Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 100 (2016) 221 – 228

Procedia Computer Science

Conference on ENTERprise Information Systems / International Conference on Project MANagement / Conference on Health and Social Care Information Systems and Technologies, CENTERIS / ProjMAN / HCist 2016, October 5-7, 2016

Analysis of Asset Specification Languages for Representation of Descriptive Data from MDE Artifacts

Fábio P. Basso[a*], Toacy C. Oliveira[a], Cláudia M. L. Werner[a], Rafael Z. Frantz[b]

[a]*COPPE, Federal University of Rio de Janeiro. Av. Horácio Macedo, 2030. Rio de Janeiro, RJ, Brazil*
[b]*Department of Exact Sciences and Engineering, Unijuí University. Rua do Comércio, 3000. Ijuí 98700-000, RS, Brazil.*

Abstract

In order to improve the adoption of Model-Driven Engineering (MDE) Artifacts (e.g., design languages and associated model transformations), the literature points out that these elements need to be first shared in Knowledge Bases (KB) to be further downloaded, compared and integrated to software projects. A common concept used as pivotal representation between software artifacts and repositories is the Asset: structured information that provides standard taxonomies to catalog artifacts from different natures. In this work, we evaluated the applicability of concepts from two important asset specification languages on the representation of MDE artifacts: The Asset Management Specification (AMS) and the Reusable Asset Specification (RAS). Our contribution is a list of benefits and drawbacks from RAS and AMS to work as a pivotal language in KBs for MDE.

Peer-review under responsibility of the organizing committee of CENTERIS 2016

Keywords: Pivotal Language; Asset; Globalization of DSL; Model-Driven Engineering; Artifacts.

1. Introduction

Artifacts for Model-Driven Engineering (MDE), also referred as MDE settings[1], aim at producing software using (semi-) automatic tasks among many phases of software development processes. Examples are model transformations and Domain-Specific Languages (DSLs)[2]. Many of these artifacts have been developed for specific

* Corresponding author. Tel.: ++55 21 2562 8672
E-mail address: fabiopbasso@cos.ufrj.br

Peer-review under responsibility of the organizing committee of CENTERIS 2016
doi:10.1016/j.procs.2016.09.145

needs along more than 15 years. The options of artifacts to include in an MDE-based process has grown to the point that it is very difficult to decide which option meets specific needs[5]. Currently, all the information that could benefit the industry in applying MDE in practice is distributed among many documents, requiring a long time from software engineers to reach those options that best meet specific needs. Thus, in order to facilitate the access and retrieval of data associated with these artifacts, the literature of the area pointed out to the need of a Knowledge Base (KB)[6,14].

A possible solution to publish and download content from artifacts into a KB is through assets[7]. An asset is anything that provides reuse and value through a reference (link), cataloged with taxonomies, described by a set of properties and owning zero or more data about artifacts[8]. Assets have been used to describe software components[9], application and domain models[10] and, more recently, tools on the cloud in the context of MDE[11]. Thus, assets are important for playing a pivotal role between artifacts reusers (clients) and service providers (KBs for MDE).

We present an analytical study on the feasibility of the current technological support for a pivotal representation language for MDE. For instance, this language should be used together with KBs. Likewise, we executed an analysis of benefits and limitations in existing asset specification languages: Reusable Asset Specification (RAS)[7] and the Asset Management Specification (AMS)[8]. We compared these specifications, representing 37 assets for each language using two DSLs generated with the Eclipse EMF. This means that these assets are models. At the end, we present research gaps that increment our understanding about the role of a pivotal language, with issues in the state-of-art for conception of a KB for MDE. Although technical data is essential for artifacts shared in KB for MDE[6,14], they are discussed elsewhere. Thus, due to the lack of contributions discussing the needed semantics associated with MDE Artifacts, this paper considered only the representation of descriptive data, including: 1) catalog information for searching DSLs based on standard data; 2) instructive information about how to use and adapt existing components and; 3) information for decision making to one decides the best option for DSLs after a search in a KB.

Next sections present the result of our analysis and are organized as follows: Section 2 provides background on the role of a pivotal language for MDE Artifacts and Section 3 introduces asset specifications; Section 4 demonstrates RAS and AMS on the modeling of descriptive information associated with MDE Artifacts, allowing the evaluation of the representativeness of these languages. Benefits and drawbacks are discussed in Section 5, complemented in Section 6 with a discussion of limitations of RAS and AMS to be explored in future research; Section 7 highlights research gaps for long-term investigations; and, Section 8 presents our conclusions.

2. The Role of a Pivotal Language for a KB for MDE Artifacts

The need for a knowledge base for MDE artifacts is recently discussed in the literature[5, 6]. Authors claim that a knowledge base is necessary for end-users to easily find and comprehend what is necessary to introduce artifacts in practice. A KB is a solution to reduce costs in producing new DSLs, reusing those already produced and shared for free by the MDE community. In this regard, the initiative initially called Globalization of Models of Computation (GEMOC, <http://gemoc.org/>), proposes that the community makes an effort for sharing artifacts such as DSLs, model transformations, scripts, tools and others in repositories[14]. Not much is understood about the requirements for the implementation of this global reuse scenario and, as the authors claimed, there are more questions than answers[14]. In this sense, we are investigating the role of pivotal representation languages for this reuse scenario. Pivot languages discussed in the next section connect clients with repositories through links and descriptions.

Representation/structures from repositories are related research topics, as illustrated in Fig.1. The Repository for Model Driven Development (ReMoDD)[15] is our current alternative for a KB. It allows, for example, the use of searches through a web platform about some artifacts associated with proposals for DSLs published in some conferences such as MODELS, ECMFA, etc. Most data that should be explicit in repository structures is available as physical files, papers and tutorials, which requires a long time to find and analyze adequate options for MDE Artifacts. Several other options for MDE repositories have been proposed in the last years, such as MDEForge[17]. Another KB that gained attention in the last years is the SEMAT[16] - an initiative in Software Engineering that starts from the principle that stored models for methods are represented with a DSL named Essence[16]. Differently, GEMOC repository and MDEForge are for any type of MDE artifacts[14], thus complementary to SEMAT.

These proposals are unconnected. A pivotal language for MDE could connect clients with many KBs, but we miss this language and its requirements. We found in a previous literature review that the closer in the state-of-art from the conception of this type of language are two asset specifications: RAS and AMS.

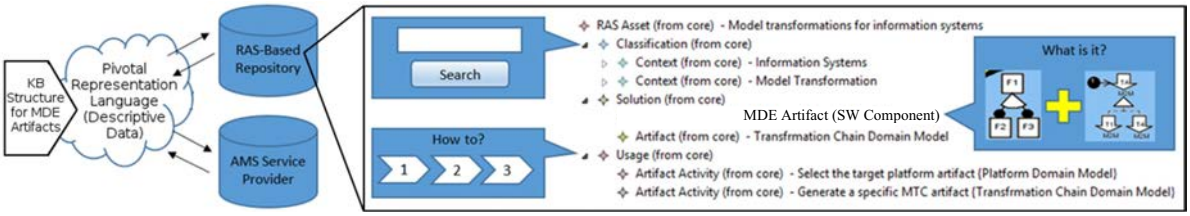

Fig. 1. Information represented in a reusable asset conforms to RAS and its descriptive role to a KB for MDE Artifacts.

So far, the acknowledged role of a pivotal language is to allow automatically publishing and downloading the data independently from the database structure adopted to store the artifacts. It also must represent descriptive information associated with artifacts in a structured and common format. However, a global reuse scenario for MDE is far more complex and needs specific analysis about the state-of-art. In this paper, we investigate two open questions: *Q1) Do RAS and AMS fulfill the needs for descriptive information from these KBs? Q2) Which are the research gaps needed to be investigated for the conception of this language within this emergent reuse scenario?*

Our contribution answers these questions. Based on the analysis of KBs, we suggest some descriptive data to represent in a pivotal language, showing how to specify them in assets. Then, we analyze the benefits and limitations in RAS and AMS, introducing some requirements for a "pivotal language" and associating specific research gaps in the area.

3. Asset Specification Languages

RAS is an OMG (Object Management Group) standard to classify, catalog and instruct the reuse of software artifacts in reuse repositories[7]. Moreover, reusable assets provide data that allows specifying, store, retrieving diverse artifacts used in software engineering processes[9] with meta-classes shown in Fig.2 (A). The goal is to detail instructive data associated with artifacts, used by end-users to learn about what should be adapted in software artifacts for different needs in software projects. Fig.1 exemplifies an asset for a scenario where RAS specifications are currently used: *to describe information on how to reuse some software component*. It describes a domain model conforms to FOMDA DSL[19], used to generate code for information systems[18]. So, the represented artifact is used by this DSL to adapt code generators represented for target platforms (e.g., to generate code for different Java APIs)[19].

We investigate the feasibility of RAS and AMS to represent data associated with FOMDA DSL (the novelty), not for this domain model (already explored and accepted in reuse practices). Anyway, in order to promote the reuse of this software artifacts in future software projects, it is always necessary to provide adequate information used in specific moments in a reuse process[7]. Thus, Fig.2 shows that an asset provides structure for descriptive information that can be used at different moments in between acquisitions from repositories (*Classification*) to the integration of the artifacts in target software projects (*Usage*). For example, part of the information provided in an asset can be used for cataloging and searching in a repository (see *Classification*) and another part is used for guiding (see *Usage*) how to adapt and integrate the artifacts represented in the *Solution* structure in a target software project. Thus, structure promoted in reusable assets is important to localize the correct information according to reuse steps.

Another option to specify an asset is AMS[8], adopted to catalog tools on clouds. AMS is part of Open Services for Lifecycle Collaboration (OSLC)[8], an industry specification that allows implementing concepts of Software-as-Service (SaaS)[11], classifying tools for MDE through asset specifications. The proposal of OSLC is slightly different from the GEMOC initiative, which beside tools must also share DSLs, model transformations, libraries, etc. So, clients must search in KBs for data that is not only limited for web services or integration ports. On the other hand, for the problem motivated in this work, the AMS allows to represent descriptive data shown in Fig.2 (B).

RAS and AMS have a role as a pivotal language for artifacts. For these representations, an artifact can be a tool, a DSL, a model transformation and others. This is the main reason why they are interesting to be used as a pivot for the implementation of a global reuse scenario. The question is whether they are enough or if need to be extended.

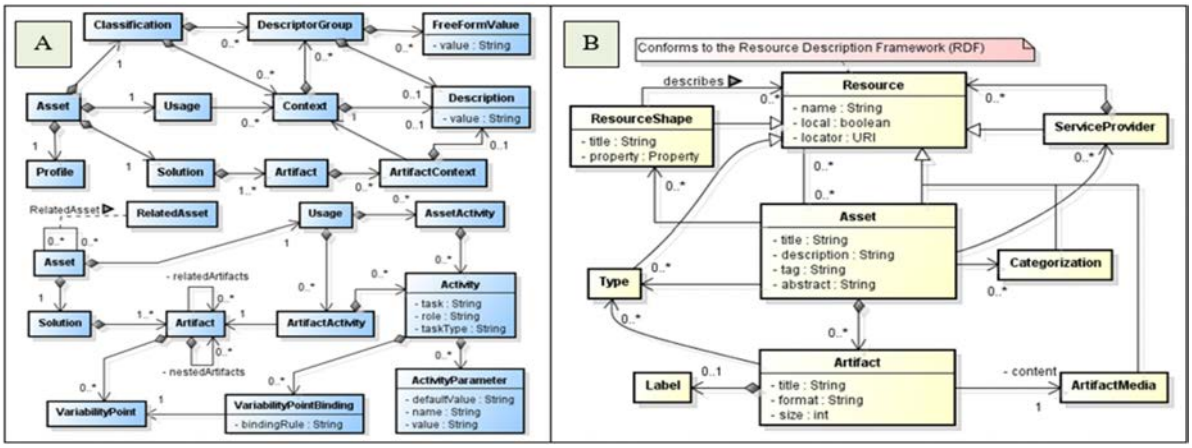

Fig. 2. Main metaclasses from RAS to represent descriptive information.

4. Evaluation of the Representativeness of RAS and AMS

In order to understand the applicability of assets to represent information on KBs for diverse MDE artifacts, we conducted a case study, representing 37 assets for some DSLs and tools for Software Product Lines (SPL)[12] found in the literature of the area. For the comparison of RAS and AMS, we adopted the criteria of representatives and structure available for descriptive information, as well as the support for local and distributed repositories that will store this information. For example, AMS is less structured than RAS. Besides, the RAS metamodel shown in Fig.2 (A) is richer with descriptive information than the one available in AMS shown in Fig.2 (B). RAS and AMS provide an element for classification, which can receive data about the contexts in which the asset is applied. This information is exemplified in the asset shown in Fig.3 (A), represented in AMS. Fig.3 (B) shows an asset specified with the standard RAS, which allows representing catalog with much more detail than those allowed by AMS.

AMS is restricted to keyword search. However, more than keyword-based search is needed. Hence, the example shown in Fig.3 (B) adds two new elements: descriptor groups and free form values, only available in RAS. Descriptors groups are used to add structured textual data to better describe a context associated with an asset. Free form values, besides a name, have a property named *value* in which long texts can be used, such as details about each pre and post condition shown in Fig.3 (B). Therefore, it is possible to specify the needed descriptive data.

We found that RAS is interesting to represent useful data to decide if a DSL or tool should be used, in which contexts they should be used, among other descriptive information, while AMS is limited to keyword search. Both specifications provide a mean to contextualize artifacts for MDE, i.e., to provide a classification and description. However, only in RAS it is possible to represent rich and structured textual content.

Fig.2 exemplified the *Solution* structure, the content-part of an asset, which is usually applied to describe software artifacts (e.g., models, components, source code), but also DSLs as illustrates Fig.4. For a DSL, an asset can have information associated with configuration files, APIs and libraries, binaries and scripts, metamodels and model transformations, among others. Besides, the specifications allow to represent artifacts and tasks (this one only in RAS), but with a very limited set of meta-classes that allows to represent the type of artifacts and tasks. However, current repositories for MDE are also limited in this sense[13], allowing extensions through similar mechanisms.

As illustrated in Fig.2, in assets that describe reusable artifacts for software components it is common to associate instruction for reuse. This is something very important for model transformations, which can be adapted by end-users[5]. However, it is not so obvious what should be represented as instruction in an asset for a DSL. Thus, we suggest that for DSLs, a pivotal language should instruct how to configure a set of libraries, components and files.

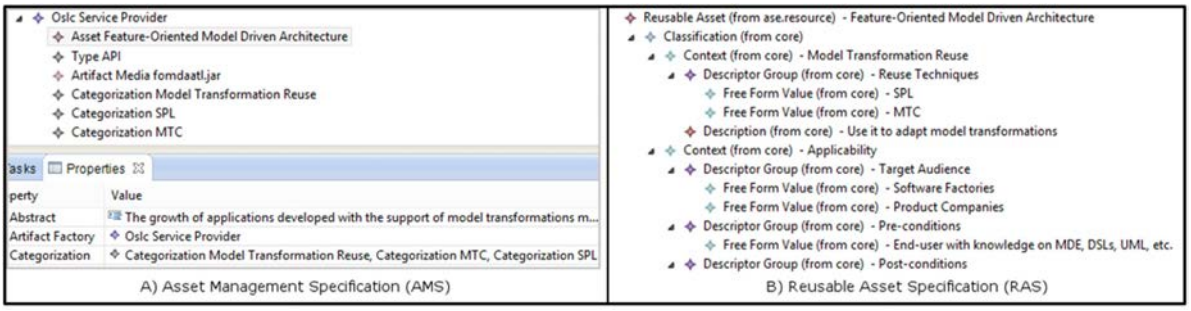

Fig. 3. Classification of a technical solution named FOMDA with AMS and RAS.

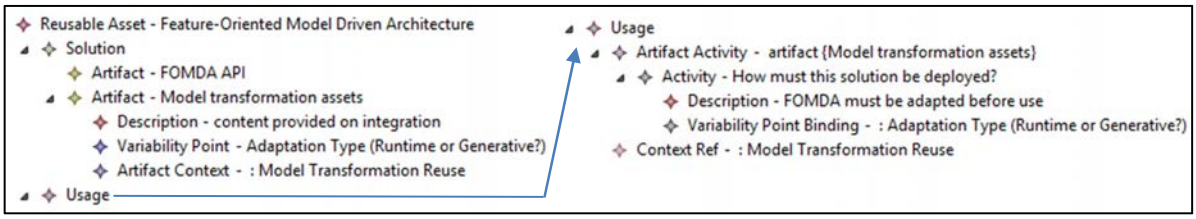

Fig. 4. Content and instruction associated with the DSL named FOMDA.

In the bottom-part of Fig.2 (A) is illustrated some of the elements provided in RAS to support such an instructive information. An artifact can receive descriptive information about variability points that highlight the main features that can be changed in that artifact. For example, model transformations can be configured to support runtime or generative adaptations over those artifacts detailed in Fig.4. This is a benefit that allows end-users understand how MDE Artifacts are built and how to change it. In RAS, the instruction on how to proceed to apply an adaptation is specified in activities in the structure named *Usage* shown in Fig.4, which illustrates the additional metaclasses from RAS not found in AMS. In this case, activities must associate a variability point binding to indicate that it is intended to support a given variability point from some artifact. Moreover, activities in RAS can receive data about the *Role* enabled to perform a given task, thus instructing how one can use the DSL or tool.

With RAS it is possible to instruct the necessary adaptations in resources through activities. Diverse activities such as how to and guidelines can be represented with instances of *Activity*. In addition, an activity can have many variability point bindings, meaning that they are instructing exclusively adaptations needed in assets. Thus, the metaclasses *Activity* and *VariabilityPointBinding* are used to guide the end-user towards adaptations in artifacts.

5. Benefits and Limitations for a Pivotal Language for MDE Artifacts

This section answers ***Q1) Do RAS and AMS fulfill the needs for descriptive information from these KBs?***

In summary, we found that both specifications provide cataloging information and can link to resources (e.g., APIs, binaries, model transformations, etc.) through artifact abstractions as required by existing proposals (ReMoDD, GEMOC, SEMAT, MDEForge …). However, RAS also adds extra-information in comparison to AMS (the *Usage*). In the following, we discuss the results of our analysis with regard to their associated benefits.

Benefits: *1) Structured descriptive information*- A visible benefit is the clarity of the information provided in reusable assets due to its structure; *2) Abstraction of a real physical content*- It is important to notice that for AMS and RAS specifications, the content (e.g., an artifact or a task) is just a meta-information that describes the real data. For example, an artifact is a link to a physical file stored in some place through URLs; *3) Interchange of information*- Both specifications allow to serialize information in XML, which can be used to post an asset into a

repository or to download it. This is important if we consider that different software engineers will use different repositories to store their assets, requiring a common representation protocol; *4) Independence from a KB/repository vendor*- In the case of artifacts described by AMS, they are specified in RDF/XML, for which open-source tools to post and download are already available under the Apache license. Thus, one can change the service provider that hosts the asset without losing information; *5) Distributed information in KBs*- Assets could enable a distributed KB for MDE resources through dependence links proposed by AMS, which contain information about the service provider; and *6) Applicable to describe many MDE Artifacts*- Any model transformation or even techniques and didactic materials for Software Engineering in general are include, i.e., some of initiatives are using assets specified with AMS to describe tools provided in the cloud such as database management systems, application servers and custom applications[11].

Limitations: *1) Lack of structured information to abstract artifacts associated with technical solutions for MDE* - Important information, usually associated with it, such as the metamodel and the serialization language, should be expressed in an appropriate structure[1]; and *2) Lack of meta-classes to represent artifacts and tasks in technical-level*- It is important to include in assets also the information associated with metamodels, model transformations and chains[3]. Asset specifications have no support for this type of information associated with model-based components. These limitations are discussed in Section 7 with research gaps.

6. Research Gaps for Conception of a Pivotal Representation Language Considering Descriptive Data

Specifications provide rich structures to describe MDE artifacts that are currently represented in some repositories. Both specifications present some differences and overlapping in classification, used to catalog and retrieve artifacts associated with the described DSLs for MDE. We reasoned that RAS is richer in meta-information than the AMS is, such as descriptor groups, free form values (i.e., long texts) and instructive data (i.e., activities). However, structures for descriptive information can be improved. Thus, this Section answers part of ***Q2) Which are the research gaps needed to be investigated for the conception of this language for this emergent reuse scenario?***

Both specifications have some limitations for MDE Artifacts that hamper their use in this context. Accordingly, it is possible to use the default versions of these languages, but acknowledging their limitations. We believe that neither RAS nor AMS are ready to represent the technical data from MDE artifacts such as settings/relationships between models, metamodels and transformations. RAS and AMS lack more technical-level information associated with components from MDE-based processes. This means that the data associated with artifacts and activities are only descriptive, thus needing adequate structures to represent semantics for MTs.

The structure for descriptive information should be based on standard data. RAS and AMS are too general specifications, allowing classification with any data. We miss adequate structures for classifying MDE artifacts, i.e., a contribution explores taxonomy for model transformations[21], but none describes taxonomy for types of DSLs and tools used in specific phases of software engineering. Thus, standard data should be explored by future researches.

Lack of structure for comparative reasons. Existing structures for descriptive data are for instruction and classification, thus not adequate to analytically compare assets. Qualified information associated with DSLs, such as return-on-investment (ROI), empirical data on adoption, minimum requirements for use, and others are needed[5]. Thus, we have been exploring, in new extensions for RAS++[20], our proposal for a pivotal language that includes the following new structures: a) *UsageCondition*, which includes representation of pre and post conditions, team skills and taxonomy; and b) *DecisionSupport*, which includes ROI, associated with adoption data and applicability.

7. Research Gaps for Conception of a Pivotal Representation Language Considering MDE Settings

This Section complements the above answers with research gaps from a technical perspective towards the representation of MDE Artifacts and Settings. Likewise, several proposals for connection MDE artifacts have emerged as DSLs for MDE Settings[1]. Examples are contributions for Model Transformation Chain (MTC)[3], Component Model for Model Transformation (CMMT)[2] and other concepts for processes[1] and reuse[19]. However, little is acknowledged about requirements for representations of disconnected artifacts, shared and reused independently from processes and component connectors. In this sense, our proposal for roles from a pivotal language includes[20]: 1) The specification of information from assets in a centralized KB that manages searches from

arbitrary repositories; 2) In a second moment, a client acquires these packages through a central KB (automatically or through a web front end), comparing each information associated with artifacts to decide the more adequate option for a specific context of MDE; and 3) In a third moment the content of the selected packages are downloaded from specific repositories and connected with a specific language for MDE Settings[19].

The first role is full of options for model and reuse repositories[11-15], but they are limited for the management of distributed information as required in a KB. The third scope is full of DSLs for MDE Settings and tool support for execution of components, chains and reuse operations[1-3,19], but they have no common concepts for a pivot language. Likewise, our research indicates is that the state-of-art for repositories[11-15] and DSLs for MDE Settings[1-3] are not enough to implement this emergent scenario for globalization of DSLs[14]. There is a vacuum in the state-of-art that imposes limitations for the advent of a common KB for MDE: *there is no standard/common representation for Artifacts and their Settings conceptualized as pivotal.* Besides, we miss requirements for a pivotal language, thus finding these needs is critical for the advent of this pivot language.

We found some technical needs as follow:

Repositories: To the best of our understanding, in general, important information associated with these artifacts is not shared in repositories such as ReMoDD[17], SEMAT[16] and MDEForge[15], opening following research gaps: *A)* Variability points and instructions are relevant, artifacts stored in these repositories are possible of adaptation, thus structures from these repositories could be incremented; *B)* The state-of-practice in MDE adopts several repositories to store their physical files, thus a relevant question is whether a KB for MDE should stores only information associated with artifacts (the focus of a pivotal language[20]) instead of physical files (the focus of repositories[17])? and *C)* Artifacts should be federated between several possible locations. However, the state-of-art is limited in this regard, thus opening a research question on which representations are needed in a KB and/or a pivotal language to federate the distribution of assets and artifacts?

Assets: The lack of a standard representation of technical data will imply in a manual the effort to publish, search and download these artifacts from KBs. We believe that this can make these initiatives for global reuse inefficient. Thus, the following research gaps are associated: *D)* We found indispensable MDE Settings in a pivotal representation, needing an investigation on what is common between Artifacts and DSLs for MDE Settings for conception of RAS++[20] … what else is needed? *E)* Currently, we need to perform a manual integration of content of artifacts retrieved from repositories with some target representations (MTCs or software process specifications), thus an open question is whether is possible to do it automatically through a pivotal representation? and *F)* not every integration of content can be automated. Meanwhile, instructive data from reusable assets are used for guidance purposes, so that end-users can follow it to adapt and connect Artifacts with Settings. Thus, it is important to evaluate whether the instructive data are essential to actively assist manual configurations.

MDE Settings: In 2007, researchers reported that the lack of critical mass is an issue that hampers a KB for MDE[17]. This issue is still a research gap[15]. We believe that the facilities in tool support can help us to surpass this issue. Thus, open questions include: *G)* Whether a pivot language and tool support help on the automatic upload of data from MDE Artifacts and their Settings? *H)* Would this tool support benefit to introduce Artifacts and Settings in target contexts? *I)* MDE Artifacts have been represented in MDE Setting by highly technical stakeholders[19], thus an open question is which “end-user profile” is suitable to upload the required data in a pivot language? and finally *J)* Which are the benefits and drawbacks from a common representation for the state-of-practice in MDE?

8. Concluding Remarks

The lack of attention on the details for the descriptive information associated with MDE Artifacts can lead to a wrong assumption that any textual data is enough for the globalizations of DSLs. This is bad to understand the requirements for a pivotal representation language to be used in the core of KBs for MDE. This work contributes with an analysis of descriptive data allowed by two pivotal representation languages: AMS and RAS.

In order to reach some requirements for a pivotal language to be used together with a KB, we represented 37 assets describing DSLs and tools for SPL with these two representations. We concluded that in special RAS can represent all the descriptive information associated with MDE Artifacts, as found in the literature of the area. Thus, although this type of information should be better structured, we concluded that RAS is expressive and provides the basis for new extensions to bridge clients and service providers of MDE Artifacts.

We pointed out to some gaps associated with a pivotal representation language that suggest the need for extension in RAS or AMS. Without these extensions, two reuse scopes remain disconnected: repositories for Artifacts and DSLs for MDE Settings. A global reuse scenario, as motivated in recent researches, needs connected scopes. Thereby, aiming at a future agreement in a pivotal representation, concepts from RAS and AMS should be considered together with extensions for the reported gaps.

References

1. R. Hebig, H. Giese, F. Stallmann, A. Seibel, On the complex nature of mde evolution, in: Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems, 2013, pp. 436-453.
2. J. S. Cuadrado, E. Guerra, J. D. Lara, A component model for model transformations, IEEE Transactions on Software Engineering, 40 (11) (2014) 1042-1060.
3. A. Yie, R. Casallas, D. Deridder, D. Wagelaar, Realizing model transformation chain interoperability. Software & Systems Modeling 11 (1) (2012) 55-75.
4. D. Batory, E. Latimer, M. Azanza, Teaching model driven engineering from a relational database perspective, in: Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems, 2013, pp. 121-137.
5. J.Whittle, J. Hutchinson, M. Rouncefield, H. Burden, R. Heldal, Industrial adoption of model-driven engineering: Are the tools really the problem?, in: Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems, 2013, pp. 1-17.
6. G. Mussbacher, D. Amyot, R. Breu, J.-M. Bruel, B. H. Cheng, P. Collet, B. Combemale, R. B. France, R. Heldal, J. Hill, J. Kienzle, M. Schöttle, F. Steimann, D. Stikkolorum, J. Whittle, The relevance of model-driven engineering thirty years from now, in: Model-Driven Engineering Languages and Systems, 2014, pp. 183-200.
7. RAS reusable asset specification version 2.2 november 2005. URL http://www.omg.org/spec/RAS/
8. Asset management specification. (2015). URL http://open-services.net/wiki/asset-management/OSLC-Asset-Management-2.0-Specification/
9. R. Hong-min, Y. Zhi-ying, Z. Jing-zhou, Design and implementation of ras-based open source software repository, in: Sixth International Conference on Fuzzy Systems and Knowledge Discovery., Vol. 2 of FSKD'09, 2009, pp. 219-223.
10. S. Park, S. Park, V. Sugumaran, Extending reusable asset specification to improve software reuse, in: Proceedings of the 2007 ACM symposium on Applied computing, SAC 23'07, 2007, pp. 1473-1478.
11. M. Elaasar, A. Neal, Integrating modeling tools in the development lifecycle with oslc: A case study, in: Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems, 2013, pp. 154-169.
12. T. Thüm, S. Apel, C. Kästner, I. Schaefer, G. Saake, A classification and survey of analysis strategies for software product lines, ACM Comput. Surv. 47 (1) (2014) 6:1-6:45.
13. F. Basciani, D. D. Ruscio, L. Iovino, and A. Pierantonio. Automated chaining of model transformations with incompatible metamodels. In Model-Driven Engineering Languages and Systems, MODELS'14. 602-618. 2014.
14. B. Combemale, J. Deantoni, B. Baudry, R. France, J.-M Jézéquel, and J. Gray. Globalizing modeling languages IEEE Computer, Institute of Electrical and Electronics Engineers, 47(6):68-71, June 2014.
15. R. France, J. Bieman, and B. Cheng. Repository for model driven development (remodd). In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4364 LNCS, pages 311-317, 2007.
16. I. Jacobson, P.-W. Ng, P. E. McMahon, I. Spence, S. Lidman, The essence of software engineering: The semat kernel. a thinking framework in the form of an actionable kernel., ACMQUEUE. Development 9. Networks 10 (10) (2012) 1-12.
17. J. D. Rocco, D. D. Ruscio, L. Iovino, and A. Pierantonio. Collaborative repositories in model-driven engineering [software technology]. Software, IEEE, 32(3):28-34, May 2015.
18. F. P. Basso, R. M. Pillat, T. C. Oliveira, F. Roos-Frantz, and R. Z. Frantz. Automated design of multi-layered web information systems. Journal of Systems and Software, 117:612-637, 2016.
19. F. P. Basso, R. M. Pillat, T. C. Oliveira, and L. B. Becker. Supporting large scale model transformation reuse. In 12th International Conference on Generative Programming: Concepts & Experiences., GPCE'13, pages 169-178, 2013.
20. F. P. Basso. A proposal for a common representation language for MDE artifacts and settings. In Proceedings of the Doctoral Symposium at Software Technologies: Applications and Foundations 2015 Conference (STAF 2015), L'Aquila, Italy, July 20, 2015., pages 21-31, 2015.
21. L. Lúcio, M. Amrani, J. Dingel, L. Lambers, R. Salay, G. M. Selim, E. Syriani, and M. Wimmer. Model transformation intents and their properties. Software & Systems Modeling, pages 1-38, 2014.