Ontologies framework for semantic driven code generation over testbed premises

Valentina Nejković*, Milorad Tošić*, Filip Jelenković*
* Faculty of Electroninc Engineering, University of Nis/Intelligent Information Systems Laboratory, Nis, Serbia
{valentina,milorad.tosic,filip.jelenkovic}@elfak.ni.ac.rs

***Abstract*— Past decade revealed testbed based evaluation approaches in networking research as highly important and primarily due to the emphasis on practicality. At the beginning, testbeds were built for a specific project or studying a specific technology. In the context of New Internet, networking and computational testbeds appear of crucial importance for conducting testing of computational tools and new technologies, describing experiments, new platforms and environments. This paper addresses networking testbed enhancements in terms of orchestration, control and virtualization capabilities. It reports about real world evaluation of federated testbed infrastructure in terms of development of ontology-based automatic code generator for experiments. The main idea is to enable experimenters to define semantic description of the experiment through a high-level specification and generate software code that is directly deployable on the testbed federation. As a prerequisite for semantic driven automatic code generation, the ontology framework that gives semantic description of particular experiment domain is developed and its design is discussed in the paper.**

Keywords: ontology, semantics, automatic code generation, 5G, networking testbed, federated testbeds

I. INTRODUCTION

A consequence of the increasing complexity of federated testbeds for 5G applications is the demand for reducing complexity and time of designing and running experiments. On the other side, emerging networking technologies, such as software defined networks (SDN) and network function virtualization (NFV), together with the diversity of radio access technologies, such as LTE, Bluetooth, and Wi-Fi, and the growing trends requiring their simultaneous use, significantly increase learning curve for wireless networking experiments. A promising approach to the problem is automatic generation of target-specific code directly from a high-level experiment description [1,2,3].

Different approaches to testbed federation management are proposed by researchers including Future Internet Research and Experimentation (FIRE) initiative [4], Slice Based Federation Architecture (SFA), FITeagle, Open Baton, jFed Experimenter Client, OMF 6 – Federated Resource Control Protocol, and OML [4,5,6,7]. Resources from different testbeds in the federation, needed for an experiment, can be accessed by the experimenter through a single access-point named FITeagle framework in the FIRE environment. User is able to discover, reserve, acquire and monitor selected resources over testbeds in a federation after authentication. Interoperability with other technologies is provided by FITeagle that implements the standard SFA interface. After allocated resources are successfully provisioned, FITeagle returns a manifest containing names and other information about resources. Using these names, experimenter can build OEDL script and/or shell script, which is then used to execute the experiment. Even though the script languages are designed with easy-to-use in mind, it is a domain specific language with a steep learning curve. Our research question is: "*How to help experimenters to more easily develop experiment code for federated testbed premises*?"

With a goal to scale the experimentation process, both in time and in number of experimenters, we propose the use of semantic based automatic code generation. To the best of our knowledge there is no previous work focusing on semantic-based automatic experiment code generation. Writing domain specific code is a knowledge intensive process that requires programming as well as domain knowledge. The required knowledge for automatic code generation is heterogeneous, including understanding of the radio-related features, knowledge about software and hardware modules available in the used testbeds, and practical knowledge of the domain specific language. We envisage that the process starts with the semantic description of the experiment components using ontology driven user interface. Based on the inputs and the semantic based ontologies framework, the experiment source code could be generated.

In this paper, ontologies are proposed to represent the knowledge in a formal manner and access it in the process of code generation as needed. Framework that is proposed is focused on experiments to be conducted on the testbed and considers network-sensing domain. It should be noted that the approach is general in a sense that it could be used for different domains and experiments conducting over any complex heterogeneous system or federation of such systems.

Section II of this paper gives background on networking testbeds and related work in ontologies usage within testbeds. Section III gives overview of automatic code generation process and set the position of OFSecGENE in such process. The main part of this paper is Section IV where we introduce and discuss the OFSecGENE. The last section is Section V, which concludes paper.

II. BACKGROUND

A. Networking Testbeds

Last decade investments in the experimentation leading towards next generation of wireless systems indicate the importance of networking testbeds [1, 2, 3]. Networking testbeds represent the important tools for testing and validating new developed protocols that address problems of the current and the future Internet (FI) architectures. The proposed testbeds are relying on the development of cloud and BigData computing infrastructures and are expected to integrate with them. The result of such integrations is high level of complexity which leads to a need for different and multidisciplinary research areas in order to improve usability and simplify experimentation. Some of these research areas drive to the creation of federations of high-performance testbed and experimentation facilities across the globe, known as Future Internet and Distributed Cloud (FIDC) testbeds [1].

Testbeds are usually used and built for studying a specific technology or for a project purposes. They provide different environmental settings and usually have heterogeneous nature. Testbeds can be federated, where each testbed maintains its own administration scope, architecture and functionality but delegates its resources into a common shared pool. There are a number networking testbeds, such as: 5GIC [8], FUSECO Playground from FOKUS Fraunhofer/TUB [9], Orbit [7], Nitos [10], etc..

New methods, algorithms, computational models, federation management, coordination protocols etc. are needed due to the novelty of the testbeds federation approach [2]. Under EU FP7 and HORIZON 2020 funding schemes several projects have just been finished or are still running in this area. For example, testbed infrastructure that supports the orchestration of resources according to the needs of different applications are investigating within SoftFIRE project [4]. In particular, this project considers federated testbeds infrastructure comprising experimental networks and programmable resources designed according to NFV/SDN principles. FLEX (FIRE LTE testbeds for open experimentation) [20] aims at the improvement of FIRE's infrastructure such as cellular access technologies and LTE. FLEX testbed was built on the tools for experiment control and monitoring with basis of current FIRE testbed management.

B. Ontologies and Networking Testbeds

Future federations of heterogeneous networks envision common features such as coordination of available resources and intelligent retrieval of available computing and networking resources. By adopting ontologies as a knowledge background for information model of resources and services, advanced manipulations such as deduction of service and infrastructure behaviors become possible. Availability of resources as well as services provisioning can be intelligently deduced if ontologies are used [18]. Examples that prove high potential of using ontologies in networking can be found within several EU FP7 and Horizon2020 founded projects such as: NOVI [18], FLEX [20], Fed4FIRE [12], TRESCIMO [13], INFINITY [11], SoftFIRE [14] etc. Existing scientific and technical achievements, solutions and theories we will rely upon in addressing the problem are in compliance with these projects.

In NOVI [19], OWL ontologies are used to formalize information model and to develop the corresponding data model for communication among system components. The information model describes resources at a conceptual level, including all components required to support operation of the system software. On the other side, the data model describes implementation details based on representation of concepts and their relations provided by the information model. FLEX [20] project, with its CoordSS subproject [15], used ontologies with the basic assumption that semantic technologies could be used to improve coordination in cognitive radio networks. In particular, FLEX directly works with the 5G heterogeneous networking challenge concerning coordination in heterogeneous networks. The spectrum sensing and coordination in such networks is represented as an interactive process consisting of communication between distributed agents and information sharing about a specific spectrum usage effectiveness [16]. Semantic technologies are used to represent conceptual agreement on vocabulary among agents in the network. The knowledge is represented in a form of ontologies, where the standardized way for this representation is used [16],[17]. Fed4FIRE, TRESCIMO, INFINITY and SoftFIRE used semantic based approaches and mechanisms with semantically annotated graphs, which allows automatic reasoning, linking, querying and validation of heterogeneous data. These projects considered individual testbeds as well as federated testbeds environments and used approach of semantic-based management of federated infrastructures. Fed4FIRE project, [17] aims to define a homogeneous way of describing heterogeneous resources within a federated testbed. Starting under the umbrella of Open-Multinet (OMN) [18], they have been working towards a set of upper ontologies to describe federated infrastructures and their underlying resources. These ontologies support a number of use cases to semantically manage the whole life-cycle of a resource: discovery, selection, reservation, provisioning, monitoring, control, termination, authentication, authorization, and trustworthiness.

Ontologies were considered for adoption to different problems within networking testbed community [19,20,21,22]. The main focus so far was on semantic resource description [23] and provisioning. Also, ontologies are successfully adopted to IoT infrastructure [24].

III. EXPERIMENTATION PROCESS WITH AUTOMATIC CODE GENERATION

We envisage that testbed experimentation framework should enable experimentator to construct experiment flow and to generate code that can execute it on the targeted testbed. We identified following processes:

- ***Experiment topology semantic description.*** Design of the experiment starts by defining experiment

topology. Experimentator imports semantic description of the experiment topology into the existing domain knowledge residing in the Experiment Flow Editor (EFE).

- ***Experiment flow creation.*** EFE provides experimentator with the intuitive graphical user interface which navigates him through the process of creating experiment flow. EFE interact with semantic knowledge of the networking domain. We envisage that EFE should contains experiment components that user can use to construct experiment by performing simple drag/drop and connect actions on these components. Each component should describe experiment flow task on the high level of conceptualization, such as: specific metrics (for example throughput), transfer (selection of one of the existing topology nodes and one interface that exists on the selected node), start/end of experiment etc.
- ***Automatic code generation.*** For each flow component automatic code generator should automatically generates code for the targeted testbed execution environment. Automatic code generator should generate commands that follow experiment flow. The code generator generates OEDL experiment code and Shell Script code. In the case of shell script execution environment (such are VNF capable testbeds) output should be provided in the form of the shell scripts files for each of the network nodes. Scripts can be included in the corresponding VNFD packages, in which case they will be executed when VNFD is deployed. For OMF capable testbeds file with OEDL code should be generated.
- ***Experiment execution***. Experiment should be executed by running generated script file.

Described steps are shown on Figure 1. Also, Figure 1 positioning OFSecGENE.

IV. Ontologies Framework For Semantic Driven Code Generation

The OFSecGENE includes a set of the domain and system ontologies for semantic descriptions of the network sensing and experiment execution flow.

OMN. We envisage that everything in experimenting environment is considered as a accessible resource. We adopted the OMN Resource ontology with a goal of knowledge reuse and interoperability. The OMN provides the concepts and methods to describe resources present in Future Internet platforms [18][25].

The OMN defines a formal information model for federated infrastructure management. Corresponding data models are developed in order to enable the communication among the various components in software architecture. This ontology also holds information on how resources are connected together in a federated infrastructure. Further, we identified OMN Monitoring ontologies as useful for the scenario of automatic code generation in wireless networking domain area. Those set of ontologies covers various monitoring services provided for federation administrators, experimenters, and testbeds federation services. The most relevant are OMN Monitoring Data Ontology and OMN Monitoring Tool Ontology.

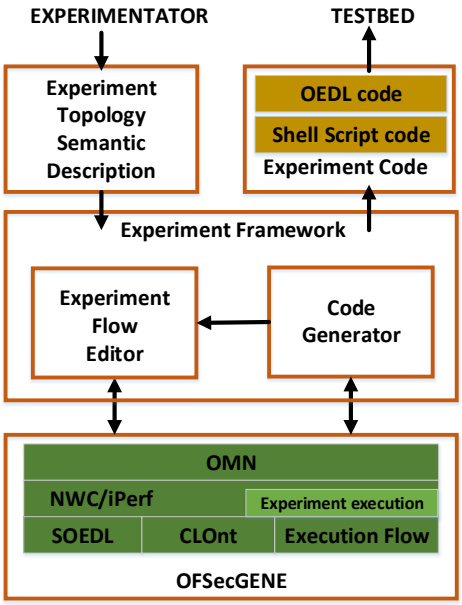

Figure 1 Processes of experimenting using automatic code generator

CoordSS ontologies. The CoordSS ontologies framework is evaluated and used for experimenting over testbeds federation and coordination in heterogeneous communication networks [16]. The CoordSS framework represents a base for coordination protocols based on semantic technologies and ontologies. The CoordSS framework comprises of ontologies: command line ontology (CLOnt), wireless ontology (WDSO), spectrum sensing capabilities ontology (SSCO), Meta System Architecture ontology (MSAO), Resource Description Framework ontology (RDFO). They specify semantic descriptions of radio spectrum, coordination, frequency selection, dynamic spectrum access, command line, wireless, and spectrum sensing capabilities of supporting software.

For automatic code generator CLOnt, MSAO and RDFO are needed. RDFO represents a very general ontology, and can be used to describe any concept. MSAO represents ontology for describing in computational systems. MSAO gives descriptions of entity, element and attribute concepts. RDFO is the most generic ontology in the framework, and describes value and property concepts.

Network Sensing Ontology. The fundamental concepts for network sensing domain are specified in the Network Sensing Ontology (NS). Network sensing measurements of the bandwidth in IP networks are practically considered in the context of one possible implementation represented by the software tool iPerf [26]. IPerf has command line arguments, receives data from nodes and writes results to the database. The NS ontology semantically describes iPerf software module that implements throughput

measuring. Perf reports the bandwidth in test process. NS uses CLOnt for semantic description of concepts used in command line, because iPerf receives arguments via command line.

Client ontology. In the experiment environment client application is responsible for communication among server and testbed nodes. Experiment client represents a console application with following arguments: URI of the agent, URI of the channel, URL of the server, subscribe, publish, register and read messages, corresponding operations. Client application is based on experiment Client Ontology (CO), which contains statements describing client application with its parameters and command line arguments. Client parameters are:

a) *server* represents the server that is responsible for sending and receiving client messages.
b) *agent* is the user of the client. It can be human user or another software component.
c) *channel* represents communication channel on which agents sends and receives messages.
d) *message* contains data that are send from one agent to another via given channel.

Experiment Flow Ontology. Flow of the experiment consists of the sequence of steps. Experiment instructions are represented by the ExecutionStep class, and the sequence of steps is created by using the TransferStep class and its sub classes. TransferStep represents the concept that is used as an abstraction of sequence. Properties passesTo and receviesFrom are used to annotate which ExecutionStep follows and which one precedes. We differentiate four types of steps, with the corresponding definition:

a) *execStart* represents the first step in experiment execution, e.g. start of the experiment.
b) *execEnd* represents final step in experiment execution.
c) *execNext* executes one ExecutionStep that is connected via passesTo property.
d) *execAll* executes all ExecutionSteps that are connected vai passesTo property. All ExecutionsSteps are executed in parallel.

SOEDL. Semantic OEDL Ontology **(**SOEDL) semantically describes basic building blocks of the OEDL language. Since we use iPerf software module for network sensing we first created definition of that module in SOEDL. Attributes for this application are set according to OMF requirements. We have connected experimenter knowledge with experiment that can be executed in OMF based testbed.

The next step is describing experiment in SOEDL, which should be automatically done based on users given descriptions of experiment topology and execution flow over Ontology driven user interface. After experiment network expressed over SOEDL ontology, this semantic description of experiment could be transformed to the OMF executable code. SOEDL ontology contains all OEDL concepts as described in [27] and by populating them with concrete instances and values semantic description for experiment can be created. After that we create experiment code that can be executed on any OMF capable testbed. Example is given on figure 2 and figure 3. Figure 2 gives an example experiment of network with two WiFis as seen in SOEDL ontology, while figure 3 gives corresponding OEDL code.

V. CONCLUSION

In this paper, we propose an ontology system that deals with heterogeneity of testbeds and technologies they cover and automatic experimental code generation. These two issues are important for experimentation with networking systems. The main impact of this paper presented ontologies framework is to facilitate experimentation based on flexibility of semantic automatic code generation. The presented framework presents the prerequisite for automatic code generation of testbed experiments. The OMF framework with OEDL language and different shell scripts could be used as the target platform for practical implementation of the automatic code generator.

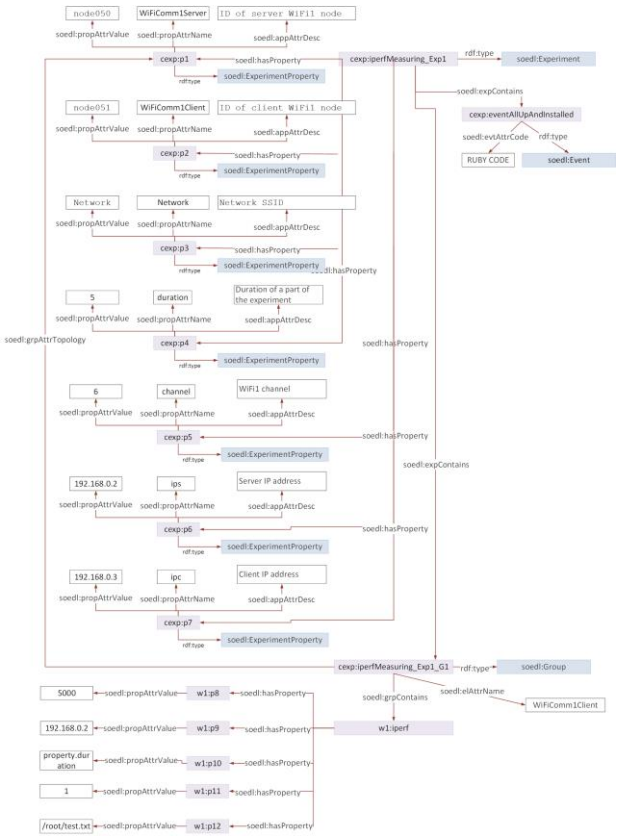

Figure 2. Semantic description of the iPerf-Sensing experiment

```
#NODES
defProperty('WiFiCommlServer', 'node050','ID of server WiFi1 node')
defProperty('WiFiCommlClient', 'node051','ID of client WiFi1 node')

#NETWORKS
defProperty('Network', 'Network','Network ESSID')

#MISC
defProperty('duration', 5, "Duration of a part of the experiment")
defProperty('channel', 6, "WiFi1 channel")
defProperty('ips', "192.168.0.2", "Server IP address")
defProperty('ipc', "192.168.0.3", "Client IP address")

#iperf application wrapper
defApplication('iperf') do |app|
  app.description = %{iperf is a traffic generator and bandwidth measurement
tool. It provides generators producing various forms of packet streams and port
for sending these packets via various transports, such as TCP and UDP.
  }
  app.path = "/usr/bin/iperf"

  app.defProperty('port', 'set server port to listen on/connect to to n (default 5001)', "-p",  :type => :integer)
  app.defProperty('server', 'run in server mode', "-s",  :type => :boolean)
  app.defProperty('client', 'run in client mode, connecting to <host>', "-c",  :type => :string,  :order => 1)
  app.defProperty('time', "time in seconds to transmit for (default 10 secs)", "-t",  :type => :integer, :unit => "second")
  app.defProperty('interval', 'seconds between periodic bandwidth reports', "-i",  :type => :double, :unit => "second", :default => "1.")
  app.defProperty('outputfn', 'Redirect output to console', ">>",  :type => :string)
end

#WiFi nodes
defGroup('WiFiCommlServer', 'omf.servername.'+property.WiFiCommlServer) do |node|
  node.addApplication("iperf", :id => "iperf") do |app|
    app.setProperty('port', 5000)
    app.setProperty('server', true)
  end
  node.net.w0.mode = "adhoc"
  node.net.w0.type = 'g'
  node.net.w0.channel = property.channel
  node.net.w0.essid = property.Network
  node.net.w0.ip = property.ips
end

defGroup('WiFiCommlClient', 'omf.servername.'+property.WiFiCommlClient) do |node|
  node.addApplication("iperf", :id => "iperf") do |app|
    app.setProperty('port', 5000)
    app.setProperty('client', '192.168.0.2')
    app.setProperty('time', property.duration)
    app.setProperty('interval', 1)
    app.setProperty('outputfn', '/root/test.txt')
  end
  node.net.w0.mode = "adhoc"
  node.net.w0.type = 'g'
  node.net.w0.channel = property.channel
  node.net.w0.essid = property.Network
  node.net.w0.ip = property.ipc
end

onEvent(:ALL_UP_AND_INSTALLED) do |event|
  info "This is WiFi iperf experiment"
  wait 30
  allGroups.startApplications
  wait property.duration+10
  allGroups.stopApplications
  info "All my Applications are stopped now."
  Experiment.done
end
```

Figure 3. OEDL code for the iPerf-Sensing experiment

REFERENCES

[1] Berman, M., Demeester, P., Lee, J.W., Nagaraja, K., Zink, M., Colle, D., Krishnappa, D.K., Raychaudhuri, D., Schulzrinne, H., Seskar, I. and Sharma, S., "Future internets escape the simulator." Communications of the ACM, vol. 58, no. 6, pp: 78-89, 2015.

[2] http://ict-forge.eu

[3] "Fourth GENI & FIRE Collaboration Workshop", Washington DC, September 17-18, 2015, Online Available:http://groups.geni.net/geni/wiki/GENIFireCollaborationWorkshopSeptember2015

[4] https://www.ict-fire.eu/

[5] Vandenberghe, W.; Vermeulen, B.; Demeester, P.; Willner, A.; Papavassiliou, S.; Gavras, A.; Sioutis, M.;Quereilhac, A.; Al-Hazmi, Y.; Schreiner, F.; et al. Architecture for the heterogeneous federation of future internet experimentation facilities. In Proceedings of the Future Network and Mobile Summit (FutureNetworkSummit), Lisbon, Portugal, 3–5 July 2013; pp. 1–11.

[6] Willner, Alexander, and Daniel Nehls. "Semantic-based management of federated infrastructures for future internet experimentation." (2016), Available: https://www.netsys2015.com/wp-content/uploads/NetSys2015_PhD-Forum_Willner.pdf.

[7] OrbitLab, http://www.orbit-lab.org; [accessed 15.01.17].

[8] 5G Innovation Centre, http://www.surrey.ac.uk/5gic; [accessed 15.01.17].

[9] FUSECO Playground, https://www.fokus.fraunhofer.de/go/de/fokus_testbeds/fuseco_playground; [accessed 15.01.17].

[10] NITlab, NITOS, http://nitos.inf.uth.gr/; [accessed 15.01.17].

[11] www.fi-infinity.eu/

[12] https://www.fed4fire.eu/,

[13] https://trescimo.eu/,

[14] https://www.softfire.eu/,

[15] http://infosys1.elfak.ni.ac.rs/coordss/

[16] M. Tošić, Z. Nikolić, V. Nejković, B. Dimitrijević, N. Milošević, Spectrum Sensing Coordination for FIRE LTE testbeds, Invited Paper, 2nd International Conference on Electrical, Electronic and Computing Engineering, IcETRAN 2015, Silver Lake, Serbia, June 8-11, 2015

[17] McGuinness, Deborah L., and Frank Van Harmelen. "OWL web ontology language overview." W3C recommendation 10.10 (2004): 2004.

[18] Wibisono Adianto; Cees de Laat a and Paola Grosso, Future Internet Ontologies:The NOVI Experience, www.semantic-web-journal.net/system/files/swj580.pdf

[19] https://www.novi.eu/,

[20] https://www.flex.eu/

[21] Al-Hazmi, Y.; Magedanz, T. Towards Semantic Monitoring Data Collection and Representation in Federated Infrastructures. In Proceedings of the 3rd International Conference on Future Internet of Things and Cloud (FiCloud), Rome, Italy, 24–26 August 2015; pp. 17–24.

[22] Fortuna, C., Tosic, M., Chwalisz, M., deValck, P., Moerman, I., & Seskar, I. (2015, May). Representation of spectrum sensing experimentation functionality for federated management and control. In Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on (pp. 1226-1229). IEEE.

[23] Giatili, M., C. Papagianni, and S. Papavassiliou. "Semantic aware resource mapping for future internet experimental facilities." Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2014 IEEE 19th International Workshop on. IEEE, 2014.

[24] Maglaris, Vasilis, et al. "Toward a holistic federated future internet experimentation environment: the experience of NOVI research and experimentation." IEEE Communications Magazine 53.7 (2015): 136-144.

[25] Lanza, Jorge, et al. "A Proof-of-Concept for Semantically Interoperable Federation of IoT Experimentation Facilities." Sensors 16.7 (2016): 1006.

[26] iPerf, https://iperf.fr/; [accessed 15.01.17].

[27] The OMF Experiment Description Language (OEDL), https://omf.mytestbed.net/projects/omf6/wiki/OEDLOMF6, [accessed 15.01.17].