Modeling as a Service: A Survey of Existing Tools

Saheed Popoola Jeffrey Carver Jeff Gray
Department of Computer Science
University of Alabama
sopopoola@crimson.ua.edu, {carver,gray}@cs.ua.edu

***Abstract*—Modeling tools are needed to deliver the promises of Model-Driven Engineering, which include reduced development time and enhanced software quality. However, users must typically install these tools locally. The tools often have complex configurations and inter-dependency requirements that discourage non-technical users or novices from adopting such tools. The local installation also hampers collaborative modeling and reuse of modeling artifacts. A solution to these challenges is to deliver modeling functionality as a service. In this paper, we present a survey of current tools that deliver modeling functionality as service. We analyzed various approaches used to develop existing tools and the functionalities exhibited by them. The results of our review show that support for collaboration and domain-specific modeling are the dominant features exhibited by the tools, but collaboration is the major feature that drives tool adoption. The paper concludes by proposing future research directions that can facilitate the wider adoption of modeling as a service.**

I. Introduction

Model-Driven Engineering (MDE) is an approach to software engineering whereby models are used as first-class artifacts throughout the stages of software development. This elevation of models to first-class entities makes it easy to focus on the problem space rather than the underlying computing environment, thereby reducing development time. MDE also enhances code reusability by providing support for capabilities such as automated code generation and model transformations.

In order to achieve the benefits of MDE, tools are needed for manipulating models. Many frameworks and tools such as ATL [33], Epsilon [35] and GEMOC [3], have been developed to support a wide range of model management activities. However, MDE is yet to be widely adopted due to two major reasons: First, most modeling tools are usually deployed as software packages that need to be installed locally. These tools often have complex configurations and inter-dependency requirements that may discourage a novice from adopting such tools [44], [54]. Second, the support for reuse of existing modeling artifacts and interoperability among existing tools is limited; hence, developers usually build similar modeling tools from scratch [39]. A potential solution toward tackling these challenges is to develop modeling tools and frameworks to follow the software-as-a-service paradigm. A service-oriented modeling platform can offer a more transparent solution to the reusablity and interoperability challenges by defining bridges across multiple platforms as services that can be executed on demand [10].

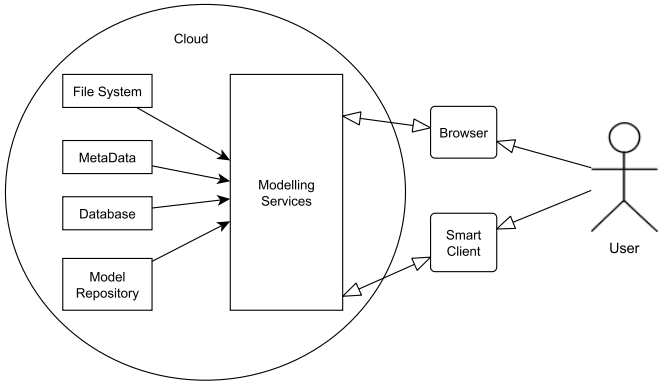

Fig. 1. A Typical MaaS Platform

The goal of this paper is to present an analysis of current tools from the literature that offer modeling (within MDE) as a service (MaaS). Figure 1 shows a typical modeling platform that offers its functionalities as a service. In this study, the aim is to answer the following research questions with respect to modeling tools that offer their functionalities as a service:

RQ1: What are the common features and functionalities exhibited by the tools?

RQ2: How important are these features towards the adoption of the tools?

RQ3: What future research is needed to facilitate the adoption of the tools?

II. Background

This section presents a brief introduction to related terms that are used throughout this paper.

A. Cloud computing

The National Institute of Standards and Technology (NIST) defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [43]. Cloud computing is projected to be a disruptive technology to the IT industry, and numerous surveys show an increasing adoption of the cloud, including the migration of numerous software to the cloud [9], [12].

A lot of studies have been conducted as exemplified by the multiple literature reviews in the field [4], [8], [24], [46].

Currently, there are many cloud platforms such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure, that offer many services ranging from online storage to operational software applications. These services can be grouped into three main categories [4]:

- Infrastructure as a Service (IaaS): This involves offering hardware-related components as services, e.g., virtual storage services. A good example is *Amazon S3*[1].
- Platform as a Service (PaaS): This involves offering development platforms, where cloud applications can run as a service. An example is *Microsoft Azure*[2].
- Software as a Service (SaaS): This offers a complete, ready-to-use software application as a service. This is the most common category of cloud services and examples of SaaS applications include Gmail, Google docs, and Eclipse Che [1].

Further explanations on the SaaS category is provided in the next section because it is the most relevant to this work.

B. Software as a Service

"Software as a Service (SaaS)" is an approach whereby software is stored remotely (e.g., on the cloud) and its capabilities are accessed via a set of APIs (i.e., as services). The underlying infrastructure, platform and software requirements/details are hidden from the users. Recently, there has been a gradual migration of software platforms and tools to the cloud. Several cloud-based alternatives for traditional desktop-based software tools are now available (for example, [1], [2]). The benefits of the software-as-a-service paradigm include efficient cost policies, minimal setup configurations, and reduced maintenance efforts. SaaS also enhances reuse of software artifacts because they are now stored on the cloud and may be accessed publicly via the internet.

III. Related Work

A number of surveys and literature reviews have been conducted in the area of cloud computing, software as a service and software modeling. Ramollari et al. conducted a survey on methodologies that are used in service-oriented computing [47]. They compared some existing methodologies against prominent features service-oriented software are expected to have. Patidar et al. and Ahmed et al. classified current technologies and tools based on how they are used [4], [46]. Zhou et al. focused on the different services that are available on the cloud [55], while Gray et al. gave a brief overview of current MaaS tools [27]. The study most similar to the one conducted in this paper was the survey on MaaS architectures by Cayirci et al. [13]; however, they focus more on the possible risks and challenges associated with the deployment of modeling operations as a service. None of these works have conducted an extensive review of the existing tools and technologies that deliver software modeling functionalities as a service.

[1]https://aws.amazon.com/s3/

[2]https://azure.microsoft.com

IV. Research Methodology

For this study, we used four research databases — Google Scholar, Scopus, SpringerLink, and IEEE Explore – to extract relevant papers and tools on modeling as a service. We used the key words — ("software modeling" OR "modeling") AND ("services" OR "cloud") — for the search. By manually reading through the paper (or abstract in some cases), we were able to identify relevant papers for this study. We selected papers that deal with how modeling tools or activities can be delivered as a service and we eliminated others (e.g., those dealing with how to model services on the cloud). We then searched through the references of the identified papers to extract additional papers that we might have missed. Due to the number of papers available, we decided to focus on tools that support at least one of the core MDE functionalities, such as code generation from models and/or automated transformation of models from one form to another.

To answer the first research question on the features exhibited by selected tools, we examined the relevant features mentioned by the paper that describes a tool, and other features affirmed by other papers that referenced the tool. For the second and third research questions on the features contribution to a tools adoption and future research areas, we answered these questions based on the information we were able to extract from the papers cited in Sections VIII and IX.

V. Approaches to Deploying Modeling as a Service

Two main approaches were observed in deploying modeling as a service. They are client-server and cloud approaches [5].

- Client-Server: In this approach, the modeling platform is installed on a local server and its capabilities are accessed by the clients connected to the server. The number of users is limited to the clients connected to the server and this approach still requires a form of installation on the local server. However, it allows for flexibility and easy customization [29]. This approach was implemented in AToMPM [15], ModelBus [29], and DSLforge [40].
- Cloud: This approach fully explores the benefits of modeling as a service. It is accessed on the web via the internet and does not require any form of installation. Its functionalities can be programmatically accessed as a service via a set of APIs or through a web browser [5]. This is the most common approach and it was used in GenMyModel [23], WebGME [42], CLOOCA [31], MORSE [32] and MDEForge [5].

VI. modeling as a Service (MaaS) tools

This section gives a brief overview of selected software modeling frameworks that deliver modeling functionalities (within MDE) as a service.

- GenMyModel is an online modeling platform that currently supports 7 modeling formats including UML, Flowcharts, and Ecore. It also offers capabilities for customized formats. The main focus is on collaboration and the ability to work simultaneously with many developers.

Hence, it provides mechanisms for conflict management and change awareness. It also provides a public repository of models to facilitate searching and reuse of public projects. A flexible pricing model is adopted where users can access the service for free if the project is public, but monthly subscriptions are required for private projects. However, GenMyModel limits the user to their own UML tools and a limited set of modeling languages [16], [23].

- WebGME is a scalable web and cloud based tool for collaborative design and synthesis of domain-specific modeling tools and associated models. The main target of the tool is large-scale complex information systems. Its capabilities can be accessed via a web-based client or a set of APIs. It supports model version control and object-oriented concepts such as composition and inheritance. The tool supports model versioning, inheritance and composition by creating a copy of the model (as a child) and establishing dependency links between the parent and child copy so that changes are easily propagated. This duplication of models makes the framework cumbersome, but it also helps the modeller to handle the inherent complexities of large models [34], [42].
- CLOOCA is a platform for developing domain-specific modeling languages and the necessary code generators. The tool adopts a client-server architecture whereby the models are stored on the server and clients can access the models via a JavaScript-enabled web browser. The client side consists of two main parts — an editor and a workbench. The workbench allows for diagrammatical creation of modeling languages and their code generators, while the editors make it easy to modify an existing language [31].
- AToMPM is a collaborative platform that provides a multi-view mechanism for multiple users that are working simultaneously on the same models. It supports the design of modeling language environments, model transformations and general model management. The tool's architecture consists of a front-end server, back-end server and the clients. The front-end server receives the clients' requests, and then forwards the request to the appropriate destination in the back-end server. The front-end server is used as a scalable routing system for managing incoming messages so that the back-end server only sees a single client irrespective of the number of users. This 2-server architecture ensures consistency and efficient change propagation [15].
- Morse framework provides a central model repository environment for managing and storing modeling artifacts. Each model and model element in the repository is assigned a unique identifier. The capabilities of these models can be accessed via a set of services that contain references to the model that created them, which makes the services aware of the models. The framework links each service with its related models via the models' identifiers. The repository supports versioning, merging, branching, tagging and sharing of modeling artifacts across multiple projects. The framework also supports heterogeneous platforms by abstracting the underlying technologies, thereby making it easy to focus on the projects at runtime [32].
- MDEForge is a modular and extensible model repository that can be used to store diverse modeling artifacts. Four main services are provided by the tool; namely, model, metamodel, transformation and editor services. These services are also easy to extend or customize [5].
- Modelbus is a framework that extends web-service interfaces to provide modeling services. Its main goal is to add model awareness to service-oriented systems. It contains a model repository, which is the central point of the framework. A user can reference or manipulate the models that are stored in the repository via each model's unique URL. The framework uses a notification system to propagate changes across the models. The framework also supports model versioning, model merging, and distributed model management operations, in a scalable and consistent manner [29].
- DSLforge is a framework that can be used to generate online textual models (for example, language grammars) and domain-specific languages. It is useful for producing extensible online editors that can be used to create transformations for diverse models [40].

TABLE I
OVERVIEW OF CURRENT MAAS TOOLS

Tool	Approach	Persistence	Extensible	I/A [1]	G/T [2]
GenMyModel	Cloud	XML	No	I	G
WebGME	Cloud	COM	Yes	Both	Both
CLOOCA	Cloud	XML	?	A	T
AToMPM	Client-Server	XML	Yes	A	Both
MORSE	Cloud	XML	Yes	I	Both
MDEForge	Cloud	REST	Yes	A	Both
ModelBus	Client-Server	Adapter Based	Yes	I	Both
DSLforge	Client-Server	XML	Yes	A	T

Table I gives a summary of these tools, the format in which the models are stored (persistence) and whether they can be easily extended. It is important that models should be stored (persisted) in a way that makes it easy to execute modeling operations on the models and also makes the tool compatible to a wide range of modeling formats [38]. A good modeling framework should also be extensible. The framework should contain a set of reusable services that are abstract enough, so that it becomes easy to extend the framework's capabilities and adapt the framework to a wide range of functionalities [45].

[1]Industrial (I) or Academic (A)
[2]Graphical (G) or Textual (T)

TABLE II
FEATURES IN EXISTING MAAS TOOLS

Tools\Functionalities	Multi-View	Collaboration	Code Generation	Versioning	DSML	Merging	Transformation
GenMyModel	Yes	Yes	Yes	Yes	No	No	No
WebGME	Yes	Yes	No	Yes	Yes	No	Yes
CLOOCA	No	No	Yes	No	Yes	No	No
AToMPM	Yes	Yes	Yes	No	Yes	No	Yes
Morse	No	Yes	No	Yes	Yes	Yes	No
MDEForge	No	Yes	Yes	No	Yes	No	Yes
ModelBus	No	Yes	No	Yes	Yes	Yes	No
DSLforge	No	Yes	Yes	No	Yes	No	Yes

TABLE III
FEATURES PRESENT IN MODELING PLATFORMS

Features	Number of Tools
Collaboration	6
DSML	6
Code Generation	4
Versioning	4
Transformation	3
Multi-View	3
Merging	2

VII. FEATURES IN EXISTING MAAS TOOLS

This section answers the first research question: "Which Features are present in existing tools"? Section VI showed that current tools offer diverse functionalities. We have synthesized these functionalities to extract the most common features that are related to modeling or service-oriented computing. Eight main features were exhibited by many of the tools and they are listed throughout this section.

Multi-View Based modeling: Since modeling is usually done at different levels of abstractions depending on its intended purpose, it is essential that a modeling platform should be able to support multiple views of the same system at different levels of abstraction [15], [20]. This feature is present in AToMPM, GenMyModel, and WebGME while other tools do not support this feature.

Collaboration: This allows teams of technical and non-technical stakeholders to work together and brainstorm on possible solutions to the problem space [17], [20], [52]. Traditional modeling tools inhibit this functionality because they are typically desktop based; hence, this feature is a good validation of delivering modeling capabilities as a service. All the tools under study except CLOOCA provide support for collaboration. Furthermore, many of the tools also support real-time collaboration in varying degrees of granularity. Tools like GenMyModel and AToMPM have a very high level of granularity and ensure that only the model element that is being edited is locked, while other parts of the same model can be edited concurrently [15].

Code Generation: This is the ability to automatically generate code from models. This helps to bridge the gap between modeling and programming, and makes it easy to programmatically manipulate models and other modeling artefacts [30]. GenMyModel, AToMPM, and MDEForge, offer built-in code generators and they also support customized generators, while CLOOCA and Morse only support built-in code generators. Other tools do not support this feature.

Model Versioning and Model Merging: Model versioning makes it easy to store and relate different versions of the same model, while model merging is the combination of two or more models with similar or different structures [32]. The implementation of these features is inherently complex due to the high level of inter-connection among elements in a model [34]. Therefore, only four tools support either model versioning or model merging.

Domain-Specific modeling Language: This is the ability to support the development of new modeling languages [42]. This feature will make it easy to adapt the framework for different purposes [18]. This is one of the most common feature exhibited by the tools and it is supported by all of the tools except GenMyModel.

Model-to-Model Transformation: This is the ability to transform a model from one form to another and it is one of the most common activity in MDE [49]. A viable tool for MDE should support this functionality. However, more than half of the tools under study do not support model-to-model transformation.

Heterogeneity and Tools Interoperability: For a wider adoption, it is necessary that a tool should be able to support a wide range of modeling artifacts that are stored in diverse formats [45]. Support for heterogeneity may be achieved by storing modeling artifacts in a central repository and allowing access to them using standardized APIs [29]. It is also important that the tools should be easy to integrate with other tools [11] (for example, backup data to Google drive).

Scalability: This is the ability to support large models (for example, models with millions of elements) without a significant impact on the tool's performance. This feature is important in determining whether a tool will be useful in an industrial context and it has been one of the major challenges associated with modeling in general [37], [38]. The computing power available in the cloud is expected to mitigate the challenges of scalability in modeling [10]. Although minimal research has been done to assess the performance of current tools in handling large models, scalability was mentioned as an important feature in GenMyModel [23], Modelbus [29], DSLforge [40], and WebGME [42].

Table II gives a summary of the the functionalities offered by each tool, while Table III shows the number of tools that support each feature. It can be observed that collaboration and

domain-specific modeling languages have the most support across the frameworks.

VIII. Features that Contribute Significantly to the Adoption of MaaS Tools

This section answers the second research question: "How relevant are these features towards the adoption of current tools"? Section VII gave a concise list of features that are present in existing tools. However, we do not know the contribution of these features to the adoption of the tools or if they are even necessary. A lot of features might make a tool cumbersome while lack of essential features is likely to affect its usability [25].

Limited statistical data is available on the number of users adopting the tools. Hence, there is little evidence to measure the impact of each feature on the adoption of the tools. However, three features that seem to have a significant effect on the adoption of these tools are highlighted below.

Collaboration: This is likely to be the most important feature driving the adoption of service-oriented modeling tools. Software development has become a collaborative activity due to the increasing complexity of software and software modeling is no exception [52]. Dirix et al. reported that it was the most demanded feature during the development stage of GenMyModel [23]. We also noticed that most of the commercial modeling tools, including those that are not included in this paper such as LucidChart[3], and creately[4], support collaboration [21], [52].

Tool Integration: Developers and users of the tools store models in different formats, and often deal with diverse frameworks and technologies. Therefore, the users often need to reuse data from an external tool [45]; hence, one of the main factors for adoption is a tool's ability to integrate seamlessly with other tools, for example, support for data stored in Microsoft Excel or Google sheets, and integration with versioning systems such as GitHub [11].

Scalability: A common theme among the commercial tools is the ability to support large models with thousands of model element [23]. The prevailing support for scalability across commercial tools is a strong sign that the feature is important for the adoption of these tools. The importance of this feature is corroborated by the fact that the need for scalability is well-established even in traditional desktop-based modeling tools [10], [37].

In summary, even though there is limited statistical data to measure the contribution of each feature to a tool's adoption rate; support for collaboration, scalability, and seamless integration with other tools, are essential to the success of a tool.

IX. Additional Features That Can Enhance the Adoption of MaaS Tools

In order to answer the third research question that deals with future research directions, we compared the functionalities exhibited by current tools with essential features expected from a modeling or cloud-based tool [45], [55].

Support for Sketches: Most tools support modeling using a standard modeling language such as UML. However, a number of works have highlighted the importance of sketches during a brainstorming section, and during the design and analysis phase of software development [14], [28], [41]. Since modeling also takes place during the design and analysis phase, tool support for sketching or handwriting models is likely to increase the adoption of a tool. Automated model management operations such as code generation may also be realized from these sketches.

Security and privacy: A major concern for any cloud-based platform is security and privacy [56]. It is necessary to design appropriate means of preventing unauthorized access. The privacy of the users' data and proper attribution of a modeling artifact to its owner, is vital to boosting user's confidence in adopting a tool [55].

Model persistence: Most modeling frameworks store models in XMI. While XMI is a good standard for model persistence because it is able to support a wide range of formats, it is not implemented consistently across different platforms; thereby, making it hard to share models across different vendors [53]. Furthermore, XMI is also verbose and may impact performance negatively [7].

Opportunities for Reverse Engineering: The advent of large open-source projects and public repositories have increased the importance of reverse engineering [48]. A lot of MDE-based frameworks have been used to extract structural and behavioural information from source code. These frameworks usually produce visual models to aid code comprehension and gain useful insights to decisions made during the design of the software [10]. A potential usefulness of MaaS is to develop capabilities to scan through online repositories and generate appropriate models without having to download the source code locally on a machine.

Incentives to share models: A major advantage of delivering modeling capabilities as a service is the opportunity to share and reuse modeling artifacts. However, it is necessary to encourage developers to publicly share their models online. A common paradigm is to offer free modeling services for developers with publicly available models [23]. However, more work is needed in terms of copyright and proper attribution of modeling artifacts to the original developer(s) [19].

Collaboration as a social activity: Collaborative software development is a social activity where developers need to communicate, send emails, and share ideas [6]. Hence, the addition of social features such as group messaging, will likely enhance the adoption of these tools [6], [50].

Pricing model: Much has not been done on viable pricing models for MaaS. Yet, a viable source of revenue is important to ensure sustainability of the platforms due to recurrent costs such as hosting and platform maintenance fees. These costs are usually not incurred in traditional desktop-based tools, because the user directly bears the cost for the underlying computing infrastructure and maintenance. However, care should be taken

[3]https://www.lucidchart.com/

[4]https://www.creately.com/

to ensure that the pricing model is not a barrier to users' adoption of the tool [55].

X. Further Observations on Existing Tool Features

A. Diverse functionalities

Our review of the MaaS tools shows that most of the tools offer varied functionalities in different ways. However, the features and functionalities discussed in Section VII tends to be more common among the tools, even though they exhibit the features in different ways. For example, GenMyModel offers a more granular form of collaboration with support for conflict management than most other tools [19]. It may also be preferable to develop frameworks that other developers or users can extend easily to suit a user's preference [45].

B. Collaboration is necessary for adoption

Section VIII shows that all the commercial platforms support collaborative modeling and it was the most requested feature by users during the development of GenMyModel [22]. This widespread support for collaboration highlights the need for collaborative features in motivating users towards adopting a tool. Many modeling technologies usually lock a model when another member of the team is editing the model, thereby making it impossible for other users to edit the same model at the same time, though they can view the changes in real-time [15], [23]. This locking mechanism is necessary for consistency management and efficient change propagation among the users [42]. However, techniques to improve granularity, which allows two people to edit different parts of the same model at the same time, is needed to enhance the collaboration experience.

C. Scalability remains a holy grail

The commercial success of a modeling tool tends to depend on its performance when handling large models (for example, models with hundreds of thousands of elements) [19]. Efficient search and query mechanisms are needed to search for large modeling artifacts and execute model management programs, such as code generation, in a scalable manner [36]. It is necessary that developers of modeling platforms should take advantage of the computing power that is available in the cloud (e.g., via distributed query techniques) to deliver a scalable framework that can handle large models without a significant impact on performance [51].

D. Centralized repository is mostly used for collaboration and tool heterogeneity

All the selected frameworks for this study that support collaboration and/or heterogeneity, store models in a central repository for consistency and easy management [23], [29], [32], [42]. However, the use of a central repository may expose the tool to a single point of failure or inefficient performance due to lack of distributed capabilities.

E. Empirical studies on tool features

Statistical data to measure the rate of a tool's adoption, or determine the contribution of each feature to the tool's success, is not available. Hence, the most significant feature that is driving the adoption a tool could not be determined. Empirical studies on tool usability is needed to determine the main features that drive a tool's adoption.

F. More features and functionalities are needed to enhance adoption

Although current tools exhibit a lot of features, Section VII shows that more features are needed to facilitate the adoption of MaaS tools. However, it should be noted that a tool should not contain too many features, since this may affect the tool's usability [26]. A good approach is to build tools that other developers can extend easily to suit user demand [45].

XI. Research Limitations

A major limitation of this work is the exclusion of modeling tools that do not offer MDE capabilities, such as code generation and model transformation. Furthermore, there may be a significant level of bias in the gathering, sorting and analysis of the tools and techniques listed in this paper. This is because these activities involved much manual input and speculations.

A second limitation is that the paper focuses on the presence (or absence) of some features in the tools, and a detailed exploration of the level of support provided for the features is not considered. Therefore, it may be possible that while a tool may support collaboration, such support may be limited or cumbersome. However, we do believe that this study is a good initial step towards a comprehensive survey of MaaS tools.

XII. Conclusion

This paper has highlighted some of the important features necessary for the adoption of a modeling tool that deliver its functionalities delivered as a service. This knowledge is useful for developers and researchers, in order to know the essential features to focus on. The first part of this paper reviewed different techniques for delivering modeling as a service and the state-of-the-art tools that offer modeling services. The findings show many differences in the functionalities offered by these tools. The study also discovered that collaboration is necessary for facilitating the adoption of these tools. Furthermore, the study revealed that important features such as pricing and security of these platforms were rarely discussed in the literature.

We have also identified some future research directions that can lead to the full realization of tools that deliver modeling functionalities as a service and facilitate the adoption of such tools. We consider work on efficient model persistence format and support for sketching as viable research directions.

In the future, we plan to carry out a comprehensive systematic review of the tools and to validate (or invalidate) the conclusions in this study, especially with regards to the second and third research questions.

REFERENCES

[1] Eclipse Che, a developer workspace server and cloud IDE. http://www.eclipse.org/che/.

[2] Eclipse Orion, a modern open source software development environment that runs in the cloud. https://orionhub.org/.

[3] GEMOC studio. www.gemoc.org/studio, 2017.

[4] M. Ahmed, A. Chowdhury, M. Ahmed, and M. M. H. Rafee. An advanced survey on cloud computing and state-of-the-art research issues. *IJCSI International Journal of Computer Science Issues*, 9(1):1694–0814, 2012.

[5] F. Basciani, J. Di Rocco, D. Di Ruscio, A. Di Salle, L. Iovino, and A. Pierantonio. MDEForge: an extensible web-based modeling platform. In *CloudMDE@ MoDELS*, pages 66–75, 2014.

[6] A. Begel, R. DeLine, and T. Zimmermann. Social media for software engineering. In *Proceedings of the FSE/SDP workshop on Future of Software Engineering Research*, pages 33–38. ACM, 2010.

[7] A. Benelallam, A. Gómez, M. Tisi, and J. Cabot. Distributed model-to-model transformation with ATL on MapReduce. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering*, pages 37–48, 2015.

[8] Y. Benslimane, M. Plaisent, P. Bernard, and B. Bahli. Key challenges and opportunities in cloud computing and implications on service requirements: Evidence from a systematic literature review. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pages 114–121, 2014.

[9] N. Bridge and G. OM. 2013 future of cloud computing 3rd annual survey results. http://www.northbridge.com/2013-cloud-computing-survey, 2013.

[10] H. Bruneliere, J. Cabot, and F. Jouault. Combining model-driven engineering and cloud computing. In *Modeling, Design, and Analysis for the Service Cloud-MDA4ServiceCloud'10: Workshop's 4th edition (co-located with the 6th European Conference on Modelling Foundations and Applications-ECMFA)*, 2010.

[11] B. Bryant, J.-M. Jézéquel, R. Lämmel, M. Mernik, M. Schindler, F. Steinmann, J.-P. Tolvanen, A. Vallecillo, and M. Völter. Globalized domain specific language engineering. In *Globalizing Domain-Specific Languages*, pages 43–69. Springer International Publishing, 2015.

[12] Capgemini. Business cloud: The state of play shifts rapidly. technical report,. Technical report, Capgemini, 2012.

[13] E. Cayirci. Modeling and simulation as a cloud service: a survey. In *Winter Simulation Conference (WSC)*, pages 389–400, 2013.

[14] M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko. Let's go to the whiteboard: how and why software developers use drawings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 557–566, 2007.

[15] J. Corley and E. Syriani. A cloud architecture for an extensible multi-paradigm modeling environment. In *PSRC@ MoDELs*, pages 6–10, 2014.

[16] R. Crocombe and D. Kolovos. Code generation as a service. In *Proceedings of the 3rd International Workshop on Model-Driven Engineering on and for the Cloud, 18th International Conference on Model Driven Engineering Languages and Systems*, pages 25–30, 2015.

[17] A. R. da Silva. Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139 – 155, 2015.

[18] J. Davis. GME: the generic modeling environment. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 82–83, 2003.

[19] J. Di Rocco, D. Di Ruscio, L. Iovino, and A. Pierantonio. Collaborative repositories in model-driven engineering [software technology]. *IEEE Software*, 32(3):28–34, 2015.

[20] D. Di Ruscio, M. Franzago, H. Muccini, and I. Malavolta. Envisioning the future of collaborative model-driven software engineering. In *Proceedings of the 39th International Conference on Software Engineering Companion*, pages 219–221. IEEE Press, 2017.

[21] M. Dirix. Awareness in computer-supported collaborative modelling. application to genmymodel. In *ECOOP Doctoral Symposium*, 2013.

[22] M. Dirix, X. Le Pallec, and A. Muller. Software support requirements for awareness in collaborative modeling. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 382–399, 2014.

[23] M. Dirix, A. Muller, and V. Aranega. Genmymodel: an online uml case tool. In *ECOOP*, 2013.

[24] P. T. Endo, G. E. Gonçalves, J. Kelner, and D. Sadok. A survey on open-source cloud computing solutions. In *Brazilian Symposium on Computer Networks and Distributed Systems*, volume 71, 2010.

[25] A. Forward and T. C. Lethbridge. The relevance of software documentation, tools and technologies: a survey. In *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 26–33, 2002.

[26] N. C. Goodwin. Functionality and usability. *Communications of the ACM*, 30(3):229–233, 1987.

[27] J. Gray and B. Rumpe. The evolution of model editors: browser- and cloud-based solutions. *Software and Systems Modeling (SoSyM)*, 15(2):303–305, 2016.

[28] J. Grundy and J. Hosking. Supporting generic sketching-based input of diagrams in a domain-specific visual language meta-tool. In *Proceedings of the 29th international conference on Software Engineering*, pages 282–291. IEEE Computer Society, 2007.

[29] C. Hein, T. Ritter, and M. Wagner. Model-driven tool integration with modelbus. In *Workshop Future Trends of Model-Driven Development*, pages 50–52, 2009.

[30] Z. Hemel, L. C. Kats, and E. Visser. Code generation by model transformation. In *International Conference on Theory and Practice of Model Transformations*, pages 183–198, 2008.

[31] S. Hiya, K. Hisazumi, A. Fukuda, and T. Nakanishi. clooca: Web based tool for domain specific modeling. In *Demos/Posters/StudentResearch@ MoDELS*, pages 31–35, 2013.

[32] T. Holmes, U. Zdun, and S. Dustdar. Morse: A model-aware service environment. In *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, pages 470–477, 2009.

[33] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev. ATL: a model transformation tool. *Science of Computer Programming*, 72(1):31–39, 2008.

[34] G. Karsai, M. Maroti, Á. Lédeczi, J. Gray, and J. Sztipanovits. Composition and cloning in modeling and meta-modeling. *IEEE Transactions on Control Systems Technology*, 12(2):263–278, 2004.

[35] D. Kolovos, L. Rose, A. Garca-Domnguez, and R. Paige. The Epsilon Book. http://www.eclipse.org/epsilon/doc/book/, 2017.

[36] D. Kolovos, L. Rose, R. Paige, E. Guerra, J. Cuadrado, J. De Lara, I. Ráth, D. Varró, G. Sunyé, and M. Tisi. MONDO: scalable modelling and model management on the cloud. In *STAF2015 Project Showcase*, 2015.

[37] D. S. Kolovos, R. F. Paige, and F. A. Polack. Scalability: The holy grail of model driven engineering. In *ChaMDE 2008 Workshop Proceedings: International Workshop on Challenges in Model-Driven Software Engineering*, pages 10–14, 2008.

[38] D. S. Kolovos, L. M. Rose, N. Matragkas, R. F. Paige, E. Guerra, J. S. Cuadrado, J. De Lara, I. Ráth, D. Varró, M. Tisi, et al. A research roadmap towards achieving scalability in model driven engineering. In *Proceedings of the Workshop on Scalability in Model Driven Engineering*, pages 2–5, 2013.

[39] I. Kurtev, J. Bézivin, F. Jouault, and P. Valduriez. Model-based dsl frameworks. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 602–616. ACM, 2006.

[40] A. Lajmi, J. Martinez, and T. Ziadi. Dslforge: Textual modeling on the web. In *DEMONSTRATIONS track of the ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems (Models)*, 2014.

[41] N. Mangano, T. D. LaToza, M. Petre, and A. Van der Hoek. How software designers interact with sketches at the whiteboard. *IEEE Transactions on Software Engineering*, 41(2):135–156, 2015.

[42] M. Maróti, T. Kecskés, R. Kereskényi, B. Broll, P. Völgyesi, L. Jurácz, T. Levendovszky, and Á. Lédeczi. Next generation (meta) modeling: Web-and cloud-based collaborative tool infrastructure. In *International Workshop on Multi-Paradigm Modeling@ MoDELS*, pages 41–60, 2014.

[43] P. M. Mell and T. Grance. Sp 800-145. the nist definition of cloud computing. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2011.

[44] P. Mohagheghi, W. Gilani, A. Stefanescu, and M. A. Fernandez. An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. *Empirical Software Engineering*, 18(1):89–116, 2013.

[45] R. F. Paige and D. Varró. Lessons learned from building model-driven development tools. *Software & Systems Modeling*, 11(4):527–539, 2012.

[46] S. Patidar, D. Rane, and P. Jain. A survey paper on cloud computing. In *Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on*, pages 394–398. IEEE, 2012.

[47] E. Ramollari, D. Dranidis, and A. J. Simons. A survey of service oriented development methodologies. In *The 2nd European Young Researchers Workshop on Service Oriented Computing*, volume 75, 2007.

[48] S. Rugaber and K. Stirewalt. Model-driven reverse engineering. *IEEE software*, 21(4):45–53, 2004.

[49] S. Sendall and W. Kozaczynski. Model transformation: The heart and soul of model-driven software development. *IEEE software*, 20(5):42–45, 2003.

[50] M.-A. Storey, C. Treude, A. van Deursen, and L.-T. Cheng. The impact of social media on software engineering practices and tools. In *Proceedings of the FSE/SDP workshop on Future of Software Engineering Research*, pages 359–364. ACM, 2010.

[51] G. Szárnyas, B. Izsó, I. Ráth, D. Harmath, G. Bergmann, and D. Varró. Incquery-d: A distributed incremental model query framework in the cloud. In *International Conference on Model Driven Engineering Languages and Systems*, pages 653–669, 2014.

[52] J.-P. Tolvanen. Metaedit+ for collaborative language engineering and language use (tool demo). In *Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering*, pages 41–45, 2016.

[53] Z. Una, Ieva, N. Oksana, and G. Konstantins. Several issues on the model interchange between model-driven software development tools. In *10th International Conference on Software Engineering Advances (ICSEA)*, 2015.

[54] R. Van Der Straeten, T. Mens, and S. Van Baelen. *Challenges in Model-Driven Software Engineering*, pages 35–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[55] M. Zhou, R. Zhang, D. Zeng, and W. Qian. Services in the cloud computing era: A survey. In *Universal Communication Symposium (IUCS), 2010 4th International*, pages 40–46. IEEE, 2010.

[56] K. Zunnurhain and S. V. Vrbsky. Security in cloud computing. In *Proceedings of the 2011 International Conference on Security & Management*, 2011.