UML-based Cloud Application Modeling with Libraries, Profiles, and Templates*

Alexander Bergmayr, Javier Troya, Patrick Neubauer,
Manuel Wimmer, and Gerti Kappel

Business Informatics Group, Vienna University of Technology, Austria
{bergmayr,troya,neubauer,wimmer,kappel}@big.tuwien.ac.at

Abstract. Recently, several cloud modeling approaches have emerged. They address the diversity of cloud environments by introducing a considerable set of modeling concepts in terms of novel domain-specific languages. At the same time, general-purpose languages, such as UML, provide modeling concepts to represent software, platform and infrastructure artifacts from different viewpoints where the deployment view is of particular relevance for specifying the distribution of application components on the targeted cloud environments. However, the generic nature of UML's deployment language calls for a cloud-specific extension to capture the plethora of cloud provider offerings at the modeling level. In this paper, we propose the *Cloud Application Modeling Language (CAML)* to facilitate expressing cloud-based deployments directly in UML, which is especially beneficial for migration scenarios where reverse-engineered UML models are tailored towards a selected cloud environment. We discuss *CAML's* realization as a UML internal language that is based on a model library for expressing deployment topologies and a set of profiles for wiring them with cloud provider offerings. Finally, we report on the use of UML templates to contribute application deployments as reusable blueprints and identify conceptual mappings between *CAML* and the recently standardized TOSCA.

Keywords: Cloud Computing, Model-Driven Engineering (MDE), Cloud Modeling, UML, Language Engineering

1 Introduction

Cloud computing has recently emerged as a new possibility how software can be made available to clients as a service. For software vendors, this is appealing as cloud environments [3] have the benefit of low upfront costs compared to a traditional on-premise solution and operational costs that scale with the provisioning and releasing of cloud offerings. They may range from low-level infrastructure elements, such as raw computing nodes, over higher level platforms, such as a Java execution environment on top of a cloud infrastructure, to ready-to-use software deployed on a platform. As a result, current cloud environments are diverse in nature and show various levels of virtualization they operate on. Recent cloud modeling approaches already capture a considerable set of domain-specific concepts to support different scenarios: description of

* This work is co-funded by the European Commission under the ICT Policy Support Programme, grant no. 317859.

cloud-based applications [17] and their deployments [5, 13, 21], optimization of such deployments [14, 18], provisioning of cloud resources [10], or automating the scalability of cloud environments [9, 11]. At the same time, general-purpose languages, such as UML, provide modeling concepts to represent software, platform and infrastructure artifacts from different viewpoints. Hence, providing extensions to UML that satisfy current cloud modeling requirements appears beneficial, especially when cloud-oriented migration scenarios [4] need to be supported where reverse-engineered UML models are tailored towards a selected cloud environment.

However, to date, effective UML-based support for modeling cloud application deployments that are wired with cloud provider offerings is still missing. As a result, on-premise deployments expressed in UML can hardly be turned into cloud-based deployments without neglecting the intended usage of UML. In the ARTIST project [4], we are particularly confronted with this problem as we work towards a model-driven engineering approach for modernizing applications by novel cloud offerings, which involves deploying them or at least some of their components on a cloud environment. Ideally, the design choices of a cloud-based deployment are expressed at the modeling level, which calls for an appropriate language support in the light of UML. While in this way, not only the full expressive power of UML can be exploited, also a seamless integration of cloud-specific models into existing UML models is ensured.

In this paper, we propose the *Cloud Application Modeling Language (CAML)* [7] to enable representing cloud-based deployment topologies directly in UML and refining them with cloud offerings captured by dedicated UML profiles. Thereby, a clear separation is achieved between cloud-provider independent and cloud-provider specific deployment models [1], which is in accordance with the PIM/PSM concept. In our case, the "platform" refers to the cloud provider. We developed profiles for two major cloud providers[1] and integrated them into a common cloud profile. Inspired from common cloud computing literature [2, 3, 12], recent cloud modeling approaches [5, 9, 15, 17, 18, 21] and cloud programming approaches[2], we developed *CAML's* model library that facilitates developing base deployment topologies to which cloud offering profiles are applied. The benefits of realizing *CAML* as an internal language of UML are threefold: (i) UML provides a rich base language for the deployment viewpoint, (ii) "cloudifying" UML models is facilitated without the need to re-model existing applications, and (iii) profiles in UML allow hiding details of cloud provider offerings from models and dynamically switching between them by (un-/re-)applying respective cloud provider profiles.

We motivate the practical value of *CAML* by means of a deployment scenario in Section 2. In Section 3, we give the design rationale of *CAML* and provide insights into its model library and the covered UML profiles whereas in Section 4, we discuss the employment of UML templates as reusable deployment blueprints. The operationalization of *CAML* by means of a mapping to the recently accepted TOSCA standard is dedicated to Section 5. Finally, in Section 6 we discuss work related to *CAML* before we conclude in Section 7.

[1] Amazon AWS: http://aws.amazon.com and Google Cloud Platform: http://cloud.google.com

[2] Deltacloud: https://deltacloud.apache.org and jclouds: http://jclouds.apache.org

2 Motivating Deployment Scenario

To motivate the benefits of employing UML as the host language for realizing *CAML*, we give an overview of UML's structural viewpoints that support representing application deployments by means of a reference application[3] of the ARTIST project. We take the viewpoint of the application components and their deployment. Figure 1a depicts some components of our application, an excerpt of their realizing classes and the manifestation of these components by deployable artifacts. A possible on-premise deployment for them is presented in Figure 1b. It covers instances of the two deployable artifacts and connects them to a Java-based middleware and a relational DBMS, which are in turn deployed onto a node with specified (virtual) machine characteristics. The model elements of the deployment are instances of the custom types defined in the component viewpoint (see Figure 1a) and the deployment viewpoint (see Figure 1c), respectively. With the emergence of cloud offerings and the demand to exploit them, deployment models need to be expressive enough to capture such offerings. This is exactly the idea of *CAML*. Because it is realized in terms of lightweight extensions to UML, *CAML* models are applicable to UML models and so to our modeled reference application as depicted in Figure 1. In Sections 3 and 4, we present cloud-based deployments for our reference application.

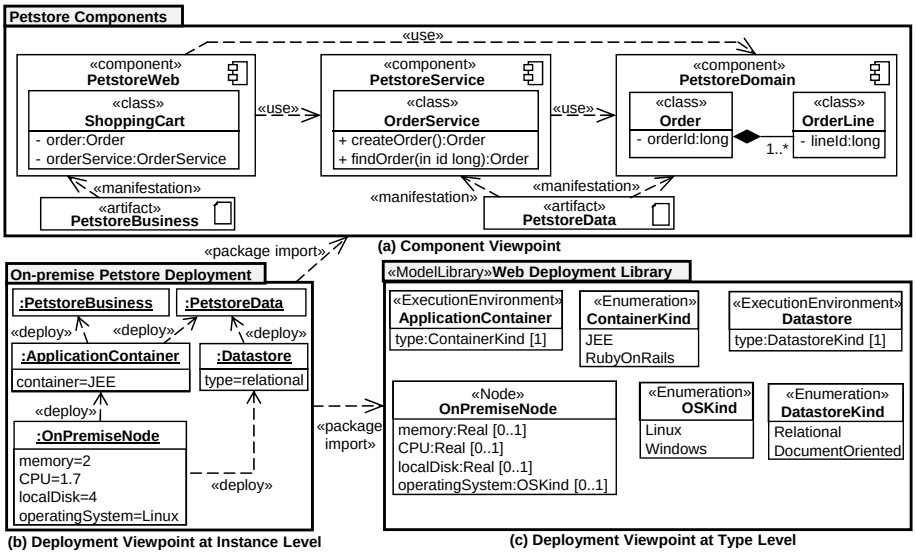

Fig. 1: *CAML* Use-Case

3 Cloud Application Modeling

With *CAML*, we propose lightweight extensions to UML for modeling cloud application deployments that are seamlessly applicable to UML models, such as component models, typically created throughout software modeling activities. The intended purpose of

[3] It is based on the Java Petstore: http://www.oracle.com/technetwork/java/index-136650.html

CAML is to express deployment topologies by common cloud modeling concepts and to enable the wiring of such models with concrete cloud provider offerings. This wiring is achieved by applying a dedicated *CAML Profile* to a deployment model expressed in terms of the *CAML Library*. As a result, a clear separation between cloud-provider independent and cloud-provider specific models is achieved. Selecting cloud provider offerings at the modeling level for a concrete deployment becomes a matter of applying the respective stereotypes. The overall set of stereotypes encompass the possible design choices provided by *CAML* regarding cloud provider offerings.

3.1 Model Library for Cloud Deployment Topologies

As presented in Figure 2, the *CAML Library* is built around the concept of *cloud offering*. It is considered as a virtual resource that is expected to be supported by a cloud environment once the wiring with a concrete cloud offering has been performed. More specifically, three offering types capture common cloud environment capabilities. A *cloud node* provides compute capacity and operates at a certain level of virtualization [3]. From an infrastructure-level perspective, cloud nodes come with an operating system, while when turning this perspective to the platform level they also provide middleware, such as a web server and an application container. In case of the latter, the platform is fully managed by a cloud environment. With dedicated scalability strategies, the elastic nature of a cloud environment is managed. For instance, cloud nodes can automatically be acquired depending on the number of incoming requests. Clearly, acquiring and releasing cloud nodes can also be manually controlled. The second offering refers to the *cloud storage* capabilities of cloud environments which provide diverse solutions for structuring application data [12] and increasing their availability by relaxing consistency [22]. Finally, a *cloud service* is considered as a ready-to-use *cloud offering* that is provisioned and managed by a cloud provider. For instance, a load balancer that distributes requests to cloud nodes is an infrastructure-related *cloud service*, while a task queue for long running processes is a platform-related *cloud service*. To represent offering-to-offering connections, *communication channels* are employed while *cloud configurations* enable modifying the assumed conventions of a cloud environment. For instance, an automatic scaling strategy can be configured with boundaries of minimum and maximum running cloud nodes. Generally, instantiated elements of the *CAML Library* are refined to concrete cloud provider offerings via dedicated stereotypes.

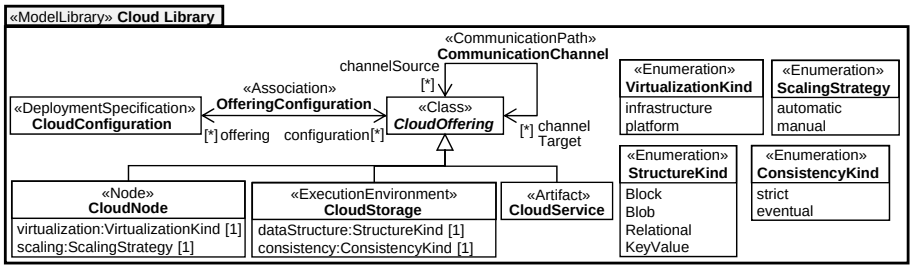

Fig. 2: Cloud Library of *CAML*

3.2 Profiles for Cloud-Provider Specific Deployments

With *CAML Profiles*, we provide a set of UML stereotypes that enable wiring cloud deployment topologies with concrete offerings of cloud providers. Basically, a stereotype embodies a concrete offering at the modeling level and captures its features in terms of properties. Figure 3 presents some stereotypes specific to the cloud offerings of the Google App Engine (GAE) and Amazon AWS. Common cloud offerings that are shared by both providers are lifted to the *common cloud profile*. Considering *instance types*, they are supposed to be applied to cloud nodes to wire them to a concrete cloud offering, such as a "Frontend Instance" (e.g., *GAEF1*) that hosts a Java-based middleware managed by Google's App Engine. In turn, cloud offerings are refined by what we call meta-profiles. With the notion of meta-profiles, we facilitate refining them with technical-related details, such as the performance of instance types, and business-related information [8], like the costs of cloud offerings.

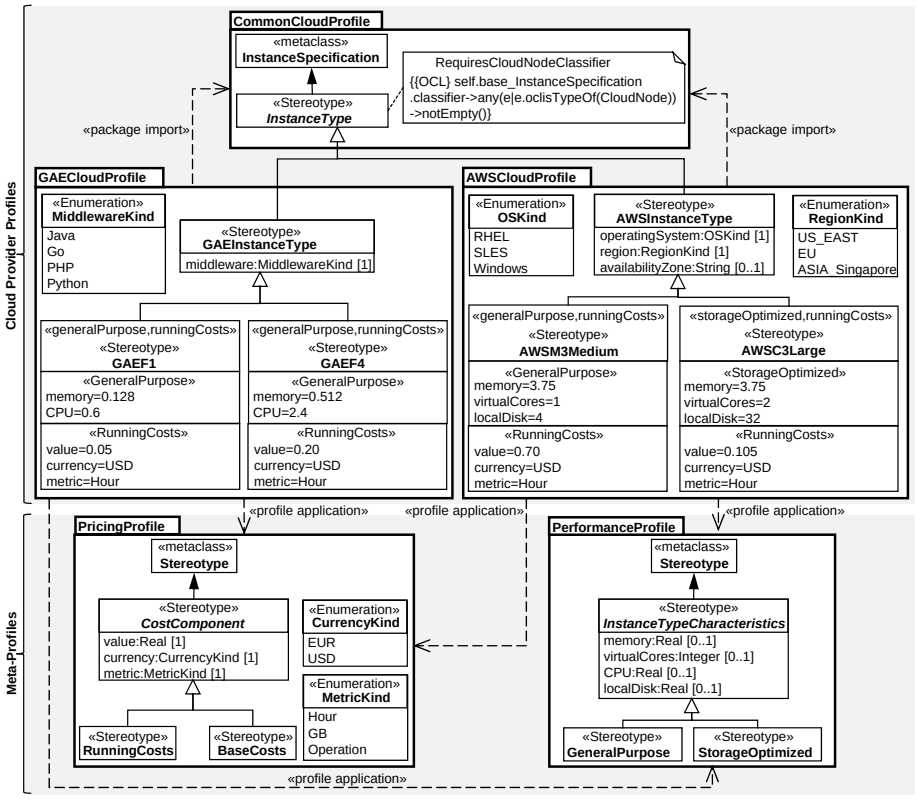

Fig. 3: *CAML* Profiles and Meta-Profiles

3.3 CAML By-Example

To demonstrate how *CAML* is applied, Figure 4 presents a possible deployment topology and refinement towards a GAE-based cloud deployment of our introduced use case (cf. Figure 1). In a first step, we modeled the deployment topology. It consists of two automatically scaled cloud nodes and a key-value cloud storage for managing the application data in an eventually consistent way. As the cloud nodes are specified as platform-level offering, we directly deployed the application components onto them. Then, in a second step, we applied the GAE profile and the respective stereotypes to refine the deployment model towards concrete cloud offerings provided by the GAE. As a result, the modeled cloud nodes refer to the *F1* and *F4* instance types that host a Java-based middleware. The configuration attached to these cloud nodes constrains the maximum number of idle cloud nodes. Finally, GAE's key-value datastore is employed for the required cloud storage capabilities.

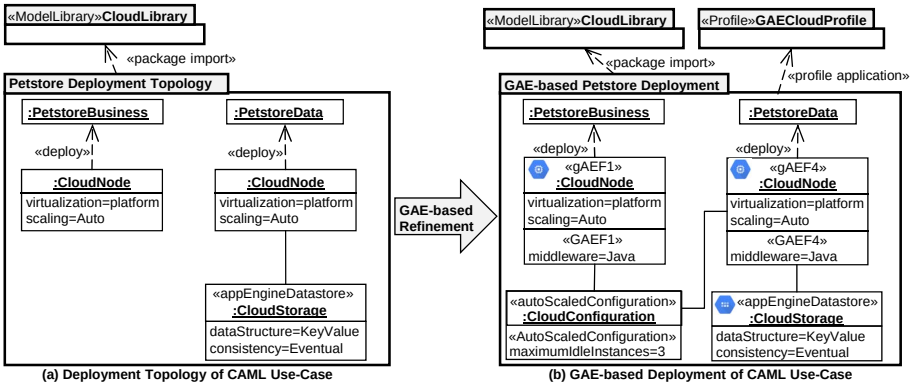

Fig. 4: Reference Application deployed onto Google App Engine

3.4 Prototypical Implementation

To show the feasibility of *CAML*, we have implemented an Eclipse-based prototype, which exploits extension points. In this way, developers can directly use *CAML* in Eclipse tools, such as Papyrus[4], or access its library and profiles in terms of a resource, which is helpful for the development of transformations. *CAML* together with all artifacts used in this paper are publicly available at our project web site [7]. In addition, together with our industrial partner SparxSystems, we have also implemented a first version of *CAML* for Enterprise Architect[5]. This provides first evidence that our proposed approach for developing a UML internal cloud modeling language based on a library and profiles is feasible and current modeling tools with UML support provide the necessary features to support *CAML* models.

[4] Papyrus: http://www.eclipse.org/papyrus
[5] Enterprise Architect: http://www.sparxsystems.at

4 Reusable Deployment Blueprints as UML Templates

As *CAML* is based on UML, its reuse mechanisms can be applied for cloud application deployments. This is particularly useful for providing frequently occurring deployment patterns as predefined UML templates. To show their usefulness and give first evidence of *CAML's* expressivity, we developed 10 templates as reusable deployment blueprints, most of them are based on Amazon's best practices[6]. We modeled their inherent topology with *CAML's* cloud library and refined them with stereotypes from the cloud profile dedicated to Amazon. The developed blueprints are available at our project website [7]. To demonstrate the use of a blueprint, we show how our reference application is bound to a template, which refers in our case to a 2-tier web architecture [12]. To reuse the predefined template, the deployable artifacts need to be bound to the template parameters. Figure 5 depicts the component viewpoint of our reference application and the respective *CAML* template. It consists of two cloud nodes that refer to the "M3Medium" offering of Amazon. Their location is required to be in Europe while the operation system needs to be Linux. For reliability reasons, they are placed in different availability zones. Requests that arrive at the cloud nodes are first handled by a load balancing service, which enables a higher fault tolerance of the application. The number of running cloud nodes is automatically managed by Amazon as expressed by the scalability strategy. Only the minimum number of running cloud nodes and their adjustment is configured. Both cloud nodes are connected to a cloud storage that in turn is replicated to improve data availability. Finally, as Amazon cloud nodes operate at the infrastructure level, the required middleware for our reference application is defined. In fact, we directly reused it from the on-premise deployment given in Figure 1.

[6] Amazon Architecture Center: https://aws.amazon.com/architecture

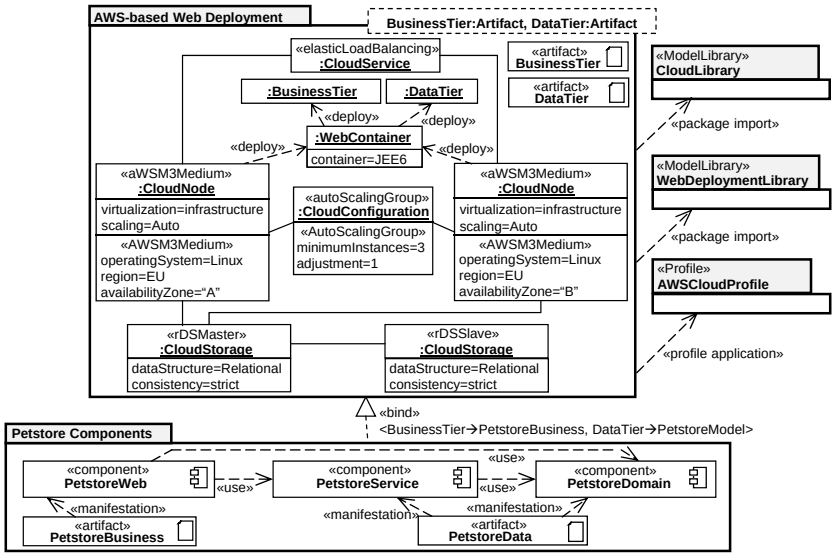

Fig. 5: Reusable Deployment Template for AWS

5 Interoperability between CAML and TOSCA

One major aspect in model-based engineering is to place models as first-class entities in the engineering process. Ideally, they should be turned into executable or interpretable artifacts. Regarding the deployment viewpoint, it appears desirable to translate the respective models into descriptors and scripts that are passed to provisioning engines for cloud environments. For instance, a GAE-based deployment requires specific descriptors for defining the assignment of application modules to a concrete instance type. This assignment can certainly be derived from a *CAML* model. At the same time, there are ongoing efforts in standardizing the representation of cloud-based application deployments. The recently accepted TOSCA standard aims at supporting portable cloud applications. With the notion of management plans, emerging TOSCA-compliant engines are capable to interpret such deployment topologies and initiate the provisioning of defined service templates [5]. Clearly, this is also of practical value for *CAML* models. For that reason, we present an initial mapping between *CAML* and a subset of *TOSCA*. Generally, in TOSCA, two modeling concepts are prevalent: *template* and *type*. Templates embody the elements of a deployment topology while types expose the properties and relationships for which concrete values are provided by templates. In this sense, types are considered as reusable entities that can inherit from each other. Figure 6 depicts a concrete TOSCA model expressed in Vino4TOSCA [6] for an excerpt of our GAE-based application deployment (cf. Figure 4). To represent the TOSCA template for the stereotyped *CAML* cloud node, the pertinent TOSCA types need to be created: "CloudNode" and "GAEF1". The latter is derived from the former as in TOSCA a template can only have a single type. Similarly, the deployed application component is represented by a TOSCA template. Finally, the deployment relationship type is required for connecting the deployed application component to the cloud node at the template level.

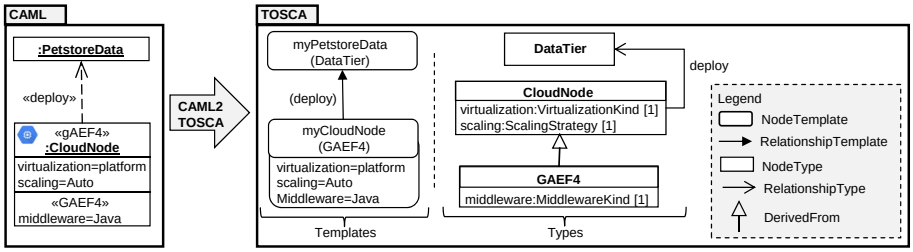

Fig. 6: Mapping between *CAML* and TOSCA

6 Related Work

Cloud modeling approaches with the purpose of achieving the wiring of applications with concrete cloud offerings are most closely related to *CAML*. Modeling concepts of these approaches [5, 9, 15, 18, 21] are reflected by *CAML* on a level of abstraction that facilitates to represent design decisions for cloud-based application deployments. As a

result, modeling concepts of these approaches, e.g., required to achieve the optimization of an application deployment (cf., [15]) or to express elasticity rules (cf., [9]), are not completely captured by *CAML*. However, *CAML* enables expressing cloud application deployments that are seamlessly applicable on UML models usually created throughout software modeling activities as it is realized as a UML internal language. As a result, well-connected modeling views on cloud applications from a cloud-provider independent perspective as well as a cloud-provider specific perspective are supported. The refinement of modeling views is enabled by profiles for cloud providers. This additional typing dimension provided by such profiles and the exploitation of a multi-viewpoint language to realize *CAML* differentiates it from existing cloud modeling approaches and the recently standardized TOSCA.

To the best of our knowledge, the only approach providing cloud modeling support within UML is MULTICLAPP [17]. It proposes a UML profile for the purpose of representing components that are expected to be deployed onto a cloud environment by applying cloud-provider independent stereotypes to them. Hence, these stereotypes do not support wiring components with cloud provider offerings, which is different to *CAML* as stereotypes are applied to achieve exactly that wiring.

CloudML-UFPE [16] provides modeling concepts to represent cloud offerings connected with the internal resources of a cloud environment. Similarly to approaches [10, 11, 19], which propose modeling concepts to represent resources internally managed by a cloud environment, the focus is set on the cloud provider perspective. As a result, such modeling approaches support cloud providers to model their environments, which is out of the scope of *CAML*.

Finally, it is worth mentioning that approaches, such as Deltacloud[7] and jclouds[8], provide an abstraction layer on top of cloud-provider specific programming libraries. They can be considered as transformation targets for cloud modeling approaches to automate the provisioning of modeled application deployments.

7 Conclusion and Future Work

We have presented *CAML* as a UML internal language based on a library, profiles, and templates. Currently, it is employed by the ARTIST project to model deployments of large applications used in practice. In this respect, cloud providers that operate at both infrastructure level and platform level are targeted. Although the realization and initial evaluation of *CAML* seems promising, several lines of future work need to be investigated. First, we aim for an automated maintenance of provider-specific profiles with, for instance, performance or pricing information based on web information extraction techniques. Second, we intend to provide a simulator for *CAML* to provide prediction about non-functional properties such as costs and performance. In this respect, we plan to explore how FUML can be employed to provide behavioral semantics for *CAML* in a similar way as we use it to define behavioral semantics for metamodels [20]. Finally, we aim for interoperability with current cloud modeling approaches by providing dedicated transformations or a UML profile.

[7] https://deltacloud.apache.org
[8] https://jclouds.apache.org

References

1. Ardagna, D., Nitto, E.D., Mohagheghi, P., Mosser, S., Ballagny, C., D'Andria, F., Casale, G., Matthews, P., Nechifor, C.S., Petcu, D., Gericke, A., Sheridan, C.: MODAClouds: A Model-Driven Approach for the Design and Execution of Applications on Multiple Clouds. In: MISE Workshop (2012)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: A View of Cloud Computing. CACM 53(4) (2010)
3. Badger, M.L., Grance, T., Patt-Corner, R., Voas, J.M.: Cloud Computing Synopsis and Recommendations. Tech. rep., NIST Computer Security Division (2012)
4. Bergmayr, A., Bruneliere, H., Cánovas Izquierdo, J.L., Gorroñogoitia, J., Kousiouris, G., Kyriazis, D., Langer, P., Menychtas, A., Orue-Echevarria Arrieta, L., Pezuela, C., Wimmer, M.: Migrating Legacy Software to the Cloud with ARTIST. In: CSMR (2013)
5. Binz, T., Breitenbücher, U., Kopp, O., Leymann, F.: TOSCA: Portable Automated Deployment and Management of Cloud Applications. In: Advanced Web Services (2014)
6. Breitenbücher, U., Binz, T., Kopp, O., Leymann, F., Schumm, D.: Vino4TOSCA: A Visual Notation for Application Topologies based on TOSCA. In: OTM (2012)
7. CAML: Project Web Site (2014), http://code.google.com/a/eclipselabs.org/p/caml
8. Cardoso, J., Barros, A., May, N., Kylau, U.: Towards a Unified Service Description Language for the Internet of Services: Requirements and First Developments. In: SCC (2010)
9. Chapman, C., Emmerich, W., Márquez, F.G., Clayman, S., Gallis, A.: Software Architecture Definition for On-Demand Cloud Provisioning. Cluster Comput. 15 (2012)
10. Chatziprimou, K., Lano, K., Zschaler, S.: Towards a Meta-model of the Cloud Computing Resource Landscape. In: MODELSWARD (2013)
11. Dougherty, B., White, J., Schmidt, D.C.: Model-Driven Auto-Scaling of Green Cloud Computing Infrastructure. FGCS 28 (2011)
12. Fehling, C., Leymann, F., Retter, R., Schupeck, W., Arbitter, P.: Cloud Computing Patterns - Fundamentals to Design, Build, and Manage Cloud Applications. Springer (2014)
13. Ferry, N., Rossini, A., Chauvel, F., Morin, B., Solberg, A.: Towards Model-Driven Provisioning, Deployment, Monitoring, and Adaptation of Multi-cloud Systems. In: CLOUD (2013)
14. Frey, S., Fittkau, F., Hasselbring, W.: Search-based Genetic Optimization for Deployment and Reconfiguration of Software in the Cloud. In: ICSE (2013)
15. Frey, S., Hasselbring, W.: The CloudMIG Approach: Model-Based Migration of Software Systems to Cloud-Optimized Applications. Advances in Software 4 (2011)
16. Gonçalves, G., Endo, P., Santos, M., Sadok, D., Kelner, J., Merlander, B., Mångs, J.E.: CloudML: An Integrated Language for Resource, Service and Request Description for D-Clouds. In: CloudCom (2011)
17. Guillén, J., Miranda, J., Murillo, J.M., Canal, C.: A UML Profile for Modeling Multicloud Applications. In: ESOCC (2013)
18. Leymann, F., Fehling, C., Mietzner, R., Nowak, A., Dustdar, S.: Moving Applications to the Cloud: An Approach Based on Application Model Enrichment. IJCIS 20(3) (2011)
19. Liu, D., Zic, J.: Cloud#: A Specification Language for Modeling Cloud. In: CLOUD (2011)
20. Mayerhofer, T., Langer, P., Wimmer, M., Kappel, G.: xMOF: Executable DSMLs based on fUML. In: SLE (2013)
21. Nguyen, D.K., Lelli, F., Taher, Y., Parkin, M., Papazoglou, M.P., van den Heuvel, W.J.: Blueprint Template Support for Engineering Cloud-Based Services. In: ServiceWave (2011)
22. Vogels, W.: Eventually consistent. CACM 52(1) (2009)