FITeagle: A Semantic Testbed Management Framework

Alexander Willner
Architekturen der Vermittlungsknoten (AV)
Technische Universität Berlin,
Berlin, Germany
alexander.willner@tu-berlin.de

Daniel Nehls
Architekturen der Vermittlungsknoten (AV)
Technische Universität Berlin,
Berlin, Germany
daniel.nehls@tu-berlin.de

Thomas Magedanz
Next Generation Network Infrastructures
Fraunhofer FOKUS
Berlin, Germany
thomas.magedanz@fokus.fraunhofer.de

ABSTRACT

Approaches to share resources across administrative domains have been developed for many years. A specific field of application is the experimental evaluation of Future Internet related research within federated facilities. Testbeds that want to offer support for this experiment life cycle have to expose multiple interfaces along with mechanisms to allow handovers between them. Existing work rest upon self-contained software components that are bound to specific protocols and schematic data models that aggravate such handovers. As a result, for a testbed the process of joining one or multiple federations is an expensive process. We propose an approach that reutilizes insights of the Semantic Web research to address parts of this issue. In this paper, we describe a framework which abstracts from specific protocols and functionalities by managing heterogeneous resources based on a semantic information model. As a result, different API calls relate to the semantically same resources which allows theses handovers and further makes this approach applicable to related fields such a federated cloud computing. We have validated our work within several research projects and further show that the implementation is standard compliant.

Categories and Subject Descriptors

C.2.0 [**Computer Systems Organization**]: GENERAL—*System architectures*; C.2.4 [**Computer Systems Organization**]: COMPUTER-COMMUNICATION NETWORKS—*Distributed Systems*; D.2.9 [**Software**]: SOFTWARE ENGINEERING—*Management*; D.2.12 [**Software**]: SOFTWARE ENGINEERING—*Interoperability*

General Terms

Management, Design, Experimentation

Keywords

Semantic, Testbed, Infrastructure, Federation, Management

1. INTRODUCTION

Sharing and granting access to geographical dispersed resources is the underlying concept in the field of Distributed Computing. Associated with this, considerable efforts have been spent on architectures to manage heterogeneous resources across multiple administrative domains. We denote this as a resource federation.

A specific field of application is the distribution of experiments between geographically distributed test environments in the context of the Future Internet (FI) research. Given the scale, complexity and heterogeneity of the Internet, one method to evaluate these new approaches is experimental validation conducted within large-scale test environments. Therefore, several independent testbeds have been established accompanied by architectures and protocols to support the whole experiment life-cycle across multiple sites.

An important challenge that arises in this context is that facilities that open access to their resources for remote experimentation have to support a number of Application Programmers Interfaces (APIs) for users from one or multiple federations. Due to the diversity and complexity of the involved interfaces their implementations are rather specialized and have been developed mainly independently of each other. Particularly they exchange information based on different tree-based data models such as XML or JSON and different protocols such as XML Remote Procedure Calls (RPCs), Extensible Messaging and Presence Protocol (XMPP), Advanced Message Queuing Protocol (AMQP) or plain TCP sockets. As a result, while managing the very same resources within a testbed, this approach aggravates the required handovers between them and makes the deployment and maintenance within a facility an expensive process.

Existing work has its origin in the Future Internet Research and Experimentation[1] (FIRE) and Global Environment for Network Innovations[2] (GENI) initiatives. Three main protocols have been chosen to support the federated experiment life cycle. To describe, discover, reserve, provision and release resources the Slice-based Federation Architecture[1] (SFA) has been applied. To control the resources within an experiment, the Federated Resource Control Protocol[2] (FRCP) is being used and related to this measurement information are transported using the OML Measurement

[1] http://ict-fire.eu
[2] http://geni.net

TRIDENTCOM 2015, June 24-25, Vancouver, Canada

DOI 10.4108/icst.tridentcom.2015.259748

Stream Protocol[3] (OMSP). Further, within the Federation for FIRE[3] (Fed4FIRE) project additional services, such as reputation management, resource orchestration and resource reservation are being implemented as additional components. Notwithstanding the above, the federation of infrastructures is also subject in similar contexts, again implementing other protocols. In particular within the Future Internet Public Private Partnership[4] (FI-PPP) the federation of nodes is implemented based on Future Internet Core Platform[5] (FIWARE) components. And the IEEE group Standard for Intercloud Interoperability and Federation[4][6] (P2302) is focusing again on other technologies to interconnect heterogeneous cloud environments.

In this paper, we assume that (i) experimentation and federation mechanisms that are already in place, are unlikely to be replaced in the medium term, (ii) a majority of the functionality of these technologies are working in principle on the same set of information, (iii) almost everything (a resource, service, testbed or personnel) is a concept that can be modeled and federated, (iv) existing testbed features (such as user databases or billing mechanisms) must be incorporated, (v) the concept of federated resource management will be adopted by more fields of application in the future, and (vi) in particular the diversity of involved resources will increase, given the emerging Intercloud[5] and 5G-related fields Internet of Things (IoT) / Machine-To-Machine Communication (M2M), Software Defined Networking (SDN) and Network Function Virtualization (NFV).

The main contribution of this paper is the presentation of a reference implementation of the Federated Infrastructure Resource Management Architecture[6] (FIRMA) using the Federated Infrastructure Description and Discovery Language[7] (FIDDLE) and Open-Multinet[8][7] (OMN) ontologies; each defined earlier by the authors. In the initial development phase, three main focus points have been selected. First, the framework supports both an SFA and a REST based API for the provisioning, monitoring, and control of resources. Second, it is designed to be notably reusable and applicable to different areas of application by supporting arbitrary protocols and heterogeneous resources. Third, to achieve this goal, the framework consists of multiple decoupled Microservices[9] that exchange messages based on a Semantic Web[10] compliant information model. The implementation allows testbed providers to offer resources by several protocols at once, which reduces maintenance costs, allows developers to efficiently implement and link management protocols and users can execute complex queries to find heterogeneus resources based on a large semantic graph that has been contructed on the server side by reasoning over the available descriptions.

We have initially validated our approach by intense discussions within the FIRE and adjoining communities, the analysis of their requirements and the inspection of related work. Further, we have integrated the reference implementation in different testbeds and projects to demonstrate its applicability and we show the SFA compliancy of the implementation. Finally, the underlying semantic concept is currently in the process of being refined and standardized by an international consortium that is independent of specific research projects or products.

The remainder of the paper is structured as follows. We give a brief overview of related work in the context of resource management in federated testbeds in Sec. 2. In Sec. 3, we describe our approach in more detail including the architecture and implementation details. The results obtained from the current implementation are discussed in Sec. 4. Finally, we close giving some conclusions and considerations and describe future work in Sec. 5.

2. RELATED WORK

The experiment life cycle consists of multiple, correlated phases (cf. Figure 1), that can be grouped into three distinct areas: (i) resource description, discovery, requirement specification, reservation, and provisioning (direct or orchestrated) and termination; (ii) experiment control and monitoring; and (iii) their underlying federated identity, authorization and trust related mechanisms (such as Service Level Agreement (SLA) related facility, infrastructure and reputation monitoring). Testbeds that want to join a FIRE federation have to support multiple APIs in this regard. Based on identified characteristics, different possible federation architectures have been evaluated[3] and the heterogeneous federation approach was recognized to be the most suitable one for the FI experimentation facilities under evaluation. In this architecture all testbeds run their native testbed management software and according related work is listed below.

For the first area, the XML RPC based SFA Aggregate Manager (AM) API has to be offered by a testbed and depending on the use case also the Slice Authority (SA) API. On the server side, a number of different implementations are available. On the one hand, each of the five competing control frameworks[11] built in the GENI context offer an SFA AM API. On the other hand, several resource specific implementations are available such as the Flowvisor OpenFlow Aggregate Manager[8] (FOAM), the GENI Rack Aggregate Manager (GRAM) or the NITOS broker. Further, more generic wrappers have been developed, namely the Python based AMsoil that was part of the OFELIA Control Framework (OCF), SFAWrap[9], and the GENI Control Framework (GCF). Users can also refer to a number of tools such as the HTML based MySlice[10] Graphical User Interface (GUI), the Java package jFed[11], the Python Command Line Interface (CLI) sfi or omni, and others.

For the second area, either plain SSH access is being offered or, for more sophisticated experiment work flows, the XMPP or AMQP based FRCP API. As implementations for the testbed side mainly Resource Controllers (RCs) are being deployed that are part of the cOntrol and Manage-

[3] http://oml.mytestbed.net/doc/oml/latest/doxygen/omsp.html
[4] http://fi-ppp.eu
[5] http://fi-ware.org
[6] http://standards.ieee.org/develop/project/2302.html
[7] http://open-multinet.info
[8] https://openflow.stanford.edu/display/FOAM
[9] http://sfawrap.info
[10] http://myslice.info
[11] http://jfed.iminds.be

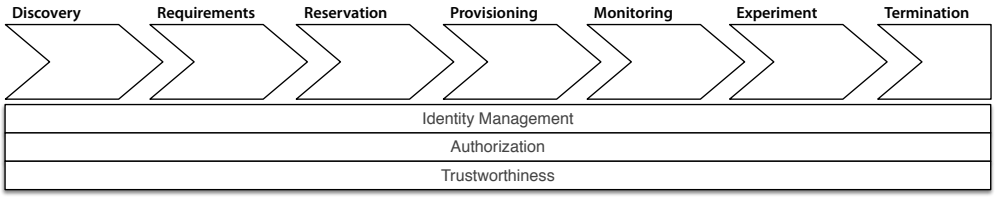

Figure 1: The experiment life cycle (based on [3])

ment Framework[12] (OMF). Experimenters can then use either OMF Experiment Controllers (ECs) or the Network Experimentation Programming Interface (NEPI).

In order to push measurements from the experiments the TCP based OMSP is available. Experimenters can integrate a library to their system under study to send monitored data to a server. Such a server is usually operated at the users premises and common implementations include the ORBIT Measurement Library[13] (OML)[12] and its semantic counterpart Semantic OML[13] (SOML).

For the third area, X.509 certificate based Public Key Infrastructures (PKIs) are usually being used to implement federated authentication and trust. While current authorization mechanisms use signed SFA user and slice credentials, more sophisticated Attributed Based Access Control[14] (ABAC)[14] is under development. The used approaches for SLA management are depending on the respective federation context and chosen implementation.

Given the above sketched situation with the different involved APIs, implementations and technologies, support for the whole experiment life cycle from a testbed provider and tool developer point of view raises a number of challenges. One that stands out in particular is the handover between the first and second area, while taking the last one into consideration.

3. FITEAGLE FRAMEWORK

To address some of the above mentioned challenges, our approach is focusing on three aspects: (i) increasing the ease of use by reducing the number of involved technologies, (ii) enabling reusability by abstracting from specific APIs, and (iii) focusing on managing semantic graphs rather than specific resources. While the proposed framework doesn't intend to offer an exhaustive solution to all federated experimentation related issues, it does offer an extensible foundation to further implement and study semantic resource management mechanisms (including their handovers).

3.1 Architecture

The underlying idea to achieve the outlined goals, is to manage the experiment life cycle on a formal information level. For this, it is important to highlight the difference of Object Models (OMs), Serializations, Data Models (DMs), Information Models (IMs) and Semantic Models (SMs) (cf. Figure 2). The documented related work, besides the Open Resource Control Architecture[15] (ORCA), is based on functional

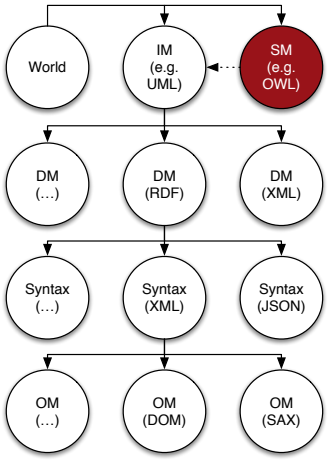

Figure 2: Levels of abstraction (based on [16, 17])

code that is mapping OMs to corresponding tree-based DMs that are usually serialized as XML or JSON. The needed logic to relate specific resources to each other, to validate descriptions and requests, or to discover available information based on given requirements is encoded in the according implementation. One possible approach to abstract this logic to a declarative level is to adopt mechanisms developed with the Semantic Web[10]. Namely, the graph-based Resource Description Framework (RDF), the semantic Resource Description Framework Schema (RDFS) and Web Ontology Language (OWL), and the SPARQL Protocol And RDF Query Language (SPARQL). In our approach, we describe, discover, interconnect, reserve, provision, control and monitor resources based on these descriptive languages. This allows us to work internally only on a semantic graph and then to adapt to specific APIs and resources.

The underlying architecture is distinctly described in [6]. In principle, required functionalities are clustered in four areas. While the management of the resource life cycle is handled in an abstracted graph-based in the *Core Modules*, the concrete user APIs are implemented as *Delivery Mechanisms*, the integration of specific resources in *Resource Adapters*, and the export and import of other information in the *Service Integration* context. The different modules are decoupled from each other by allowing internal communication only through the exchange of semantic information models over a message bus. This architecture allows reusability by the separation of concerns[18] and combines established design patterns such as Data, Context and Interaction[19] (DCI) and

[12] https://oml.mytestbed.net
[13] https://github.com/alco90/soml
[14] http://abac.deterlab.net

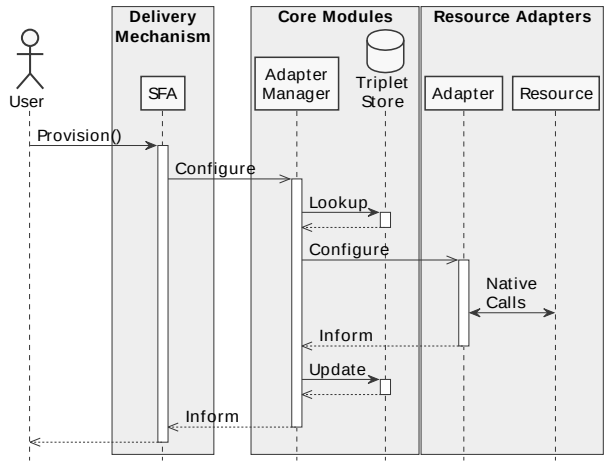

Figure 3: A Provision() method call sequence

modern approaches such as Microservices[9], with Semantic Web[10] based technologies.

3.2 Implementation

The concrete instantiation of this architecture and proof-of-concept of the underlying concept is subject of this paper. In Figure 3 a simplified sequence diagram is shown that illustrates the relationship of some of the afore mentioned components. While each Microservice could have been implemented using different programming languages, it was chosen to use Java 8 for the sake of consistency. The overall framework is a Message Oriented Middleware (MOM) and divided into several Apache Maven 3.1[15] modules, that are hosted at GitHub[16], and tested after each commit using JUnit 4.12 and the Travis Continous Integration (CI) environment. The modules are deployed within the Java Platform Enterprise Edition (J2EE) application server Wildfly 8.2[17], the OWL based information model is serialized using turtle (see [7] for details), transported using the Java Message Service (JMS) provider HornetQ 2.4, parsed by Apache Jena 2.12[18], persisted with Sesame 2.8[19] and converted from/into GENI Resource Specifications (RSpecs) using the latest OMN library.

Within this environment a number of reusable Microservices are deployed. Each module can offer its service by a number of APIs, while the JMS interface is compulsory and required default functionalities are implemented in abstract classes. If the service needs external components, this should be abstracted as well to allow emulate their functionalities for testing purposes. The allowed message types are depicted in Table 1 and the message body is described using FIDDLE and OMN, respectively.

By default, a native REST, a GENI SFA AM and a ProtoGENI SFA SA *Delivery Mechanisms* are included. Besides others, further mechanisms that have either been initially implemented or are envisioned include an FRCP interface for resource control (AMQP support is already enabled), an OMSP interface to receive monitoring information about resources (as shown in [20]), an IEEE P2302 compliant interface for Intercloud federations (as shown in [21]), a Topology and Orchestration Specification for Cloud Applications[22] (TOSCA) interface (the needed translation has already been implemented in the OMN context), or a JSON with Padding (JSONP) interface to be used with LodLive[23][20].

Further, a number of Microservices are included as part of the *Core Modules*. A user management for authorization processes, a resource adapter manager for handling available resources, a reservation module to manage allocation requests and an orchestrator for resource provisioning and configuration requests. Also, as a *Service Integration* demonstration, a message bus logger service is being deployed.

Finally, in order to handle resources, two *Resource Adapters* are deployed automatically. The first one for testing purposes represents a virtual *Garage* that instantiates and manages a number of virtual *Motors* with properties such as Revolutions per Minute (RPM). The second one for demonstration purposes allows to create public-key based SSH logins on a compute node. Further, more adapters are available to be installed for resources such as an OpenStack or TOSCA compliant servers.

4. EVALUATION

In order to demonstrate the applicability of the proposed work, three approaches have been followed. First, the complexity of the framework setup has been streamlined; second, the standard conformity against GENI SFA AM and SA APIs has been verified; and third, the performance of the given implementation has been examined.

4.1 Bootstrapping

To bootstrap the environment and to setup an SFA compliant testbed under four minutes, only a single command has to be issued on a Linux/Unix compliant machine. As shown in Listing 1 the initial command first verifies the environment for software requirements and then downloads, configures, compiles, installs and starts all needed software components. After the installation is completed, a J2EE environment should run on the designated machine and accept SFA method calls.

Listing 1: FITeagle bootstrapping

```
$ time (curl -fsSL fiteagle.org/bootstrap | bash -s init
     deployFT2 deployFT2sfa)
Checking environment...
 * Checking for 'java'...OK
 * Checking for 'javac'...OK
 * Checking for 'mvn'...OK
...
Getting FITeagle bootstrap sources...OK
Downloading container...
Installing container...
...
real 2m46.679s
user 3m56.167s
sys 0m27.237s
```

[15] http://maven.apache.org
[16] https://github.com/fiteagle
[17] http://wildfly.org
[18] http://jena.apache.org
[19] http://rdf4j.org
[20] http://en.lodlive.it

Table 1: FITeagle messages in comparison

SFA	FITeagle	REST	FRCP	SPARQL	OMN	Description
GetVersion	Get	GET	Request	Describe	*omn:Infrastructure*	Describe testbed
ListResources	Get	GET	Request	Describe	*omn:parentOf*	List resources
Register	Create	PUT	Create	Insert	*omn:Topology*	Create a slice
Allocate	Create	PUT	Create	Insert	*omn:Reservervation*	Reserve an instance
Describe	Get	GET	Request	Describe	*Graph*	Describe a slice
Renew	Configure	POST	Configure	Del/Ins	*omn:Reservervation*	Extend a reservation
Provision	Configure	POST	Configure	Del/Ins	*owl:NamedIndividual*	Create instance
Status	Get	GET	Request	Describe	*omn:Status*	Get instance status
Perf.Op.Act.	Configure	POST	Configure	Del/Ins	*omn:Attribute*	Change instance
Delete	Release	DELETE	Release	Delete	*Graph*	Delete instance
-	Inform	-	Inform	-	-	Push update

4.2 Functional Evaluation

The installation of the SFA module contains a script to automatically test its conformity by executing the jFed low level test *TestAggregateManager3*. As shown in Listing 2 the script downloads the required binaries and runs 20 test against the installation on *localhost*.

Listing 2: jFed Test Results

```
$ ./sfa/src/test/bin/runJfed.sh
...
   Tested Authority:localhost
      URN (connect):urn:publicid:IDN+localhost+authority+
           root
      URN (rspec):urn:publicid:IDN+localhost+authority+root
      Hrn:localhost
...
Running testGetVersionXmlRpcCorrectness... SUCCESS
Running testGetVersionResultCorrectness... SUCCESS
Running testGetVersionResultApiVersionsCorrectness...
     SUCCESS
Running testGetVersionResultNoDuplicates... SUCCESS
Running testListAvailableResources... SUCCESS
Running testStatusBadSlice... SUCCESS
Running testListResourcesBadCredential... SUCCESS
Running createTestSlices... SUCCESS
Running testStatusNoSliverSlice... SUCCESS
Running testDescribeNoSliverSlice... SUCCESS
Running testAllocate... SUCCESS
Allocated sliver for "urn:publicid:IDN+localhost+
     slice+soVS1424678247"
Running testProvision... SUCCESS
Provisioned sliver for "urn:publicid:IDN+localhost+
     slice+soVS1424678247" manifestRspec=...
Running testSliverBecomesProvisioned... SUCCESS
testSliverBecomesReady -> sliver is now fully provisioned:
      operational_state=geni_ready
Running testPerformOperationalAction... WARN
Running testStatusExistingSliver... SUCCESS
Running testDescribeProvisionedSliver... SUCCESS
Running testSliverBecomesStarted... SUCCESS
testSliverBecomesReady -> sliver ready: geni_ready
Running testDescribeReadySliver... SUCCESS
Running testNodeLogin... SUCCESS
Running testRenewSliver... SUCCESS
Running testDeleteSliver... SUCCESS
DeleteSliver for slice urn:publicid:IDN+localhost+slice+
     soVS1424678247
```

4.3 Performance Evaluation

While the influence of the chosen serialization has already been analyzed in [6], in Figure 4 the overall responsiveness of the implementation is being visualized. Depicted are end to end response times of the SFA AM method calls GetVersion(), ListResources() and Allocate(), as well as SFA SA method calls GetCredentials() and Register() using the jFed[21] library. In order to obtain significant results, the measurements were repeated 500 times with 100 millisecond breaks in between and 5 repetitions were executed before (warmup) to filter out possible start up, initialization and compilation outliers.

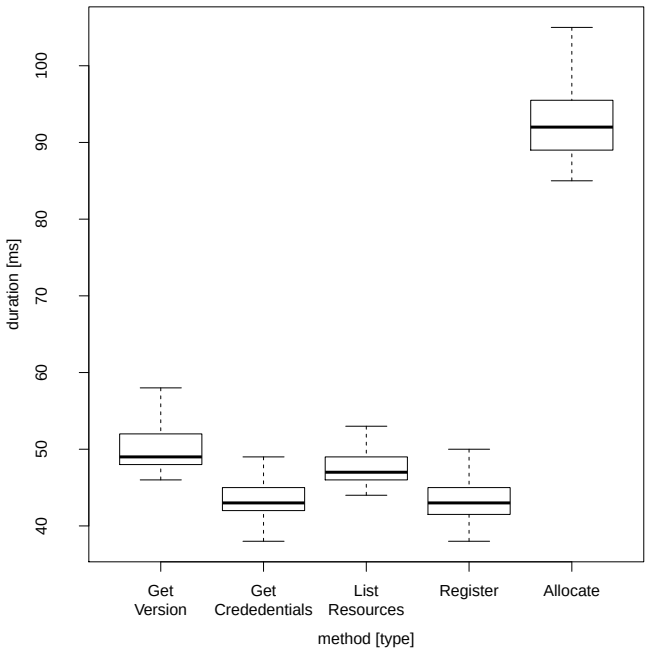

Figure 4: Performance of selected SFA method calls

In brief, it can be emphasized that the repetitive execution does not influence the overall stability, the duration of non-idempotent operations do not differ significantly, and the tested idempotent operation is about twice as slow.

5. CONCLUSION AND FUTURE WORK

We have presented the concept of a semantic-based framework to manage testbed resources within FIRE and GENI and to some extend also other federations. The crucial result is a proof-of-concept and reference implementation that demonstrates the management of resources on a semantic

[21] http://jfed.iminds.be

information layer using potentially multiple interfaces in a federated environment.

This work can be used by infrastructure owners to join multiple federations, describe their resources and services in a semantic manner, and to publish this information using different APIs. Users have means to build complex queries, can precisely define their requirements and reuse the same information model for different phases of the experiment life cycle. Developers can extend the framework to support further APIs, services, resources or core functionalities, while extracting the meaning out of the information models without functional code.

Further work includes the enhancement of the SFA compliancy and the extension of the API to exploit the advantages gained by the underlying semantic model in short term (e.g. for resource discovery). To maintain the required ontology the W3C Community Group Federated Infrastructures[22] has been established and will continue to add required concepts and properties. The midterm goal is to add support for FRCP based resource control and OMSP based export of monitoring information. The long-term objective is to extend the framework to cover further fields of applications such as Interclouds.

Acknowledgments

Research for this paper was partially financed by the European Union's FP7 grant agreement no. 318389 Fed4FIRE Project. We thank our project partners for their contributions and their collaboration on this research work.

References

[1] L. Peterson et al. *Slice-based Federation architecture*. Tech. rep. GENI, 2009.

[2] T. Rakotoarivelo, G. Jourjon, and M. Ott. "Designing and Orchestrating Reproducible Experiments on Federated Networking Testbeds". In: *Computer Networks, Special Issue on Future Internet Testbeds* (2014).

[3] W. Vandenberghe et al. "Architecture for the Heterogeneous Federation of Future Internet Experimentation Facilities". In: *Future Network and Mobile Summit (FNMS)*. Lisboa, Portugal, 2013, pp. 1–11.

[4] D. Bernstein and V. Deepak. *Draft Standard for Intercloud Interoperability and Federation (SIIF)*. Tech. rep. IEEE P2303, 2014.

[5] N. Grozev and R. Buyya. "Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey". In: *Software: Practice and Experience* 44.3 (2012), pp. 369–390.

[6] A. Willner and T. Magedanz. "FIRMA: A Future Internet Resource Management Architecture". In: *Workshop on Federated Future Internet and Distributed Cloud Testbeds (FIDC)*. Karlskrona, Sweden: IEEE Xplore Digital Library, Sept. 2014.

[7] A. Willner, R. Loughnane, and T. Magedanz. "FIDDLE: The Federated Infrastructure Description and Discovery Language". In: *4th Int. WS on Cloud Computing; Interclouds, Federations, and Interoperability*. Tempe, Arizona: IEEE, 2015.

[8] A. Willner et al. "The Open-Multinet Upper Ontology - Towards the Semantic-based Management of Federated Infrastructures". In: *10th Int. Conf. on Testbeds and Research Infrastr. for the Dev. of Netw. & Comm. (to appear)*. Vancouver, Canada: IEEE, 2015.

[9] S. Newman. *Building Microservices*. O'Reilly Media, 2015, pp. 1–250.

[10] T. Berners-Lee, J. Hendler, and O. Lassila. "The Semantic Web". In: *Scientific American* 284.5 (May 2001), pp. 34–43.

[11] T. Magedanz and S. Wahle. "Control framework design for Future Internet testbeds". In: *e&i Elektrotechnik und Informationstechnik* (2009).

[12] T. Rakotoarivelo et al. "OMF: a control and management framework for networking testbeds". In: *Operating systems review* (2009).

[13] M. Singh et al. "ORBIT Measurements framework and library (OML): motivations, implementation and features". In: *1st Int. Conf. on Testbeds and Research Infrastr. for the Dev. of Netw. & Communities*. 2005, pp. 146–152.

[14] N. Li, J. C. Mitchell, and W. H. Winsborough. "Design of a role-based trust-management framework". In: *Symposium on Security and Privacy*. IEEE, 2002, pp. 114–130.

[15] J. Chase. *ORCA control framework architecture and internals*. Tech. rep. Duke University, 2009.

[16] A. Pras and J. Schoenwaelder. *On the Difference between Information Models and Data Models*. RFC 3444 (Informational). 2003.

[17] A. Pras et al. "Key research challenges in network management". In: *Communications Magazine, IEEE* 45.10 (2007), pp. 104–110.

[18] R. C. Martin. *Agile Software Development: Principles, Patterns, and Practices*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2003.

[19] J. O. Coplien and G. Bjørnvig. *Lean Architecture: for Agile Software Development*. Wiley, 2010, p. 376.

[20] Y. Al-Hazmi et al. "An Automated Health Monitoring Solution for Future Internet Infrastructure Marketplaces". In: *26th International Teletraffic Congress - Workshop on Future Internet and Distributed Clouds (FIDC)*. Karlskrona, Sweden: IEEE Xplore Digital Library, Sept. 2014.

[21] A. Willner et al. "Towards an Ontology-based Intercloud Resource Catalog - The IEEE P2302 Intercloud Approach for a Semantic Resource Exchange". In: *4th Int. WS on Cloud Computing; Interclouds, Federations, and Interoperability*. Tempe, Arizona: IEEE, 2015.

[22] D. Palma and T. Spatzier. *Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1*. Nov. 2013.

[23] D. V. Camarda, S. Mazzini, and A. Antonuccio. "LodLive, exploring the web of data". In: *Proceedings of the 8th International Conference on Semantic Systems - I-SEMANTICS '12*. New York, New York, USA: ACM Press, Sept. 2012, p. 197.

[22] https://www.w3.org/community/omn/