PLATFORM-AS-A-SERVICE (PAAS): THE ADOXX METAMODELLING PLATFORM

Dimitris Karagiannis and Niksa Visic

University of Vienna, Knowledge Engineering Research Group, Brünnerstr. 72, A-1210 Vienna, Austria
{dk, nv}@dke.univie.ac.at

Abstract. This paper researches the synergies between metamodelling platforms and cloud computing paradigm. In particular, a classification of services that belong to different levels of abstraction, or cloud layers (SaaS, PaaS), is given, using ADO*xx* – complex, distributable, scalable and component-based metamodelling platform – as an example. Furthermore, the feasibility of porting desktop metamodelling platforms to the cloud is explored, including the possible advantages and disadvantages the cloud may provide.

Keywords: PaaS, SaaS, ADO*xx*, Metamodelling, Platform

1 Introduction

Metamodelling approaches are an active research field and in the past 20 years serious application areas in the software and information technology industries have been found. It is only logical to assume that metamodelling approaches will expand to other application areas, not only in software and information system engineering, but also to other disciplines in- and outside of computer science. To support the rapid expansion and popularity of metamodelling approaches we should view existing modelling and metamodelling software as legacy software that needs to be evolved by taking advantage of the benefits cloud computing has to offer.

The basic explanation about the notions *modelling method* and *cloud computing* follows.

Modelling Method: The Concept

A modelling method [1] consists of two components: a modelling technique, which is divided in a modelling language and a modelling procedure, and mechanisms & algorithms working on the models described by the modelling language (see Figure 1). The modelling language contains the elements with which a model can be described: *syntax*, *semantics* and *notation*. The modelling procedure describes the steps applying the modelling language to create results, i.e., models. Algorithms and mechanisms provide *"functionality to use and evaluate"* models described by a modelling language. When such functionalities, enabling structural analysis as well as simulation of models are defined for existing modelling technique, the modelling methods are formed [2].

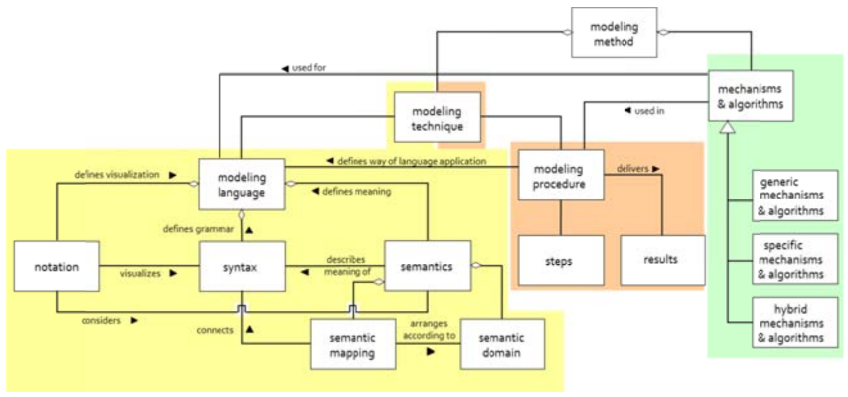

Figure 1. Modelling methods, mechanisms and algorithms (Karagiannis & Kühn, 2002)

To support this concept, metamodelling platform should be realized on a component-based, distributable, and scalable architecture [1]. This kind of architecture also has advantages when transporting the platform into the cloud. Additionally, one of the most important elements of the metamodelling platforms, the *meta-metamodel*, needs to contain all the general concepts (*metamodel*, *classes*, *relations*, *attributes*, *model types*, etc.) for method definition and method application.

The Cloud: Layers and Services

Cloud computing has been coined as an umbrella term to describe a category of sophisticated on-demand computing services initially offered by commercial providers, such as Amazon, Google, and Microsoft. It denotes a model on which a computing infrastructure is viewed as a *"cloud"*, from which businesses and individuals access applications from anywhere in the world on demand [3].

The *cloud* is a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load, allowing optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guaranties are offered by the infrastructure provider by means of customized SLAs (Service Level Agreements) [4].

Cloud computing services are divided into three classes, according to the abstraction level of the capability provided and the service model of providers, namely: (i) Infrastructure-as-a-Service (IaaS), (ii) Platform-as-a-Service (PaaS), and (iii) Software-as-a-Service (SaaS) [5]. These abstraction levels can be viewed as layered architecture where services of higher layer can be composed from services of the underlying layer.

The *Infrastructure-as-a-Service* (IaaS) layer provides low-level, virtualized resources, such as storage, networks, and other fundamental computing resources via self-service to the user. In general, the user can deploy and run arbitrary software, which usually includes operating systems as well as applications [6].

The *Platform-as-a-Service* (PaaS) layer provides capability to deploy custom applications on top of the cloud’s infrastructure. These applications are deployed using the programming languages, development tools and APIs defined by the cloud provider. The process of implementing and deploying a cloud application becomes more accessible and simplified by removing the need to manage the underlying software and hardware infrastructure (including complex programming details, scalability, load balancing, etc.) and allowing the developer to focus on important issues [6].

The Software-as-a-Service (SaaS) layer is the highest layer in the proposed cloud model [5]. SaaS layer provides ready-to-run services that are deployed and configured for the user. All data manipulated in such systems is held in the cloud. One of the most prominent advantages of SaaS layer applications is universal accessibility regardless of the client system’s software availability [6].

2 Related Work

The notion of exploring synergies between metamodelling approaches and cloud computing is still very young. One of the first mentions of Method-as-a-Service paradigm appears in a keynote speech from Rolland C. [7], where he proposes to adopt a service-based paradigm analog to

SaaS. The aim was in developing method engineering approach driven by the needs of method clients, whereas implementation details of method services should remain under the control of method providers. The implementation is further discussed in [8].

There is also an ongoing research conducted by AtlanMod[1] team [9] in which they introduce the notion of Modelling-as-a-Service as a way to provide modelling and model-driven engineering services from the cloud. A similar concept, or rather a small part of it, is also present in the industry (The Enterprise Architect[2] blog), called Model-Execution-as-a-Service, where the ultimate goal is to support the agile application lifecycle, from a first idea to a working application, and from a working application to long-term business agility (i.e., the evolution of an application along with the business) trough simple and fast model-driven development & deployment in the cloud.

Research in these topics is still in a preliminary phase, resulting in lack of related scientific literature and concrete results. The first web-based modelling tools (sometimes also marketed as diagram drawing software) have started to emerge during the last few years, such as Gliffy[3], Cacoo[4], Creately[5], Diagramly[6], LucidChart[7], etc. Most recently, there is a trend of cloud-enabling present in the industry, especially with BPM (Business Process Management) and office software solutions, where desktop applications are transferred into the cloud and offered as a service for a recurring subscription fee (The Business Software Centre[8]). This model of cloud deployment is also known as "*SaaS deployment without re-development*"[9]. In most of the cases ported applications appear to run as if locally installed on a client computer.

Searching for the web-based metamodelling tools did not bring many results. One of the most prominent contributions is GEMSjax [10], a web-based metamodelling tool for collaborative development of domain specific languages. By employing modern Web 2.0 technologies (Ajax and REST services), it allows simultaneous web browser-based creation and modification of metamodels and model instances, as well as remote model access over a simple web-based interface.

[1] http://www.emn.fr/z-info/atlanmod/index.php/Main_Page
[2] http://www.theenterprisearchitect.eu/
[3] http://www.gliffy.com/
[4] https://cacoo.com/
[5] http://creately.com/
[6] http://www.diagram.ly/
[7] http://www.lucidchart.com/
[8] http://www.businesssoftwarecentre.com/
[9] http://www.realtechsolutions.co.uk/

3 Integrating ADO*xx* with the Cloud and Cloudlike Infrastructure

ADO*xx* [2] is an extensible, repository-based metamodelling platform, which offers a three-step modelling hierarchy with a rich meta-metamodel. ADO*xx* can be customized using metamodelling techniques and extended with custom components to build a modelling environment for a particular application domain. The ADO*xx* platform kernel provides basic modules for managing models and metamodels. In addition, the ADO*xx* generic components for graphical and tabular model editing, for model analysis, for simulation, or for model comparison can be reused and customized in all solutions derived from ADO*xx*. Each ADO*xx*-based solution contains a solution-specific modelling language and may have additional set of solution specific components.

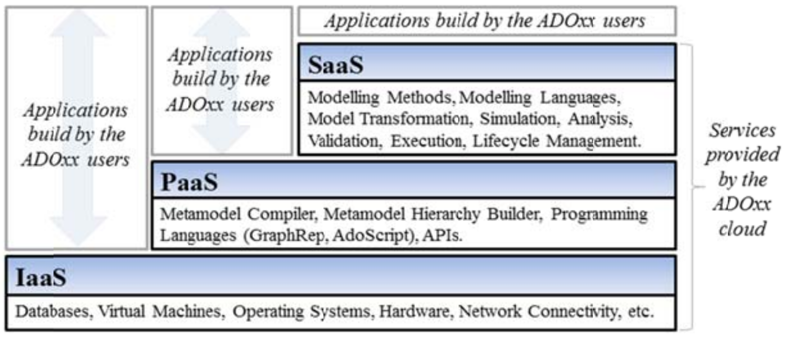
Applications build by the ADOxx users
Applications build by the ADOxx users
Applications build by the ADOxx users
SaaS
Modelling Methods, Modelling Languages, Model Transformation, Simulation, Analysis, Validation, Execution, Lifecycle Management.
PaaS
Metamodel Compiler, Metamodel Hierarchy Builder, Programming Languages (GraphRep, AdoScript), APIs.
IaaS
Databases, Virtual Machines, Operating Systems, Hardware, Network Connectivity, etc.
Services provided by the ADOxx cloud

Figure 2. Overview of ADO*xx* PaaS and SaaS Level Services

Because of the architecture with three levels of abstraction (meta-metamodel, metamodel, model), ADO*xx* can be seamlessly integrated into the cloud, where metamodelling and modelling can be provided as a service – metamodelling services on PaaS and modelling solutions, including modelling methods, on SaaS level, introducing a new cloud computing notions: Modelling-as-a-Service and Method-as-a-Service. On PaaS level ADO*xx* contains a set of specialized components and frameworks for defining a modelling language (metamodel, domain specific modelling language) and modelling method without or very little hand coding: dialog-based metamodel hierarchy builder/explorer for defining abstract syntax and semantics of a modelling language, special programming language called *GraphRep* for defining notation (visual or graphical syntax), and scripting language called *AdoScript* that provides mechanisms to define specific behavior and functionality of a modelling method. ADO*xx* also has a variety of APIs that can be used and extended to support additional functionality and services on PaaS (metamodel editing, method editing, method integration, method extension, etc.), as well as on SaaS level (model transformation, model editing, model simulation, model analysis, model lifecycle management, model validation, model execution, etc.). For further details see Figure 2.

One of the prerequisites of cloud computing, especially on SaaS level, is that services are accessible from anywhere on any machine without worrying about software requirements on the client side. To improve ADO*xx* platform integration with the cloud, which means switching from cloudlike infrastructure to real cloud infrastructure, new web-based interface development is a necessity. ADO*xx* is using remote desktop technology limited to Windows operating systems only. By switching to web-based access to the platform, employing only a web browser, full cloud integration can be achieved.

4 Conclusion

Transferring any software from *“desktop”* form to *“as-a-service”* form is hard work. A new growing branch in the IT industry dedicated only to consulting enterprises regarding cloud migration and offering help with application porting to the cloud is proof enough to grasp the complexity of this process. Metamodelling platforms, which are in most cases very complex software systems composed from multiple interconnected components, are no exception. By reverse engineering and dividing metamodelling platforms into small specialized components, each dedicated to a relatively independent task that can be offered as a service, and arranging those in appropriate cloud computing layers (PaaS, SaaS) the porting process can be simplified. Regardless, by using this procedure it is generally only possible to make cloud-like applications that will in most cases perform inferiorly as opposed to applications engineered from scratch with cloud architecture in mind (true cloud applications). Extensions (new web-based GUIs, etc.) and modifications (communication between internal components, data management, etc.) to the desktop applications are sometimes necessary if we wish to use all the advantages that cloud has to offer.

Method-, Modelling-, and Metamodelling-as-a-Service are still very young research topics, which makes them even more interesting, but – because of their pioneering status – also risky to tackle with. Serviceorientation and model-driven engineering, as outlined in [9], are two of the most dominant software engineering paradigms, followed by metamodelling paradigms, including language engineering and method engineering. It can be concluded with certainty that the need for web-based modelling and metamodelling solutions will only grow, which will also initiate even bigger research interest in these topics.

References

[1] H. Kühn and D. Karagiannis, “Metamodelling Platforms,” *Lecture Notes in Computer Science*, vol. 2455, no. 2455, pp. 182–182, 2002.

[2] D. Karagiannis and N. Visic, “Next Generation of Modelling Platforms,” in *Lecture Notes in Business Information Processing, BIR 2011*, Riga, Latvia, 2011, vol. 90, pp. 19-28, (*In Press*).

[3] W. Voorsluys, J. Broberg, and R. Buyya, “Introduction to Cloud Computing,” in *Cloud Computing: Principles and Paradigms*, R. Buyya, J. Broberg, and A. Goscinski, Eds. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011, pp. 1-41.

[4] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, Dec. 2008.

[5] T. Grance and P. Mell, “The NIST Definition of Cloud Computing,” *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.

[6] S. Jha, D. S. Katz, A. Luckow, A. Merzky, and K. Stamou, “Understanding Scientific Applications for Cloud Environments,” in *Cloud Computing: Principles and Paradigms*, R. Buyya, J. Broberg, and A. Goscinski, Eds. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011, pp. 345-371.

[7] C. Rolland, “Method Engineering: Towards Methods as Services,” in *Making Globally Distributed Software Development a Success Story*, vol. 5007, Q. Wang, D. Pfahl, and D. M. Raffo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 10-11.

[8] A. Iacovelli, C. Souveyet, and C. Rolland, “Method as a Service (MaaS),” in *Second International Conference on Research Challenges in Information Science, 2008. RCIS 2008*, 2008, pp. 371-380.

[9] H. Brunelière, J. Cabot, and F. Jouault, “Combining Model-Driven Engineering and Cloud Computing,” 15-Jun-2010. [Online]. Available: http://hal.inria.fr/hal-00539168_v1/. [Accessed: 17-Aug-2011].

[10] M. Farwick, B. Agreiter, J. White, S. Forster, N. Lanzanasto, and R. Breu, “A Web-Based Collaborative Metamodeling Environment with Secure Remote Model Access,” in *Web Engineering*, vol. 6189, B. Benatallah, F. Casati, G. Kappel, and G. Rossi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 278-291.