Collaborative Model-Driven Software Engineering: reflections on the past and visions of the future

Mirco Franzago[1], Davide Di Ruscio[1], Ivano Malavolta[2], Henry Muccini[1]
[1]DISIM Department, University of L'Aquila, Italy
[2]Vrije Universiteit Amsterdam, The Netherlands
[1]{firstname.lastname}@univaq.it, [2]i.malavolta@vu.nl

ABSTRACT

The adoption of Model-driven Software Engineering (MDSE), to develop complex software systems in key application domains such as automotive and aerospace, is being supported by the maturation of model-driven platforms and tools. However, empirical studies show that challenges to a wider adoption of MDSE technologies still exist. One limiting factor is related to the limited support for *collaborative MDSE*.

This NIER paper makes an overview of collaborative MDSE approaches by considering three peculiar aspects, i.e., model management, collaboration, and communication means. Reflections on emerging results about such dimensions are given and research directions together with related opportunities and challenges are drawn.

1. INTRODUCTION

Terms like the Internet-of-Things, petascale datacenters, Industry 4.0, Autonomous Vehicles continuously pop up in the media. All those terms are proxies for software systems with millions of lines of code, running everyday, 24 hours a day. Said that, it is understandable that the growing complexity and ubiquity of software systems give place to a number of development and management issues that necessarily have to be addressed. In software engineering, software complexity has been addressed from many perspectives, ranging from static or dynamic software verification, domain-specific programming languages, advanced testing techniques, etc.

One of the most pursued strategies for taming complexity in a realistic manner is to use appropriate methods of abstraction [20]. Model-driven software engineering (MDSE) [5, 16] is one of the proposed approaches aiming at taming the complexity of software systems. In MDSE models are considered as first-class entities, which are employed both for descriptive and prescriptive purposes. Many are the success stories of MDSE, ranging from the development of large-scale enterprise Web applications, to clinical data management, and to public authority data interchange [9, 16].

On a different line of research, the prominence of agile methods, open source software projects, and global software development techniques allows software development teams (but also external and non-technical stakeholders like customers and end users) to seamlessly collaborate to engineer complex systems. Such techniques belong to the domain of *collaborative software engineering* (CoSE) [23].

The focus of this paper is on **collaborative model-driven software engineering**, which can be positioned as the intersection between MDSE and CoSE. Collaborative MDSE is gaining a growing interest in both academia and industry [6, 18]. A number of research initiatives are being run to enable large teams of modelers to work on (large) models in a collaborative manner [18], each of them proposing different building blocks and perspectives about collaborative MDSE. Among many, Morsa[1] is a NoSQL-based model repository aimed at client scalability with querying capabilities; EMF-Collab[2] is a light-weight solution to let multiple users edit a single model concurrently; GenMyModel[3] supports the design and sharing of design diagrams in real-time.

In this paper we elicit and explore the main constituent elements of collaborative MDSE. In particular, Section 2 and Section 3 provide the (main) elicited building blocks synthesized from (i) a systematic mapping study on collaborative MDSE [11, 13], (ii) a through discussion occurred during a dedicated workshop[4] held at the 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016), and (iii) the interaction with industrial partners about collaborative MDSE. Section 4 elaborates on visions of the <u>future</u> of collaborative MDSE by discussing emerging needs, challenges, and open issues about the surely intriguing synergies between CoSE and MDSE. Section 5 concludes the paper.

2. COLLABORATIVE MDSE

Model-Driven Software Engineering (MDSE) provides suitable techniques and tools for specifying, manipulating, and analyzing software modeling artifacts including metamodels, models, and transformations. Collaborative MDSE focuses on those approaches in which several distributed technical and/or non-technical stakeholders collaborate to produce models of a software system, working in a local or remote shared workspace, either synchronously or asynchronously. Collaborating stakeholders can include, but are not limited to, technical figures (modelers, designers, developers), domain experts, non-technical managers, customers, and users of software systems.

Collaborative MDSE is gaining growing interests in both academia and practice, and during the last few years several studies scattered across different independent research areas have been exploring the "building blocks", which are required to support collaborative MDSE. Modeling editors, model repositories, model versioning systems, and model differencing are only few examples of techniques and tools that are

[1]http://modelum.es/trac/morsa
[2]http://qgears.com/products/emfcollab
[3]https://www.genmymodel.com
[4]http://cs.gssi.infn.it/commitmde2016

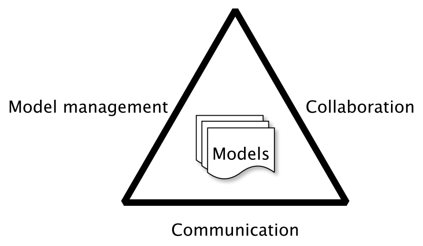

Figure 1: Collaborative MDSE dimensions

required when employing collaborative MDSE approaches. Starting from this body of knowledge, it is possible to identify the following three complementary dimensions, which can be considered as the peculiar aspects building up a collaborative MDSE approach (see Figure 1) [13]:

▷ *model management infrastructures* for managing the life cycle of models (e.g., modeling tools, modeling repositories);

▷ *collaboration means* for allowing involved stakeholders to work on the modelling artifacts collaboratively (e.g., model versioning systems, model merging mechanisms, systems for visualizing and managing model conflicts);

▷ *communication means* for allowing involved stakeholders to be aware of what the other stakeholders collaborating with them are doing (e.g., chat, social network, wiki, asynchronous messaging system, issue tracker).

In the following, after an overview and some reflections on emerging results about the three dimensions shown in Fig. 1, research directions and related opportunities that we envision about this topic are drawn.

3. REFLECTIONS ON THE PAST

How are stakeholders involved in the collaborative specification of modeling artifacts? What are the technological means currently available for supporting collaborative processes? How can collaborating stakeholders communicate about the changes to be performed on the model being developed? This section answers such questions by drawing an overview of the techniques and tools currently available to support collaborative MDSE.

3.1 Model Management

Model management provides the infrastructure for managing the life cycle of models. It includes a (possibly distributed) repository for managing the persistence of the models and their related metadata, a modelling tool [6] for creating, editing, or deleting models, interchange formats for sharing models across projects and organizations, etc.

In the last few years the modeling community started to face several model management challenges, especially from a collaborative point of view. One of these challenges is to lower the technical entry barriers to the modeling process: embedding the modeling environment into the *web-browser*, for instance, is one of the proposed solutions [22, 25], so there is no need to install software. There are also approaches that try to involve non-technical stakeholders even into the *metamodeling activity*, like FlexiSketch [28] and Collaboro [17]. In particular, FlexiSketch gives the users the possibility to sketch free-form diagrams and to formalize later the drawings by means of metamodel inference mechanisms. Collaboro allows the collaborative construction of DSMLs through example-driven techniques.

Beside these initial explorations of the model management dimension, several challenges due to the emergence of delocalized users working on shared resources still remain to be properly addressed. Examples of such issues are security, usability, scalability, management of multi-view specification, etc. We detail our vision about these aspects in Section 4.

3.2 Collaboration

The kinds of collaboration that stakeholders can employ to work on shared resources can be twofold [10]: (i) **synchronous**, multiple users share one or more modeling artifacts and *concurrent* edit operations performed on the same artifacts are propagated to all participants immediately, in a (near) real-time fashion; (ii) **asynchronous**, multiple users share one or more modeling artifacts, changes are not propagated in real-time to other users, who instead have to explicitly update their local copies similarly to what happens in textual versioning systems like SVN and GIT.

Both kinds of collaboration share relevant issues related to the management of conflicting changes, which necessarily have to be detected and reconciled. In the last decade, both industry and academia have invested a lot of effort on *conflict management* and on *model versioning* systems especially employed in asynchronous collaborative scenarios. For an exhaustive model versioning approaches survey and details of the different aspects, please refer to [1].

Different attempts to deal with specific synchronous collaboration issues have been also investigated: further than conflict management aspects, it is relevant to manage the propagation of editing operations by considering also possible network problems. Locking mechanisms [25] or notification systems [22] are just two possible tentative solutions for a still open research problem.

3.3 Communication

Collaboration intrinsically implies involvement of a (potentially heterogeneous) set of stakeholders that need communication support to be aware of each others' work and to coordinate their own editing actions. Communication dimension has two main aspects: (i) **workspace awareness**, defined as the up-to-the-moment understanding of other stakeholders' interaction with a shared workspace [15], and (ii) **communication means**, the set of support mechanisms and associated tools for the unstructured natural language communications between stakeholders [27]. Historically, software engineers have adopted a wide range of *communication tools and technologies*, such as telephone, teleconferences, voice mail, instant messaging, VoIP, etc. [27].

In the context of collaborative MDSE, besides these "traditional" communication means, the concept of workspace awareness plays a key role: with the term *workspace* we refer to the modelling environment provided by the collaborative MDSE approach (e.g., the editor in which stakeholders create and modify models) and all its additional facilities like shared dashboards, integration with wiki-based engines, or special notification areas in the adopted tool. As discussed in the next section, additional effort is required in the context of collaborative MDSE to properly integrate communication means with workspace awareness.

4. VISIONS OF THE FUTURE

In this section we make an overview of the needs, challenges, and open issues we envision for the future of collaborative MDSE. Such findings are built on top of academic knowledge and the analysis of current practices (coming from the systematic study, the workshop, and the analysis of ongoing projects and initiatives). In the following we details the envisioned aspects with respect to the three collaborative MDSE dimensions, concluding with (one of) the main orthogonal findings.

4.1 Model Management

Modeling one single small model by means of a standalone tool is surely a thing from the past. In the future we envision a *distributed modeling environment* in which *multiple stakeholders* of different types (e.g., developers, end users, sales) edit models, possibly in *real-time.* This scenario opens for many challenges, such as the support of *lightweight, Web-based modeling environments* (it is unthinkable to ask a team member to install a modeling environment locally today), accessible by following a *multi-device experience*, which facilitates collaboration in *mobility.*

High-volumes of data are generated by everything around us everyday, meaning that models of this data (potentially generated at run-time) will follow the same trend in the very recent future. Even if there are some approaches advocating the capability of managing large scale models [2, 19], we will need more advanced techniques to store, maintain, and edit models in a *scalable* manner, and *cloud-based services* for manipulating or dynamically extracting parts of them (loading and editing a full model on the client side is just a thing from the past).

Initial steps in these directions include: partial loading of models [26], model pre-fetching [8], incremental model transformations [24], browser-based model editors with cloud services [14], etc. A good point of inspiration in a different domain is the Eclipse Che project[5]. It is an on-demand, Web-based collaborative IDE where the tools, code dependencies, and the workspace can be dynamically provisioned. It has a microservice-based backend and a plug-in based architecture.

4.2 Collaboration

More than ten years ago we all have been amazed by the real-time collaborative features of Google Docs. In the future, we envision a similar revolution in MDSE: different stakeholders will collaborate either in *real-time* or in an *asynchronous* manner on shared models, which will belong to shared *ecosystems of modeling artifacts.* Models in those ecosystems will be *queried*, edited, *mapped* to each other, *transformed in a light-weight manner*, or even injected into the running system (e.g., via *model interpretation or compilation*). In this scenario it will be fundamental to have *inconsistency-tolerant environments* for making collaboration smooth, *model repositories* with extraordinary capacity and performance, advanced engines for *detecting and resolving inconsistencies*, *branching* and *merging* capabilities, possibly with as much automation as possible.

Interesting initial steps in this directions include: inconsistency management via process transformation, access control rules at the model element level [4], incremental bidirectional model transformations [24], model indexing and efficient querying [7], NoSQL databases for model repositories with nearly instant read/write time [3].

Also, when talking about collaborative modeling we are implicitly bringing up the concept of *multi-view modeling*, where multiple stakeholders may be working on multiple views of the same system at the same time. In turn, multi-view modeling brings up the well-known tension between the *projective* and the *synthetic* multi-view principles. If on one side projective approaches may simplify collaboration (mainly because the common metamodel is the only source or target for the enactment of editing operations), synthetic approaches may introduce accidental complexity (mainly due to the potentially large number of interdependences among modeling artifacts). In any case, each multi-view principle may be embraced for many other reasons, either organizational or technical, thus additional scientific effort must be devoted to both of them, even by suitably combining them. As an example, first results in this direction have been proposed with the orthographic modeling paradigm at the basis of the Vitruvius approach [21].

[5]http://www.eclipse.org/che

4.3 Communication

As discussed in Section 3, in the past the vast majority of collaborative MDSE approaches assumed that only technical stakeholders collaborate on models. However, software modelling will be performed by a team consisting of several members each with *very different technical knowledge and background.* Without proper communication means, this situation may lead to mismatches in the used language, understanding of the to-be system, additional design and development iterations, thus resulting in (unforeseen) waste of resources, budget and time.

Under this perspective, we see a big research gap in terms of *support for collaboration between technical and non- technical stakeholders.* This trend will be more evident in domains where soft skills are more needed; for instance, stakeholders that are typically involved in the development of mobile applications include user experience designers, graphic designers, information architects, developers, users, and customers [12].

So far many collaborative MDSE approaches are performing well in terms of *workspace awareness*, however there is still room for improvement here, especially for synchronous approaches, where modeling elements cannot simply disappear or move within the editing environment. In the future we will see advanced facilities for workspace awareness in terms of *who* is performing an editing operation (e.g., role-based editing) on *what* model or modeling element (e.g, real-time model updates), and *why* the change has been performed (e.g., quick recording of design decisions and their rationale). As an example, WebGME [22] supports well the *what* dimension via tool status notifications and real-time updates in the modelling editor, but those updates just "appear" in the editor without any indication about who made those updates on which parts of the models.

Finally, it will be fundamental to ensure *traceability* between the design decisions discussed in communication- oriented contents (e.g., the text of a chat, the page of a Wiki, or a recorded design decision) and modeling artifacts (e.g., a model, a specific model element, or even previous versions of a model). Indeed, those (potentially typed or structured) links will be a precious asset for reusing technical and non-technical knowledge about the models developed within large software organizations.

4.4 Main Orthogonal Challenge

During the previously described analysis and reasoning we came across a common future challenge, which is orthogonal

Flexibility aspects
Support for importing, linking, and editing models conforming to several and integrated domain-specific modeling languages
Pluggable external editors to edit modelling artifacts (e.g., models, metamodels)
Support for including an open set of plug-ins providing vertical functionalities (e.g., for performing a specific type of analysis)
Traceability engines for external artifacts, possibly with links at the model element level
Concrete syntax redefinition or customization
Pluggable domain-specific conflict detection/resolution engines
Pluggable domain-specific consistency checkers, both within and across models

Table 1: Flexibility aspects for future collaborative MDSE approaches

to all the dimensions previously discussed: **flexibility** shall be a cornerstone of any collaborative MDSE approach. It is difficult to figure out a totally generic collaborative MDSE approach that perfectly fits any application domain (e.g., automotive vs Web apps), any organization (large corporates vs startup-like companies), any development process (certification-oriented development vs SCRUM-based agile development), etc. In this respect, Table 1 shows the identified emerging types of flexibility.

Modular and plugin-based architectures will surely play a role when architecting a collaborative MDSE tool. We envision flexible and agile modeling infrastructures where any stakeholder will be able to add, customize, or remove the functionalities of the modeling environment at wish, thus adapting it to the (ever-changing) specific project and organizational needs.

5. CONCLUSIONS

Collaborative MDSE envisions a new way to develop complex systems by putting in synergy collaborative software engineering and model-driven software engineering. Even though the benefits related to the adoption of collaborative MDSE are evident, it is far from clear about how to get the best out of it. In this paper, we analyzed three peculiar aspects underpinning collaborative MDSE, identified related opportunities and challenges.

References

[1] K. Altmanninger, M. Seidl, and M. Wimmer. A survey on model versioning approaches. *Int. Journal of Web Information Systems*, 5(3):271–304, 2009.

[2] F. Basciani et al. MDEForge: an Extensible Web-Based Modeling Platform. In *Procs. of the 2nd CloudMDE@MoDELS*, pages 66–75, 2014.

[3] A. Benelallam et al. Neo4emf, a scalable persistence layer for emf models. In *European Conf. on Modelling Foundations and Applications*, pages 230–241. Springer, 2014.

[4] G. Bergmann et al. Query-based access control for secure collaborative modeling using bidirectional transformations. In *Procs of the 19th Int. Conf. on MoDELS*. ACM, 2016.

[5] J. Bézivin. On the unification power of models. *Softw Syst Model*, 4(2):171–188, may 2005.

[6] M. Brambilla, J. Cabot, and M. Wimmer. *Model-driven software engineering in practice*, volume 1. Morgan & Claypool Publishers, 2012.

[7] G. Daniel, G. Sunyé, and J. Cabot. Mogwaï: a framework to handle complex queries on large models. In *IEEE Int. Conf. on Research Challenges in Information Science*, 2016.

[8] G. Daniel, G. Sunyé, and J. Cabot. Prefetchml: a framework for prefetching and caching models. In *Procs of the ACM/IEEE 19th Int. Conf. on MoDELS*. ACM, 2016.

[9] D. Di Ruscio, R. F. Paige, and A. Pierantonio. Guest Editorial to the Special Issue on Success Stories in Model Driven Engineering. *Sci. Comput. Program.*, 89(PB):69–70, 2014.

[10] C. A. Ellis, S. J. Gibbs, and G. Rein. Groupware: Some issues and experiences. *Commun. ACM*, 34(1), Jan. 1991.

[11] M. Franzago et al. Protocol for a Systematic Mapping Study on Collaborative Model-Driven Software Engineering. Technical Report TR-001-2016, DISIM - University of L'Aquila, 2016. URL https://arxiv.org/pdf/1611.02619.pdf.

[12] M. Franzago, H. Muccini, and I. Malavolta. Towards a collaborative framework for the design and development of data-intensive mobile applications. In *Procs of MOBILESoft 2014, Hyderabad, India*, pages 58–61, 2014.

[13] M. Franzago, D. D. Ruscio, I. Malavolta, and H. Muccini. Collaborative model-driven software engineering: a classification framework and a research map. *IEEE Transactions on Software Engineering*, PP(99):1–1, 2017. URL https://tinyurl.com/ya5srlvd.

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

[15] C. Gutwin and S. Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)*, 2002.

[16] J. Hutchinson, M. Rouncefield, and J. Whittle. Model-driven engineering practices in industry. In *2011 33rd Int. Conf. on Software Engineering (ICSE)*, pages 633–642. IEEE, 2011.

[17] J. L. C. Izquierdo et al. Engaging end-users in the collaborative development of domain-specific modelling languages. In *Int. Conf. on Cooperative Design, Visualization and Engineering*, pages 101–110. Springer, 2013.

[18] D. S. Kolovos et al. A Research Roadmap Towards Achieving Scalability in Model Driven Engineering. In *Procs of the Workshop on Scalability in Model Driven Engineering*, BigMDE '13, New York, NY, USA, 2013. ACM.

[19] D. S. Kolovos et al. MONDO: scalable modelling and model management on the cloud. In *Joint Procs of the Doctoral Symposium and Projects Showcase at STAF 2016*, 2016.

[20] J. Kramer. Is abstraction the key to computing? *Communications of the ACM*, 50(4):36–42, 2007.

[21] M. E. Kramer, E. Burger, and M. Langhammer. View-centric engineering with synchronized heterogeneous models. In *Procs of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*, page 5. ACM, 2013.

[22] M. Maróti et al. Next generation (meta) modeling: Web- and cloud-based collaborative tool infrastructure. In *MPM@ MoDELS*, pages 41–60, 2014.

[23] I. Mistrík et al. Collaborative software engineering: Challenges and prospects. In *Collaborative Software Engineering*. Springer, 2010.

[24] O. Semeráth et al. Iterative and incremental model generation by logic solvers. In *Int. Conf. on Fundamental Approaches to Software Engineering*. Springer, 2016.

[25] C. Thum, M. Schwind, and M. Schader. Slim: A lightweight environment for synchronous collaborative modeling. In *Int. Conf. on Model Driven Engineering Languages and Systems*, pages 137–151. Springer, 2009.

[26] R. Wei et al. Partial loading of xmi models. In *Procs of the ACM/IEEE 19th Int. Conf. on MoDELS*. ACM, 2016.

[27] J. Whitehead, I. Mistrík, J. Grundy, and A. Van der Hoek. Collaborative software engineering: concepts and techniques. In *Collaborative Software Engineering*. Springer, 2010.

[28] D. Wuest, N. Seyff, and M. Glinz. Flexisketch team: Collaborative sketching and notation creation on the fly. In *IEEE/ACM 37th ICSE*, 2015.