installation of research software using a self-contained and portable packaging format.

[8] `https://schema.org/`

[9] `https://dataverse.org/`. For an overview of all components of ReSUS see section 4

[10] `https://www.fossology.org`

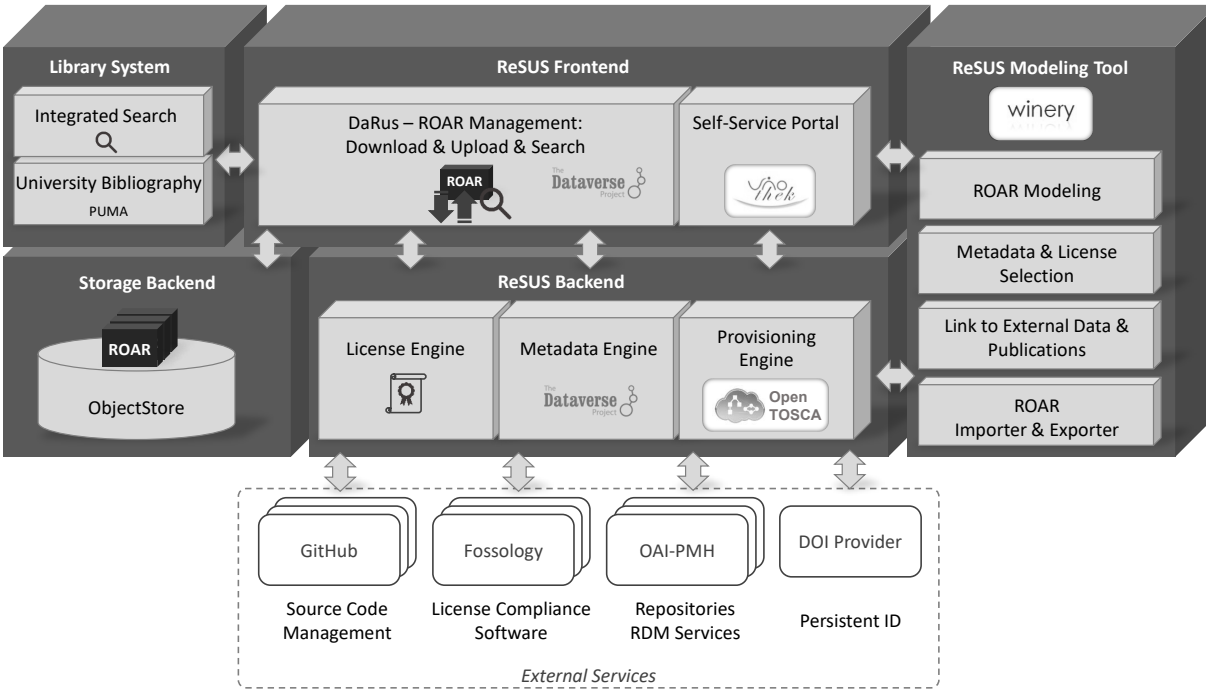

Figure 1: System architecture of the ReSUS platform.

In addition, all important information of research software, especially its technical dependencies, related research data, descriptive metadata, licenses, and references to corresponding publications can be bundled within a ROAR.

The ReSUS Modeling Tool is based on the OpenTOSCA modeling tool Winery [13] and extends it with additional required functionalities regarding the management and creation of ROARs. Here, the topology of the application with all required dependencies can be modeled and managed. Moreover, metadata can be added to describe the ROAR and a license can be selected. Required data can either be packaged directly within the ROAR, or referenced to a remote location. Likewise, related publications can be referenced. The ROARs can be exported from Winery as well as imported. They are self-describing and contain all artifacts and information necessary for the automated provisioning [14].

For modeling the topology of the application, the Topology and Orchestration Specification for Cloud Applications (TOSCA) [15, 16] is used. The OASIS standard TOSCA allows to define the deployment of applications by topology models as well as management plans. A topology model consists of the components of the application as well as their relations to each other. For example, it can be defined that a web application shall be hosted on an application server and also shall be connected to a database, where it reads data from. Such modeled applications can be executed automatically by a corresponding TOSCA runtime environment, like for example the TOSCA-based provisioning engine OpenTOSCA Container [17].

An extended version of the OpenTOSCA provisioning engine is part of the ReSUS Backend and is able to consume the ROARs exported from Winery and interpret the topology model describing the application. Furthermore, it executes all required steps in order to provision an instance of the modeled application [18].

Beside the modeling tool Winery and the provisioning engine Container, the self-service portal Vinothek [19] is also part of the OpenTOSCA Ecosystem.

In our ReSUS platform, the Vinothek is part of the ReSUS Frontend, and allows the end-user to manage and initiate the provisioning of the modeled application contained in a selected ROAR. If required, user specific as well as use-case specific variables, such as credentials to a cloud service or the location of data to be processed, are requested to be entered here by the end-user.

In addition to the OpenTOSCA ecosystem, with Dataverse and DaRUS we also build on further existing software. DaRUS[11] is part of the ReSUS Frontend for managing the ROARs. It is based on Dataverse and allows to upload, download, and search for ROARs. In the ReSUS Backend, our Metadata Engine is also based on Dataverse (see section 3). It supplies all components with required metadata and provides standardized interfaces for retrieving metadata and registering persistent IDs. The persistent ID ensures the citability and findability of the research software. The License Engine, which is also part of the ReSUS Backend, checks and manages the licenses of the research software. To do this, it uses external license checking software, such as Fossology or ScanCode[12]. Moreover, the License Engine is intended to assist the researcher in the selection of a license by suggesting a compatible license (see section 2).

On the left side of figure 1, the Library System is shown. The goal is to connect the ReSUS platform with the existing Library System in order to be able to search for ROARs and integrate them into the already available bibliography. Moreover, the depicted Storage Backend is used in order to store the ROARs in a sustainable way.

To sum up, we want to implement our concepts of the ReSUS platform based on standards and already existing software, such as TOSCA, the OpenTOSCA Ecosystem, and Dataverse. In the ReSUS project, we plan on extending and adapting the mentioned components by adding additionally functionality regarding the handling of ROARs, especially the linking of data, referencing of publications, creation of IDs, adding of metadata, and selection of licenses. The presented software components are open-source and can be obtained from GitHub[13,14].

[11] `https://darus.uni-stuttgart.de`
[12] `https://github.com/nexb/scancode-toolkit`
[13] `https://github.com/OpenTOSCA`
[14] `https://github.com/IQSS/dataverse`

5 Conclusion

In this paper, we outlined our goals and concepts for an operable solution to make research software more discoverable, accessible, interoperable, and reusable. Therefore, in this work, we first illustrated the current problems in the area of research software, for example, regarding the selection of an appropriate license in order to be able to publish it. Furthermore, we highlighted the importance of metadata for research software in order to fulfill the FAIR principles. Moreover, we presented our ideas and concepts in order to tackle the illustrated issues. Finally, we depicted an architecture overview and presented the single components of our planned ReSUS platform, which is based on existing software components and standards, such as the OpenTOSCA ecosystem and the TOSCA standard. In the future work, we focus on refining and implementing the presented concepts.

Acknowledgements

This work was partially funded by the German Research Foundation (DFG) project "ReSUS (Reusable Software University of Stuttgart)" (GEPRIS 425911815).

Bibliography

[1] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3:160018–, March 2016. URL: `http://dx.doi.org/10.1038/sdata.2016.18`.

[2] Mark D Wilkinson, Susanna-Assunta Sansone, Erik Schultes, Peter Doorn, Luiz Olavo Bonino da Silva Santos, and Michel Dumontier. A design framework and exemplar metrics for FAIRness. *bioRxiv*, 2017. URL: `https://www.biorxiv.org/content/early/2017/12/01/225490`, `arXiv:https://www.`

biorxiv.org/content/early/2017/12/01/225490.full.pdf, https://doi.org/10.1101/225490

[3] Christopher Erdmann, Natasha Simons, Reid Otsuji, Stephanie Labou, Ryan Johnson, Guilherme Castelao, Bia Villas Boas, Anna-Lena Lamprecht, Carlos Martinez Ortiz, Leyla Garcia, Mateusz Kuzak, Paula Andrea Martinez, Liz Stokes, Tom Honeyman, Sharyn Wise, Josh Quan, Scott Peterson, Amy Neeser, Lena Karvovskaya, Otto Lange, Iza Witkowska, Jacques Flores, Fiona Bradley, Kristina Hettne, Peter Verhaar, Ben Companjen, Laurents Sesink, Fieke Schoots, Erik Schultes, Rajaram Kaliyaperumal, Erzsébet Tóth-Czifra, Ricardo de Miranda Azevedo, Sanne Muurling, John Brown, Janice Chan, Niamh Quigley, Lisa Federer, Douglas Joubert, Allissa Dillman, Kenneth Wilkins, Ishwar Chandramouliswaran, Vivek Navale, Susan Wright, Silvia Di Giorgio, Mandela Fasemore, Konrad Förstner, Till Sauerwein, Eva Seidlmayer, Ilja Zeitlin, Susannah Bacon, Katie Hannan, Richard Ferrers, Keith Russell, Deidre Whitmore, and Tim Dennis. Top 10 FAIR Data & Software Things, February 2019. https://doi.org/10.5281/zenodo.2555498

[4] Daniel S. Katz, Morane Gruenpeter, and Tom Honeyman. Taking a fresh look at FAIR for research software. *Patterns*, 2(3), March 2021. https://doi.org/10.1016/j.patter.2021.100222

[5] Anna-Lena Lamprecht, Leyla Garcia, Mateusz Kuzak, Carlos Martinez, Ricardo Arcila, Eva Martin Del Pico, Victoria Dominguez Del Angel, Stephanie van de Sandt, Jon Ison, Paula Andrea Martinez, Peter McQuilton, Alfonso Valencia, Jennifer Harrow, Fotis Psomopoulos, Josep Ll. Gelpi, Neil Chue Hong, Carole Goble, and Salvador Capella-Gutierrez. Towards FAIR principles for research software. *Data Science*, pages 1–23, November 2019. URL: https://doi.org/10.3233%2Fds-190026, https://doi.org/10.3233/ds-190026

[6] Hartwig Anzt, Felix Bach, Stephan Druskat, Frank Löffler, Axel Loewe, Bernhard Y. Renard, Gunnar Seemann, Alexander Struck, Elke Achhammer, Piush Aggarwal, Franziska Appel, Michael Bader, Lutz Brusch, Christian Busse, Gerasimos Chourdakis, Piotr Wojciech Dabrowski, Peter Ebert, Bernd Flemisch, Sven Friedl, Bernadette Fritzsch, Maximilian D. Funk, Volker Gast, Florian Goth, Jean-Noël Grad, Jan Hegewald, Sibylle Hermann, Florian Hohmann, Stephan Janosch, Dominik Kutra, Jan Linxweiler, Thilo Muth, Wolfgang Peters-Kottig, Fabian Rack, Fabian H.C. Raters, Stephan Rave, Guido Reina, Malte Reißig, Timo Ropinski, Joerg Schaarschmidt, Heidi Seibold, Jan P. Thiele, Benjamin Uekermann, Stefan Unger, and Rudolf Weeber. An environment for sustainable research software in germany and beyond: current state, open challenges, and call for action. *F1000Research*, 9:295, January 2021. URL: https://doi.org/10.12688%2Ff1000research.23224.2, https://doi.org/10.12688/f1000research.23224.2

[7] Christian Collberg, Todd Proebsting, and Alex M. Warren. Repeatability and benefaction in computer systems research. studie, 2015. URL: http://reproducibility.cs.arizona.edu/v2/RepeatabilityTR.pdf.

[8] Daniel A. Almeida, Gail C. Murphy, Greg Wilson, and Mike Hoye. Do software developers understand open source licenses? In *Proceedings of the 25th International Conference on Program Comprehension*, ICPC '17, pages 1–11. IEEE Press, 2017. https://doi.org/10.1109/ICPC.2017.7

[9] Michael C. Jaeger, Oliver Fendt, Robert Gobeille, Maximilian Huber, Johannes Najjar, Kate Stewart, Steffen Weber, and Andreas Würl. The FOSSology Project: 10 Years Of License Scanning. *Journal of Open Law, Technology & Society*, 9(1):9–18, 2018. URL: https://www.jolts.world/index.php/jolts/article/view/123.

[10] Arfon M. Smith, Daniel S. Katz, Kyle E. Niemeyer, and FORCE11 Software Citation Working Group. Software citation principles. *PeerJ Computer Science*, 2(e86), 2016. URL: https://www.force11.org/software-citation-principles, https://doi.org/10.7717/peerj-cs.86

[11] Matthew B. Jones, Carl Boettiger, Abby Cabunoc Mayes, Arfon Smith, Peter Slaughter, Kyle Niemeyer, Yolanda Gil Gil, Martin Fenner, Krzysztof Nowak, Mark Hahnel, Luke Coy, Alice Allen, Mercè Crosas, Ashley Sands, Neil Chue Hong, Patricia Cruse, Dan Katz, and Carole Goble. Codemeta: an exchange schema for software metadata. version 2.0. 2017. https://doi.org/10.5063/schema/codemeta-2.0

[12] Michael Zimmermann, Uwe Breitenbücher, Jasmin Guth, Sibylle Hermann, Frank Leymann, and Karoline Saatkamp. Towards Deployable Research Object Archives Based on TOSCA. In *Papers from the 12th Advanced Summer School on Service-Oriented Computing (SummerSoC 2018)*, pages 31–42. IBM Research Division, October 2018.

[13] Oliver Kopp, Tobias Binz, Uwe Breitenbücher, and Frank Leymann. Winery - A Modeling Tool for TOSCA-based Cloud Applications. In *Proceedings of 11th International Conference on Service-Oriented Computing (ICSOC'13)*, volume 8274 of *LNCS*, pages 700–704. Springer Berlin Heidelberg, December 2013. https://doi.org/10.1007/978-3-642-45005-1_64

[14] Michael Zimmermann, Uwe Breitenbücher, Lukas Harzenetter, Frank Leymann, and Vladimir Yussupov. Self-Contained Service Deployment Packages. In *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER 2020)*, pages 371–381. SciTePress, May 2020.

[15] OASIS. *Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0.* Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[16] OASIS. *Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0.* Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[17] Uwe Breitenbücher, Christian Endres, Kálmán Képes, Oliver Kopp, Frank Leymann, Sebastian Wagner, Johannes Wettinger, and Michael Zimmermann. The OpenTOSCA Ecosystem - Concepts & Tools. *European Space project on Smart*

Systems, Big Data, Future Internet - Towards Serving the Grand Societal Challenges - Volume 1: EPS Rome 2016, pages 112–130, December 2016. https://doi.org/10.5220/0007903201120130

[18] Michael Zimmermann, Uwe Breitenbücher, Michael Falkenthal, Frank Leymann, and Karoline Saatkamp. Standards-based Function Shipping - How to use TOSCA for Shipping and Executing Data Analytics Software in Remote Manufacturing Environments. In *Proceedings of the 21st IEEE International Enterprise Distributed Object Computing Conference (EDOC 2017)*, pages 50–60. IEEE, October 2017.

[19] Uwe Breitenbücher, Tobias Binz, Oliver Kopp, and Frank Leymann. Vinothek - A Self-Service Portal for TOSCA. In *Proceedings of the 6th Central-European Workshop on Services and their Composition (ZEUS 2014)*, pages 69–72. CEUR-WS.org, February 2014.