A Smart Development Environment for Infrastructure as Code

Jesús Gorroñogoitia[1], Zoe Vasileiou[2], Emilio Imperiali[3], Indika Kumara[4], Dragan Radolović[5] and Georgios Meditskos[2]

[1]*ATOS, Spain*
[2]*Information Technologies Institute, Centre for Research and Technology Hellas, Greece*
[3]*Politecnico di Milano, Italy*
[4]*Jheronimus Academy of Data Science, Tilburg University, The Netherlands*
[5]*XLAB Research, Slovenia*

Abstract

Cloud computing is a mature paradigm that has evolved to accommodate ever-increasing complex applications such as in the AI and HPC domain. If applications are complex, infrastructure can be even more, spanning over hybrid architectures. As such, producing a less error-prone deployment while offering high performance requires application and infrastructure awareness, and also deep knowledge of the IaC languages. In this paper, we present the SODALITE IDE, a suite that assists the users in the authoring of application deployment topology and infrastructure models for IaC. With focus on quality and performance, the IDE enables the faster and simpler development of IaC by offering features such as in-sync multiple model viewpoints , smart context-aware content assistance and semantic validation, powered by a Knowledge Base.

Keywords

Infrastructure as Code, TOSCA, Ansible, IDE, Semantic, Ontology

1. Introduction

In recent years, the global market has seen a tremendous rise in utility computing serving as the backend for practically any new technology, methodology or advancement from healthcare to aerospace. SODALITE addresses complex tasks of configuration, deployment, and operation of complex applications. The development of these tasks implies knowledge of multiple IaC scripting languages and being able to manage the whole development process of IaC. Given those intricacies, the simplification and abstraction of those DevOps processes is uppermost. To this end, SODALITE provides tools for a simpler and faster development of IaC, deployment and execution of heterogeneous applications in HPC, Cloud & Software-Defined(SW) defined computing environments, with particular focus on quality, performance, and manageability. Following this vision, SODALITE offers smart modeling capabilities to help non-expert DevOps teams in defining *Abstract Application Deployment Models* (AADMs). The main novelty of

First workshop on trustworthy software and open source, March 23-25, 2021, Virtual Conference
jesus.gorronogoitia@atos.net (J. Gorroñogoitia); zvasilei@iti.gr (Z. Vasileiou); emilio.imperiali@mail.polimi.it (E. Imperiali); i.p.k.weerasinghadewage@tilburguniversity.edu (I. Kumara); dragan.radolovic@xlab.si (D. Radolović); gmeditsk@iti.gr (G. Meditskos)

SODALITE regarding the support to the definition of AADMs is its ability to simplify this task by offering semantic-based guidance to the end users.

The purpose of this paper is to describe the SODALITE Integrated Development Environment (SODALITE IDE) highlighting its ability to create trustworthy AADMs. The paper is structured as follows: Section 2 presents the state of the art in the creation of AADMs; Section 3 presents the design and implementation of the SODALITE IDE in detail; Section 4 provides a description of the preliminary evaluation of the IDE; Finally, Section 5 draws the conclusion and delineate the plan for future work.

2. Background and Related Work

2.1. TOSCA and IaC

TOSCA (The Topology and Orchestration Specification for Cloud Applications) [1] is an OASIS standard language that enables the interoperable description of applications and cloud infrastructure services, their relationships, and the operational behavior of those services. The TOSCA metamodel supports the definition and description of those services through a topology definition which consists of *Node Templates* representing components of the application such as databases, virtual machines or web servers. These templates are instances of the *Node Types* that define the schema of the infrastructural or application component such as which properties, and requirements can be used. Regarding the operational behavior, node types might define lifecycle operations to implement the behavior that an orchestration engine can invoke at runtime when instantiating a node. These operations are used to configure a database or start a server, and are backed by script implementation artifacts such as Chef or Ansible.

2.2. Related Work

Several modeling languages have been proposed for supporting the specification of complex application topologies and their deployment into infrastructures including Cloud [1] [2] [3] [4] and HPC [5] [6]. TOSCA is a fast emerging standard in the Cloud realm, but lacks visual notation, leading to the appearance of authoring tools with their own non-standardized visual notation. For addressing this limitation, researchers have designed their proposals to standardize the visual notation for TOSCA.

Winery [7] is a Web TOSCA visual editor, that can be also included, as a plugin. It separates modeling concerns to support not only resource experts on the specification of TOSCA types, but also application owners on the definition of their application topologies. CloudCAMP DSML [3], a web-based editor, supports the generation of IaC deployment models from users' abstract business-oriented requirements for the creation of application component topologies by utilizing TOSCA node templates and relationships. Alien4Cloud[1], a Web-based editor, enables application owners to design their deployment topology, as an orchestration of components instantiated from types retrieved from a common TOSCA types Catalogue.

The main novelty of the SODALITE IDE is to support the complete specification of both application deployment topologies, and of the resources the application requires on the target

[1] https://alien4cloud.github.io/

infrastructure as model instances of the SODALITE DSL. This DSL has been designed as an abstraction that leverages TOSCA to facilitate the export of AADM topologies as TOSCA blueprints into the SODALITE IaC environment. Moreover, the SODALITE IDE includes support for the creation of Ansible Models (AMs) explicitly associated with the resource node types defined in a Resource Model (RM) and application node types defined in AADMs, thus fully covering design and operation of the node types.

3. SODALITE IDE: A Smart Environment for Developing Trustworthy IaC

3.1. Design Goals

The primary goals for the SODALITE IDE are as follows:

- **Portability.** A application should be deployed over various types of infrastructures with little or no modification to the descriptions of the application's deployment model. To support portability, we use the TOSCA standard and the IaC approach to defining and enacting application's deployment, the containerization to packaging application components, and the MDE (model-driven engineering) approach to converting the abstract deployment models into concrete TOSCA and IaC scripts.
- **Reduce Coding Effort.** The end-users with little or no expertise in heterogeneous infrastructures or complex IaC should be able to design the deployment models of the applications with little effort. To reduce coding effort, IDE provides a high-level programming model (DSL), and helps the users with the context assistance and recommendations for various modeling tasks.
- **Support Developing Trustworthy Deployment Codes.** The end-users should be able to develop the deployment artifacts that do not contain known smells, bugs, and errors. To achieve this goal, the SODALITE IDE uses a various types of the reasoning support for TOSCA and IaC, from syntax validation to smell detection.

3.2. Architecture of SODALITE IDE

Figure 1 shows the architecture of the SODALITE IDE, which consists of four main layers: deployment modeling, deployment preparation, orchestration, and intelligence.

3.2.1. Deployment Modeling Layer

This layer contains the main interface with the user through the SODALITE IDE. All the model representations are instances of the SODALITE DSL, which has been designed as a meta-model for creating model instances based on that schema. All the models are designed to collect from users the minimum set of deployment information needed for synthesizing of the TOSCA blueprint based on the assistance provided by the Intelligence Layer. Additionally, the IDE is providing various dashboards giving, in such a way, access to the whole lifecycle of the application.

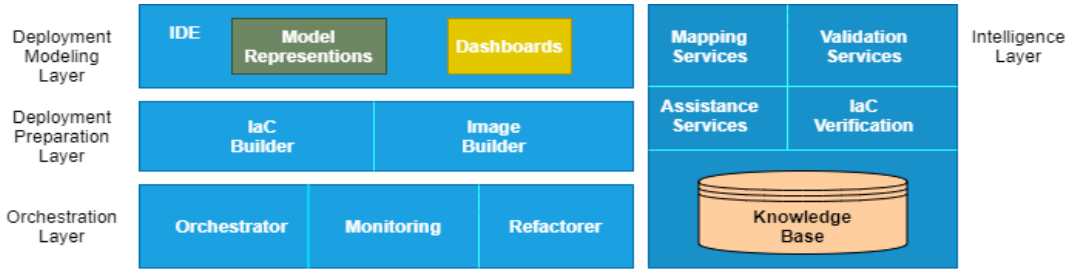

Figure 1: The SODALITE generic architecture framework

3.2.2. Intelligence Layer

This layer is vertical to all the other layers as it is the main storage of all the resources. In this layer, the cloud applications and infrastructures are semantically represented as abstracted reusable patterns. Those semantic abstractions are realised in the form of RDF Knowledge graphs, aiming at the formal representation and linking of the information that enables a semantic reasoning framework to be developed on of those RDF graphs to support search, discovery, validation and reuse. Through the mapping services, the DSL models given as input from the Deployment Modeling layer are transformed to RDF graphs. For having less error-prone deployments, in this layer, the IaC, such as TOSCA blueprints and Ansible scripts, are verified against smells/bugs using data-driven and rule-based techniques.

3.2.3. Deployment Preparation Layer

This layer is the glue of the Deployment Modeling and Intelligence layers with the runtime execution of the models (Orchestration layer). The Image Builder supports pre-building of images for targeting an OS virtualizer such as Docker, while the IaC builder produces a complete and executable TOSCA blueprint based on the AADM verified by he Intelligence layer.

3.2.4. Orchestration Layer

This layer is in charge of the deployment of the applications into heterogeneous infrastructures through the Orchestrator which receives the application deployment model as a TOSCA blueprint from the Deployment Preparation Layer. Also, in order to improve the application's performance, this layer is responsible for refactoring the application based on monitoring parameters. The Refactorer searches for the best alternative deployment variant utilizing the reasoning capabilities (matchmaking) of the Knowledge Base (the intelligence layer).

3.3. Using SODALITE IDE to Develop Trustworthy IaC

One of the main workflows supported by the SODALITE platform is the authoring of an AADM for preparing the deployment of a complex application. The Resource Models(RMs) provide TOSCA related modeling concepts supporting the definition of types for infrastructure resources, and AADMs support the specification of application component instances, as TOSCA templates.

For fostering re-usability, during the authoring of the AADM, we reap the benefit of the resource types stored in a common, shared SODALITE *knowledge base* (KB). The Intelligence of the IDE mainly derives from the semantic Knowledge Base (KB) for assisting users developing the model with hints, suggestions, and validation during the process of AADM development.

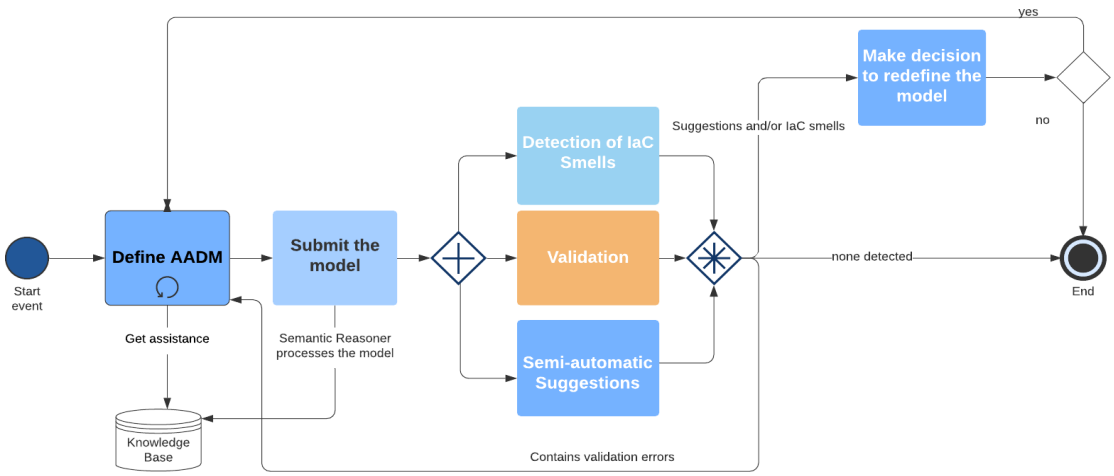

Figure 2: BPMN model describing the AADM authoring workflow.

Figure 2 shows the AADM authoring process using a BPMN workflow diagram. Initially, the user defines the AADM in the SODALITE DSL language by getting content iterative assistance both for syntactic, and semantic content (provided by the KB). This assistance could include retrieving compatible concepts, such as properties, and attributes, with the type schema of each template, locating compatible hosts for an application or reusing resources matching component's requirements. Subsequently, the user submits the model, and in the back end, the Semantic Reasoner processes the model. This process includes the mapping of the DSL model to RDF Knowledge Graphs, and then three activities are performed:

- **Validation.** In TOSCA, the type system supports inheritance as a type can extend another, inheriting all its concepts (e.g. properties, capabilities). Each template of the AADM is an instance of a specific type, namely an infrastructure resource, and gets validated against this type definition. Some of the validation cases are: (i) Verification of TOSCA Application Topologies in respect of all inter-component relationships [8] (ii) Required properties are missing
- **Semi-automatic Model Completion.** To further abstract the model, some information can be omitted by the user, and be suggested or auto-completed by the Semantic Reasoner. For instance, Semantic Reasoner suggests node templates that could serve as requirements, according to the type definition of the template, representing connections such as where the application/middleware component can be hosted or dependencies with other components. This missing information can be either suggested on the design time, or

auto-completed on the deployment time.

- **Quality Assurance of IaC.** To assure the quality and trustworthiness of TOSCA and IaC programs, we support calculating different quality metrics for them, and detecting various smells and linguistic issues in them using the ontology reasoning and the machine learning and NLP (Natural Language Programming) based techniques. More information about for our support be found in our earlier publications [9, 10, 11].

Then, the output of the aforementioned activities can lead to different paths. If the validation process results in detected errors, then the model is not saved in the KB, the errors are returned, and the user should redefine the model. If no errors are detected, the model is saved in KB. If suggestions and/or IaC smells are returned then the user can optionally accept them, and adjust the model by following again the same workflow.

A similar process is followed in the design of the RMs and the *optimization models* (OMs). The modelling of OMs are related with the static optimization of the application, for which recommendations with quick-fixes are provided. Additionally, the modelling of *Ansible models* (AMs) is supported representing the operation implementations as Ansible playbooks of the infrastructure resources. This KB-powered assistance is planned for the AMs.

3.4. Implementation

The SODALITE IDE is a part of the SODALITE H2020 project [12]. It was designed, and implemented in Eclipse as a set of plugins, offers editors for the models, described in Section 3.3, namely, RMs, AADMs, OMs, and AMs constituting instances compliant with the SODALITE DSLs. The RM and AADM DSLs are closely related to the TOSCA specification. This separation of modeling concerns, targeting different modeling roles, simplifies the TOSCA complexity and promotes the reusability of resources stored in the shared *knowledge base* (KB).

The current release of the IDE[2] offers textual (XText) editors for all the SODALITE DSLs, and a graphical view representation (Sirius) for the AADM, which is automatically generated (Figure 3). Sirius technology permits the creation of multiple graphical viewpoints of the same DSL. These representations are synchronised, making changes in one representation propagate to the others sharing the same underlying model. The IDE textual editors offer most the features included by XText, remarkably the syntax coloring, and validation, formatting and auto-completion, cross-reference navigation, quick-fix proposals, outline view, and others.

Smart context-aware content-assistance and semantic validation is largely supported through the symbolic inference capabilities of the KB. These RDF graphs capture both the structural and semantics relationships of the models, promoting reusability and interoperability using standard ontology languages (OWL2) and reasoning tools. In Figure 4, a part of a custom node type is depicted as a UML graph inheriting other TOSCA normative types and containing a description with various concepts. More information about SODALITE ontologies can be found in an earlier publication[13].

AADMs and RMs are stored into the KB, enabling the reuse of the resource instance and types defined in them. Additionally, the SODALITE IDE provides support to browse the KB, retrieve, modify, and delete models stored therein. It also supports other SODALITE processes,

[2]https://github.com/SODALITE-EU/ide

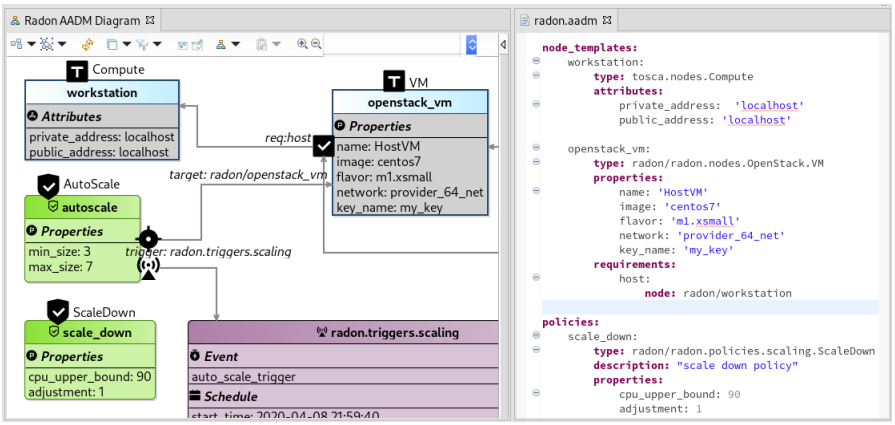

Figure 3: AADM graphical and textual view representation

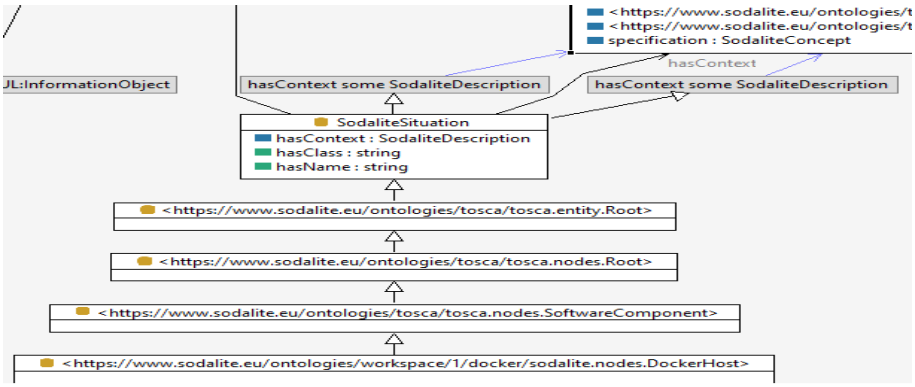

Figure 4: *sodalite.nodes.DockerHost* custom node type as a UML graph in Topbraid ME Composer

notably the deployment of AADMs into target infrastructures, the creation of resource images, and the governance of deployments.

Ansible Models are used to implement the operations defined in the RM or AADM. The IDE provides support for this integration between TOSCA and Ansible, by allowing to couple an AM with an operation from a RM or AADM node definition. In this way, while the user is defining the AM, the IDE suggests the available inputs provided by the TOSCA operation, that can be used in the AM. Furthermore, the AM DSL, while being related to the Ansible specification, conceptually groups the Ansible attributes in categories based on their usage, for better organization of the code. The IDE takes care of generating the concrete Ansible playbook from the abstract AM, that can then be executed for the deployment in the target infrastructures. Two demo videos, one for AADM and OM editors, and one for Ansible DSL editors are available at figshare[3].

[3]https://figshare.com/projects/SODALITE/101705

4. Evaluation

We first evaluated the feasibility of the SODALITE IDE using it to implement the three industrial case studies of the SODALTE project, namely clinical trials, vehicle IoT, and Snow. Clinical trials case study focuses the development of a simulation process chain supporting in-silico clinical trials of bone-implant-systems. Vehicle IoT case study deploys a distributed system for processing vehicular data over Cloud and Edge environments. Snow case study deploys a workflow that processes snow images from multiple data sources to derive information on mountain snow coverage. For each use case, using the SODALITE IDE, we developed the AADM models for the application, and deploy the application based on the developed models. The case study resources are in our GitHub repository.

We next performed controlled experiments with two different types of users, namely non-expert users (9 participants) and TOSCA experts (5 participants), with the objective to receive feedback on perceived ease of use, usefulness and intention to use of the DSL and IDE. The mentioned factors are usually determining the user acceptance of a particular technology. All experiment participants have been asked to focus on the development of AADMs for a Machine Mearning (ML) application, which consists of 1) a database that stores training data, 2) a component that trains a ML model 3) a repository that stores trained machine learning models, 4) a component that makes predictions/inferences based on the trained models. The use case owners used their corresponding use cases.

The experiments with TOSCA experts showed the SODALITE can help to achieve 27.73% improvement over the baseline consisting in using their typical textual editor for creating a TOSCA topology. The experiments with both groups showed that the users consider the SODALITE IDE very useful, easy to use, and think it has a high adoption potential. As a weak point, some missing features of the IDE , the partial stability, and incomplete documentation of the IDE have negatively impacted on the user's effort and time. We plan to work on these aspects in the next period.

5. Conclusion

In this paper, we presented an overview of the SODALITE IDE, a smart suite for IaC, providing full support for the modeling of infrastructure resources, application deployment topologies, application optimizations, and operation implementations, and also for the management of deployed applications at runtime. In future, we intend to: (i) provide more advanced context-assistance and validation services (ii) abstract the application model by permitting the concretization of the AADM at deployment time by relying on Semantic Reasoning services (iii) improve the AADM visual modelling authoring from palettes, (iv) enhance OM authoring and (v) further interconnect the different DSLs.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 825480 (H2020), SODALITE. We thank all members

of the SODALITE consortium for their inputs and feedbacks to the development of this paper.

References

[1] Oasis, Oasis topology and orchestration specification for cloud applications version 1.0, 2013. URL: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf.

[2] M. Carlson, et al., Cloud application management for platforms (2012). URL: https://www.oasis-open.org/committees/download.php/47278/CAMP-v1.0.pdf.

[3] A. Bhattacharjee, Y. D. Barve, A. Gokhale, Cloudcamp : A model-driven generative approach for automating cloud application deployment and management, 2017.

[4] N. Ferry, et al., Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems, in: 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, 2013, pp. 887–894.

[5] M. Palyart, D. Lugato, I. Ober, J.-M. Bruel, Mde4hpc: An approach for using model-driven engineering in high-performance computing, in: SDL 2011: Integrating System and Software Modeling, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 247–261.

[6] M. Śmiałek, K. Rybiński, R. Roszczyk, K. Marek, Towards a unified requirements model for distributed high performance computing, Springer International Publishing, Cham, 2020, pp. 1–20.

[7] O. Kopp, T. Binz, U. Breitenbücher, F. Leymann, Winery - a modeling tool for tosca-based cloud applications, in: ICSOC, 2013.

[8] A. Brogi, A. D. Tommaso, J. Soldani, Sommelier: A tool for validating tosca application topologies, in: 5th International Conference on Model-Driven Engineering and Software Development, 2017, pp. 1–22.

[9] I. Kumara, G. Quattrocchi, D. Tamburri, W.-J. Van Den Heuvel, Quality assurance of heterogeneous applications: The sodalite approach, in: Advances in Service-Oriented and Cloud Computing, Springer International Publishing, 2021, pp. 173–178.

[10] I. Kumara, Z. Vasileiou, G. Meditskos, D. A. Tamburri, W.-J. Van Den Heuvel, A. Karakostas, S. Vrochidis, I. Kompatsiaris, Towards semantic detection of smells in cloud infrastructure code, WIMS 2020, Association for Computing Machinery, 2020, p. 63–67.

[11] N. Borovits, I. Kumara, P. Krishnan, S. D. Palma, D. Di Nucci, F. Palomba, D. A. Tamburri, W.-J. van den Heuvel, Deepiac: Deep learning-based linguistic anti-pattern detection in iac, MaLTeSQuE 2020, Association for Computing Machinery, 2020, p. 7–12.

[12] E. Di Nitto, J. Gorroñogoitia, I. Kumara, G. Meditskos, D. Radolović, K. Sivalingam, R. S. González, An approach to support automated deployment of applications on heterogeneous cloud-hpc infrastructures, in: 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2020, pp. 133–140.

[13] G. Meditskos, Z. Vasileiou, A. Karakostas, S. Vrochidis, I. Kompatsiaris, A pattern-based semantic lifting of cloud and hpc applications using owl 2 meta-modelling, in: Special Session on High Performance Services Computing and Internet Technologies, HPCS, 2020.