MDE-Cloud Deployment Comparison

In this paper we will explore three different tools available in terms of model conversion or transformation. The main idea is to start with a Platform Specific Model (PSM) recovered from the original Legacy System and obtains a Platform Independent Model (PIM) that can be easily transformed to a PSM for a specific Cloud platform. The three available tools are PIM4Cloud from the REMICS project, the ARTIST project's CAML and CloudML.

REMICS-PIM4Cloud

REMICS proposes to provide support for application deployment to cloud platforms via the use of a domain-specific language (DSL) as an integral part of their PIM4Cloud interpreter [1]. This DSL models applications in terms of components, of their properties and their topologies. The purpose of the interpreter is actually to match the description of an application to the description of the platform it will be deployed on. Subsequently, it proceeds to the actual deployment, and returns a model of the application containing annotations with run-time properties. It relies on CloudML to model and provide the cloud resources needed to deploy and run the application. They do so by defining templates that specifies the characteristics of each nodes and the topology. The templates are written using the JavaScript Object Notation (JSON). Providing these templates as input to the CloudMLEngine, cloud-computing nodes are created from the given specification. The CloudMLEngine builds on top of the jClouds framework, thus it provides, through a Java API, abstraction of the environmental features specific to the cloud.

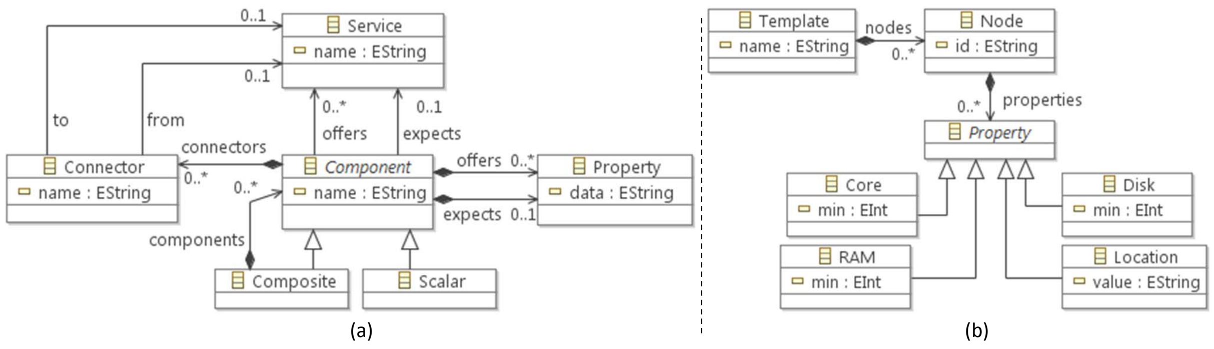

Figure 1 (a) Modeling with PIM4Cloud (b) Templates in context [4]

In the figure above, in part (a) we can observe how the component-based modeling looks as an integral part of the propose solution. We can observe also that a *Component* can offer and expect services, and that the *Connectors* allow for linking components via *Services*. In part (b) we can appreciate an overview of how the *Templates* are used to model the *Nodes* in the context of CloudML jointly with PIM4Cloud. In other words and simpler terms, the templates are defined by the model created in part (a) and used in part (b) to model the nodes that are going to be deployed on the cloud.

In conclusion this tool, PIM4Cloud in conjunction with CloudML, are excellent to model and deploy at the Infrastructure as a Service level, mainly because of the aspect of templates and the usage of the CloudMLEngine. It can also provide sufficient means to model and deploy at the Platform as a Service Level and Software as a Service Level but it remains more suited for the former.

ARTIST-CAML

The ARTIST project proposes a modelling language named CAML [3] (Cloud Application Modelling Language) that enables the representation of models from the reverse engineering and the forward-engineering perspectives. A primary objective of CAML is to provide guidance in terms of choosing optimization patterns that can be used in order to transform the cloud-independent models (PIM in this case) into cloud-specific models (Cloudified-PSM). These PSMs can in turn be used to generate cloud-optimized application code that can serve as a pre-requisite for the deployment on the selected cloud environment. More specifically, it addresses Non-Functional Cloud Consumer Concerns, which represents the Service Level Profile (SaaS, PaaS, IaaS), Pricing Profile and the Performance Profile. It also addresses the Functional Cloud Consumer Concerns, which represents the different UML profiles that are vendor-specific (Microsoft Azure, Amazon AWS and Google App Engine). Final element of CAML is the Cloud Deployment Modeling Library (CDML), and its purpose is to express cloud-based software from a deployment perspective that is independent from the cloud service providers.

Generally speaking, CAML operates on all the service models (IaaS, PaaS, SaaS) and seem to provide an extensive and comprehensive approach to model deployment of applications, in a MDE effort, to a cloud environment.

The UML Profiles for functional cloud consumer concerns are structured using five views, providing particular perspectives from the modelled cloud environments. The first of these views is the Core View, which depicts the runtime environment such as, Java, PHP, Python etc. and the usage type whether it is Free, Paid, or any other usage type. The second view is called the Instances View, and it tries to capture details pertaining the running instance on the cloud environment such as the utilization type (light, medium, heavy), the contract type (one year, three years, undefined), the instance category (on demand, reserved, spot), the region as well as the operating system. The third view is the Storage view, depicts the types of storage available to the consumer, as an example, RDBS, SimpleDB or any other storage type made available by the cloud service provider. The fourth view is Services View, it contains all the services offered by the cloud service provider, for example Google App Engine offers a TaskQueue service as well as many others, such as E-mail and Search just to name a few. The fifth and final view is the Configuration View, it consist of encapsulating the configuration of the deployment and the environment of the deployment, either in terms of nodes or of backend-frontend configurations.

For brevity purposes we won’t go into further details about the UML Profiles for the non-functional cloud consumer concerns.

Also CAML is made available through an Eclipse plugin/add-on or an Enterprise Architect plugin/add-on, and thus makes it for an easy integration in the normal workflow of application development.

In conclusion, the idea of developing UML Profile that is tailored for each of the major cloud service providers is a very neat concept. It makes for a modelling tool that is comprehensive and careful of the environment of deployment specifically.

CloudML

CloudML stands for Cloud Modelling Language [5] and is an XML-based approach to model the architecture and deployment specificities from an infrastructure perspective. It models the different cloud providers' features as services, which are in turn specified using profiles. Profiles represent the different nodes or links, made available by the cloud service providers, in terms of their hardware and functional capabilities and they are aggregated with respect to the type of service they render. One of the key features of CloudML is the ability to represent every physical and virtual cloud resource and their current state. This is achieved with the usage of XML schemas that characterize the node characteristics such as the service descriptions and also by capturing the state of the virtual machines. Another important feature is the ability to specify the service requests in terms of the services offered by the cloud service provider.

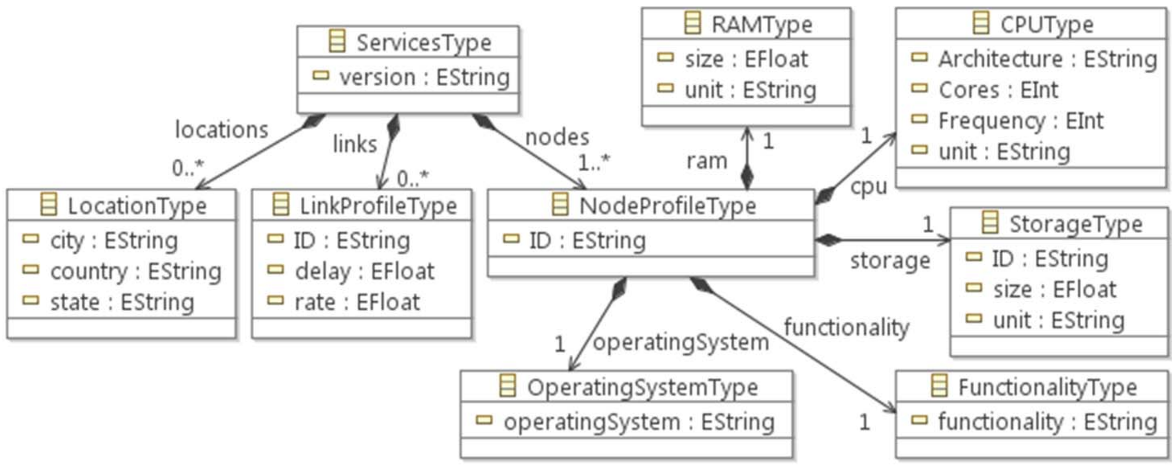

Figure 2 CloudML Service Description Model [4]

In the figure below, we can observe how one could model a service using the Cloud Modeling Language and we notice that modeling elements to model a virtual or a physical resource are almost identical, even though CloudML explicitly differentiate between the two of them. Furthermore, in the next figure the physical or virtual modeling elements are abstracted to provide what they refer to as the Resource Description model. This enables the modeller to treat the different resources in an agnostic fashion and thus makes for more generalized models. The third diagram is to showcase the Request Description modelling capabilities of this modeling language. It provides a simple clean and neat modeling artefact that truly captures the necessary gaps between the resources, the request and delivery.

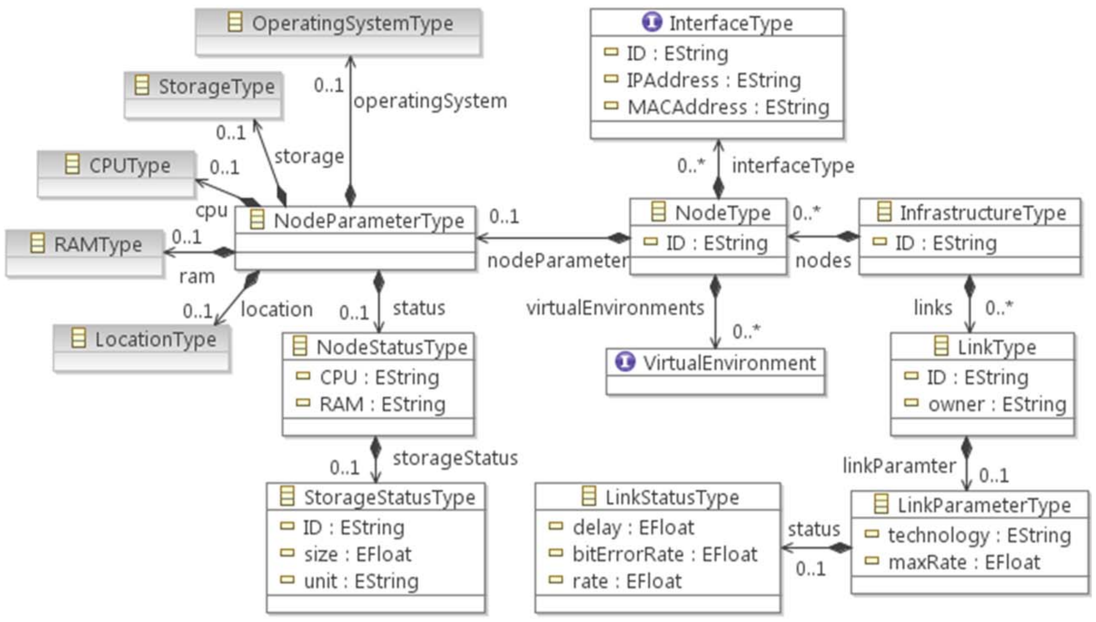

Figure 3 CloudML Resource Description Model [4]

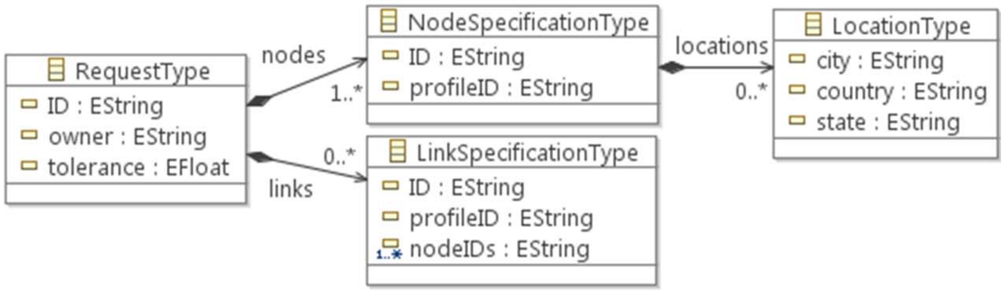

Figure 4 CloudML Request Description Model [4]

In the figure below is the overview of the whole CloudML process. It captures all the actors involved in the process of migrating an application to the cloud platform, from the Business inclined person to the Cloud (more technically inclined) Expert to the User (or the person using the CloudML process that is in charge of the provisioning). We can observe the different stages of the CloudML process and it gives a nice overview of it mechanics. First the business person provides a set of requirements to the Cloud expert based on the application and their current and future needs. From these the Cloud Expert defines different possible node topologies and proceeds to write the templates for those in XML. The next step involves selecting which template, which the User does, and feeding it into the CloudMLEngine. The engine consists of a five steps process. The first one consists of converting the template into a native format, and the second one consist of converting pure nodes into instances ready for provisioning. The third step is to connect to all the desired providers, and the fourth step is to propagate the instances. Finally the fifth step consists of producing models@run.time of the instances being propagated, which provides meta-data about the propagated instances back to the user and are updated in an asynchronous manner.

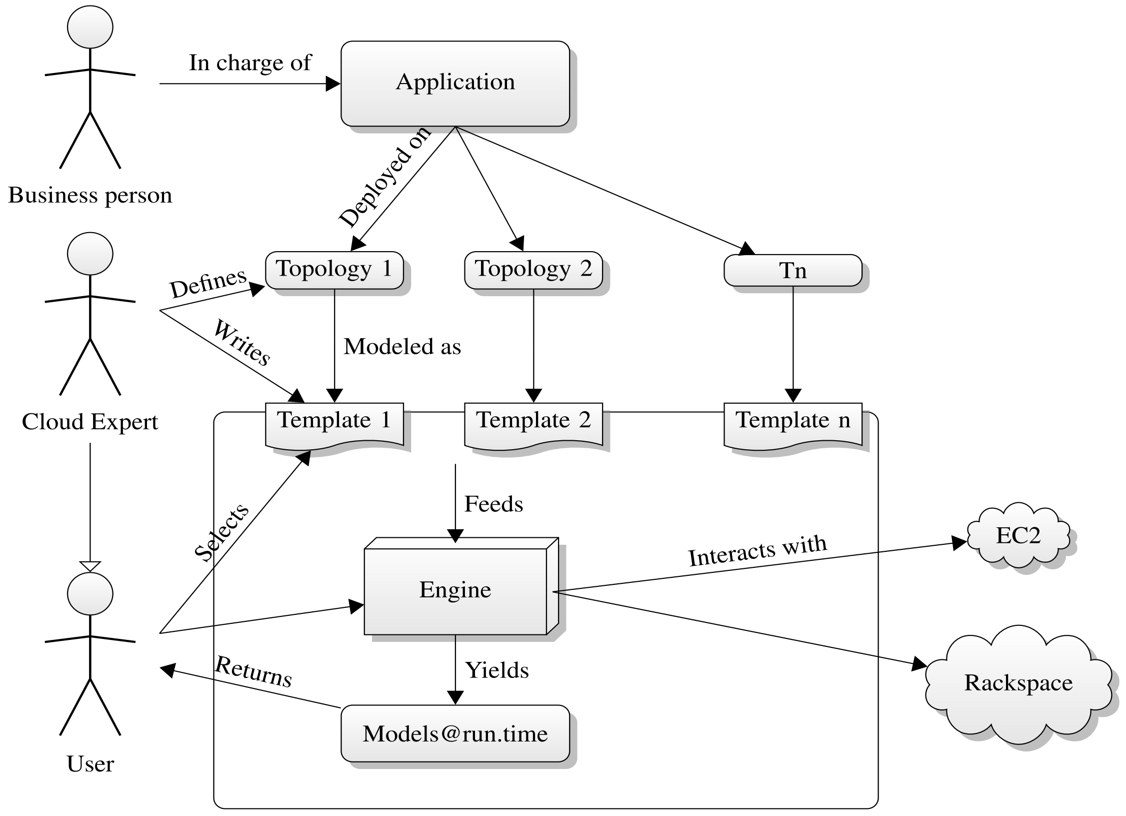

Figure 5 Overview of the process involved in CloudML [5]

Conclusion

The three tools presented in this paper all tackles the problem of modelling deployment differently, each one using more or less a different perspective. REMICS addresses this by using a DSL to model the deployment to generate templates that will be used as an input to the CloudMLEngine. Thus CloudML is the underlying technology of REMICS, and we could see REMICS PIM4Cloud as a higher-level process compared to CloudML, which is more flexible and extensible, but much lower level in the grand scheme of modernization or migration frameworks or methodologies. As for ARTIST CAML, it was a much more ambitious proposal, and it was justified by realizing a state of the art survey of all the deployment modelling technologies available in [4], which notably included REMICS PIM4Cloud and CloudML itself. In that literature review they've identified several shortcomings of all the methods surveyed, from which they derived a proposed methodology, which is CAML. Perhaps the most complete out-of-the-box solution presented, and the most recent one too. It focuses at providing cloud provider specific UML Profiles to the modellers and does it in an elegant and modular fashioned, which makes for a more accessible and less obscure solution.

Possible Future Work

None of these methodologies or frameworks really addresses the service delivery model that is Software as a Service, as described in [2]. We think that this could be a direction for future work, at least in the experimental sense, since the SaaS model is so heterogeneous in terms of deployment scenarios and environment.

Another prospect in terms of future research would be in how could we reduce the need for the user to understand the cloud infrastructure or to resort to a cloud expert. Since in all of the proposed solutions, a third party cloud expertise is needed to bridge the gap from PIM to PSM that are cloud specific that can be readily deployed on the cloud. More specifically automate the process of going from a PIM that is built with a SOA, to a PSM that accounts for the cloud deployment environment.

REFERENCES

[1] Sadovykh A.A.A., Srirama, S., Jakovits, P., Smialek, M., Nowakowski, W., & Chauvel, F. (2012). Deliverable D4.1 REMICS PIM4Cloud.

[2] Mell,P., & Grance, T. (2009). The NIST definition of cloud computing. *National Institute of Standards and Technology,* 53(6), 50.

[3] Bergmayr A., Wimmer M. (2013). ARTIST Deliverable D9.2 Modeling Language and Editor for Defining Target Specifications.

[4] Bergmayr A., Wimmer M., Grossniklaus M. (2013). ARTIST Deliverable D9.1 State of the art in modelling languages and model transformation techniques.

[5] Brandtzæg, E. (2012). CloudML: A DSL for model-based realization of applications in the cloud.